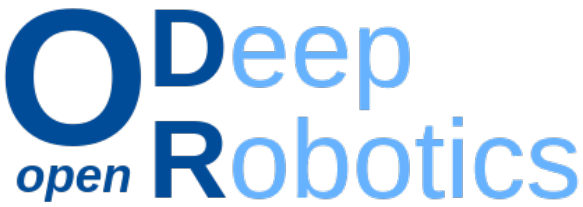


OpenDR — Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D4.2: Second report on deep environment active perception and cognition

Date of delivery: 31 December 2021

Contributing Partners: AU, AUTH, ALU-FR, TUD, TAU, AGI
Version: v2.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	**D4.2: Second report on deep environment active perception and cognition**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Report
Dissemination Level:	**PU**blic
Authors	Vödisch (ALU-FR), Tim Welschehold (ALU-FR), Abhinav Valada (ALU-FR), Wolfram Burgard (ALU-FR), Illia Oleksiienko (AU), Lukas Hedegaard Morsing (AU), Alexandros Iosifidis (AU), Anton Muravev (TAU), Jenni Raitoharju (TAU), Moncef Gabbouj (TAU), Jelle Luijkx (TUD), Osama Mazhar (TUD), Jens Kober (TUD), Robert Babuška (TUD), Anastasios Tefas (AUTH), Nikos Nikolaidis (AUTH), Maria Tzelepi (AUTH), Nikolaos Passalis (AUTH), Paraskevi Nousi (AUTH), Angelos Nalmpantis (AUTH), Charalampos Symeonidis (AUTH), Konstantinos Tsampazis (AUTH), Alea Scovill (AGI), Dennis Larsen (AGI)
Lead Beneficiary	AU (Aarhus University)
WP	4
Doc ID:	OPENDR_D4.2.pdf

Document History

Version	Date	Reason of change
v0.1	29/9/2020	Deliverable structure template ready
v1.0	3/12/2021	Initial version of deliverable submitted for review
v1.1	8/12/2021	Reviewer comments sent and minor comments added in the document
v2.0	13/12/2021	Review comments addressed and deliverable revised

Contents

Executive Summary

This document presents the status of the work performed for **WP4–Deep environment active perception and cognition** during the second year of the project. This work package consist of four main tasks, which are *Task 4.1–Object detection/recognition and semantic scene segmentation and understanding*, *Task 4.2–2D/3D object localization and tracking*, *Task 4.3–Deep SLAM and 3D scene reconstruction*, and *Task 4.4–Sensor information fusion*. After a general introduction that provides an overview of the individual chapters with a link to the main objectives of the project, the document dedicates a chapter to each of the tasks. Each chapter *(i)* provides an overview on the state of the art for the individual topics and existing toolboxes, *(ii)* details the partners' current work in each task with initial performance results, and *(iii)* describes the next steps for the individual tasks. Finally, a concluding chapter provides a final overview of the work and the planned future work for each individual task.

1 Introduction

This document describes the work progress of the second year of the project in the four major tasks of *WP4–Deep environment active perception and cognition*, namely *Task 4.1–Object detection/recognition and semantic scene segmentation and understanding*, *Task 4.2–2D/3D object localization and tracking*, *Task 4.3–Deep SLAM and 3D scene reconstruction* and *Task 4.4–Sensor information fusion*. In this section, a brief description of the work conducted by the consortium in these tasks is provided, along with a short description on how this work contributes to the Objectives of the project. A more detailed description of the conducted work is provided in the following sections.

1.1 Object detection/recognition and semantic scene segmentation and understanding (T4.1)

Performance of recent methods for object detection/recognition and semantic scene segmentation and understanding can be impressive, when the underlying deep learning models used are of high number of parameters. To make such methods applicable in the context of OpenDR, one needs to use high-performing lightweight deep learning models. Thus, emphasis needs to be placed in the training process used to optimize the parameters of such models. To this end, OpenDR participants developed a variety of methods and tools towards tackling challenges that arise in deep environment active perception and cognition. These methods target improving performance and efficiency of deep learning models used for classification or (binary) semantic segmentation of images, via improved knowledge distillation and hashing, or by allowing for modeling the model's uncertainty related to its response.

To that end, AUTH proposed a pseudo-active sensory refinement method (Section 2.1) that works by applying a number of neural transformation layers on the sensor data. This allows for refining the sensory input, without having to reacquire the sensor data. In contrast with traditional image processing operations, such as histogram equalization, contrast corrections, etc., the proposed method is *end-to-end* trainable and formulated as a series of neural layers. As a result, the proposed method can be fully integrated in DL end-to-end training pipelines. However, at the same time, it provides significant advantages, since a) it can be directly used with any DL model, without requiring any model-specific training or any platform-specific adjustments, b) it does not require support by the underlying hardware, and c) it avoids the need to reacquire a new frame for processing by the employed DL model. As a result, the proposed method provides a solid step towards developing practical and powerful tools that can be directly deployed in a wide variety of systems, tasks and conditions, increasing the perception accuracy, as also demonstrated in the conducted experimental evaluation. Moreover, AUTH developed a single-stage self-distillation method, called Probabilistic Online Self-Distillation (POSD) for improving the performance of any deep neural model, in an online fashion (Section 2.2). The proposed method allows training compact yet effective models for classification purposes. To achieve this goal, it argues that considering a classification problem, apart from the hard labels that express explicit concepts, there are also so-called latent labels that express implicit concepts. These concepts reflect similarities among data, regardless of the classes. Thus, our target is to maximize the mutual information between the data samples and the latent labels. In this fashion, we are able to mine additional knowledge from the model itself, in an online manner. The effectiveness of the proposed method is demonstrated through extensive experiments on six datasets.

AUTH also worked on further developing an online self-distillation method for circumventing the flaws of the conventional offline Knowledge Distillation (KD) (i.e., time consuming, complex and memory and computationally demanding process) (Section 2.3). The proposed method, named Multilayer Online Self-Acquired Knowledge Distillation (MOSAKD), uses k-nn non-parametric density estimation for estimating the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is, either intermediate layers or the output later. In this way, we can directly estimate the posterior class probabilities of the data samples and use them as soft labels that explicitly express the similarities of the data with the classes, negligibly affecting the computational cost. Interestingly, apart from the output layer, useful information can be acquired from the intermediate layers. Indeed, the experimental validation on six datasets, including a dataset of synthetic images, indicated the effectiveness of the MOSAKD method. In addition, AUTH worked on improving the performance of BiseNet segmentation model, which achieves considerable performance considering the speed-accuracy trade-off (Section 2.4). More specifically, AUTH investigated ways to improve the performance of the BiseNet model, both in terms of deployment speed and segmentation accuracy, considering binary segmentation problems. Towards this end, first, a lightweight version of the model was proposed. Subsequently, the available losses were exploited. As it is experimentally validated, the modified version of the model is faster than the baseline model, while improved accuracy considering binary segmentation problems is achieved using hinge loss.

Furthermore, AUTH worked on improving object recognition methods, by incorporating label embedding criteria into the learning objective of lightweight neural networks (Section 2.5). The proposed label embeddings aim to capture general class similarities as well as instance-specific resemblances between samples. The general class embedding learns a linear mapping between the one-hot encoding of the classes and the resulting soft labels are used as the groundtruth in a cross entropy criterion. For the instance-specific similarities, an external network in the form of an Autoencoder is used, to generate soft embeddings which can first reconstruct the samples themselves and secondly act as targets in the cross entropy loss function. The resulting soft labels lead the networks to better generalization ability and better recognition accuracy. AUTH also worked on further developing a deep supervised hashing algorithm that optimizes the learned codes using an information theoretic measure, the Quadratic Mutual Information (QMI), providing a useful tool for many robotics application where similar objects must be retrieved from a database (Section 2.6). This can include many applications, ranging from few-shot learning applications to mining training samples for semi-supervised learning. The proposed method has been adapted to the needs of large-scale hashing and information retrieval leading to a novel information-theoretic measure, the Quadratic Spherical Mutual Information (QSMI), that is inspired by QMI but leads to significantly better retrieval precision. Indeed, the effectiveness of the proposed method is demonstrated under several different scenarios that include multiple large-scale datasets and network architectures, and it outperforms the existing deep supervised hashing techniques.

AU proposed a type of neural networks called Variational Neural Networks (VNNs) (Section 2.7), which can be used for the estimation of the deep learning model uncertainty with higher quality than other approaches, such as Monte Carlo Dropout [63] or Bayes By Backprop [14] methods, according to tests on Epistemic Neural Networks framework [152]. Layers of VNNs are composed of two identical layers with different weights to generate mean and variance predictions, which are used to sample from a Gaussian distribution with these parameters. This means that the model is stochastic and gives different outputs for the same set of weights and inputs. Predictions and uncertainty of such networks are computed by calculating the mean

and variance of $N \geq 1$ runs of the network. The usage of qualitative uncertainty estimation in robotics could lead to higher system stability and could be used to improve performance of an active perception task.

TAU proposed Sparse Representation Matching (SRM), a method to transfer intermediate knowledge obtained from one Convolutional Neural Network (CNN) to another one by utilizing sparse representation learning (Section 2.8). This is done by extracting sparse representations of the hidden features of the teacher CNN, which are then used to generate both pixel-level and image-level labels for training intermediate feature maps of the student network. To obtain a trainable model, a neural processing block is used which can be efficiently optimized using stochastic gradient descent and integrated into any CNN in a plug-and-play manner. TAU also conducted research towards better understanding of learnt feature spaces in image classification tasks in deep learning (Section 2.9). Training of deep neural networks can be seen as a two-stage process of feature learning followed by the task-specific prediction of the corresponding output, which is generally achieved by the last linear layer. Given that successful prediction requires successful linear classification of learnt features, it is expected that models achieving good performance converge to feature spaces with highly-separable compact classes. We quantify this notion of compactness and perform evaluations of different image classification models on several datasets and find that better compactness is not necessarily associated with better performance, as well as that compactness ratio is more often maximized than minimized during training. Further development of tools based on these findings remain for future work.

1.2 2D/3D object localization and tracking (T4.2)

Localization and tracking of objects in (2D) color images and/or in the 3D space requires the development of efficient methods that can operate using different input data types, i.e. image, point clouds, and being able to operate in real time. OpenDR participants developed methods for 2D/3D object localization and tracking operating on different types of input data, i.e. lidar-generated point cloud, spherical images, and color images. AU worked towards 3D object detection based on spherical projection images of point cloud data (Section 3.1). This data representation is more natural than voxelization or Birds-Eye-View projections as it uses all data points in their original representation to create an ordered set of points which can be directly processed by 2D CNNs. The proposed method follows FairMOT's [254] approach for 2D object tracking and predicts a heatmap of object centers in the spherical image. The features of the center point are used to regress 3D size, rotation and position offset relative to this point. AU also worked on a method for single 3D object tracking (Section 3.2) based on a Siamese architecture [11] of a PointPillars network. This model uses first frame detections to initialize and then tracks the object in Axis-Aligned Bounding Box (AABB) format in PointPillars's pseudo-image space by using target and search features and computing cross-correlation scores of target and search regions' convolutional features. The position with the highest cross-corellation score is selected as the new target position. Based on the predicted AABB position, an additional regression model predicts full 3D location, size and rotation of the object of interest.

AUTH developed a 2D tracking methodology (Section 3.3) that focuses on active object tracking, where a tracker receives an input visual observation and directly outputs the most appropriate control actions in order to follow and keep the target in its field of view, unifying in this way the task of visual tracking and control. This is in contrast with conventional tracking approaches, as typically developed by the computer vision community, where the problem of detecting the tracked object in a frame is decoupled from the problem of controlling the camera

and/or the robot to follow the object. The main contribution of the conducted work was a vision-based active tracking method that employs DRL, along with an appropriately designed reward shaping approach for active tracking problems. The developed methods were evaluated using Webots, demonstrating good generalization on various dynamic trajectories of moving objects under a wide range of different setups.

1.3 Deep SLAM and 3D scene reconstruction (T4.3)

For the field of mapping and localization, in this project, OpenDR aims to use the latest advances in deep learning to integrate network structures into classic SLAM methods to make learning more robust and efficient. Lately, several methods have put some effort into embedding traditional SLAM structures into deep network architectures. This approach has been shown to have more robust performance and better generalization capabilities compared to methods that approximate the whole procedure with a black box function approximator. For example, Neural SLAM [252] embeds the motion model, mapping, and localization procedures [21] into a completely differentiable deep neural network. The network includes an external memory architecture which is used as a "map" for the agent to learn to write onto and read from. The method in [90] encodes particle filtering processes into a network [179]. Active neural localizer incorporates traditional filtering-based localization methods into a completely differentiable network and demonstrates the effectiveness of the learned policy in both 2D and 3D simulated environments [32]. Following these ideas, we will train separate deep learning modules for each component of the SLAM pipeline. In this way, each module could either be trained end-to-end, or they could still use traditional methods but take deep features as inputs. For all the methods, the use of a lightweight deep learning methodology will be explored to allow deployment on embedded hardware. First preliminary steps to this end are presented in Section 4.

In particular, in Section 4.1, ALU-FR introduces their work on LiDAR-based panoptic segmentation EfficientLPS [195] as the continuation of the image-based EfficientPS [140] method, presented in the previous deliverable D4.1. It sets a new state-of-the-art for predicting both semantic and instance segmentation maps for 3D point clouds. The main idea is to project the 3D data into the 2D domain and apply a CNN-based backbone followed by subtask-specific heads. Finally, semantic and instance predictions are fused and reprojected to 3D. Moreover, in Section 4.2, ALU-FR presents an extension of their previous work CMRNet++ [27], addressing the task of monocular visual localization in maps created by a LiDAR sensor. Using a convolutional neural network, the method advances the state-of-the-art by enabling vision-based localization in 3D maps independent of the camera parameters. It further shows superior performance compared to other methods by effective generalization to unseen places, i.e., allowing for accurate localization without the need to retrain.

AGI continued to improve their SLAM plant row detection algorithm. To reduce costs for the farmer, the second forward facing normal lens camera on Robotti was removed. The plant row algorithm has been updated to handle the fisheye lens. To increase speed of the algorithm, it has been moved to C++ and ROS. To increase accuracy, confidence levels have been added, so the system will be able to make decisions based on expected accuracy.

1.4 Sensor information fusion (T4.4)

In the field of sensor information fusion the project aims to develop environment perception solutions, which are capable of simultaneously utilizing the data from heterogeneous sources

(e.g. RGB imagery, depth imagery, audio, various sensory outputs). The emphasis is placed on efficiency and robustness, specifically the ability to detect and use the information redundancies between different data streams. This would allow to both reduce the resource consumption of the model during normal operation and to minimize the ultimate information loss in case of input stream disruptions.

To that end, TAU has continued its work on the multimodal sensor fusion framework for manipulation tasks. The formulation of the representation learning task was extended with additional objective functions to support input reconstruction and cross-modal compensation. Input reconstruction allows the model to detect an input signal being out-of distribution, which could indicate the sensor failure or data distortion in a given modality. The system is thus capable of reacting to corrupted inputs without assumptions or prior knowledge of which input is affected and how. Cross-modal compensation then allows the model to drop the corrupted input entirely and still recover a feature encoding that is similar (in terms of L2 norm) to the full encoding. Together these additions allow for the feature encoder to continue operating under degradation or absence of certain modalities, which is in line with the goal of achieving robustness. Additionally, TAU is working on optimizing the structures of the fusion modules and the feature encoder as a whole, using neural architecture search. Being autonomous (i.e. requiring no expert input), the neural architecture search is capable of discovering neural structures that are superior in terms of task-oriented performance, computational costs, or both. While the search process itself is computationally expensive, it is performed in an offline fashion, not running directly on the devices targeted by OpenDR. The discovered models can be subsequently trained and deployed in OpenDR environments, where their improved properties would bring practical advantages.

Computer vision is another essential part of robotics, as visual information from the environment is widely utilized in numerous successful solutions. Deep learning algorithms in particular have shown powerful capabilities of solving tasks such as image classification, face identification and object detection. However, they remain sensitive to input variations, which can be caused by illumination changes or other harsh visual conditions. Therefore, robotic vision can also benefit greatly from the performance boost and increased redundancy provided by the sensor information fusion. Effective fusion methodologies are crucial for achieving robustness against asymmetric sensor failures and to minimize uncertainties in the predictions. To that end, TUD has developed efficient sensor fusion strategies in the context of object detection for RGB and IR/depth sensors for harsh lighting conditions [131]. The proposed fusion strategies learn to exploit sensor redundancy in extreme lighting conditions by intelligently determining the scalar/mask weights for the dominant sensor modality. TUD has also exploited Gumble-softmax trick to obtain discrete samples from the categorical distribution given the class probabilities of each element in the extracted feature volumes of the input modalities. The learned weights/masks can also be visualized for each prediction to understand the underlying strengths of the opted sensor modalities in the given conditions. The work is built upon a transformer-based end-to-end object detector which eliminates the need for hand-crafted components like anchors and non-maximum suppression. Our multi-modal design converges fairly quickly thus reducing training time and optimizing computational resources for large-scale datasets. A novel data augmentation scheme, namely Random Shadows-Highlights (RSH) [132], has been proposed to mimic such extreme conditions. This is performed by randomly creating masks over the input images, while adjusting the pixel values in the masked regions. RSH is a learning-free lightweight generic algorithm that allows exploiting large scale multi-modal datasets to train object detectors or any other image classification models with an improved performance.

1.5 Connection to Project Objectives

The work carried out by the consortium in the context of WP4, as summarized in the previous subsections, is directly related to the overarching project goals. In particular, the work described here progressed the state-of-the-art towards meeting the project goals O1 and O2, as will be presented in detail below.

O1 *To provide a modular, open and non-proprietary toolkit for core robotic functionalities enabled by lightweight deep learning*

O1a *To enhance the robotic autonomy exploiting lightweight deep learning for on-board deployment*

AUTH developed online distillation methods for efficiently training lightweight models for robotics applications, ranging from a Probabilistic Online Self-Distillation methodology that has been employed (Section 2.2) to an efficient self-distillation approach (Section 2.3). Also, AUTH explored ways for ameliorating the performance of BiseNet semantic segmentation model both in terms of deployment speed and segmentation accuracy, considering binary segmentation problems (Section 2.4). Also, AUTH developed a supervised hashing methodology (Section 2.6) that enables retrieving objects similar to a query image, minimizing inference and query time and providing a useful tool both for few-shot learning, as well as for mining training data when few training data exist.

AU has proposed methods for 3D object detection and tracking which can be used in robotics applications where lidar sensors are used to sense the robot's environment. A method for 3D object detection uses spherical projection images (Section 3.1) for a natural ordering of point clouds and uses a heatmap-based approach to find central object points and regress a final 3D bounding box from features of this point. A single object 3D tracking method (Section 3.2) was developed that uses a Siamese PointPillars architecture to find an object in the generated pseudo-image space and then regress its 3D bounding box based on the features of the found region. AU also introduced a new type of neural networks, called Variational Neural Networks (Section 2.7), with an inherent ability to express uncertainty.

TAU proposed a knowledge distillation method to transfer intermediate knowledge obtained from one Convolutional Neural Network (CNN) to another one by utilizing sparse representation learning (Section 2.8). This approach can be integrated into any CNN in a plug-and-play manner, in order to improve performance of light-weight CNNs for efficient image/scene classification.

O1b *To provide real-time deep learning tools for robotics visual perception on high-resolution data*

AUTH also worked on improving accuracy in classification problems using lightweight networks which can run in real-time on embedded devices for high-resolution inputs, by incorporating label embedding criteria into the learning objective of these networks (Section 2.5).

TAU has worked towards investigating the relationships between properties of learnt feature spaces in deep neural networks and performances of corresponding models in image

classification tasks (Section 2.9), which can be used for understanding the training process of deep learning models when applied on visual perception tasks and can enable development of methods in further work of the project.

O2 *To leverage AI and Cognition in robotics: from perception to action*

O2a *To propose, design, train and deploy models that go beyond static computer perception, towards active robot perception*

AUTH proposed pseudo-active sensory refinement method (Section 2.1) that works by applying a number of neural transformation layers on the sensor data, allowing for improving the perception accuracy of DL models without having to re-acquire sensor data. Also, AUTH worked towards developing end-to-end active tracking approaches. (Section 3.3.

O2c *To provide tools for enhanced robot navigation, action and manipulation capabilities that can exploit if needed deep environment active robot perception*

ALU-FR proposed two new methods, namely Efficient LiDAR Panoptic Segmentation (EfficientLPS) [195] (Section 4.1) and CMRNet++ [27] (Section 4.2). The EfficientLPS architecture allows to jointly assign semantic classes and distinguish instances of the same class in LiDAR point clouds, enabling in-depth scene understanding, which is a key requirement for many robot action capabilities. Facilitating advanced robot navigation, the proposed CMRNet++ enhances monocular vision-based localization in 3D LiDAR maps and shows superior performance compared to other methods by effective generalization to unseen places, i.e., allowing for accurate localization without the need to retrain.

TAU has further expanded its modular sensor fusion framework for robotic manipulation tasks (Section 5.1), incorporating detection and cross-modal compensation of damaged input signals, as well as leveraging the neural architecture search to autonomously discover more efficient fusion models. These extensions improve efficiency and robustness of DRL-based controllers in practical robotic manipulation scenarios.

TUD has developed sensor fusion strategies for multi-modal object detection that efficiently exploit sensors redundancy in harsh lighting conditions (Section 5.2). TUD also proposed a lightweight learning-free data augmentation method which creates random highlights and shadows to mimic such harsh conditions (Section 5.2).

AGI improved their SLAM plant row guidance algorithm, which will enable the field robot to be able to react to changes in the plant rows and in its environment.

2 Object detection/recognition and semantic scene segmentation and understanding

2.1 Pseudo-Active Vision For Improving Deep Visual Perception Through Neural Sensory Refinement

2.1.1 Introduction, objectives and summary of state of the art

Active vision allows for appropriately controlling the camera of a perception system in order to improve the perception accuracy [7]. It is worth noting that a camera can be controlled both regarding its external parameters, e.g., pan and tilt, as well as some of its internal parameters, e.g., exposure and color profile, etc. Even through there is an increasing amount of literature for handling the former [183, 138, 161, 19, 247], less focus has been given to the latter (with respect to the performance of DL models). Indeed, most DL algorithms implicitly assume that the heavy pre-processing that is involved in most digital camera sensors, e.g., color constancy algorithms [52, 12, 83, 105], will mitigate the effect of varying illumination conditions. As a result, most DL models do not explicitly deal with these effects. However, as we experimentally demonstrate in this work, DL models are especially prone to both varying illumination conditions, as well as changes in contrast, brightness, and slight color shifts. This is not a surprising finding, since adversarial attacks, as well as studies of the effect of color shifts on the accuracy of DL models, have indeed demonstrated the vulnerability of DL models to such transformations [3, 166].

Apart from distribution shifts that arise from the environmental factors described above, using different hardware for the deployment can also reduce the visual perception accuracy, if the sensors are not adequately calibrated. The typical solution to this problem is to manually tune the hardware, as well as employ the appropriate software pre-processing modules in order to ensure that the models will perform as expected. However, it is obvious that this process requires a significant amount of effort, often slowing down the integration in many real robotics applications.

Active perception algorithms can be employed to tackle the aforementioned challenges, i.e., to appropriately control the internal camera parameters in order to maximize perception accuracy for a given DL model. This process has many similarities with the visual perception systems developed in many biological organisms, such as many mammals. In these cases, different mechanisms exist for adjusting various parameters that affect the signal reaching to sensor cells, well before propagating the information to the visual cortex. Perhaps the most well-known example of such mechanism is the adaptive response of the iris to different illuminations conditions, altering the amount of light eventually reaching the retina [196]. Despite their potential, active vision methods come with two important drawbacks. First, they typically lead to the (complete or partial) loss of (at least) one frame acquired by the camera, which can negatively affect the latency of the system. Second, such algorithms are not easily deployed, since the output of the active perception algorithm must be first adequately translated in order to match the underlying hardware of each system, i.e., to translate the model's output to control signals.

To overcome these limitations, we propose a pseudo-active sensory refinement method that works by applying a number of neural transformation layers on the sensor data. This allows for refining the sensory input, without having to reacquire the sensor data. In contrast with traditional image processing operations, such as histogram equalization, contrast corrections,

etc., the proposed method is *end-to-end* trainable and formulated as a series of neural layers. As a result, the proposed method can be fully integrated in DL end-to-end training pipelines. However, at the same time, it provides significant advantages, since a) it can be directly used with any DL model, without requiring any model-specific training or any platform-specific adjustments, b) it does not require support by the underlying hardware, and c) it allows for avoiding the need to reacquire a new frame for processing by the employed DL model. As a result, the proposed method provides a solid step towards developing practical and powerful tools that can be directly deployed in a wide variety of systems, tasks and conditions, increasing the perception accuracy. Indeed, we demonstrate the effectiveness of the proposed method using two different tasks and datasets, i.e., image recognition on ImageNet dataset (ILSVRC 2012) [101] and object detection on PASCAL VOC 2007 dataset [56].

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.1:

1. [163] N. Passalis and A. Tefas, "*Pseudo-Active Vision for Improving Deep Visual Perception through Neural Sensory Refinement*", International Conference on Image Processing, 2021

2.1.2 Description of work performed so far

Let $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ be an input image, where W, H and C denote its width, height and number of channels respectively. Typically, the input image $\mathbf{x}$ is fed into a DL model $f(\mathbf{x})$ in order to perform visual information analysis, e.g., object recognition. The proposed method works by employing a series of neural transformation layers, as shown in Fig. 1, before feeding the input into the DL model. The proposed method is composed of three separate modules, each one performing a different *learnable* transformation on the input. Each of these modules operate on a dynamic differentiable color histogram compiled through the previous stage. Therefore, during the first stage, the image is globally shifted and scaled according to the input histogram. Then, the second stage performs the same transformation, but on channel level. These two modules combined aim at correcting global and per color channel brightness and contrast. Then, the third module is responsible for refining local regions of the input image, while also taking into account its global histogram, in order to recover information that is potentially lost during image acquisition, e.g., due to over-exposure.

More specifically, the proposed method works as follows. First, we compile a differentiable histogram with adaptive bins using a Neural Bag-of-Feature-based formulation [156]. To this end, we quantize the input image features $[\mathbf{x}]_{ij} \in \mathbb{R}^C$ using a codebook $\mathbf{V} = [\mathbf{v}_1; \ldots; \mathbf{v}_{N_K}]^T \in \mathbb{R}^{N_K \times C}$ as:

$$\mathbf{h} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \mathbf{u}_{ij} \in \mathbb{R}^{N_K} \tag{1}$$

where N_K is the number of codewords, while the similarity vector $\mathbf{u}_{ij}$ for the codewords $\mathbf{v}_k$ and the input features $\mathbf{x}_{ij}$ is calculated as

$$[u_{ij}]_k = \text{sigm}(\mathbf{v}_k^T \mathbf{x}_{ij}), \tag{2}$$

where $\text{sigm}(\cdot) \in (0,1)$ denotes the sigmoid function. Three different histograms are compiled, as shown in Fig. 1: a) one at the input ($\mathbf{h}_1$), b) one at the output of the first module ($\mathbf{h}_2$) and c) one at the end of the second module ($\mathbf{h}_3$). Note that the first histogram operates on the averaged

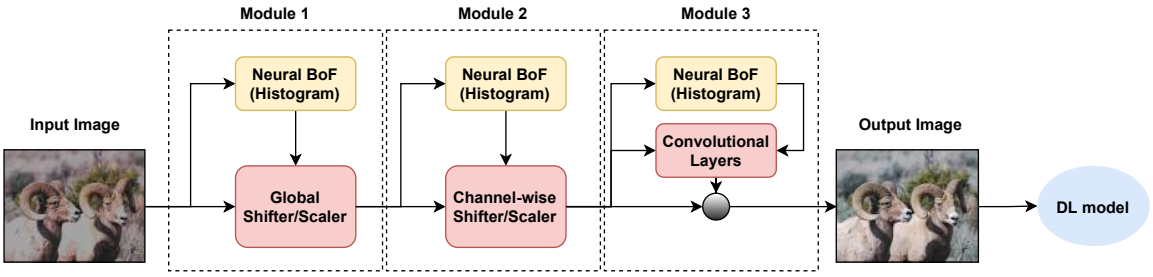


Figure 1: The proposed method is composed of three separate neural modules. Each one is responsible for learning how to apply a different kind of image transformation in order to gradually refine the input image in order to maximize the accuracy of the subsequent DL model.

color intensities, i.e., $C = 1$, instead of separate color channels, since its aim is to capture the global information regarding the brightness distribution in the image.

After compiling the first histogram, the proposed method employs two linear layers to appropriately shift and scale the input:

$$\mathbf{x}' = (\mathbf{x} - \mathbf{h}_1^T \mathbf{W}_{11})/(1 + \mathbf{h}_1^T \mathbf{W}_{12}), \tag{3}$$

where $\mathbf{W}_{11} \in \mathbb{R}^{N_K \times 1}$ denotes the weights of the shifting sub-layer and $\mathbf{W}_{12} \in R^{N_K \times 1}$ denotes the weights of the scaling sub-layer. Then, this process is repeated in the second module, but the shifting and scaling is now applied per-channel:

$$[x'']_{ijk} = ([x']_{ijk} - [\mathbf{h}_2^T \mathbf{W}_{21}]_k)/(1 + [\mathbf{h}_2^T \mathbf{W}_{22}]_k), \tag{4}$$

where $\mathbf{W}_{21} \in \mathbb{R}^{N_K \times C}$ denotes the weights of the shifting sub-layer and $\mathbf{W}_{22} \in \mathbb{R}^{N_K \times C}$ denotes the weights of the scaling sub-layer.

Finally, the third module works by first projecting the third histogram into a lower dimensional space:

$$\mathbf{h}_x = \tanh(\mathbf{h}_3^T \mathbf{W}_p) \in \mathbb{R}^{N_P} \tag{5}$$

where $\mathbf{W}_p \in \mathbb{R}^{N_K \times N_P}$ and $\tanh(\cdot)$ denotes the hyperbolic tangent function. Then, the vector $\mathbf{h}_x$ is upsampled into a $W \times H \times N_P$ tensor and concatenated (channel-wise) with the output of the previous layer to form the $\mathbf{x}_p \in \mathbb{R}^{W \times H \times (C+N_P)}$ tensor. The output of the final layer is then calculated as:

$$\mathbf{x}''' = \mathbf{x}'' \odot (1 + \tanh(g(\mathbf{x}_p))), \tag{6}$$

where $g(\cdot) \in \mathbb{R}^{W \times H \times (C)}$ is a series of convolution and deconvolution layers and $\odot$ denotes the Hadamard (element-wise) product between two tensors. In this work we use two 5×5 convolution layers with 16 and 8 filters, followed by two symmetric deconvolution layers (the last layer has C filters). The $\tanh(\cdot)$ non-linearity is used for all the layers. Note that we assumed that the input is already normalized into a specific (limited) range. If this assumption does not hold, then the output of the last layer must be also scaled using a trainable parameter.

All the parameters of the proposed method are learned by first employing an image transformation on the original image $\mathbf{x}$, e.g., brightness, contrast or hue adjustment, leading to a *corrupted* image $\tilde{\mathbf{x}}$. Then, all the layers are trained using the back-propagation algorithm in order to recover the original image to the output of the proposed sequence of neural layers,

denoted by $h(\cdot)$, by minimizing the following loss function:

$$\mathscr{L} = \frac{1}{WHC} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{C} ([\mathbf{x}''']_{ijk} - [\mathbf{x}]_{ijk})^2, \tag{7}$$

where $\mathbf{x}''' = h(\tilde{\mathbf{x}})$.

2.1.3 Performance evaluation

The proposed method was first extensively evaluated on the ImageNet dataset (ILSVRC 2012) [101] using an image recognition setup and a ResNet-18 architecture [76]. The proposed method was also compared to a) directly using the trained ResNet-18 architecture (denoted as "Baseline") with the default normalization scheme (global standardization), b) employing a histogram equalization approach (denoted as "Hist. Equalization") and c) maximizing the contrast (denoted as "Autocontrast"). We also used different approaches to deteriorate the original images: a) increasing the contrast by 30% (+) and 40% (++), b) increasing (+/++) or decreasing (-/–) the brightness by 30% and 40% respectively, c) increasing (+) or decreasing (-) the hue by 10%. Also, we evaluated the methods in a more challenging combined setup where the contrast was increased by 20%, brightness was decreased by 20% and hue was increased by 5%. The proposed model was trained by randomly applying different transformations during the training process and then recovering the original image.

The experimental results for the ImageNet dataset are reported in Table 1. Several interesting conclusions can be drawn from the reported results. First, note that all the applied transformations, i.e., increasing the contrast, increasing/reducing the brightness, increasing/decreasing the hue, and/or combining multiple transformations on the input image, lead to a significant reduction in recognition accuracy. For example, even a mild shift in the hue by 10% leads to a reduction in top-1 accuracy from 69.76% to 62.11%. At the same time, employing non-linear histogram equalization does not improve the recognition accuracy. In contrast, this type of image enhancement actually reduces the recognition accuracy, possibly by introducing a catastrophic distribution shift. Similar results were observed for the rest of transformations as well, but omitted from Table 1, due to lack of space. On the other hand, maximizing the contrast in the input image can lead to improved recognition accuracy in virtually all the evaluated cases. Employing the proposed method can further improve the recognition results, by being able to learn how to appropriately refine the original images in order to compile new transformed images that would be appropriate for the DL model at hand. It is worth noting that for some cases, e.g., for contrast transformations, using the proposed method can improve the top-1 recognition by over 3% (relative improvement).

To further demonstrate the generality of the proposed method, we conducted additional experiments for one other task, i.e., object detection. To this end, we employed the Single Shot MultiBox Detector (SSD) [124], with a MobileNet backbone [82] for object detection. The proposed method was not trained on the corresponding detection dataset. Instead, we directly employed the model trained on the ImageNet dataset for a recognition task in order to evaluate how it generalizes on this related task. The experimental results are reported in Table 2. Indeed, using the proposed method again improves the perception accuracy over the evaluated baselines. Quite interestingly maximizing the contrast does not work so well for this task, while the proposed method still leads to impressive improvements (e.g., the mean Average Precision (mAP) increases by over 1.5% in some cases). These results highlight the ability

Table 1: Pseudo-Active Perception: Object recognition evaluation (recognition accuracy (%) is reported) on ImageNet dataset (ILSVRC 2012)

Method	Transformation	top-1	top-5	Method	Transformation	top-1	top-5
Baseline	No	69.76	89.08	Baseline	Brightness(-)	60.52	82.64
Baseline	Contrast (+)	62.18	83.94	Autocontrast	Brightness(-)	61.02	82.99
Hist. Equalization	Contrast (+)	54.69	77.74	Proposed	Brightness(-)	**61.67**	**83.58**
Autocontrast	Contrast (+)	62.18	83.93	Baseline	Brightness(–)	53.13	76.38
Proposed	Contrast (+)	**63.43**	**84.95**	Autocontrast	Brightness(–)	53.78	77.15
Baseline	Contrast (++)	57.49	80.23	Proposed	Brightness(–)	**54.78**	**78.01**
Hist. Equalization	Contrast (++)	50.00	73.48	Baseline	Hue (+)	62.11	84.38
Autocontrast	Contrast (++)	57.47	80.23	Autocontrast	Hue (+)	62.17	84.39
Proposed	Contrast (++)	**59.23**	**81.78**	Proposed	Hue (+)	**62.37**	**84.70**
Baseline	Brightness (+)	64.55	85.92	Baseline	Hue (-)	62.63	85.02
Autocontrast	Brightness (+)	66.81	87.26	Autocontrast	Hue (-)	62.69	85.05
Proposed	Brightness (+)	**67.03**	**87.35**	Proposed	Hue (-)	**63.41**	**85.46**
Baseline	Brightness(++)	60.09	82.48	Baseline	Combined	55.31	78.35
Autocontrast	Brightness(++)	64.10	85.14	Autocontrast	Combined	55.17	78.23
Proposed	Brightness(++)	**64.31**	**85.38**	Proposed	Combined	**55.60**	**78.80**

of the proposed method to be directly employed and combined with virtually any DL model and further increase its robustness by providing an efficient pseudo-active vision approach. Additional ablation experiments are provided in Appendix 7.1.

2.2 Probabilistic Online Self-Distillation

2.2.1 Introduction, objectives and summary of state of the art

Over the recent years, deep learning (DL) models have eclipsed previous solutions in a wide range of challenging computer vision tasks, [73]. State-of-the-art DL models are generally parameter-heavy, containing millions or even billions of parameters [84], assisted to some extent by the availability of increasingly powerful GPUs. That is, their exceptional performance stems from their depth and complexity. Therefore, an apparent need for developing compact yet effective models, diminishing the storage requirements and the computational cost, has been arisen. During the recent years, substantial research work has been performed to achieve this goal [37]. Amongst them, *Knowledge Distillation* (KD) [80] has emerged as a highly promising approach to address this issue by transferring the knowledge from one, usually larger, model to a more compact model.

KD methods fall into two categories: *online* and *offline* KD. The multi-stage procedure of initially training a powerful teacher model and then distilling the knowledge to a weaker student model stands for *offline* KD. However, offline distillation is accompanied by some flaws. That is, it is a long-lasting, complex, and computationally and memory-wise demanding process, since it requires to train first a powerful and heavyweight model and after the convergence to transfer the acquired knowledge to a faster model. Thus, in the recent literature several online KD methods have been proposed, in order to circumvent the aforementioned flaws of offline KD. Online KD describes a procedure where the teacher and the student models are trained concurrently, that is without the stage of pre-training the teacher model.

Online KD includes a method that trains multiple (student) models mutually from each other [255], where each model acts as a teacher to other model, as well as a method that trains k copies

Table 2: Pseudo-Active Perception: Evaluation on the VOC2007 (object detection) dataset (%)

Method	**Transform.**	**VOC2007 mAP**
Baseline	-	75.51
Baseline	Contrast (+)	62.24
Autocontrast	Contrast (+)	62.13
Proposed	Contrast (+)	**63.19**
Baseline	Contrast (++)	66.95
Autocontrast	Contrast (++)	66.86
Proposed	Contrast (++)	**67.44**
Baseline	Brightness (++)	70.30
Autocontrast	Brightness (++)	71.36
Proposed	Brightness (++)	**71.45**
Baseline	Brightness (–)	56.84
Autocontrast	Brightness (–)	55.81
Proposed	Brightness (–)	**56.85**
Baseline	Combined	56.64
Autocontrast	Combined	56.39
Proposed	Combined	**57.29**

of a target (student) model in parallel by adding a distillation term to the loss function of the *i*-th model to match the average prediction of the other models [5]. Subsequently, an online distillation approach builds a multi-branch version of the network by adding identical branches, each of which constitutes an independent classification model with shared low level layers, and to create a strong teacher model utilizing a gated logit ensemble of the multiple branches [107].

A recent work [94], combines the previous works [255] and [107], by proposing an online mutual knowledge distillation method for enhancing both the performance of the fusion module and the sub-networks. That is, when different sub-networks are used, the sub-networks are trained similar to [255], while when identical sub-networks are used, the low level layers are shared and a multi-branch architecture similar to [107] is used. Finally, a two-level distillation methodology, named Online Knowledge Distillation with Diverse peers (OKDDip), where two types of students are involved, i.e., multiple auxiliary peers and one group leader, is proposed in [35]. Distillation is conducted among auxiliary peers with a mechanics for preserving diversity, and then an ensemble of predictions of these peers is further distilled to the group leader.

A novel probabilistic single-stage self-distillation method, called *Probabilistic Online Self-Distillation* (POSD), is proposed. The proposed method is model-agnostic, that is, it improves the performance of any deep neural model considering object classification tasks (i.e., lightweight and common heavyweight models, as it is demonstrated through the conducted experiments) in an online manner. Thus, as compared to conventional offline KD methods, it reduces the computational and memory requirements. It should be emphasized that the proposed method derives the additional knowledge from the model itself -that is without any powerful teacher model- as in [108], but also in an online manner.

More specifically, we argue that in each classification problem there are *explicit concepts*, expressed with the *hard labels*, but there are also *implicit concepts*, expressed with so-called *latent labels*. These implicit concepts reflect potential sub-clusters formed by data samples

that share specific attributes or similarities among data samples regardless of the class labels. That is, they are cross-label concepts, which convey useful information about the relationships between data samples.

Thus, while conventional KD argues that it is useful to maintain the similarities of the samples with the classes (that express the explicit concepts) during the training process, we argue that there are samples that also share other semantic attributes, extending beyond the explicit classes. Therefore, we aim to discover these similarities during the training process and then appropriately employ them to transfer additional knowledge. To achieve this goal, we use a well-established metric in Information Theory, that is *Mutual Information* (MI), which allows us to quantify the information regarding these implicit concepts. Since mutual information is in general impractical and inefficient to be estimated in high dimensional spaces, we utilize a variant of mutual information, that is *Quadratic Mutual Information* [211], which allows us to efficiently optimize the network toward the aforementioned objective. That is, our goal is to maximize the MI between the distribution of the data samples and the latent labels. In this way, the model is not only able to discriminate between different classes, but also to better encode the geometric relationships between them.

A summary of this work is provided hereafter. The corresponding technical report is listed below, and can be found in Appendix 7.2:

1. M. Tzelepi, N. Passalis and A. Tefas, "*Probabilistic Online Self-Distillation*", Technical Report.

2.2.2 Description of work performed so far

In each classification problem, there are *explicit concepts*, expressed with the *hard labels*, but there are also *implicit concepts*, associated with the *latent labels*. DL models transform the probability distribution of the data, layer by layer, learning increasingly complex layer representations. As the distribution of the data is being transformed through the layers during the training procedure, we aim to derive useful information about the relationships among the data samples which is ultimately ignored, as the samples are forced by the conventional supervised loss to suppress the implicit concepts. To this aim, we propose to introduce an auxiliary objective aiming to encode the useful implicit concepts, that reflect similarities among the data, from the model itself. These implicit concepts are unknown, and they can be discovered through several ways. For example, they could be pre-calculated using clustering, or discovered by exploiting the local manifold structure of the space, etc. In this work, without loss of generality, we consider the degenerated case where each sample defines a different implicit concept.

More specifically, we consider a O-class classification problem and the labeled data $\{\mathbf{x}_i, \mathbf{l}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \Re^D$ an input vector and D its dimensionality, while $\mathbf{l}_i \in \mathscr{Y}^O$ corresponds to its O-dimensional one-hot class label vector (hard label). For an input space $\mathscr{X} \subseteq \Re^D$ and an output space $\mathscr{F} \subseteq \Re^O$, we consider as $\phi(\cdot;\mathscr{W}) : \mathscr{X} \rightarrow \mathscr{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of parameters $\mathscr{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_{N_L}\}$, where $\mathbf{W}_L$ are the weights of a specific layer L, which transforms its input vector to a O-dimensional probability vector. That is, $\phi(\mathbf{x}_i;\mathscr{W}) \in \mathscr{F}$ corresponds to the output vector of $\mathbf{x}_i \in \mathscr{X}$ given by the network ϕ with parameters $\mathscr{W}$.

Thus, considering the typical classification problem, we seek for the parameters $\mathscr{W}^*$ that minimize the cross entropy loss, $\mathscr{J}_{ce}$, between the predicted and hard label distributions:

$$\mathscr{W}^* = \arg\min_{\mathscr{W}} \sum_{i=1}^{N} \mathscr{J}_{ce}(\mathbf{l}_i, \phi(\mathbf{x}_i;\mathscr{W})), \tag{8}$$

The cross entropy loss for a set of N samples is formulated as:

$$\mathscr{J}_{ce} = -\sum_{i=1}^{N}\sum_{m=1}^{O} l_i^m log(z_i^m), \tag{9}$$

where l_i^m is the m-th element of $\mathbf{l}_i$ one-hot label vector, and z_i^m is defined as the output of the softmax operation on the O-dimensional network's output:

$$z_i^m = \frac{\exp(\phi(\mathbf{x}_i;\mathscr{W})^m)}{\sum_{j=1}^{C}\exp(\phi(\mathbf{x}_i;\mathscr{W})^j)}, \tag{10}$$

where the notation $\phi(\mathbf{x}_i;\mathscr{W})^m$ is used to refer to the m-th value of the output vector of the network. The cross entropy loss generally suppresses the aforementioned implicit concepts, and thus our goal is to circumvent this by enforcing the data samples to maintain the MI with these concepts.

For simplicity, we consider Y to be a random variable representing the image representations of the feature space generated by a specific deep neural layer L, that is $\mathbf{y}_i = \phi(\mathbf{x}_i;\mathscr{W}_L)$. We also consider a discrete-value variable C that represents the latent labels. Each feature representation $\mathbf{y}$ defines an implicit concept and it is associated with a latent label c.

MI measures dependence between random variables, that is, it measures how much the uncertainty for the latent label c is reduced by observing the feature vector $\mathbf{y}$. Let $p(c)$ be the probability of observing the latent label c, and $p(\mathbf{y},c)$ the probability density function of the corresponding joint distribution. The MI between the two random variables is defined as:

$$MI(Y,C) = \sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y},c)\log\frac{p(\mathbf{y},c)}{p(\mathbf{y})P(c)}d\mathbf{y}, \tag{11}$$

where $P(c) = \int_{\mathbf{y}} p(\mathbf{y},c)d\mathbf{y}$. Instead of utilizing MI, we use Quadratic Mutual Information [211], since directly calculating MI is not computationally tractable. That is, MI can be interpreted as a Kullback-Leibler divergence between the joint probability density $p(\mathbf{y},c)$ and the product of marginal probabilities $p(\mathbf{y})$ and $P(c)$. Thus, QMI is derived by replacing the Kullback-Leibler divergence by the quadratic divergence measure [211]. That is:

$$QMI(Y,C) = \sum_{c=1}^{N}\int_{\mathbf{y}} (p(\mathbf{y},c) - p(\mathbf{y})P(c))^2 d\mathbf{y}. \tag{12}$$

And thus, by expanding eq. (12) we arrive at the following equation:

$$\begin{aligned} QMI(Y,C) = &\sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y},c)^2 d\mathbf{y} + \sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y} \\ &-2\sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y},c)p(\mathbf{y})P(c)d\mathbf{y}. \end{aligned} \tag{13}$$

The quantities appearing in eq. (13) are called *information potentials* and they are defined as follows: $V_{IN} = \sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y},c)^2 d\mathbf{y}$, $V_{ALL} = \sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y})^2 P(c)^2 d\mathbf{y}$, $V_{BTW} = \sum_{c=1}^{N}\int_{\mathbf{y}} p(\mathbf{y},c)p(\mathbf{y})P(c)d\mathbf{y}$,

and thus, the QMI between the data samples and the corresponding latent labels can be expressed as follows utilizing the information potentials:

$$QMI = V_{IN} + V_{ALL} - 2V_{BTW}. \tag{14}$$

As we have previously mentioned, we consider that each sample defines an implicit concept, expressed with a latent label, and thus there are N different latent labels. Under the manifold assumption [10], we consider that the implicit concepts are expressed in the feature space as the geometric proximity between the data samples. That is, nearest neighbors, in terms of Euclidean distance, in the feature space for each sample defining an implicit concept share the same concept. Note, that the nearest neighbors are updated at each iteration, and thus throughout the network's training we obtain more useful nearest neighbors, since the procedure is driven by the supervised loss. It is also noteworthy that manifold assumption allows us to infer only similarities, not dissimilarities. Therefore, we can infer that two neighbors are similar because they are close to each other, however, being far apart does not guarantee that they do not share the same implicit concept. We should also highlight that as the network is optimized to discriminate between the different classes (hard labels), the intermediate representations and the corresponding manifolds are distorted in a way that facilitates the task at hand. Therefore, some implicit concepts that are not relevant to the task at hand are suppressed, while others, relevant to the task at hand, are possibly reinforced.

Thus, each of N different latent labels consists of k_p samples (i.e., a certain number of nearest samples and the sample itself), and the class prior probability for the c_p latent label is given as: $P(c_p) = \frac{k_p}{N}$, where N corresponds to the total number of samples. Kernel Density Estimation (KDE) can be used to estimate the joint density probability: $p(\mathbf{y}, c_p) = \frac{1}{N}\sum_{j=1}^{k_p} K(\mathbf{y}, \mathbf{y}_j^p; \sigma^2)$, for a symmetric kernel K, with width σ, where we use the notation $\mathbf{y}_j^p$ to refer to the j-th sample of the p-th latent label, that is the j-th nearest neighbor, as well as the probability density of Y as $p(\mathbf{y}) = \sum_{p=1}^{k_p}(\mathbf{y}, c_p) = \frac{1}{N}\sum_{j=1}^{N} K(\mathbf{y}, \mathbf{y}_j; \sigma^2)$.

Therefore, the information potentials that appear in (14) can be efficiently calculated as:

$$V_{IN} = \frac{1}{N^2}\sum_{p=1}^{N}\sum_{k=1}^{k_p}\sum_{l=1}^{k_p} K(\mathbf{y}_k^p, \mathbf{y}_l^p; 2\sigma^2), \tag{15}$$

$$V_{ALL} = \frac{1}{N^2}\left(\sum_{p=1}^{N}(\frac{k_p}{N})^2\right)\sum_{k=1}^{N}\sum_{l=1}^{N} K(\mathbf{y}_k, \mathbf{y}_l; 2\sigma^2), \tag{16}$$

$$V_{BTW} = \frac{1}{N^2}\sum_{p=1}^{N}\frac{k_p}{N}\sum_{j=1}^{k_p}\sum_{k=1}^{N} K(\mathbf{y}_j^p, \mathbf{y}_k; 2\sigma^2). \tag{17}$$

The pairwise interactions described above between the samples can be interpreted as follows:

- V_{IN} expresses the interactions between pairs of samples sharing each implicit concept
- V_{ALL} expresses the interactions between all pairs of samples, regardless of the latent label

- V_{BTW} expresses the interactions between samples of each implicit concept against all other samples

The kernel function $K(\mathbf{y}_i, \mathbf{y}_j; \sigma^2)$ expresses the similarity between two samples i and j. There are several choices for the kernel function [189]. For example, in [211] the Gaussian kernel is used, while in [158] the authors utilize a cosine similarity based kernel to avoid defining the width, in order to ensure that a meaningful probability estimation is obtained, since fine-tuning the width of the kernel is not a straightforward task, [38]. In this work, we use the power kernel. Power kernel, K_P , also known as unrectified triangular kernel, is defined as follows:

$$K_P = ||\mathbf{y}_i - \mathbf{y}_j||^d \tag{18}$$

Our goal is to maximize the QMI between the distribution of the data samples sharing implicit concepts with the distribution of the latent labels. This process can be accelerated by observing that V_{BTW} term includes only the distant neighbors to each implicit concept and, as a result, it typically has a negligible contribution to the optimization compared to V_{IN}. Furthermore, the V_{ALL} term is just a contractive term that tends to shrink all the samples in the feature space, contributing equally to all data samples, regardless their label. Given that our goal is not to accurately estimate the exact value of MI, but to enhance the implicit concepts that appear in the data, we propose accelerating the optimization by using V_{IN} as a proxy to QMI. This approximation allows for accelerating the training process, while having only a negligible effect on the optimization.

Thus, the overall loss, J can be formulated as:

$$J = J_{CE} - \lambda J_{POSD}, \tag{19}$$

where $J_{POSD} = V_{IN}$, and λ balances the importance between the hard and the latent labels.

Simple SGD is utilized to train the model:

$$\Delta \mathscr{W} = -\eta \frac{\vartheta J}{\vartheta \mathscr{W}}. \tag{20}$$

In this way, the model is trained synchronously both with the conventional supervised loss (hard labels) so as to discriminate between different classes, and distillation loss so as maximize the QMI of the samples with the latent labels.

2.2.3 Performance evaluation

The proposed method is extensively evaluated in Appendix 7.2, including experiments on six datasets using lightweight models, as well as comparisons against competitive online distillation methods using common models (e.g., ResNet, Wide Resnet). In this Section, we provide representative experiments using lightweight models on three datasets (i.e., Cifar-10, SVHN, and Fashion MNIST) using different number of nearest neighbors (NN), and also considering various mini-batch sizes (i.e., of 32, 64, and 128 samples). The performance is measured using test accuracy (i.e., Top-1 accuracy). The experimental results are provided in Tables 3-5. Best results are printed in bold.

First, in Table 3, we can observe that the proposed online distillation method improves the baseline performance for all the values of nearest neighbors on all the utilized datasets, considering mini-batch of 32 samples. We can also see that better performance is accomplished

utilizing the maximum considered number of nearest neighbors, that is 10NN. The corresponding results setting the mini-batch size to 64 for the three utilized datasets are presented in Table 4. Similar remarks are drawn in this set of experiments. The proposed method achieves improved performance over the baseline in all considered cases. In this case, we can observe that better results are obtained for fewer nearest neighbors as compared to the previous case, that is considering 4NN and 8NN. Finally, the evaluation results considering 2NN, 4NN, 8NN and 10NN for mini-batch size set to 128 are provided in Table 5. As we can notice the enhanced performance of the proposed distillation approach against the baseline is also confirmed in this case. It is also shown that as the mini-batch size increases, better performance is achieved utilizing fewer nearest neighbors.

Method	**Cifar-10**	**SVHN**	**Fashion MNIST**
W/o Distillation	64.826% ± 0.573%	88.822% ± 0.217%	91.278% ± 0.141%
POSD-2NN	65.508% ± 0.729%	89.223% ± 0.404%	91.770% ± 0.160%
POSD-4NN	65.674% ± 0.526%	89.483% ± 0.279%	91.810% ± 0.115%
POSD-8NN	65.622% ± 0.595%	89.478% ± 0.154%	91.822% ± 0.127%
POSD-10NN	**66.280% ± 1.190%**	**89.512% ± 0.379%**	**91.882% ± 0.108%**

Table 3: Test Accuracy - Batch Size: 32

Method	**Cifar-10**	**SVHN**	**Fashion MNIST**
W/o Distillation	64.734% ± 0.654%	88.706% ± 0.306%	91.214% ± 0.141%
POSD-2NN	65.472% ± 0.914%	89.625% ± 0.307%	91.624% ± 0.111%
POSD-4NN	**66.782% ± 0.691%**	89.534% ± 0.500%	91.650% ± 0.138%
POSD-8NN	66.140% ± 1.013%	**89.912% ± 0.250%**	**91.730% ± 0.157%**
POSD-10NN	66.336% ± 0.699%	89.522% ± 0.260%	91.548% ± 0.172%

Table 4: Test Accuracy - Batch Size: 64

Method	**Cifar-10**	**SVHN**	**Fashion MNIST**
W/o Distillation	65.048% ± 0.620%	88.013% ± 0.083%	91.058% ± 0.130%
POSD-2NN	66.248% ± 0.592%	89.387% ± 0.433%	**91.728% ± 0.175%**
POSD-4NN	66.362% ± 0.726%	**89.946% ± 0.433%**	91.636% ± 0.154%
POSD-8NN	**66.760% ± 0.680%**	89.491% ± 0.363%	91.724% ± 0.121%
POSD-10NN	66.304% ± 0.919%	89.655% ± 0.334%	91.592% ± 0.151%

Table 5: Test Accuracy - Batch Size: 128

2.3 Efficient Multilayer Online Self-Acquired Knowledge Distillation for Image Classification

2.3.1 Introduction, objectives and summary of state of the art

Over the recent years, deep learning (DL) models have been successfully utilized in order to address a wide range of computer vision problems, gradually displacing previous solutions [73].

Following the evolution of DL, it is evident that DL models are becoming increasingly heavier (e.g., the older LeNet-5 model [111] has approximately 60k parameters, while the more recent VGG-16 model [194] has approximately 138M parameters). The increasing complexity of DL models is accompanied by specific computational and memory requirements, obstructing their application for robotics applications. These limitations have fueled the need for developing lightweight, fast, and effective models. Several solutions have been proposed to accomplish this goal [37]. Amongst them *Knowledge Distillation* (KD) [80] (also known as *Knowledge Transfer*) has emerged as a very effective way to tackle this challenge.

The seminal (offline) KD approach refers to the process where a high-capacity and powerful model, known as teacher, which achieves considerable performance, transfers its knowledge to a more compact and fast model, known as student. Thus, apart from the regular supervised loss, the student model is concurrently trained to mimic the so-called *soft labels* produced by the teacher model. To this aim, the parameter of temperature is introduced at the softmax activation function on the output layer of the network. The rationale behind this practice is that, compared to the hard labels, the soft labels relay more information about the way the model learns to generalize, trying implicitly to uncover similarities over the data.

KD methods can be classified into two broad categories: *offline* and *online* KD. Offline KD refers to the multi-stage process that previously described, while online KD refers to the single-stage process of training the teacher and the student models concurrently, that is by omitting the stage of pretraining the teacher model. Several online distillation methods have been proposed in the recent literature. For example, a framework of collaborative learning which trains several classifier heads of the same network at the same time, on the same training data, is proposed in [197]. The framework generates a population of classifier heads during the training process, where each head learns, apart from the hard labels, from the soft labels produced by the whole population. Furthermore, the method involves an intermediate-level representation sharing with backpropagation rescaling that aggregates the gradient flows for all the heads. Next, a two-level distillation methodology, named Online Knowledge Distillation with Diverse peers, where two types of students are involved, i.e., multiple auxiliary peers and one group leader, is proposed in [35]. Distillation is conducted among auxiliary peers with a mechanics for preserving diversity, and then an ensemble of predictions of these peers is further distilled to the group leader.

In our work, we propose a single-stage online self-acquired knowledge distillation method, namely *Multilayer Online Self-Acquired Knowledge Distillation* (MOSAKD), for ameliorating the performance of any deep neural model for classification tasks, based on our previous work published in [221]. The overarching motivation of this work is to effectively train fast and lightweight models with extra supervision, apart from the regular supervised loss, that reveals further knowledge beyond the hard labels, from the model itself and also in an online manner, so as to overcome the inherent limitations of offline KD caused by the multi-step training pipeline (i.e., time consuming, complex, and memory-wise and computationally demanding process).

The proposed method is established on the following remarks. First, considering a regular classification task, a probability distribution over the classes is produced by the softmax activation function at the output of neural network. Then, the softmax classifier inherently suppresses all the activations apart from the highest one. Hence, the conventional (offline) KD methodology manifests that the class probability distribution of a powerful teacher carries useful information on the similarities of the data with all the classes, and thus it is advantageous to retain these similarities during the training process, instead of merely training the model with the hard labels [80]. Furthermore, it has been shown that useful information can also be obtained even by transferring the knowledge from a teacher of identical capacity with the student [62]. Finally,

it has been demonstrated that compact models usually have the same representation capacity as their heavier counterparts but they are harder to train [6].

Therefore, our goal is to obtain additional knowledge about the similarities of the samples with the classes from the model itself and also in an online manner, in order to train fast and lightweight yet effective DL models. To accomplish this goal, we propose to use the k-nn non-parametric density estimation in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either by intermediate layers or by the output layer. In this fashion, we are capable of directly estimating the posterior class probabilities of the data samples, based on the neighborhood of each sample, and use them as soft labels. These soft labels explicitly express the similarity of each sample with all the classes. It should be emphasized that the proposed method is able to acquire additional knowledge from any layer, and, as shown, useful information can be distilled even from the shallower layers.

The experimental evaluation on six datasets validates the effectiveness of the MOSAKD method. Amongst them, a synthetic dataset for discriminating between humans and non humans has been created. Synthetic data has become an increasingly useful tool in training DL models, accompanied by a series of benefits [149], with a wide range of robotics applications, e.g., [120]. A few of those benefits are that synthetic data 1) provide detailed annotations, since these are automatically generated without containing errors that usually occur in the manual annotation procedure; 2) are usually large in scale, since they are procedurally generated; 3) minimize the risk of DL methods deployed in simulation environments in robotics to exhibit unstable behaviours or complete failures, due to not having been adapted to the visual differences between the virtual and the real world data. In our work, since we are focusing on robotics applications, we use a fully convolutional model which is capable of operating in real-time ($\sim$ 25 Frames Per Second - FPS) utilizing a low-power GPU [220] for high resolution input. The target is to provide semantic heatmaps of human presence on real data. That is, we train the real-time model using the proposed online distillation method on the synthetic data (only 100 real human images were used), and we test the model on unseen images that contain real humans, providing semantic heatmaps. The procedure for producing semantic heatmaps is explained in [220] (this work was included in Deliverable D4.1).

The main contributions of our work can be summarized as follows:

- We propose a novel *Multilayer Online Self-Acquired Knowledge Distillation* method.
- The MOSAKD method mines additional knowledge in an online manner from the model itself, rendering it more efficient. That is, the MOSAKD method does not require using multiple models to teach each other or copies of a given model like the existing online KD methods that leads to multiple times more computationally expensive training processes.
- The MOSAKD method is capable of mining additional information about the similarities of the samples with the classes employing intermediate and the output layers.
- The proposed method can be applied to various architectures varying from lightweight models to more complex ones.
- The self-nature of the MOSAKD method ensures that advantageous knowledge will be extracted which is a crucial issue, since the compatibility between the teacher and student models can significantly affect the effectiveness of the distillation procedure, as demonstrated in [137].

- The experimental evaluation on six dataset, including a dataset of synthetic data, validates the MOSAKD method can ameliorate the classification performance of simple and deeper models, while comparison results confirm the superiority of the method over the state-of-the-art methods.

A summary of this work is provided hereafter. The corresponding technical report is listed below and can be found in Appendix 7.3:

1. M. Tzelepi, C. Symeonidis, N. Nikolaidis and A. Tefas, "*Efficient Multilayer Online Self-Acquired Knowledge Distillation for Image Classification*", Technical Report.

2.3.2 Description of work performed so far

In our work, a novel self-distillation method is proposed that allows for developing fast-to-execute yet effective models that can comply with applications (e.g., robotics applications) with memory and computational restrictions. The proposed method allows for acquiring additional knowledge about the similarities of the samples with the classes from the model itself and also in an online manner.

More specifically, a C-class classification problem and the labeled data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \Re^D$ is an input vector and D its dimensionality, while $\mathbf{y}_i \in \mathscr{Y}^{\mathscr{C}}$ corresponds to its C-dimensional one-hot class label vector are considered. We also consider, for an input space $\mathscr{X} \subseteq \Re^D$ and an output space $\mathscr{F} \subseteq \Re^C$, as $\phi(\cdot;\mathscr{W}) : \mathscr{X} \to \mathscr{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of parameters $\mathscr{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_{N_L}\}$, where $\mathbf{W}_L$ are the weights of a specific layer L, which transforms its input vector to a C-dimensional probability vector. That is, for an input vector $\mathbf{x}_i \in \mathscr{X}$, $\phi(\mathbf{x}_i;\mathscr{W}) \in \mathscr{F}$ is the output vector given by the network ϕ with parameters $\mathscr{W}$.

Thus, considering the regular classification task, the goal is to find the parameters $\mathscr{W}^*$ that minimize the cross entropy loss, ℓ_{ce}, between the output vector $\phi(\mathbf{x}_i;\mathscr{W})$ and the one-hot class label vector $\mathbf{y}_i$:

$$\mathscr{W}^* = \arg\min_{\mathscr{W}} \sum_{i=1}^{N} \ell_{ce}(\mathbf{y}_i, \phi(\mathbf{x}_i;\mathscr{W})), \tag{21}$$

The cross entropy loss for a sample i is defined as:

$$\ell_{ce}(\mathbf{y}_i, \mathbf{z}_i) = \sum_{m=1}^{C} y_i^m \log(z_i^m), \tag{22}$$

where y_i^m is the m-th element of $\mathbf{y}_i$ one-hot label vector, and z_i^m is defined as the output of the softmax operation on the C-dimensional network's output:

$$z_i^m = \frac{\exp(\phi(\mathbf{x}_i;\mathscr{W})^m)}{\sum_{j=1}^{C} \exp(\phi(\mathbf{x}_i;\mathscr{W})^j)}. \tag{23}$$

Our goal is to mine additional knowledge from the model itself in an online fashion. To achieve this goal, we propose to use k-nn non-parametric density estimation in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either any intermediate layer or the output layer. The idea of non-parametric

density estimation is based, in general, on the fact that the probability P that a vector $\mathbf{x}$ will fall in a region R is given by:

$$P = \int_R p(\mathbf{x}')d\mathbf{x}'. \tag{24}$$

Thus, P is a smoothed version of the density function $p(\mathbf{x})$ and this smoothed value of p can be estimated by estimating the probability P. If we consider that N samples $\{\mathbf{x}_i\}_{i=1}^N$ are drawn independently and identically distributed according to $p(\mathbf{x})$, then the probability that k of these N lie inside in R is given by the binomial law:

$$P_k = \binom{N}{k} P^k (1-P)^{N-k}, \tag{25}$$

and the expected value for k is given by:

$$\mathbb{E}[k] = NP. \tag{26}$$

Furthermore, this binomial distribution for k peaks very sharply about the mean, thus we expect that the ratio $\frac{k}{N}$ will be a good estimate for the probability P, and hence for the smoothed density function. The estimate becomes more accurate as N increases. Making also the assumption that $p(\mathbf{x})$ is continuous and the region R is so small that p does not significantly vary within it, we arrive at the equation:

$$\int_R p(\mathbf{x}')d\mathbf{x}' \simeq p(\mathbf{x})V, \tag{27}$$

where $\mathbf{x}$ is a point within R and V is the volume enclosed by R. By combining Eq. (24), (26), and (27) we arrive at the following estimate for $p(\mathbf{x})$:

$$p(\mathbf{x}) = \frac{\frac{k}{N}}{V}. \tag{28}$$

This equation can be, in general, exploited in two different manners: either to define the number of nearest neighbors, k, and to adjust the volume V from the data, which stands for the k-nn density estimation, or to define the volume V and to observe how many points k fall into the region, which gives rise to the parzen windows. The essential advantage of the first one is that it allows for directly estimating the posterior probabilities $P(c_m|\mathbf{x})$ of the class being c_m, $m = \{1,\cdots,C\}$, from a set of N labeled data by using the samples to estimate the densities involved. To do this, we apply the k-nn density estimation to each class separately and then we make use of the Bayes rule. More specifically, assuming that we have a dataset consisting of N_m samples that belong to class c_m, with N samples in total, so that $\sum_m N_m = N$, we place a sphere of volume V around $\mathbf{x}$ and capture k samples, k_m of which belong to the class c_m. Then, the estimate for the probability $p(\mathbf{x}|c_m)$ is:

$$p(\mathbf{x}|c_m) = \frac{k_m}{N_m V}, \tag{29}$$

and similarly the unconditional density is given by:

$$p(\mathbf{x}) = \frac{k}{NV}, \tag{30}$$

while the priors can be approximated by:

$$P(c_m) = \frac{N_m}{N}. \tag{31}$$

Thus, we apply the Bayes rule to compute the posterior class probabilities:

$$P(c_m|\mathbf{x}) = \frac{p(\mathbf{x}|c_m)P(c_m)}{p(\mathbf{x})} = \frac{\frac{k_m}{N_m V}\frac{N_m}{N}}{\frac{k}{NV}} = \frac{k_m}{k}. \tag{32}$$

Considering the output of a specific layer, L, and for a specific number of nearest neighbors, k, at the feature space generated by the layer L, the posterior probabilities can be estimated and used as soft labels. These soft labels explicitly encode information about the similarities of the samples with the classes. That is, the soft label for a sample i is given by:

$$\mathbf{s}_i^L = [\frac{k_1^L}{k}, \frac{k_2^L}{k}, \ldots, \frac{k_C^L}{k}], \tag{33}$$

where k_m^L, $m = \{1, \cdots, C\}$ is the number of nearest neighbors that belong to the class c_m, $m = \{1, \cdots, C\}$ at the layer L.

Therefore, in the MOSAKD training procedure, the goal is to find the parameters $\mathscr{W}^*$ that minimize the overall loss of cross entropy, ℓ_{ce} and self-distillation, ℓ_{sd}:

$$\mathscr{W}^* = \arg\min_{\mathscr{W}} \sum_{i=1}^{N} [\ell_{ce}(\mathbf{y}_i, \phi(\mathbf{x}_i; \mathscr{W})) + \lambda \ell_{sd}(\mathbf{s}_i^L, \phi(\mathbf{x}_i; \mathscr{W}))], \tag{34}$$

where λ controls the relative importance between the two losses. We use Mean Squared Error (MSE) as self-distillation loss, however Kullback-Leibler (KL) divergence could also be used.

Consequently, in the MOSAKD training process the input images are fed to the network, and for each sample the predictions for belonging to each of the classes are produced. Concurrently, the soft labels are computed based on the neighborhood of each sample, considering the feature space generated by a specific neural layer, according to the procedure described above. Then, the network is trained using the cross entropy loss with the hard labels, and concurrently using the distillation loss to regress the produced soft labels, encouraging it to regard the similarity of each sample with the classes. The MOSAKD training procedure is also illustrated in Fig. 2.

2.3.3 Performance evaluation

The MOSAKD method is extensively evaluated in Appendix 7.3. In this Section, we first provide indicative experiments on Cifar-10 and Synthetic-Human datasets, utilizing a simple lightweight model and the real-time VGG-1080p model [220], respectively. Performance is measured using test accuracy (classification accuracy). The experimental results for each utilized layer and for various numbers nearest neighbors (NN) for computing the soft labels are presented in Tables 6 and 7. As it is demonstrated, the MOSAKD method improves the baseline performance of training without distillation for all the utilized layers and also for all the values of NN.

Subsequently, we compare the performance of the MOSAKD method with state-of-the-art online distillation methods using a common model, that is Wide ResNet 20-08 (WRN-20-08) model on Cifar-100 dataset. The experimental results are presented in Table 8. As it shown, the MOSAKD method achieved superior performance over existing methods.

Finally, we use the proposed trained model on the synthetic human dataset to generate heatmaps on unseen high-resolution images that contain real humans. That is, unseen images of

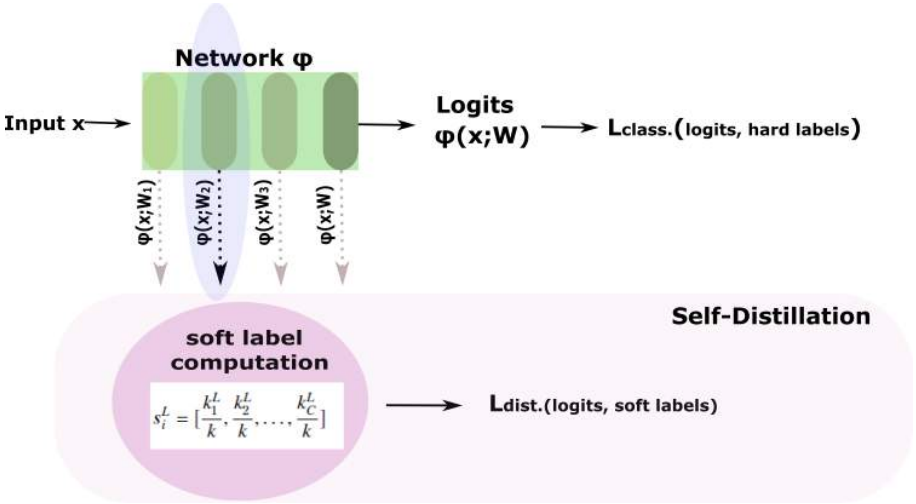


Figure 2: MOSAKD Training Procedure: The input images are fed to the network, and for each sample the predictions for belonging to each of the classes are produced. Concurrently, considering a specific neural layer, the soft labels are computed based on the neighborhood of each sample in the feature space generated by the specific layer. Then, the network is trained at the same time using the cross entropy loss with the hard labels and using the distillation loss with the produced soft labels.

Table 6: Cifar-10: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: $64.734\% \pm 0.654\%$.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	66.204% ± 0.768%	66.076% ± 0.241%	65.818% ± 0.560%	65.994% ± 0.659%	66.022% ± 0.469%
Layer 2	**66.307% ± 0.687%**	65.660% ± 0.710%	**66.756% ± 0.953%**	66.423% ± 1.100%	**66.584% ± 0.742%**
Layer 3	66.150% ± 0.593%	65.752% ± 0.534%	65.318% ± 1.035%	65.856% ± 0.638%	66.304% ± 0.883%
Layer 4	65.210% ± 0.764%	**66.176% ± 0.979%**	66.561% ± 0.635%	66.468% ± 0.587%	65.332% ± 0.511%
Layer 5	66.056% ± 0.808%	65.729% ± 0.424%	65.962% ± 0.841%	**66.570% ± 0.563%**	66.472% ± 0.260%

Table 7: Synthetic Human: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: $97.048\% \pm 0.476\%$.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	97.052% ± 0.716%	**98.275% ± 0.550%**	98.192% ± 0.801%	98.223% ± 1.288%	**98.592% ± 0.744%**
Layer 2	97.866% ± 0.684%	98.096% ± 1.129%	**98.322% ± 0.738%**	97.914% ± 0.641%	97.982% ± 0.672%
Layer 3	97.515% ± 0.636%	96.814% ± 1.916%	97.084% ± 1.113%	98.046% ± 0.908%	97.842% ± 1.009%
Layer 4	**97.982% ± 0.882%**	97.264% ± 1.482%	98.192% ± 1.082%	97.404% ± 1.129%	98.426% ± 0.981%
Layer 5	96.962% ± 0.997%	97.946% ± 0.899%	98.268% ± 0.556%	**98.376% ± 0.611%**	97.960% ± 0.711%

Table 8: Comparisons Against Online Distillation Methods on Cifar-100 utilizing the WRN-20-8 architecture.

Method	Classification Accuracy
WRN-20-8	77.50% ± 0.44%
DML [255]	79.79 ± 0.11%
ONE [107]	78.81 ± 0.12%
CL-ILR [197]	79.56 ± 0.13%
OKDDip [35]	80.37 ± 0.07%
DFL [148]	80.51% ± 0.49%
MOSAKD - Layer 1	**80.79% ± 0.20%**
MOSAKD - Layer 2	**80.76% ± 0.21%**
MOSAKD - Layer 3	**80.72% ± 0.23%**
MOSAKD - Layer 4	**81.01% ± 0.21%**

Figure 3: Heatmap on real-image containing humans utilizing the human detection model trained on synthetic humans.

size 1920×1080 are fed to the network, and for every window 64×64 we compute the output of the network at the output layer. Indicative evaluation results are illustrated in Figs. 3 and 4. As it shown, the model which is trained on synthetic images (only 100 real images depicting humans are used) can successfully detect humans on real images.

2.4 Improving Binary Semantic Scene Segmentation for Robotics Applications

2.4.1 Introduction, objectives and summary of state of the art

Semantic scene segmentation refers to the task of assigning a class label to each pixel of an image, and hence it is also known as pixel-level classification. Semantic scene segmentation is a challenging task involved in numerous robotics applications, such as autonomous driving [4]. Robotics applications are accompanied by particular computational requirements. That is, the utilized models should be able to effectively operate at sufficient speed, on embedded low power GPUs, while also considering high-resolution input.

Recent advances in Deep Learning (DL) have provided effective models for addressing the general problem of semantic scene segmentation [67], however, without considering the issue of deployment (inference) speed. That is, most of the existing state-of-the-art DL segmentation models are computationally heavy, and hence ill-suited for robotics applications. Thus, in the

Figure 4: Heatmap on real-image containing humans utilizing the human detection model trained on synthetic humans.

recent literature there have been works that also focus on the deployment speed, providing real-time segmentation models, considering mainly high power GPUs [36, 171, 31, 244, 55, 245]. A comparative study of current semantic segmentation models considering the inherent computational restrictions in the context of robotics applications is provided in [222] (this work was presented in Deliverable D4.1). More specifically, extensive experiments have been conducted on different embedded platforms (e.g., AGX Xavier, NVIDIA TX-2), and also for various input resolutions, ranging from lower to higher ones. From the conducted experiments, it is evident that the Bilateral Segmentation Network (BiseNet) [245] model achieves considerable performance considering the segmentation accuracy-speed trade-off. Towards this end, in this work we employ BiseNet model and we address the problem of binary semantic segmentation considering robotics applications.

In our work, we explore ways of improving the performance of BiseNet model both in terms of deployment speed and segmentation accuracy, considering binary segmentation problems. To this aim, we first propose a lightweight version of the model. That is, we propose a lightweight network instead of ResNet-18 that is used in the so-called *context path*. Subsequently, we exploit the available losses. Specifically, apart from the widely used cross entropy (softmax) loss, which is also used in the initial version of BiseNet, we apply hinge loss, since as it is shown in the recent literature, it provides improved accuracy considering binary classification problems [220].

A summary of this work is provided hereafter. The corresponding technical report is listed below, and can be found in Appendix 7.4:

1. M. Tzelepi, N. Tragkas and A. Tefas, "*Improving Binary Semantic Scene Segmentation for Robotics Applications*", Technical Report.

2.4.2 Description of work performed so far

The BiseNet model consists of two paths, that is *Spatial Path* and *Context Path*. The spatial path is used in order to preserve the spatial information and generate high resolution features, and the context path with a fast downsampling strategy is used in order to obtain sufficient receptive field. Furthermore, the model includes two modules, that is a Feature Fusion Module and an Attention Refinement Module, in order to further improve the accuracy with acceptable cost. Finally, apart from the principal cross entropy loss which supervises the output of the BiseNet model, two auxiliary losses are utilized to supervise the output of the context path.

In our work, we propose to replace the ResNet-18 model used in the context path, with

a more lightweight model. The proposed model, which is based on the VGG model [194], consists of five pairs of convolutional layers followed by batch normalization and ReLU activation. A max-pooling layer follows each convolutional block. As it will be presented the modified BiseNet model utilizing the proposed lightweight model in the context path achieves considerable speed improvements.

Subsequently, since lightweight models usually have inferior performance as compared to the heavyweight counterparts, we explore ways to improve their performance. To achieve this goal, we focus on the loss functions for training the segmentation model. More specifically, even though cross entropy (softmax) loss, is a widely used loss function in DL, in a recent work [220] it has been demonstrated that considering binary classification problems, hinge loss can achieve improved classification performance. Motivated by the aforementioned observation, in this work, we extend this investigation on binary segmentation problems. Thus, we utilize hinge loss so as to supervise the output of the whole model. Hinge loss per pixel is defined as:

$$\ell_h = \sum_{j=1}^{N_c} max(0, 1 - \zeta\{c = j\} y_j^{last}) \tag{35}$$

where $c \in [1, \cdots, N_c]$ indicates the correct class among the N_c classes, y_j^{last} indicates the score with respect to the j-th class, and

$$\zeta\{condition\} = \begin{cases} 1 & \text{, if condition} \\ -1 & \text{, otherwise} \end{cases}$$

In our case, $N_c = 2$, since we deal with binary segmentation problems.

2.4.3 Performance evaluation

The proposed method is extensively evaluated in Appendix 7.4. In this Section, we first provide indicative experiments on the deployment speed in terms of Frames Per Second (FPS) for the modified BiseNet model, as well as on the performance in terms of mean Intersection Over Union (mIOU) for the hinge loss. Experiments conducted on two binary datasets created from the CityScapes [41] dataset, exploiting only the Human and Vehicle classes.

The evaluation results with respect to the deployment speed of the modified model against the BiseNet model with ResNet-18 on an NVIDIA 2-80 Ti, and on an NVIDIA Jetson AGX Xavier, for various input sizes are presented in Tables 9 and 10. As it is shown, the modified model is considerably faster in any considered case.

Next, the experimental results for evaluating the hinge loss against cross entropy loss, utilizing the proposed lightweight BiseNet model, on the two binary segmentation datasets are presented in Table 11. Best results are printed in bold. As it is demonstrated, hinge loss accomplishes superior performance as compared to the cross entropy loss, considering binary segmentation problems.

Finally, some qualitative results are presented utilizing the proposed modified model trained for human segmentation and vehicle segmentation in Fig. 5.

2.5 Soft Label Embedding Methods for Improved Object Recognition

2.5.1 Introduction, objectives and summary of state of the art

The popular one-hot encoding method, typically used in classification problems in conjuction with the cross-entropy loss function, fails to capture the *inherent uncertainty* in data labeling

Table 9: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA 2080 Ti.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	66.51	**75.23**
1280×720	70.69	**84.06**
800×800	98	**119.76**
600×600	163.91	**215.61**
640×360	216.37	**305.89**
400×400	269.58	**353.59**

Table 10: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA Jetson AGX Xavier.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	11.58	**15.32**
1280×720	12.23	**16.96**
800×800	16.38	**23.40**
600×600	26.78	**40.12**
640×360	40.13	**60.66**
400×400	52.42	**77.83**

Table 11: Evaluation on segmentation performance in terms of mIOU (%) utilizing the modified lightweight BiseNet model, for evaluating hinge loss against cross entropy loss.

Dataset	Cross Entropy Loss	Hinge Loss
Human Vs Non-Human	87.82	**89.23**
Vehicle Vs Non-Vehicle	94.82	**95.03**

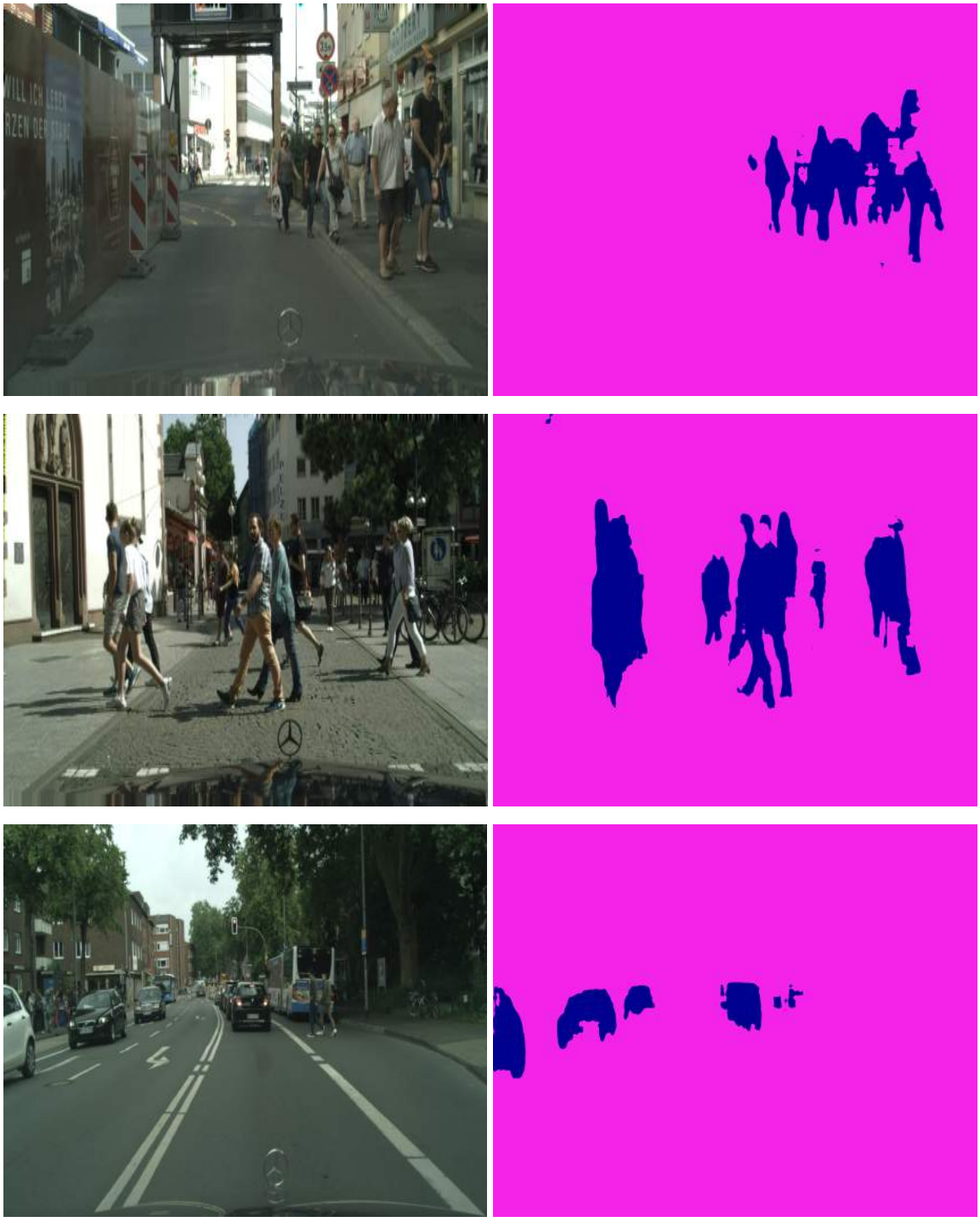


Figure 5: Predictions of the modified lightweight BiSeNet model trained on Human Vs Non-Human and Vehicle Vs Non-Vehicle datasets.

processes. Among the sources of uncertainty is the similarity between classes in a dataset, e.g., a class of objects can resemble another class or be very dissimilar from others. Furthermore, certain instances may be ambiguous, even for human annotators. One-hot encoding, due to its crisp nature, can also lead to overfitting more easily. As an example, consider a binary classification example, where a difficult sample may be correctly classified with a predicted probability of 0.8. Despite the prediction being correct, most recent approaches to this problem would further modify the network weights such that the predicted probability lies closer to 1. In aiming to learn this crisp probability, the network loses some of its generalization ability.

Figure 6 illustrates general-class and instance-specific similarities using the CIFAR10 dataset [100]. Two dimensional representations are obtained using t-SNE [225] and a DenseNet [84] classifier, using the representations of the penultimate layer. The general class similarities, like those between the *dog* and *cat* classes, or the *automobile* and *truck* classes, reflect the actual similarity between these real categories. Instance specific similarities exist as well, as shown in the marked misclassified samples: (a) a dog misclassified as a cat, (b) a truck misclassified as a car, (c) a deer misclassified as a horse, (d) a bird misclassified as a ship, (e) a bird misclassified as a ship, (f) a deer misclassified as a horse, and (g) a dog misclassified as a bird.

Soft labels have been studied in the past, in the context of simpler classifiers like the k-Nearest Neighbor one [53], and recently scientific research in this task has resurfaced. In [58], a method for modeling subjectivity in emotion recognition was proposed. An ensemble method, where different networks are trained with different annotators, was compared against a single network trained on soft labels, generated by averaging the labels from different annotators. In [226], soft labels were learned in a meta learning fashion, by treating them as learnable parameters, modeling both class-level and instance-level similarities. Soft labels have also recently been linked to knowledge distillation methods [221, 219].

In this work, we tackle the shortcomings associated with the one-hot encoding by introducing a framework for learning soft label embeddings. We model relationships both at a class-level and an instance-level into the proposed label embedding methods. Two variants are proposed: first, a learnable general-class embedding which aims to capture information regarding inter-class similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly classified ones. The concept of this framework is similar to real-life learning procedures, where uncertainty rules over many subjects.

A summary of this work is provided hereafter. The corresponding technical report is listed below, and can be found in Appendix 7.5:

1. P. Nousi and A. Tefas, "*Self-Learned Label Embeddings for General Class and Instance Specific Similarity Modeling in Classification*", Technical Report.

2.5.2 Description of the work performed so far

Let $\mathbf{x}_i \in \mathbb{R}^D$ denote an input sample and $\mathbf{y}_i \in \{0,1\}^K$ such that $\sum_{k=1}^{K} y_{i,k} = 1$ be its one-hot encoding, in a set of $i = 1,\ldots,N$ training samples spanning over K distinct classes. A standard classifier $f(\mathbf{x})$ is trained to map the input samples $\mathbf{x}_i$ to their corresponding one-hot labels and to make predictions on unseen samples. Typically, the Cross Entropy loss function is used for this purpose:

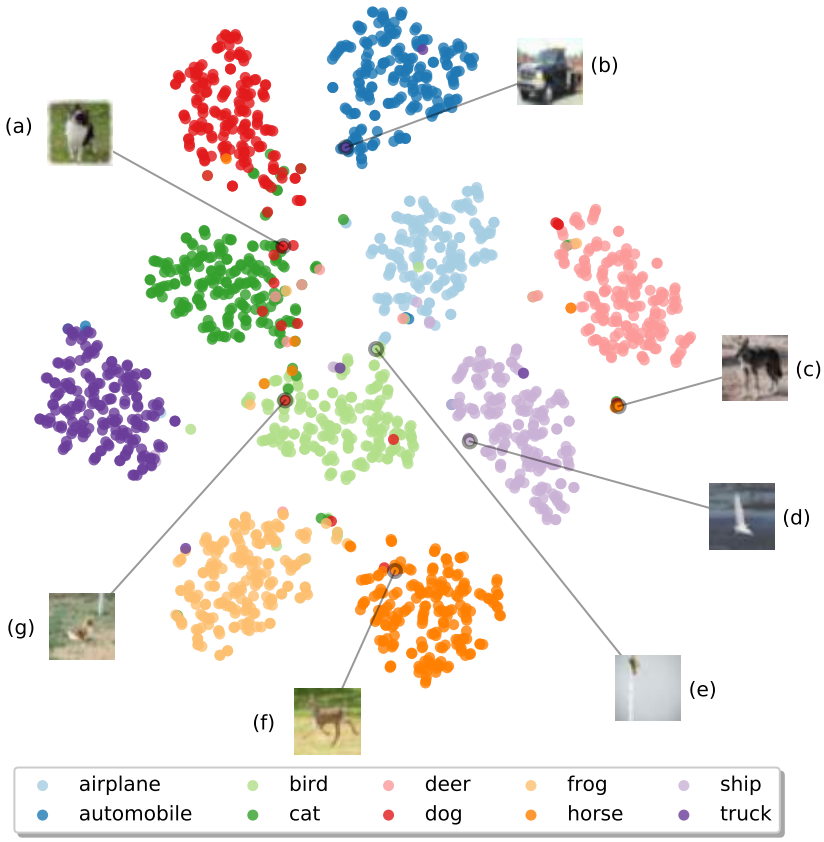


Figure 6: Visualization of feature vectors from the CIFAR10 dataset using t-SNE. General class similarities exist in this dataset as shown by the proximity of the *dog* and *cat* classes, as well as that between the *automobile* and *truck* classes. In contrast, the *deer* and *frog* classes are less similar and less likely to be confused for one another. Instance specific similarities exist as well, as shown in the marked misclassified samples: (a) a dog misclassified as a cat, (b) a truck misclassified as a car, (c) a deer misclassified as a horse, (d) a bird misclassified as a ship, (e) an airplane misclassified as a bird, (f) a deer misclassified as a horse, and (g) a dog misclassified as a bird.

$$CE(\mathbf{y},\mathbf{p}) = -\sum_{k=1}^{K} y_k \cdot \log(p_k) \tag{36}$$

where $\mathbf{p}$ are the probabilities predicted by $f(\mathbf{x})$, obtained as the output of a softmax function on the output, or logits, of the classifier. We propose the use of soft labels $\hat{\mathbf{y}}_i \in \mathbb{R}^K$ such that $\sum_{k=1}^{K} \hat{y}_{i,k} = 1$ for all samples in the dataset, such that they capture not only the groundtruth label of each sample but also inter-class similarities as well as similarities between each sample and other samples present in the dataset. Our hypothesis is that by using soft labels, a neural classifier can learn to generalize better, as the sum of errors from correctly-classified but difficult samples will decrease. In terms of the representation learned, less effort will be consumed towards separating such samples from their similar neighbors.

The new learning task is formulated as:

$$CE(\hat{\mathbf{y}},\mathbf{p}) = -\sum_{k=1}^{K} \hat{y}_k \cdot \log(p_k) \tag{37}$$

where $\hat{\mathbf{y}}$, i.e., the soft labels, are given by a differentiable function $g(\cdot)$, the parameters of which can be learned in conjunction with the parameters of the classifier during training.

General Class Soft-Label Embedding We formulate the function $g(\cdot)$ for the case of general class similarities as a simple embedding of the form $g(\mathbf{y}) = \hat{\mathbf{y}} = \mathbf{W} \cdot \mathbf{y}$ where $\mathbf{W} \in \mathbb{R}^{K \times K}$ is a learnable weights matrix. This can be viewed as a neural network with a single linear hidden layer. We are interested in modeling the generic case where each class label is given as a linear combination of all classes, including itself which should intuitively hold a larger weight:

$$\hat{\mathbf{Y}} = g(\mathbf{Y}) = \mathbf{W} \cdot \mathbf{Y} = \mathbf{W}. \tag{38}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{K \times K}$ is the soft label matrix, i.e., each row $\hat{\mathbf{y}}_k$ corresponds to the soft labels of the k-th class, and $\mathbf{Y} = \mathbf{I}_K$ holds the one-hot encodings of all classes. That is, the new label vector for the k-th class is given by:

$$\hat{\mathbf{y}}_k = g(\mathbf{y}_k) = \mathbf{W} \cdot \mathbf{y}_k = [W_{k1}, W_{k2}, \ldots, W_{kk}, \ldots, W_{kK}] \tag{39}$$

i.e., practically the k-th row of the weight matrix $\mathbf{W}$, as $\mathbf{y}_k$ is the one-hot encoding of the k-th class. As mentioned, the weights matrix should optimally conform to:

$$W_{kk} \geq \sum_{l \neq k}^{K} W_{kl}, \tag{40}$$

so as to retain the groundtruth information. A softmax function is applied on $\hat{\mathbf{y}}_k$ to ensure that $\sum_{l=1}^{K} \hat{y}_{kl} = 1$.

The classifier and label embedding network are trained in parallel by optimizing the cross entropy loss given by Eq. (37) for all input samples. The new classification targets are set to be a weighted combination of the one-hot encoding and their linear combinations given by $g(\mathbf{y})$, i.e.:

$$\hat{\mathbf{y}}_k = \alpha \cdot g(\mathbf{y}_k) + (1-\alpha) \cdot \mathbf{y}_k \tag{41}$$

where $\alpha \in (0,1)$ controls the softness of the labels.

Instance Specific Soft-Label Embedding In the instance-specific case, the soft labels for each instance $\mathbf{x}_i$ are a function of the instance itself, that is:

$$\hat{\mathbf{y}}_i = h(\mathbf{x}_i). \tag{42}$$

The criterion from Eq. (37) still applies in this case and the function $h(\cdot)$ can take the form of a network which takes $\mathbf{x}_i$ as input and outputs $\hat{\mathbf{y}}_i$.

An *external* representation of $\mathbf{x}_i$ can be extracted using an Autencoder [151], where the intermediate representation $\mathbf{z}_i$ is set to be K-dimensional. Let $g(\cdot)$, $f(\cdot)$ denote the encoding and decoding functions respectively, then $\hat{\mathbf{x}}_i \in \mathbb{R}^D$ is the network's output, which is trained to approximate the input. The intermediate representation $\mathbf{z}_i = g(\mathbf{x}_i)$ is given by the encoder. To ensure a proper mapping between the AEs latent dimensions and the one-hot encoding, a cross entropy criterion is added to the objective. The final learning objective is:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} CE(\mathbf{y}_i, \sigma(\mathbf{z}_i)) + CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \lambda \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{43}$$

where λ is a constant introduced to weigh the classification and reconstruction losses.

In other words, the intermediate representation of the AE is trained to match the groundtruth labels using a softmax function $\sigma(\cdot)$, and the cross entropy criterion. The AE is also trained using the MSE between the input and its prediction. Finally, the classifier's predictions are matched to the soft targets, which are given by the intermediate representation. In practice, an equation similar to Eq. 41 is used to generate the soft targets, so as to maintain the groundtruth information:

$$\hat{\mathbf{y}}_k = \beta \cdot \sigma(\mathbf{z}_k/T) + (1-\beta)\cdot \mathbf{y}_k \tag{44}$$

where T denotes the softmax temperature.

Combined General-Class and Instance-Specific Label Embedding The instance-specific case can be combined with the general-class similarity embedding with the addition of the general-class label embedding network $g(\mathbf{y})$. The learning objective is modified accordingly:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} CE(f(\mathbf{y}_i), \mathbf{z}_i) + \beta \cdot CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \gamma \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{45}$$

where β and γ weigh the classification and reconstruction losses. In this case, it is crucial to properly initialize the general-class embedding network so as to output labels as close as possible to the groundtruth one-hot encoding. In practice, the soft labels used to train the classifier are a linear combination of the one-hot groundtruth labels, the general-class embeddings and the instance-specific embeddings, weighted by two hyperparameters to control the effect of each embedding on the final targets.

2.5.3 Performance Evaluation

The results of our extensive experiments are presented in the technical report in Appendix 7.5. In this Section, we provide indicative experiments using a ResNet-18 model and the CIFAR100 dataset, presented in Table 12, in terms of baseline accuracy, accuracy using each compared method, and relative improvement in accuracy, to account for different training hyperparameters

for each method. The proposed method provides a significant relative improvement, comparable to similar methods.

Network	Method	Acc	Acc w/ Method	Rel. Impr.
ResNet-18	Proposed, GC		77.40	2.09
	Proposed, IS	75.81	77.11	1.71
	Proposed, GC+IS		77.27	1.92
	Tf-KD [246]	75.87	77.10	1.62
	DynLS [226], Class	77.1	78.8	2.20
	DynLS [226], Instance	77.1	79.2	2.72
	LabelEmb [203]	72.65	76.03	4.65
ResNet-56	OLS [250]	74.73	76.09	1.81
	LS [143]	72.1	72.7	0.83

Table 12: Relative improvement in accuracy using the proposed label embedding methods for the CIFAR100 dataset.

2.6 Quadratic Mutual Information Modeling for Efficient Hashing

2.6.1 Introduction, objectives and summary of state of the art

The vast amount of data available nowadays, combined with the need to efficiently and promptly provide answers to users' queries led to the development of several *hashing* techniques [51, 122]. Hashing provides a way to represent objects using compact codes, that allow for performing fast and efficient queries in large object databases, lowering the computational requirements and accelerating many information retrieval-based applications. The increasing need to perform inference on the edge using embedded devices with limited processing power [109, 159], along with the vast amount of multimedia data collected from various sensors [169, 153], further stress the need for developing efficient hashing methods.

Early hashing methods, e.g., Locality Sensitive Hashing (LSH) [43], focused on extracting generic codes that could, in principle, describe every possible object and information need. However, it was later established that *supervised hashing*, that learns compact hash codes that are tailored to the task at hand, can significantly improve the retrieval precision, as well as reduce the size of the extracted codes. In this way, it is possible to learn even smaller hashing codes, since the extracted code must only encode the information needs for which the users are actually interested in. However, note that the extracted hash codes must also encode part of the semantic relationships between the encoded objects, to allow for providing a meaningful ranking of the retrieved results. Many supervised and semi-supervised hashing methods have been proposed [106, 236, 253]. However, many deep supervised hashing techniques ignore several information-theoretic aspects of the process of information retrieval, often leading to sub-optimal results. For example, many methods employ the pairwise distances between the images [106, 236, 261] or are based on sampling *triplets* that must satisfy specific relationships according to the given ground truth [253, 258]. On the other hand, *information-theoretic* measures, such as mutual information [172], have been proven to provide robust solutions to many

machine learning problems, e.g., classification [172]. However, very few steps towards using these measures for supervised hashing tasks have been made so far.

In this work, we provide a connection between an information-theoretic measure, the Mutual Information (MI) [172], and the process of information retrieval. More specifically, we argue that mutual information can naturally model the process of information retrieval, providing a solid framework to develop efficient retrieval-oriented supervised hashing techniques. Even though MI provides a well-defined theoretical formulation for the problem of information retrieval, applying it in real scenarios is usually intractable, since there is no efficient way to calculate the actual probability densities, that are involved in the calculation of MI. The great amount of data as well as their high dimensionality further complicate the practical application of such measures.

The main contribution of this work is the proposal of an efficient deep supervised hashing algorithm that is capable of extracting short and efficient codes using an extension of an information-theoretic measure, the Quadratic Mutual Information (QMI) [211]. The architecture of the proposed method is shown in Fig. 7. To derive a practical algorithm that can efficiently scale to large datasets:

1. We adapt QMI to the needs of supervised hashing by employing a similarity measure that is close to the actual distance used for the retrieval process, i.e., the Hamming distance. This gives rise to the proposed *Quadratic Spherical Mutual Information* (QSMI). It is also experimentally demonstrated that the proposed QSMI is more robust compared to the classical Gaussian-based Kernel Density Estimation used in QMI [211], while it does not require careful tuning of any hyper-parameters.

2. We propose using a more smooth optimization objective employing a novel square clamping approach. This allows for significantly improving the stability of the optimization, while reducing the risk of converging to bad local minima.

3. We adapt the proposed approach to work in batch-based setting by employing a method that dynamically estimates the prior probabilities, as they are observed within each batch. In this way, the proposed method can efficiently scale to larger datasets.

4. We demonstrate that the proposed method can be readily extended to efficiently handle different scenarios, e.g., retrieval of unseen classes [182].

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.6:

1. [162] N. Passalis and A. Tefas, "*Deep Supervised Hashing using Quadratic Spherical Mutual Information for Efficient Image Retrieval*", Signal Processing: Image Communication, 2021

2.6.2 Description of work performed so far

Let $\mathscr{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ be a collection of N images, where $\mathbf{y}_i \in \mathbb{R}^n$ is the representation of the i-th image extracted using an appropriate feature extractor, e.g., a deep neural network. Also, note that even though the term "information need" is used to described a textual query in traditional Information Retrieval, in this Section we use this term to describe any possible need that arises from a user's query, which is not necessarily limited to textual queries. Therefore,

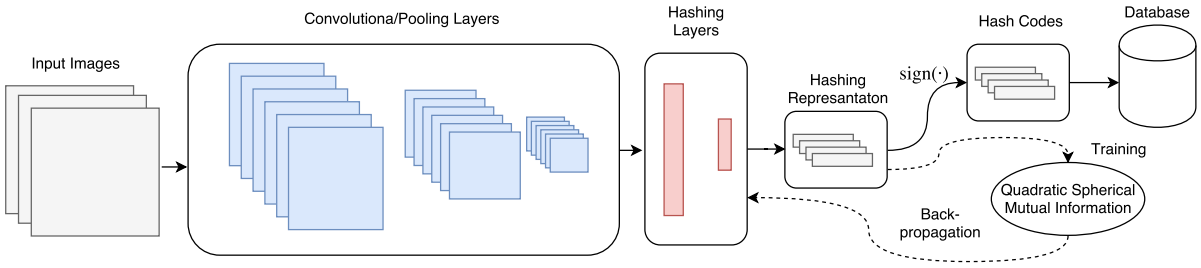


Figure 7: Pipeline of the proposed hashing method: A deep convolutional neural network (CNN) is used to extract a representation that can be used to directly obtain a compact binary hash code. The network is optimized using the proposed Quadratic Spherical Mutual Information loss that is adapted towards the needs of efficient image hashing.

without loss of generality, we focus on the case of content-based image retrieval (CBIR) [115], where each information need is expected by an image query. The problem of information retrieval can be then defined as follows: *Given an information need q retrieve the images of the collection $\mathscr{Y}$ that fulfill this information need and rank them according to their relevance to the given information need.* Since this work focuses on content-based information retrieval, the information need q is expressed through a *query image*, which is usually not a part of the collection $\mathscr{Y}$.

To be able to measure how well an information retrieval system works, a ground truth set that contains a set of information needs and the corresponding images that fulfill these information needs is usually employed. Let M be the number of information needs $\mathscr{Q} = \{q_1, q_2, ..., q_M\}$. Then, for each information need q_i, a set of images $\mathscr{Q}_i = \{\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, ..., \mathbf{y}_{N_i}^{(i)}\}$, where $\mathbf{y}_j^{(i)} \in \mathbb{R}^n$ is the representation of the j-th image that fulfills the i-th information need, is given. Please note that the notation $\mathbf{y}_j$ is used to refer to the representation of the j-th image (out of N images) of the dataset, while the notation $\mathbf{y}_j^{(i)}$ is used to refer to the representation of the j-th image (out of N_i images) that fulfills the information need q_i. We use this overloaded notation to simplify the presentation of the proposed approach. Also, note that since all images contained in $\mathscr{Q}_i$ fulfill the same information need, they can be all used as queries to express the information need q_i. However, there are also other images, which are usually not known beforehand, which also express the same information need and they can be also used to query the database. The distribution of the images that fulfill the i-th information need can be modeled using the *conditional probability density* function $p(\mathbf{y}|q_i)$.

Let $\mathscr{Y}$ be a random vector that represents the images and $\mathscr{Q}$ be a random variable that represents the information needs. The Shannon's entropy of information needs, which expresses the uncertainty regarding the information need that a randomly sampled image fulfills, is defined as [172]:

$$H(\mathscr{Q}) = -\sum_q P(q) log(P(q)), \tag{46}$$

where $P(q)$ is the prior probability of the information need q, i.e., the probability that a random image of the collection fulfills the information need q. Note that above definition implicitly assumes that the information needs are mutually exclusive, i.e., $\sum_q P(q) = 1$, or equivalently, that each image satisfies only one information need. This is without loss of generality, since it is straightforward to extend this definition to the general case, where each image can satisfy

multiple information needs, e.g., by measuring the entropy of each information need separately:

$$H(\mathscr{Q}) = -\sum_{q}\Big(P(q)log(P(q)) + (1-P(q))log(1-P(q))\Big). \tag{47}$$

To simplify the presentation of the proposed method, we assume that the information needs are mutually exclusive. Nonetheless, the proposed approach can be still used with minimal modifications, even when this assumption does not hold.

When the query vector is known, then the uncertainty of the information need that it fulfills can be expressed by the conditional entropy:

$$H(\mathscr{Q}|\mathscr{Y}) = -\int_{\mathbf{y}} p(\mathbf{y}) \left(\sum_{q} p(q|\mathbf{y})log(p(q|\mathbf{y}))\right) d\mathbf{y}. \tag{48}$$

Mutual information (MI) is defined as the amount by which the uncertainty for the information needs is reduced after observing the query:

$$\begin{aligned} I(\mathscr{Q},\mathscr{Y}) &= H(\mathscr{Q}) - H(\mathscr{Q}|\mathscr{Y}) \\ &= \sum_{q}\int_{\mathbf{y}} p(q,\mathbf{y})log(\frac{p(q,\mathbf{y})}{P(q)p(\mathbf{y})})d\mathbf{y}. \end{aligned} \tag{49}$$

It is desired to maximize the MI between the representation of the images $\mathscr{Y}$ and the information needs $\mathscr{Q}$, since this ensures that the uncertainty regarding the information need, which a query image expresses, is minimized. However, it is usually intractable to directly calculate the required probability density $p(\mathbf{y}|q_i)$ and the corresponding integral in Eq. (49), limiting the practical applications of MI.

Quadratic Mutual Information: When the aim is not to calculate the exact value of MI, but to optimize a distribution that maximizes the MI, then a quadratic divergence metric, instead of the Kullback-Leibler divergence, can be used. Despite its advantages. QMI still suffers from several limitations: For example, QMI involves the calculation of the pairwise similarity matrix between all the images of a collection. This quickly becomes intractable, as the size of the collection increases. Also, selecting the appropriate width for the Gaussian kernels is not always straightforward, as a non-optimal choice can distort the feature space and slow down the optimization. To overcome these limitations, we propose the *Quadratic Spherical Mutual Information* (QSMI). Instead of relying on the Euclidean distance between two samples, as in the regular QMI, the angle between two samples is used. In this case, the Gaussian distribution can be replaced with an appropriate circular distribution, e.g., the von Misses distribution [61]. However, such distributions significantly complicate the process of deriving efficient solutions for calculating QMI. Instead of this, the proposed QSMI directly replaces the Gaussian kernel, used for calculating the similarity between two images in the information potentials, with the cosine similarity:

$$S_{cos}(\mathbf{y}_1,\mathbf{y_2}) = \frac{1}{2}\left(\frac{\mathbf{y}_1^T\mathbf{y}_2}{\|\mathbf{y}_1\|_2\|\mathbf{y}_2\|_2} + 1\right), \tag{50}$$

where $\|\cdot\|_2$ is the l^2 norm of a vector. In that way, we maintain the computationally efficient QMI formulation and avoid the need for manually tuning the width parameter of the Gaussian kernel. It is worth noting that the proposed QSMI method is not mathematically equivalent

Table 13: NUS-WIDE Hashing Evaluation (the mAP for different hash code lengths is reported)

Method	8 bits	12 bits	24 bits	36 bits	48 bits
DSH	0.660	0.659	0.671	0.689	0.694
DPSH	0.735	0.748	0.759	0.758	0.755
Proposed	**0.746**	**0.753**	**0.766**	**0.764**	**0.763**

to replacing the Gaussian-based density estimation with cosine-based kernels. Instead, it is inspired by the QMI formulation (see Section 2.2.2), employing the convenient properties of Gaussian kernels to extract an efficient closed-form estimation, which is then substituted by a different kernel. Therefore, QSMI is defined as:

$$I_T^{cos}(\mathscr{Q},\mathscr{Y}) = V_{IN}^{cos}(\mathscr{Q},\mathscr{Y}) + V_{ALL}^{cos}(\mathscr{Q},\mathscr{Y}) - 2V_{BTW}^{cos}(\mathscr{Q},\mathscr{Y}), \tag{51}$$

where

$$V_{IN}^{cos}(\mathscr{Q},\mathscr{Y}) = \frac{1}{N^2}\sum_{k=1}^{M}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k} S_{cos}(\mathbf{y}_i^{(k)},\mathbf{y}_j^{(k)}), \tag{52}$$

$$V_{ALL}^{cos}(\mathscr{Q},\mathscr{Y}) = \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{N_k}{N}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i,\mathbf{y}_j), \tag{53}$$

and

$$V_{BTW}^{cos}(\mathscr{Q},\mathscr{Y}) = \frac{1}{N^2}\sum_{k=1}^{M}\left(\left(\frac{N_k}{N}\right)\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)},\mathbf{y}_j)\right). \tag{54}$$

Note that instead of directly optimizing the QSMI, we propose using a "square clamp" around the similarity matrix $\mathbf{S}$, smoothing the optimization surface, as we further explain in Appendix 7.6, while we also employ an adaptive way to adjust the calculations using the in-batch priors instead of relying on the statistics of the whole dataset.

2.6.3 Performance evaluation

The proposed method is extensively evaluated in Appendix 2.6, using both an ablation study and comparing it to other state-of-the-art methods. We provide a brief summary of the experimental evaluation hereafter. The proposed method was evaluated using the larger-scale NUS-WIDE dataset that contains 269,648 images that belong to 81 different concepts. Following [117], we used the images that belong to the 21 most frequent concepts, i.e., 195,834 images. Note that an image might belong to more than one concept, i.e., fulfill multiple information needs. Furthermore, instead of using a subsample of the training set, we used the whole training set (193,734 randomly sampled images) to learn the hash codes (all the methods were used in a batch setting), and a test set of 2,100 randomly sampled queries was employed to evaluate the methods. Since there are many differences in the evaluation protocol used by different papers for this dataset, we compared the proposed method to the DSH [122] and DPSH [117] methods using the same network and evaluation setup. The evaluation results are shown in Table 13. The proposed method outperforms the rest of the evaluated methods for any code length. The time required for training and performing inference using three different methods is reported in

Table 14: Training and inference time hashing evaluation (time per batch with 128 samples is reported)

Method	**Feed-forward**	**Back-propagation**
	(inference)	**(training)**
DSH	2.9 ms	0.7 ms
DPSH	4.1 ms	0.7ms
Proposed	3.4 ms	0.7ms

Table 15: ILSVRC Hashing Evaluation (20-way experiments, the mAP for different hash code lengths is reported)

Method	**16 bits**	**32 bits**	**48 bits**
DSH	0.922 ± 0.006	0.941 ± 0.004	0.945 ± 0.003
DPSH	0.782 ± 0.036	0.909 ± 0.049	0.939 ± 0.020
DCH	0.944 ± 0.012	0.911 ± 0.016	0.902 ± 0.005
Proposed	$\mathbf{0.945 \pm 0.004}$	$\mathbf{0.951 \pm 0.004}$	$\mathbf{0.953 \pm 0.003}$

Table 14. More specifically, the time needed for performing one weight update in the last layer, i.e., calculating the loss and back-propagating the gradients in the last layer, is 3.4 ms for the proposed method, 2.9 ms for the DSH method and 4.1 ms for the DPSH method. Note that the actual overhead of the proposed approach during training is small (less than 20% compared to DSH method), while there is no overhead during inference, since the same architecture is used (0.7 ms is required per batch for feed-forwarding through the hashing layer of the network).

The proposed method was also evaluated using the ILSVRC dataset. A DenseNet backbone pre-trained on the same dataset was used [84], while the same hashing architecture and experimental setup as the one used for the NUS-WIDE dataset were employed. Note that a 20-way retrieval setup, that was repeated 20 times, was used for the evaluation. The experimental results are reported in Table 15. Again, the proposed method outperforms the other evaluated method, i.e., DSH [122], DPSH [117] and DCH [24], leading to both larger mAP, as well as smaller standard deviation among the different evaluation runs.

2.7 Variational Neural Networks

2.7.1 Introduction, objectives and summary of state of the art

Uncertainty estimation in neural networks allows us to know when the model is certain in its predictions, and when it is not, it allows for making it possible to include additional resources or human effort into processing of critical tasks, when this is needed. The use of uncertainty estimation methods for robot perception can improve a robot's safety for itself and its surroundings, and it can be a good indicator for its operation in active perception tasks. In the following, we introduce a type of neural networks, called Variational Neural Networks, that can be used

for creating deep learning models with inherent ability to express such uncertainty.

Bayesian Neural Networks [128, 33] consider a distribution over their weights $p(w|a)$ and a distribution over its hyperparameters $p(a)$. Predictive distribution over an output y for a data point x can be obtained as

$$p(y|x) = \int\int p(y|x,w)p(w|a)p(a)\,da\,dw, \tag{55}$$

integrating over all possible hyperparameters and model weights.

Given a dataset $D = (X_t, Y_t)$, the distribution of weights can be derived from Bayes' theorem as

$$p(w|D,a) = \frac{p(Y_t|X_t,w)p(w|a)}{p(D)}, \tag{56}$$

and predictive distribution has the form

$$p(y|x,D) = \int\int p(y|x,w)p(w|D,a)p(a)\,da\,dw. \tag{57}$$

Classical neural networks can be viewed as Bayesian neural networks with $p(a) = \delta(a-\hat{a})$ and $p(w|D,a) = \delta(w-\hat{w}_a)$ [33], where $\hat{a}$ are selected model hyperparameters, $\hat{w_a}$ are parameters optimized by training the model, and $\delta(x)$ is a Dirac delta function which has values 0 everywhere except at $x = 0$ where it equals to 1. In this case, predictive distribution is expressed by

$$\begin{aligned} p(y|x,D) &= \int\int p(y|x,w)p(w|D,a)p(a)dadw, \\ &= \int\int p(y|x,w)\delta(w-\hat{w}_a)\delta(a-\hat{a})dadw, \\ &= p(y|x,\hat{w}_{\hat{a}}), \end{aligned} \tag{58}$$

which is a distribution dictated by the loss of the network.

Usually, hyperparameters are considered fixed at point $\hat{a}$ and weights are optimized either by maximum likelihood estimation (MLE)

$$\begin{aligned} w_{mle} &= \arg\max_w \left[log\, p(D|w,\hat{a})\right] \\ &= \arg\max_w \left[\sum_i log\, p(Y_i|X_i,w,\hat{a})\right], \end{aligned} \tag{59}$$

or by maximum a posteriori (MAP)

$$\begin{aligned} w_{map} &= \arg\max_w [log\, p(w|D,\hat{a})] \\ &= \arg\max_w [log\, p(D|w,\hat{a}) + log\, p(w)]. \end{aligned} \tag{60}$$

where $log\, p(w)$ is called a regularization term.

Posterior weights distribution $p(w|D,a)$ is often intractable and is replaced by $q_m(w|a)$, which is selected to be tractable and close to the original distribution. To determine how close a distribution q_m is to p, Kullback-Leibler (KL) divergence [103] is used.

$$KL(q_m||p) = \int q_m(w|a) log \frac{q_m(w|a)}{p(w|D,a)} dw, \tag{61}$$

2.7.2 Description of work performed so far

We define a Variational Layer (VL) that takes an input $\mathbf{x}$ as

$$\begin{aligned} \mathscr{VL}(\mathbf{x}) &= \alpha_{\mathscr{N}}(f(\mathbf{x})), \\ f(\mathbf{x}) &\sim \mathscr{N}(L_\mu(\mathbf{x}), diag[L_\sigma^2(\mathbf{x})]), \end{aligned} \tag{62}$$

where L is a regular neural network layer such as fully connected, convolutional or recurrent. L_μ and L_σ represent instances of the same layer with different values of parameters and activation functions α_μ, α_σ, respectively. Activation function $\alpha_{\mathscr{N}}$ can be used to apply nonlinearity to randomly sampled values $f(\mathbf{x})$. By selecting which of α_μ, α_σ, $\alpha_{\mathscr{N}}$ are set to identity functions and which are set to be real activation functions, such as the Rectified Linear Unit (ReLU), we can create networks that are described by different mathematical models.

Epistemic uncertainty Fully connected and convolutional layers can be described as the following operation:

$$L_\lambda(\mathbf{x}) = \alpha_\lambda(W_\lambda \mathbf{x} + b_\lambda), \tag{63}$$

where W_λ and b_λ, α_λ are weights, bias and activation function, respectively.

By selecting α_μ and α_σ to be identity functions, the above can be reformulated as

$$\begin{aligned} \mathscr{VL}(\mathbf{x}) &= \alpha_{\mathscr{N}}(f(\mathbf{x})), \\ f(\mathbf{x}) &\sim \mathscr{N}(L_\mu(\mathbf{x}), diag[L_\sigma^2(\mathbf{x})]), \\ &= \mathscr{N}(W_\mu \mathbf{x} + b_\mu, diag[(W_\sigma \mathbf{x} + b_\sigma)^2]). \end{aligned} \tag{64}$$

Using the reparametrization trick [95] we can write $f(\mathbf{x})$ as

$$\begin{aligned} \varepsilon &\sim \mathscr{N}(0, I), \\ f(\mathbf{x}) &= W_\mu \mathbf{x} + b_\mu + diag[(W_\sigma \mathbf{x} + b_\sigma)]\varepsilon, \\ &= (W_\mu + diag[W_\sigma]\varepsilon)\mathbf{x} + (b_\mu + diag[b_\sigma]\varepsilon), \\ &= W\mathbf{x} + b, \\ W &\sim \mathscr{N}(W_\mu, diag[W_\sigma^2]), \\ b &\sim \mathscr{N}(b_\mu, diag[b_\sigma^2]). \end{aligned} \tag{65}$$

This means that a Variational Layer without any inner activations (α_μ, α_σ) is identical to a regular layer with a distribution over weights, which is a Bayesian Neural Network. The resulting distribution over weights is Gaussian, which makes this a Bayes by Backprop (BBB) [15] network.

Aleatoric uncertainty We consider a Variational Layer with $L_\sigma(\mathbf{x}) = \sigma, \sigma \in \mathbb{R}$, which can be directly achieved by setting $W_\sigma = 0, b_\sigma = \sigma, \alpha_\sigma =$ identity. Following a similar approach as in (65), we can write $f(\mathbf{x})$ as

$$\begin{aligned} \varepsilon &\sim \mathscr{N}(0, I), \\ f(\mathbf{x}) &= \alpha_\mu(W_\mu \mathbf{x} + b_\mu) + \sigma\varepsilon, \\ &= L_\mu(\mathbf{x}) + \varepsilon_\sigma, \\ \varepsilon_\sigma &\sim \mathscr{N}(0, \sigma I). \end{aligned} \tag{66}$$

In this formulation, ε_σ represents aleatoric uncertainty.

Output uncertainty estimation We consider a neural network $F(x,w)$ with a parametric distribution over weights $q_m(w)$. We assume the choice of q in a form

$$q_m(w) = Q(m, p(z)), \tag{67}$$

where $p(z)$ is a non-parametric distribution and Q is a deterministic function, parametrized by m. For BBB models Q is defined as

$$Q(m = (\mu, \sigma), p(z)) = \mu + \sigma \mathcal{N}(0, I). \tag{68}$$

Defining an epistemic index $z \sim p(z)$ [152], we can formulate a deterministic neural network as

$$F_d(x, m, z) = F(x, Q(m, z)). \tag{69}$$

With this formulation, a predictive distribution (57) with fixed hyperparameters is defined as

$$\begin{aligned} p(y|x,D) &= \int p(y|x,w) q_m(w)\, dw = \int p(y|x,m,z) p(z)\, dz, \\ \mathbb{E}[y] &= \int y\, p(y|x,D)\, dy \approx \frac{1}{T} \sum_i^T F_d(x,m,z_i), \\ Var[y] &= \int (y - E[y])(y - E[y])^T p(y|x,D)\, dy, \\ &\approx \frac{1}{T} \sum_i^T (E[y] - F_d(x,m,z_i))(E[y] - F_d(x,m,z_i))^T. \end{aligned} \tag{70}$$

The expectation and the variance are computed using Monte Carlo integration, which can be viewed as an approximation of $p(z)$ with $\sum_{i=0}^{T} \frac{\delta(z-z_i)}{T}$,' $z_i \sim p(z)$, $i \in 1...T$ [231].

Variational Neural Networks can also be formulated as a deterministic function $F_d(x,w,z)$ with a variational index $z \sim p(z)$ which, in the case of identity inner activations, transforms into an epistemic index and in the case of constant L_σ, transforms into an aleatoric index. Otherwise, it is an index of a distribution that combines both aleatoric and epistemic uncertainties. Predictive distribution, expectation and variance are formulated similarly to (70).

2.7.3 Performance evaluation

We conducted experiments following the experimental protocol used by Epistemic Neural Networks [152] (ENN) to evaluate the quality of the model's uncertainty on artificially generated data. A Deep Gaussian Process is trained on the generated data and is used to generate ground-truth predictions of mean and variance of a particular point. For each method, ENN framework evaluates a set of predictions and finds mean and variance of the predicted outputs that are later compared with ground-truth by Kullback-Leibler divergence

$$KL(P||GT) = \log \frac{\sigma_p}{\sigma_{gt}} + \frac{\sigma_p^2 + (\mu_p - \mu_{gt})^2}{2\sigma_{gt}^2} - 1/2, \tag{71}$$

where μ_p, σ_p and μ_{gt}, σ_{gt} represent mean and variance of prediction and ground-truth, respectively. The estimated KL value represents the quality of a model's uncertainty. Lower values mean better predictions. Each experiment is repeated for different values of input dimensions $(10, 100, 1000)$, data ratio $(1, 10, 100)$ and noise scale $(0.01, 0.1, 1)$.

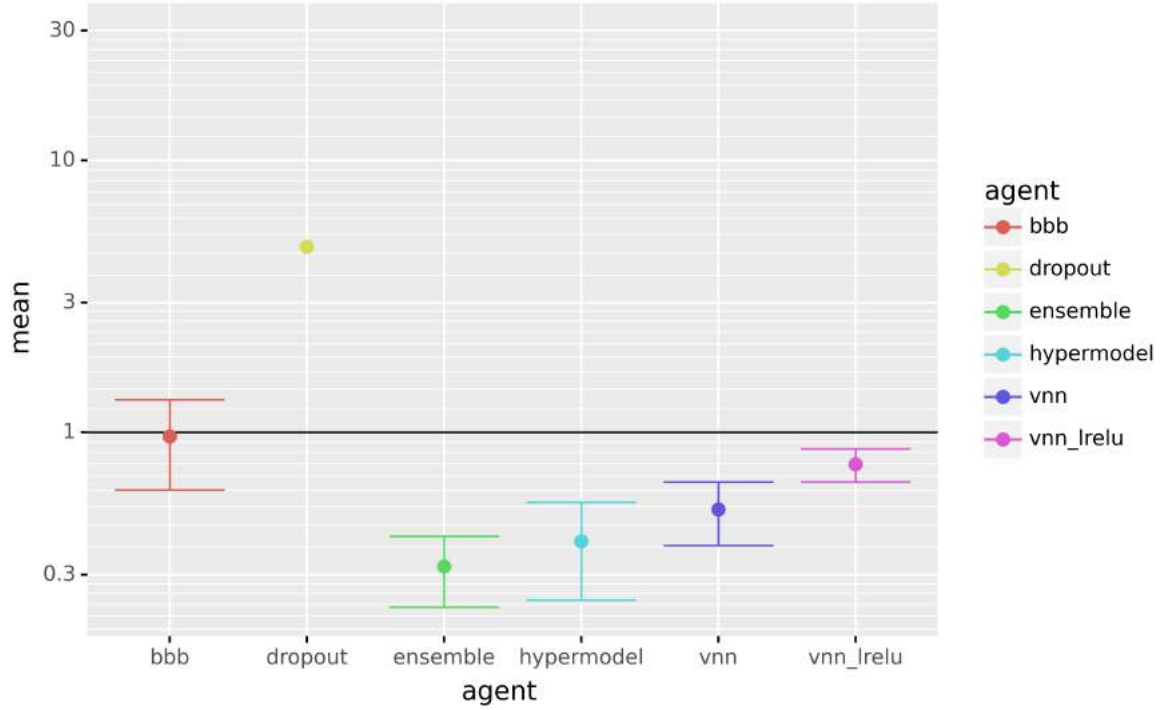


Figure 8: Estimated KL values for different ENN agents: Bayes by Backprop (bbb), Monte Carlo Dropout (dropout), Ensemble, Hypermodel, Best VNN model (vnn) and best VNN model with Leaky ReLU activation (vnn_lrelu)

According to the experimental results, the best VNN model uses no activations, and both mean and variance are fully trainable. The best model with activations leads to lower performance, but it is still better than Bayes by Backprop [14] and Monte Carlo Dropout [63] models, as can be seen from Fig. 8. Fig. 9 illustrates the mean and variance of KL values for different configurations of VNN models and fixed number of input dimensions of 100.

2.7.4 Future Work

We formulated Variational Neural Networks in general, and evaluated them in Epistemic Neural Networks benchmark along with adversarial attacks evaluation for classification tasks. Future work includes adoption of such networks for image classification, and/or other perception tasks, like 2D/3D object localization and tracking, with possible adversarial attacks for robustness evaluation.

2.8 Knowledge Distillation by Sparse Representation Matching

2.8.1 Introduction, objectives, and summary of state-of-the-arts

Over the past decade, deep neural networks have become the primary tools to tackle learning problems in several domains, ranging from machine vision [178, 176] and natural language processing [46, 174] to biomedical analysis [96, 216]. Of those important developments, Convolutional Neural Networks have evolved as a de facto choice for high-dimensional signals, either as a feature extraction block or the main workhorse in a learning system. Initially de-

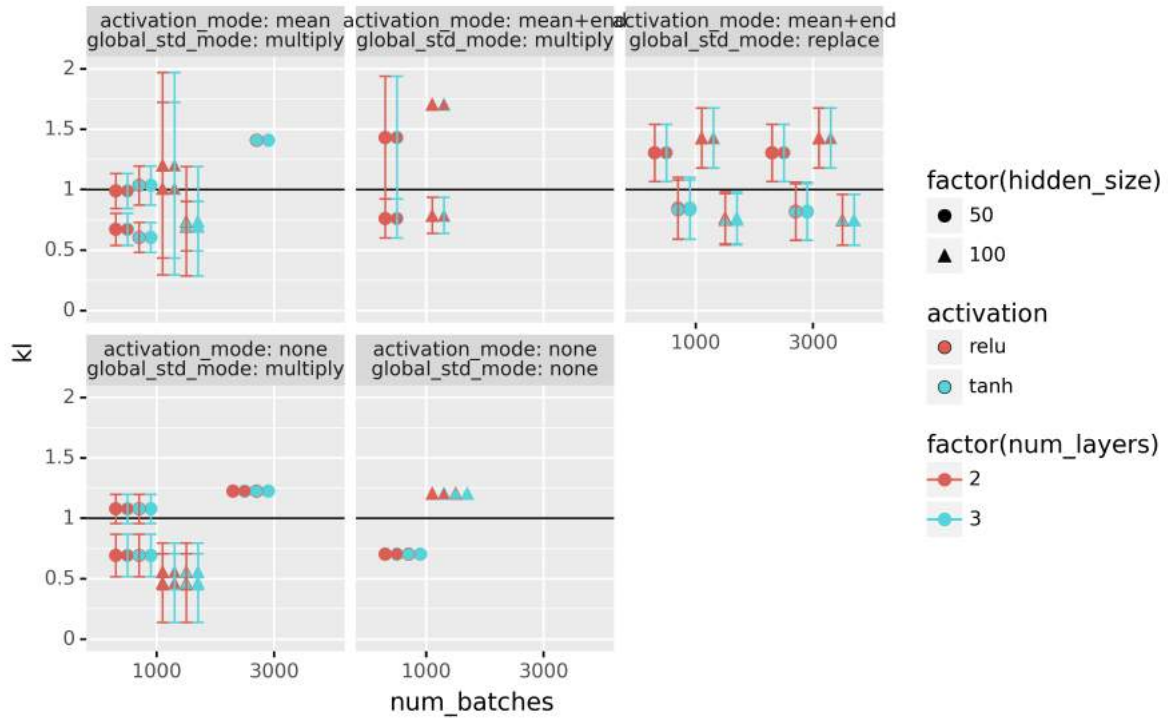


Figure 9: Estimated KL values and variances for different VNN configurations

veloped in the 1990s for handwritten character recognition using only two convolutional layers [111], state-of-the-art CNN topologies nowadays consist of hundreds of layers, having millions of parameters [84, 237]. In fact, not only in computer vision but also in other domains, state-of-the-art solutions are mainly driven by very large networks [46, 174], which limits their deployment in practice due to the high computational complexity.

The promising results obtained from maximally attainable computational power has encouraged a lot of research on developing smaller and light-weight models while achieving similar performances. This includes efforts on designing more efficient neural network families (both automatic and hand-crafted) [82, 213, 214, 263, 215], compressing pretrained networks through weight pruning [130, 218], quantization [85, 260], approximation [44], as well as transferring knowledge from one network to another via knowledge distillation [80]. Of these developments, Knowledge Distillation (KD) [80] is a simple and widely used technique that has been shown to be effective in improving the performance of a network, given the access to one or many pretrained networks. KD and its variants work by utilizing the knowledge acquired in one or many models (the teacher(s)) as supervisory signals to train another model (the student) along with the labeled data. Thus, there are two central questions in KD:

- How to represent the knowledge encoded in a teacher network?
- How to efficiently transfer such knowledge to other networks, especially when there are architectural differences between the teacher and the student networks?

In the original formulation [80], soft probabilities produced by the teacher represent its knowledge and the student network is trained to mimic this soft prediction. Besides the final predictions, other works have proposed to utilize intermediate feature maps of the teacher as

additional knowledge [180, 248, 64, 78, 158]. Intuitively, intermediate feature maps contain certain clues on how the input is progressively transformed through layers of a CNN, thus can act as a good source of knowledge. However, we argue that the intermediate feature maps by themselves are not a good representation of the knowledge encoded in the teacher to teach the students. To address the question of representing the knowledge of the teacher CNN, instead of directly utilizing the intermediate feature maps of the teacher as supervisory signals, we propose to encode each pixel (of the feature maps) in a sparse domain and use the sparse representation as the source of supervision.

Prior to the era of deep learning, sparse representations attracted a great amount of interest in computer vision community and formed the basis of many important works [257]. Sparse representation learning aims at representing the input signal in a domain where the coefficients are sparsest. This is achieved by using an overcomplete dictionary and decomposing the signal as a sparse linear combination of the atoms in the dictionary. While the dictionary can be pre-specified, it is often desirable to optimize the dictionary together with the sparse decomposition using example signals. Since hidden feature maps in CNN are often smooth with high correlations between neighboring pixels, they are compressible, e.g., in Fourier domain. Thus, sparse representation serves as a good choice for representing information encoded in the hidden feature maps.

Sparse representation learning is a well-established topic in which several algorithms have been proposed [257]. However, to the best of our knowledge, existing formulations are computationally intensive to fit a large amount of data. Although learning task-specific sparse representations have been proposed in prior works [129, 198, 142], we have not seen its utilization for knowledge transfer using deep neural networks and stochastic optimization. In this work, we formulate sparse representation learning as a computation block that can be incorporated into any CNN and be efficiently optimized using mini-batch update from stochastic gradient-descent based algorithms. Our formulation allows us to take advantage of not only modern stochastic optimization techniques but also data augmentation to generate target sparsity on-the-fly.

A summary of this work is provided hereafter. The corresponding technical report is listed below, and can be found in Appendix 7.7:

1. D. T. Tran, M. Gabbouj and A. Iosifidis, "*Knowledge Distillation by Sparse Representation Matching*", Technical Report (arXiv:2103.17012).

2.8.2 Description of the work performed so far

Knowledge Representation Given the n-th input image $\mathscr{X}_n$, let us denote by $\mathscr{T}_n^{(l)} \in \mathbb{R}^{H_l \times W_l \times C_l}$ the output of the l-th layer of the teacher CNN, with H_l and W_l being the spatial dimensions and C_l is the number of channels. In the following, we used the subscript $\mathscr{T}$ and $\mathscr{S}$ to denote a variable that is related to the teacher and student networks, respectively. In addition, we also denote by $\mathbf{t}_{n,i,j}^{(l)} = \mathscr{T}_n^{(l)}(i,j,:) \in \mathbb{R}^{C_l}$ the pixel at position (i,j) of $\mathscr{T}_n^{(l)}$. The first objective in Sparse Representation Matching (SRM) is to represent each pixel $\mathbf{t}_{n,i,j}^{(l)}$ in a sparse domain. To do so, SRM learns an overcomplete dictionary of M_l atoms: $\mathbf{D}_{\mathscr{T}}^{(l)} = [\mathbf{d}_{\mathscr{T},1}^{(l)}, \ldots, \mathbf{d}_{\mathscr{T},M_l}^{(l)}] \in \mathbb{R}^{C_l \times M_l}$ ($M_l > C_l$), which is used to express $\mathbf{t}_{n,i,j}^{(l)}$ as a linear combination of $\mathbf{d}_{\mathscr{T},m}^{(l)}$ as follows:

$$\mathbf{t}_{n,i,j}^{(l)} = \sum_{m=1}^{M_l} \psi_k(\mathbf{t}_{n,i,j}^{(l)}, \mathbf{d}_{\mathscr{T},m}^{(l)}) \cdot \kappa(\mathbf{t}_{n,i,j}^{(l)}, \mathbf{d}_{\mathscr{T},m}^{(l)}) \cdot \mathbf{d}_{\mathscr{T},m}^{(l)} \tag{72}$$

where

- $\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m})$ denotes a function that measures the similarity between $\mathbf{t}^{(l)}_{n,i,j}$ and atom $\mathbf{d}^{(l)}_{\mathscr{T},m}$. We further denote by $\mathbf{k}^{(l)}_{\mathscr{T},n,i,j} = [\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},1}), \ldots, \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},M_l})]$ the vector that contains similarities between $\mathbf{t}^{(l)}_{n,i,j}$ and all atoms in the dictionary $\mathbf{D}^{(l)}_{\mathscr{T}}$.

- $\psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{m})$ denotes the indicator function that returns a value of 1 if $\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m})$ belongs to the set of top-k values in $\mathbf{k}^{(l)}_{\mathscr{T},n,i,j}$, and a value of 0 otherwise.

The decomposition in Eq. (72) basically means that $\mathbf{t}^{(l)}_{n,i,j}$ is expressed as the linear combination of k most similar atoms in $\mathbf{D}^{(l)}_{\mathscr{T}}$, with the coefficients being the corresponding similarity values. Let $\lambda^{(l)}_{n,i,j,m} = \psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m}) \cdot \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m})$, then the sparse representation of $\mathbf{t}^{(l)}_{n,i,j}$ is then defined as:

$$\tilde{\mathbf{t}}^{(l)}_{n,i,j} = [\lambda^{(l)}_{n,i,j,1}, \ldots, \lambda^{(l)}_{n,i,j,M_l}] \in \mathbb{R}^{M_l} \tag{73}$$

By construction, there are only k non-zero elements in $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, and k defines the degree of sparsity, which is a hyper-parameter of SRM. In order to find $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, we simply minimize the reconstruction error in Eq. (72) as follows:

$$\underset{\mathbf{D}^{(l)}_{\mathscr{T}}}{\operatorname{argmin}} \sum_{n,i,j} \|\mathbf{t}^{(l)}_{n,i,j} - \sum_{m=1}^{M_l} \psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m}) \cdot \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathscr{T},m}) \cdot \mathbf{d}^{(l)}_{\mathscr{T},m}\|_2^2 \tag{74}$$

There are many choices for the similarity function κ such as linear kernel, RBF kernel, sigmoid kernel and so on. In our work, we used the sigmoid kernel $\kappa(\mathbf{x}, \mathbf{y}) = sigmoid(\mathbf{x}^T\mathbf{y} + c)$ since the dot-product makes it computationally efficient and the gradients in the backward pass are stable. Although the RBF kernel is popular in many works, we empirically found that RBF kernel is sensitive to the learning rate, which easily leads to numerical issues.

Knowledge Transfer Let us denote by $\mathscr{S}^{(p)}_n \in \mathbb{R}^{H_p \times W_p \times C_p}$ the output of the p-th layer of the student network given input image is $\mathscr{X}_n$. In addition, $\mathbf{s}^{(p)}_{n,i,j} \in \mathbb{R}^{C_p}$ denotes the pixel at position (i, j) of $\mathscr{S}^{(p)}_n$. We consider the task of transferring knowledge from the l-th layer of the teacher to the p-th layer of the student. To do so, we require that the spatial dimensions of both networks match ($H_p = H_l$ and $W_p = W_l$) while the channel dimensions might differ.

Given the sparse representation of the teacher in Eq. (73), a straightforward way is to train the student network to produce hidden features at spatial position (i, j), having the same sparse coefficients as its teacher. However, trying to learn exact sparse representations as produced by the teacher is a too restrictive task since this enforces learning the absolute value of every point in a high-dimensional space. Instead of enforcing an absolute constraint on how each pixel of every sample should be represented, to transfer knowledge, we only enforce a relative constraints between them in the sparse domain. Specifically, we propose to train the student to only approximate sparse structures of the teacher network by solving a classification problem with two types of labels extracted from the sparse representation $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ of the teacher: pixel-level and image-level labels.

Pixel-level labeling: for each spatial position (i, j), we assign a class label, which is the index of the largest element of $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, i.e., the index of the closest (most similar) atom in $\mathbf{D}^{(l)}_{\mathscr{T}}$. This basically means that we partition all pixels into M_l disjoint sets using dictionary $\mathbf{D}^{(l)}_{\mathscr{T}}$, and the student network is trained to learn the same partitioning using its own dictionary $\mathbf{D}^{(p)}_{\mathscr{S}} = [\mathbf{d}^{(p)}_{\mathscr{S},1}, \ldots, \mathbf{d}^{(p)}_{\mathscr{S},M_l}] \in \mathbb{R}^{C_p \times M_l}$. Let $\mathbf{k}^{(p)}_{\mathscr{S},n,i,j} = [\kappa(\mathbf{s}^{(p)}_{n,i,j}, \mathbf{d}^{(p)}_{\mathscr{S},1}), \ldots, \kappa(\mathbf{s}^{(p)}_{n,i,j}, \mathbf{d}^{(p)}_{\mathscr{S},M_l})]$ denote the vector that contains similarities between pixel $\mathbf{s}^{(p)}_{n,i,j}$ and M_l atoms in $\mathbf{D}^{(p)}_{\mathscr{S}}$. The first knowledge transfer objective in our method using pixel-level label is defined as follows:

$$\underset{\Theta_{\mathscr{S}}, \mathbf{D}^{(p)}_{\mathscr{S}}}{\operatorname{argmin}} \sum_{n,i,j} \mathscr{L}_{CE}(c_{n,i,j}, \mathbf{k}^{(p)}_{\mathscr{S},n,i,j}) \tag{75}$$

where $\Theta_{\mathscr{S}}$ denotes parameters of the student network. $\mathscr{L}_{CE}$ denotes the cross-entropy loss function, and $c_{n,i,j} = \arg\max(\tilde{\mathbf{t}}^{(l)}_{n,i,j})$. Here we should note that the idea of transferring the structure instead of the absolute representation is not new. For example, in [155] and [208], the authors proposed to transfer the relative distance between the embeddings of samples. In our case, the pixel-level labels provide supervisory information on how the pixels in the student network should be represented in the sparse domain so that their partition using the nearest atom is the same.

Image-level labeling: given the sparse representation $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ of the teacher, we generate an image-level label by averaging $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ over the spatial dimensions. While pixel-level labels provide local supervisory information encoding the spatial information, image-level labels provide global supervisory information, promoting the shift-invariance property. Image-level labeling bears some resemblances to the Bag-of-Feature model [157], which aggregates the histograms of image patches to generate an image-level feature. The second knowledge transfer objective in our method using image-level labels is defined as follows:

$$\underset{\Theta_{\mathscr{S}}, \mathbf{D}^{(p)}_{\mathscr{S}}}{\operatorname{argmin}} \sum_{n} \mathscr{L}_{BCE}\left(\frac{\sum_{i,j} \tilde{\mathbf{t}}^{(l)}_{n,i,j}}{H_l \cdot W_l}, \frac{\sum_{i,j} \mathbf{k}^{(p)}_{\mathscr{S},n,i,j}}{H_l \cdot W_l}\right) \tag{76}$$

where $\mathscr{L}_{BCE}$ denotes the binary cross-entropy loss. Here we should note that since most kernel functions output a similarity score in $[0, 1]$, elements of $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ and $\mathbf{k}^{(p)}_{\mathscr{S},n,i,j}$ are also in this range, making the two inputs to $\mathscr{L}_{BCE}$ in Eq. (76) valid.

To summarize, the procedure of our SRM method is similar to FitNet [180], which consists of the following steps:

- **Step 1**: Given the source layers (with indices l) in the teacher network $\mathscr{T}$, find the sparse representation by solving Eq. (74).
- **Step 2**: Given the target layers (with indices p), optimize the student network $\mathscr{S}$ to predict pixel-level and image-level labels by solving Eq. (75), (76).
- **Step 3**: Given the student network obtained in Step 2, optimize it using the original KD algorithm.

All optimization objectives in our algorithm are solved by stochastic gradient descent.

2.8.3 Performance Evaluation

CIFAR100 Experiments The first set of experiments was conducted on CIFAR100 dataset [100] to compare our SRM method with other KD methods: KD [80], FitNet [180], AT [248], PKT [158], RKD [155] and CRD [208]. The results are shown in Table 16. It is clear that the proposed distillation method outperformed other competing methods.

Table 16: Performance on CIFAR100

Model	Test Accuracy (%)
DenseNet121 (teacher)	75.09 ± 00.29
AllCNN	67.64 ± 01.87
AllCNN-KD	73.27 ± 00.20
AllCNN-FitNet	72.03 ± 00.27
AllCNN-AT	70.88 ± 0.29
AllCNN-PKT	72.22 ± 0.35
AllCNN-RKD	70.39 ± 0.17
AllCNN-CRD	72.70 ± 0.18
AllCNN-SRM (our)	$\mathbf{74.73} \pm 00.26$

Transfer Learning Experiments Since transfer learning is key in the success of many applications that utilize deep neural networks, we conducted experiments in 5 transfer learning problems (Flowers [150], CUB [227], Cars [99], Indoor-Scenes [173] and PubFig83 [168]) to assess how well the proposed method works under transfer learning setting compared to others.

In our experiments, we used a pretrained ResNext50 [237] on ILSVRC2012 database as the teacher network, which is finetuned using the training set of each transfer learning task. We then benchmarked how well each knowledge distillation method transfers both pretrained knowledge and domain specific knowledge to a *randomly initialized* student network using domain specific data. Both residual (ResNet18 [76], 11.3M parameters, 1.82G FLOPs) and non-residual (a variant of AllCNN [199], 5.1M parameters, 1.35G FLOPs) architectures were used as the student.

Full transfer setting: in this setting, we used all samples available in the training set to perform knowledge transfer from the teacher to the student. The test accuracy achieved by different methods is shown in Table 17. It is clear that the proposed SRM outperforms other methods on many datasets, except on Cars and Indoor-Scenes datasets for AllCNN student. While KD, AT, PKT, CRD and SRM successfully led the students to better minima with both residual or non-residual students, FitNet was only effective with the residual one. The results suggest that the proposed intermediate knowledge transfer mechanism in SRM is robust to architectural differences between the teacher and the student networks.

Few-shot transfer setting: we further assessed how well the methods perform when there is a limited amount of data for knowledge transfer. For each dataset, we randomly selected 5 samples (5-shot) and 10 samples (10-shot) from the training set for training, and kept the validation and test set similar to the full transfer setting. Since the Flowers dataset has only 10 training samples in total (the original split provided by the database has 20 samples per class, however, we used 10 samples for validation purpose), the results for 10-shot are similar to full transfer learning setting. The test performance (% in accuracy) under 5-shot and 10-shot transfer learning settings are reported in Table 18 and Table 19, respectively. Under this

Table 17: Full transfer learning using AllCNN (ACNN) and ResNet18 (RN18) (test accuracy %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
ACNN	40.80 ± 02.33	47.26 ± 00.18	61.93 ± 01.38	35.82 ± 00.43	78.47 ± 00.17
ACNN-KD	46.14 ± 00.39	51.80 ± 00.41	66.12 ± 00.17	38.44 ± 00.99	81.54 ± 00.09
ACNN-FitNet	30.10 ± 02.41	44.30 ± 01.25	60.20 ± 03.83	30.87 ± 00.35	77.61 ± 00.87
ACNN-AT	51.62 ± 00.69	51.74 ± 00.39	**73.89** ± 00.06	**43.56** ± 00.52	81.11 ± 01.51
ACNN-PKT	47.12 ± 00.52	47.60 ± 00.85	70.16 ± 00.51	37.71 ± 00.64	82.03 ± 00.26
ACNN-RKD	42.00 ± 01.10	39.99 ± 00.61	56.99 ± 02.44	30.94 ± 00.45	75.44 ± 00.50
ACNN-CRD	46.99 ± 01.13	51.12 ± 00.34	68.89 ± 00.47	42.82 ± 00.21	**83.02** ± 00.11
ACNN-SRM	**51.72** ± 00.58	**54.51** ± 01.72	71.44 ± 04.76	43.09 ± 00.60	82.89 ± 01.78
RN18	44.25 ± 00.42	44.79 ± 00.68	57.17 ± 01.95	36.72 ± 00.29	79.08 ± 00.36
RN18-KD	48.26 ± 00.33	54.91 ± 00.33	75.29 ± 00.46	43.84 ± 00.67	84.49 ± 00.28
RN18-FitNet	48.29 ± 01.24	61.28 ± 00.48	**85.01** ± 00.10	45.93 ± 01.00	89.78 ± 00.20
RN18-AT	51.49 ± 00.42	53.13 ± 00.40	77.14 ± 00.15	44.13 ± 00.55	83.60 ± 00.19
RN18-PKT	45.32 ± 00.51	45.24 ± 00.39	71.24 ± 02.23	37.27 ± 00.79	82.00 ± 00.10
RN18-RKD	42.32 ± 00.28	36.29 ± 00.58	56.87 ± 02.64	29.57 ± 00.90	70.90 ± 01.79
RN18-CRD	47.67 ± 00.05	53.25 ± 00.83	76.51 ± 00.74	43.76 ± 00.40	84.43 ± 00.40
RN18-SRM	**67.46** ± 01.06	**63.11** ± 00.45	84.40 ± 00.67	**54.59** ± 00.38	**89.79** ± 00.53

Table 18: 5-shot transfer learning using AllCNN (ACNN) and ResNet18 (RN18) (test accuracy %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
ACNN	32.91 ± 00.94	13.19 ± 00.46	05.03 ± 00.11	09.21 ± 00.81	05.15 ± 00.45
ACNN-KD	35.98 ± 00.76	21.60 ± 00.83	10.61 ± 00.52	14.81 ± 00.67	11.97 ± 01.40
ACNN-FitNet	28.73 ± 01.18	14.78 ± 00.60	06.11 ± 00.37	08.36 ± 00.52	08.25 ± 01.36
ACNN-AT	38.21 ± 00.88	17.69 ± 00.94	08.52 ± 01.50	08.76 ± 00.39	08.02 ± 00.70
ACNN-PKT	33.25 ± 00.31	11.30 ± 00.20	06.16 ± 00.29	10.75 ± 01.04	06.13 ± 00.18
ACNN-RKD	30.27 ± 00.27	09.55 ± 00.34	04.82 ± 00.31	09.68 ± 00.39	04.82 ± 00.16
ACNN-CRD	35.01 ± 00.57	18.09 ± 00.91	06.77 ± 00.55	09.73 ± 00.49	06.33 ± 00.28
ACNN-SRM	**41.14 ± 01.09**	**22.89 ± 01.18**	**11.71 ± 01.21**	**16.63 ± 00.34**	**13.96 ± 00.52**
RN18	32.95 ± 00.37	11.55 ± 00.10	05.00 ± 00.27	09.04 ± 00.44	04.98 ± 00.22
RN18-KD	38.07 ± 01.47	25.53 ± 00.35	11.37 ± 00.58	14.61 ± 00.88	10.69 ± 00.99
RN18-FitNet	39.17 ± 00.62	26.50 ± 00.46	12.36 ± 01.00	13.72 ± 01.07	11.49 ± 00.66
RN18-AT	37.36 ± 01.23	18.22 ± 00.47	08.96 ± 00.92	09.93 ± 00.73	08.76 ± 00.34
RN18-PKT	33.24 ± 00.47	10.63 ± 00.18	05.88 ± 00.18	11.03 ± 00.82	05.34 ± 00.11
RN18-RKD	29.97 ± 00.55	08.83 ± 00.40	04.98 ± 00.13	08.41 ± 00.49	04.49 ± 00.24
RN18-CRD	34.24 ± 00.32	18.36 ± 02.09	06.95 ± 00.26	09.01 ± 00.23	06.36 ± 00.21
RN18-SRM	**51.24 ± 00.48**	**34.67 ± 00.63**	**26.63 ± 01.82**	**21.18 ± 01.29**	**29.31 ± 00.51**

restrictive regime, the proposed SRM method performs far better than other tested methods for both types of students, largely improving the baseline results.

ImageNet Experiments For ImageNet [181] experiments, we followed an experimental setup similar to that in [39, 248]: ResNet34 and ResNet18 were used as the teacher and student networks, respectively.

Table 20 shows top-1 classification errors of SRM and related methods, including KD [80], AT [248], Seq. KD [240], KD+ONE [262], ESKD [39], ESKD+AT [39], CRD [208], having the same experimental setup. The teacher network (Resnet34) achieves top-1 error of 26.70%.

Table 19: 10-shot transfer learning using AllCNN (ACNN) and ResNet18 (RN18) (test accuracy %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
10-Shot					
ACNN	40.80 ± 02.33	25.53 ± 01.07	09.50 ± 00.82	16.23 ± 00.46	10.09 ± 00.19
ACNN-KD	46.14 ± 00.39	34.63 ± 00.35	23.43 ± 01.47	21.76 ± 00.53	28.11 ± 00.50
ACNN-FitNet	30.10 ± 02.41	29.40 ± 00.63	15.81 ± 02.16	15.71 ± 01.32	17.89 ± 00.67
ACNN-AT	51.62 ± 00.69	30.26 ± 00.35	25.22 ± 02.90	17.90 ± 00.40	26.27 ± 01.76
ACNN-PKT	47.12 ± 00.53	24.93 ± 01.92	13.82 ± 01.18	16.78 ± 00.59	10.67 ± 00.26
ACNN-RKD	42.00 ± 01.10	19.36 ± 00.50	09.60 ± 00.25	12.40 ± 00.76	06.99 ± 00.57
ACNN-CRD	46.99 ± 01.13	29.72 ± 00.25	16.96 ± 00.77	17.60 ± 00.85	17.24 ± 02.98
ACNN-SRM	**51.72 ± 00.58**	**35.86 ± 01.39**	**36.28 ± 04.33**	**24.82 ± 00.47**	**31.47 ± 01.84**
RN18	44.25 ± 00.42	22.72 ± 00.91	11.76 ± 01.35	15.16 ± 00.54	08.92 ± 00.73
RN18-KD	48.26 ± 00.33	40.57 ± 00.53	35.44 ± 01.13	23.85 ± 00.78	29.67 ± 00.91
RN18-FitNet	48.29 ± 01.24	43.83 ± 00.47	51.88 ± 01.72	24.62 ± 00.47	36.79 ± 01.27
RN18-AT	51.49 ± 00.42	30.47 ± 00.55	27.70 ± 02.72	17.48 ± 00.38	29.80 ± 01.41
RN18-PKT	45.32 ± 00.52	20.62 ± 02.56	11.00 ± 00.63	15.76 ± 00.62	08.80 ± 00.54
RN18-RKD	42.32 ± 00.28	17.73 ± 00.14	08.73 ± 00.71	11.82 ± 00.34	06.49 ± 00.27
RN18-CRD	47.67 ± 00.05	30.04 ± 00.13	17.11 ± 00.66	17.82 ± 00.49	16.08 ± 01.67
RN18-SRM	**67.46 ± 01.06**	**48.42 ± 01.86**	**61.22 ± 06.84**	**32.21 ± 01.21**	**51.99 ± 02.95**

Table 20: Top-1 Error of ResNet18 on ImageNet. (*) indicates results obtained by 110 epochs

KD	AT	Seq. KD	KD+ONE	ESKD	ESKD+AT	CRD*	SRM (our)
30.79	29.30	29.60	29.45	29.16	28.84	28.62*	28.79

Using SRM, we can successfully train ResNet18 to achieve 28.79% classification error, which is lower than existing methods (except CRD, which was trained for 110 epochs), including the recently proposed ESKD+AT [39] that combines the early-stopping trick and Attention Transfer [248].

2.9 Compactness in Deep Neural Networks

2.9.1 Introduction, objectives and summary of state of the art

Deep learning has achieved remarkable success in a variety of tasks over the recent years. Training of deep neural networks can be formulated as a two-stage process, where the first stage corresponds to feature learning, and the seconds stage denotes task-specific output prediction. In classification tasks, the prediction is generally performed by applying a linear dense layer on top of the learned feature representation and followed by a softmax activation to transform the predictions into a class probability distribution. Hence, a prediction in such a model can be seen as a linear classification in the learned feature space. Classical statistical intuition states that a good set of features should be class-wise invariant [1, 241]. In other words, it is expected that the model learns to minimize the distances between the sample representations of each class in the feature space in order to learn compact class manifolds.

We investigate this phenomenon for deep neural networks. We first analyze the relationship between the features' class-wise variance and the performance of the model on a variety of architectures and datasets. We show that in the context of deep neural networks, a more class-wise

invariant feature space, i.e., one with lower relative variance, does not necessarily imply better performance. Then, we analyze the class-wise variance of the learned features throughout the training and discover that over-parameterized neural networks, surprisingly, do not necessarily implicitly minimize it. On the contrary, models typically converge to regimes with higher variances.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.8.

- [40] Chumachenko, K., Laakom, F., Raitoharju, J., Iosifidis, A., and Gabbouj, M, "*Rethinking compactness in deep neural networks*", ICML Workshops: Overparameterization, Pitfalls and Opportunities 2021.

2.9.2 Description of work performed so far

To analyze the relationship between the class compactness and model performance, we quantify it using the class-wise variance in the feature space. Formally, the class-wise variance can be defined as follows:

Let A be a set of data samples defined as $A = \{(X_1, y_1), \cdots, (X_N, y_N)\} \in \mathscr{X} \times \mathbb{R}$, where $y_i \in \{1, \cdots, c\}$ is the corresponding class label of X_i. Let $\{\phi(X_1), \cdots, \phi(X_N)\}$ be the feature representation of the samples in A. Then, the class-wise variance of the feature space is defined as follows:

$$\mathbf{v}_c = \frac{1}{N} \sum_i ||\phi(X_i) - \mu_i||_2^2 \tag{77}$$

where μ_i is the mean of the feature representations of the data belonging to class y_i, i.e., $\mu_i = \frac{1}{\sum_j \delta(y_j == y_i)} \sum_j \phi(X_j) \delta(y_j == y_i)$.

As can be seen, the class-wise variance measures the average distance of the feature representations of the samples to the corresponding class mean. Intuitively, this quantifies the compactness of the features learned by a model. Lower class-wise variance implies that the model has learned an invariant representation of each class, where samples with the same labels are mapped close to each other, whereas higher class-wise variance implies that the learned manifolds of the classes are non-compact and outspread in the feature space. Additionally, we define a normalized variant of the class-wise variance, where it is scaled using the global variance in order to quantify the relative volume of the classes compared to the total volume in the feature space. Formally, the normalized class-wise variance can be defined as follows:

Let A be a set of data samples defined as $A = \{(X_1, y_1), \cdots, (X_N, y_N)\} \in \mathscr{X} \times \mathbb{R}$, where $y_i \in \{1, \cdots, c\}$ is the corresponding class label of X_i. Let $\{\phi(X_1), \cdots, \phi(X_N)\}$ be the feature representation of the samples in A. Then, the normalized class-wise variance of the feature space is defined as follows

$$\bar{\mathbf{v}}_c = \frac{\frac{1}{N} \sum_i ||\phi(X_i) - \mu_i||_2^2}{\frac{1}{N} \sum_i ||\phi(X_i) - \mu||_2^2}, \tag{78}$$

where $\mu = \frac{1}{N} \sum_j \phi(X_j)$ is the global mean of the data in the feature space.

The intuition that compactness is a desired property has been utilized in various classical machine learning methods in the past using the Fisher ratio [241, 87], which is closely related to the normalized class-wise variance. We investigate the link between the (normalized) class-wise variance and the performance of over-parameterized neural network models. Surprisingly, we find that the compactness intuition does not hold in this case and that lower class-wise variance values do not necessarily result in better accuracy. We perform experimental evaluation

Table 21: Average test accuracy, class-wise variance, and normalized class-wise variance of validation set on different models trained on MNIST.

	Accuracy	$\mathbf{v}_c$	$\bar{\mathbf{v}}_c$
Large CNN	98.59	47659	0.58
Dense-1	94.88	15.01	0.44
Dense-2	96.14	13.43	0.32
Dense-3	96.63	15.94	0.26
Dense-4	96.73	20.43	0.23
CNN-1	96.30	336.91	0.65
CNN-2	98.16	1117.68	0.59
CNN-3	98.13	347.39	0.50

Table 22: Average test accuracy, class-wise variance, and normalized class-wise variance of validation set on different models trained on CIFAR10.

	Accuracy	$\mathbf{v}_c$	$\bar{\mathbf{v}}_c$
ResNet18	91.29	7.76	0.46
ResNet50	92.12	7.37	0.40
Dense-1	52.80	86.88	0.84
Dense-2	53.04	48.37	0.76
Dense-3	52.90	36.45	0.70
Dense-4	51.74	29.66	0.68
CNN-1	66.36	292.41	0.89
CNN-2	74.53	222.07	0.89
CNN-3	75.68	290.47	0.76

of several deep neural network models using MNIST and CIFAR10 datasets on different deep architectures of different depths and structures, including multilayer perceptrons with solely fully-connected layers and convolutional networks with VGG-like structure, where exact network structure can be found from the corresponding paper [40]. Additionally, on CIFAR dataset we quantify the compactness of ResNet18 and ResNet50 architectures. On both datasets, for each model, we report the test accuracy along with the class-wise variance and the normalized class-wise variance of the feature space prior to the output layer over the validation set. We report the average scores across three training iterations.

As can be seen in Table 21, better model performance is not necessarily associated neither with lower class-wise variance nor with normalized class-wise variance. For example, CNN-2 outperforms CNN-3, while having lower compactness (higher $\mathbf{v}_c$ and $\bar{\mathbf{v}}_c$). At the same time, all the Dense models have lower normalized class-wise variances, hence, higher class compactness compared to any CNN model, while achieving lower accuracy.

Similar results can be observed in models trained on CIFAR10, as shown in Table 22, where Dense-4 and Dense-3 models have lower class-wise variance compared to Dense-2 model despite having worse performance. Similarly, CNN-2 outperforms CNN-1, although its class-wise variance is lower. At the same time, their normalized class-wise variances are equal despite the large gap in the test accuracy. To better understand this phenomenon in the over-parameterized

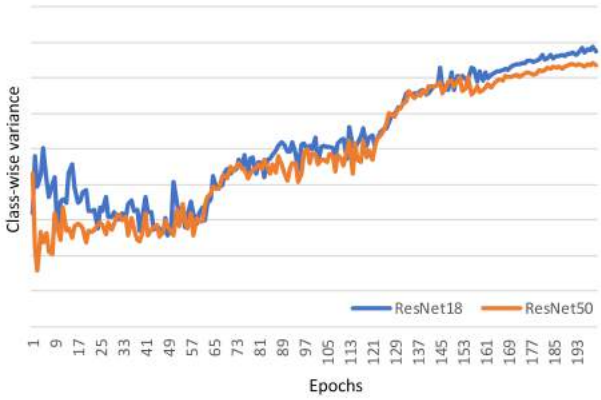


Figure 10: Class-wise variance evolution during training.

regime, we further investigate the evolution of the class-wise variance $\mathbf{v}_c$ and its normalized counterpart $\bar{\mathbf{v}}_c$ throughout the training in several different neural network architectures using CIFAR10 and MNIST. We find that generally all the models do not minimize the class-wise variance.On the contrary, $\mathbf{v}_c$ is increasing during the training. One example is given in Fig. 10 that shows the evolution of class-wise variance throughout the training in ResNet18 and ResNet50 trained on CIFAR10 dataset. Further examples can be found in the corresponding paper [40].

2.9.3 Future Work

To this point, we investigated the link between class-wise feature variance and classification performance. We studied the relationship between class-wise compactness of the learned features and the corresponding model performance in the deep neural network context and identified certain properties of this relationship. Further work is targeted at utilizing the discovered properties towards improving performance of deep learning models based on compactness in classification tasks.

3 2D/3D object localization and tracking

3.1 Spherical images for 3D object detection

3.1.1 Introduction, objectives and summary of state of the art

Previously described (D4.1) and integrated into OpenDR voxel-based 3D object detection methods [110, 125] use voxel/pillar representation of the point cloud and create a 2D pseudo-image that is processed by 2D convolutional neural networks to generate final predictions. Instead of voxelization, projection-based methods that create such 2D images by structuring points directly can be used. Projection-based methods avoid structural transformations and only try to represent a set of points on a plane (or a set of planes) based on a particular projection rule. This mitigates the structural information loss, compared to voxel-based methods.

Projection-based methods try to use different types of point-cloud projections to create structured data that can be processed by standard 2D CNNs. MVCNN [200] and GSL + AMTC [77] use multiple plane projections to create separate features and combine them to perform 3D shape recognition. VeloFCN [116] uses cylindrical projection that creates a 2D image with size

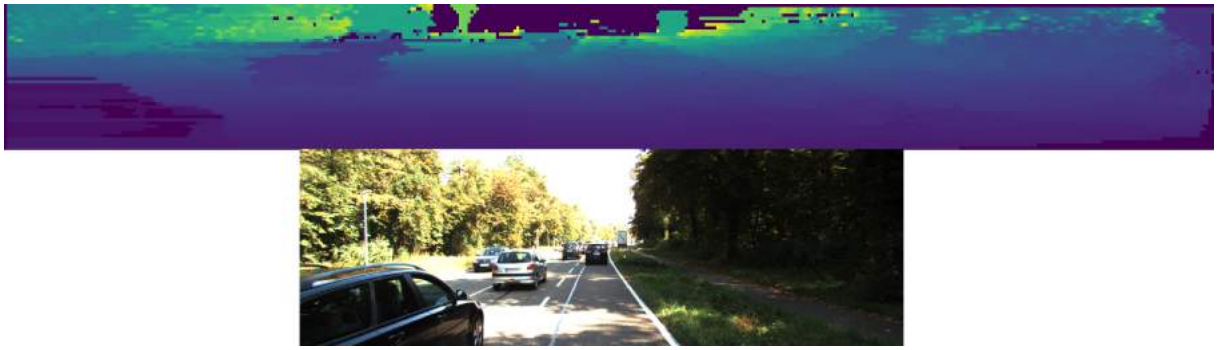


Figure 11: An example of a spherical projection depth map calculated using a Lidar point cloud from the KITTI dataset (top), with the corresponding camera image (bottom). Each pixel represents a point in the 3D scene. Lighter colors correspond to points at a higher distance from the sensor, while darker colors correspond to points at a lower distance. Absence of a point is indicated by the darkest color and can be found in the top-center part of the projection image.

that depends on the properties of the Lidar sensor. Given the point (x,y,z) and the average horizontal and vertical angle differences $\Delta\theta$ and $\Delta\phi$, the projected position (r,c) can be obtained by:

$$\begin{aligned} \theta &= atan2(y,x), \\ \phi &= arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right), \\ r &= \lfloor \theta/\Delta\theta \rfloor, \\ c &= \lfloor \phi/\Delta\phi \rfloor. \end{aligned} \tag{79}$$

The resulting projection is a 2D image that is processed by a Fully Convolutional Network (FCN) [191] which generates point-wise box and class predictions. SqueezeSeg [233] uses a slightly different spherical projection:

$$\begin{aligned} \theta &= arcsin\left(\frac{z}{\sqrt{x^2+y^2+z^2}}\right), \\ \phi &= arcsin\left(\frac{y}{\sqrt{x^2+y^2}}\right), \\ r &= \lfloor \theta/\Delta\theta \rfloor, \\ c &= \lfloor \phi/\Delta\phi \rfloor, \end{aligned} \tag{80}$$

and the resulting 2D image is processed by a CNN to produce a point cloud segmentation output. An example 2D image created by using the spherical projection in Eq. (80) can be seen in Figure 11.

3.1.2 Description of work performed so far

We use spherical projection to create a 2D pseudo image of an input Lidar, called Spherical Pseudo Image (SPI), and process it by a 2D CNN to create object predictions. We follow the approach used in FairMOT [254], i.e. heatmap-based object detection.

Figure 12: Example of a heatmap generated by an SPI model. Red color represents the depth of a point, green color represents a target heatmap, and blue color represents the predicted heatmap. The left-most target heatmap is fragmented due to the absence of points in the center of a car, seated on the edge of the camera-visible range.

FairMOT creates a 2D heatmap for each class independently by placing Gaussian-distributed values at the center of an object, leading to high values directly at the center of the object and lower values at locations close to it. This leads to a heatmap that has mostly zero values for an uncrowded image. In order to adapt this approach to the 3D case, we project the center and the edges of the target box to spherical coordinates and define the size of a Gaussian distribution based on the projected 2D box size, which will be placed at the projected center. A limitation of this approach is that on a small number of objects that are directly to the side of an ego object (the object on which the sensor is placed) and are mostly out of the camera field of view, this approach leads to a fragmented heatmap area, leading to centers of such objects having no points on them and therefore having no chance to place a heatmap on them. An example of such case can be seen in Fig. 12. This problem occurs on the KITTI dataset for which a frontal field of view is used to create the point cloud annotations, while it does not occur on the 360-degrees annotated point cloud used by NuScenes [22] dataset.

For the 2D case, knowing the center of an object and its size is enough to output the prediction, but in our case, we need to predict a 3D pose. To accomplish this, on top of the heatmap, for each point we predict the size vector $(width, height, depth)$, the offset vector (d_x, d_y, d_z) and the rotation angle α. The final prediction is a box with predicted size, the position $(p_x + d_x, p_y + d_y, p_z + d_z)$, where (p_x, p_y, p_z) is the position of a point that is classified as the object's center based on heatmap values. This means that a point at the front of a car can be responsible for the object prediction, and the actual position should be offset towards the real object center. An example is shown in Fig. 13, where the predicted box for a car is based on a frontal point, which means that the position of the predicted offset has to be defined at roughly half the object's size in the frontal direction. This can be solved by classifying responsible points as those lying in either side of an object and using this to offset the initial box position closer to a true object center.

3.1.3 Performance evaluation

Table 23 contains results for different configurations of models that use spherical pseudo images. Careful selection of a backbone, learning rate drop, augmentations and other hyperparameters lead to a big difference in performance, which can be further improved by training the model on a bigger dataset such as NuScenes [22].

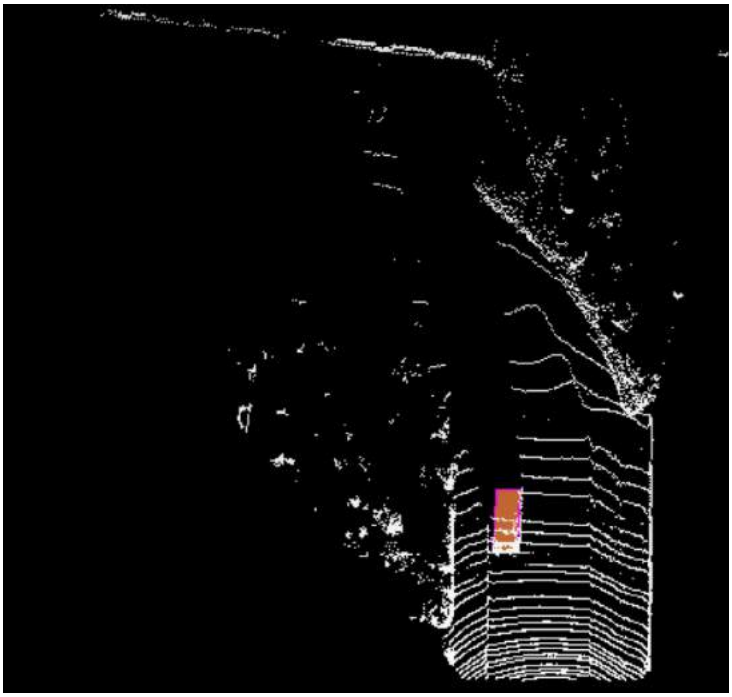


Figure 13: Example of an output of an SPI model projected to Bird's-Eye View (BEV) for better visualization.

Table 23: Results of Car 3D Object Detection on KITTI evaluation split. Easy object category is presented at different IoU thresholds for positive detections: 0.5 and 0.7. Evaluations of bounding box (BBox), Birds-Eye-View(BEV) and 3D boxes (3D) are presented. SPI models use spherical pseudo images as inputs. FPN/DLA34 specifies which backbone was used. The number 10k/60k/100k represents a number of steps to decrease the learning rate. WA means that worse data augmentations were used. Processing speed (FPS) is measured on NVIDIA GeForce RTX 2080 Ti.

Method	**FPS**	**Easy mAP @ 0.7**			**Easy mAP @ 0.5**		
		BBox	**Bev**	**3D**	**BBox**	**Bev**	**3D**
SPI-FPN-60k	25	79.04	80.93	80.62	79.04	67.97	52.35
SPI-FPN-100k	25	77.59	79.82	79.07	77.59	64.51	42.06
SPI-DLA34-100k	91	46.23	69.66	61.46	46.23	35.17	18.55
SPI-FPN-10k-WA	25	53.12	56.50	54.59	53.12	37.62	24.48
PointPillars	38	90.60	90.75	90.75	90.60	90.04	80.68

3.1.4 Future Work

We plan to explore better hyperparameters and datasets/augmentations in order to improve performance. We also plan to investigate the problem of a possible high distance between the responsible point and the center of the object by classifying this point as lying on the frontal, backward or other parts of the target object and shifting the initial bounding box center closer to the actual object center which results in smaller variance in predicted offset values, helping the network to return accurate position predictions.

3.2 Single object 3D tracking with Siamese PointPillars

3.2.1 Introduction, objectives and summary of state of the art

Multiple 3D Object Tracking (MOT) is focused on detecting and assigning unique object IDs for each object of interest over multiple data frames. These IDs should be different for different objects and identical for the same object across all frames. The Single Object Tracking (SOT) task performs tracking of only one object of interest, but problems that arise in MOT and SOT are similar. Given perfect object detections for MOT, tracking problems mainly come from losing the object due to either inability to connect the predictions at the current frame with previous history, or due to identification switches, where close or occluding objects are switching their IDs. The Single Object Tracking problem cannot have an ID switch problem directly, but if the object of interest is close to, or occluded by, another similar in appearance background object, the method may start identifying this background object as the target and the actual target object as background. This results in an ID switch scenario. When the target object cannot be identified in a frame that it is actually present, the method loses its track in the same way as in the MOT problem.

3D Multiple Object Tracking is usually performed in either by following a detection-based approach, or a simultaneous detection and tracking approach. Detection-based methods rely on an external object detector to find the location of the objects, while the method tries to assign IDs to the detected objects. One of such approaches, AB3DMOT [230], uses 3D Kalman filter to predict an object's state and the Hungarian matching algorithm with 3D IoU metric to match between tracklets predictions and provided detections.

Simultaneous detection and tracking methods mainly use internal representations for both object detections and prediction of re-identification (RE-ID) features. A low distance metric between RE-ID features of two objects indicates high probability of these objects being identical and is used to assign same IDs for objects across frames. CenterPoint [209] represents objects as a central point and uses its features to regress final position, 3D box size, rotation and velocity estimate. This method follows a similar approach as in PointPillars [110] and creates a 2D pseudo-image based on pillar representation of the input point cloud. To perform tracking, the central point is projected on the previous frame by adding a scaled negative velocity estimate of this object. The projected point is matched with the closest object from the previous frame.

Single Object Tracking methods take detection of the target in the first frame and then track the object of interest through consequent frames without the use of a detector. In [72] a 3D Siamese network is used to auto-encode the shape of a target with a point-wise network. A Siamese network is also used in [57, 190] with a Point-wise MLP or Transformer architectures.

3.2.2 Description of work performed so far

We follow the approach used in [11] for single object tracking, and apply it on a pseudo-image extracted by PointPillars [110] or another pseudo-image-based 3D Object Detection method such as TANet [125]. The part of the pseudo-image, generated by the model's head, is used as an input image for a Siamese network. We consider two ways of performing Siamese tracking. One follows Axis-Aligned Bounding Box (AABB) tracking, and the second one follows rotation-based search. In the AABB case, the object of interest is first detected in the pseudo-image space in the form of an Axis-Aligned Bounding Box and then an additional head uses features extracted from this position to perform final 3D Bounding Box regression. An example of AABB data for tracking is shown on Fig. 14.

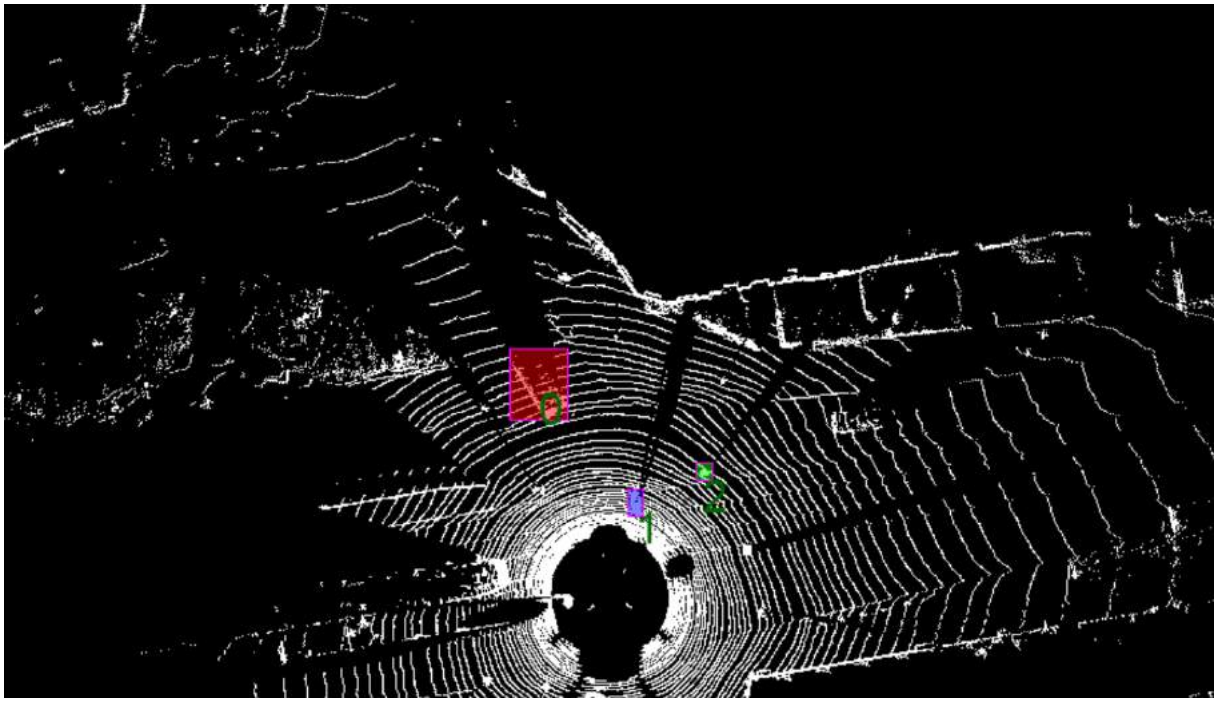


Figure 14: Example of AABB tracking annotations generated for KITTI tracking dataset. The car with index zero is rotated, but the corresponding AABB box cannot accurately capture its shape due to its rotation with respect to the frame's axes.

Training is performed by extracting target and search regions for each object in the dataset and applying a cross-correlation operation based on either 2D convolution, or Bhattacharyya distance. The scores extracted by the cross-correlation operation are compared to the ground truth using Binary Cross Entropy loss. The search region is two times bigger than the target region and has a random offset as an augmentation. Label elements around the target center are considered positive, and all other elements are considered negative. The feature extraction branch is based on the convolutional part of the PointPillars' Region Proposal Network (RPN), which is pretrained on the KITTI detection dataset.

Tracking is performed by first applying detection on the initial frame to find the object of interest. The target model is then extracted by applying the model's feature extractor on a target region, which is created by increasing the size of the target by a small number to add contextual information. The search region is defined in the same way as in training and is used in the next frame to search for a new position of the target. Features of the search region in the new frame and the initial frame's target region are compared to create similarity scores, and the position with the highest similarity score is used as the new target position.

Multiscale search, as used in 2D single object tracking, is not required in the case of 3D object tracking based on pseudo-images due to the constant object shape at different distances from the lidar sensor. However, depending on the rotation of the object in AABB tracking, the size of the final AABB will be varying. This means that there is a need to perform a free-aspect-ratio scale search. This is done by retrieving image features both from the target's location at the first frame and at the previous frame. These features are compared with the search region at the current frame, and the target with the highest score is selected as a new initial target.

For the rotation search case, we use an original bounding box and crop a rotated part of the pseudo-image to generate target and search images. These images are resized to a constant size

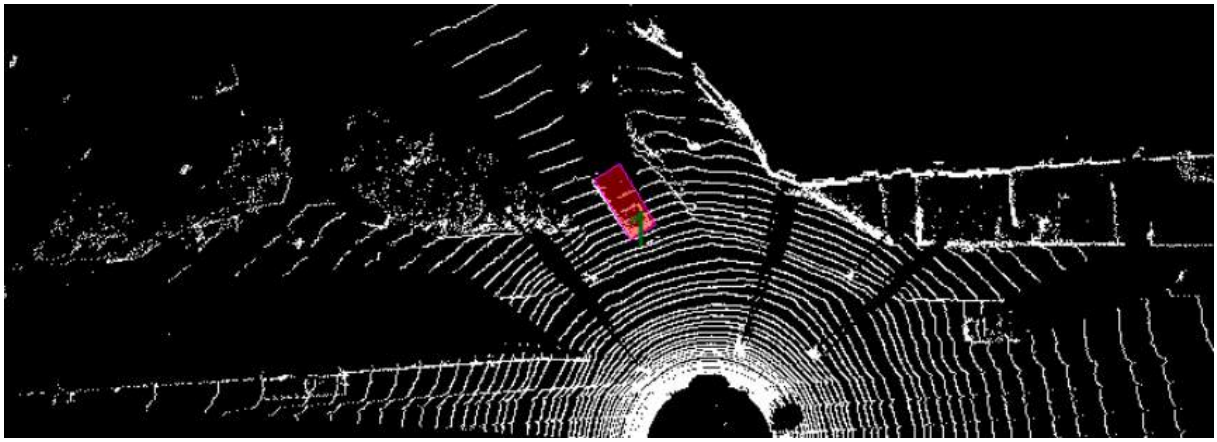


Figure 15: An example of rotated tracking prediction generated for KITTI tracking dataset.

of $(127, 127)$ and $(255, 255)$, respectively. These images are processed to create feature maps and compared in the same way as in the AABB case, but final displacements are also scaled down to original sizes. An example of a prediction obtained by following this approach can be seen in Fig. 15.

Instead of a multiscale search, we take crops of the search region with a slightly changed rotation angles and compare those to the features of the initial target. The rotation with the best score (multiplied by a penalty for rotated search regions) is selected as the new object rotation. The dimensions of the object are kept constant, as they do not change in Lidar space for rigid objects like cars.

3.2.3 Performance evaluation

We evaluate this approach on KITTI Tracking dataset, which is a Multiple Object Tracking dataset. In order to create a Single Object Tracking task from it, we extract separate tracklets and evaluate the mean AABB IoU (mIoU) between prediction and ground truth boxes for the object category Car. We also count the average percentage of tracked (mTracked) objects by counting the number of frames where IoU is positive and taking an average of that across tracklets and scenes. The results are given in Table 24, where it can be seen that both AABB and Rotated models achieve similar performance in all metrics. Since the evaluated models are in the initial stage of development, we expect that further experimentation for hyperparameter tuning can lead to improved performance. The processing speed can be improved by creating the pseudo-image of only the relevant part of the point-cloud and altering the convolutional layer structure. For the full 3D evaluation, Rotated models are expected to be faster, as there are no additional regressions needed.

3.2.4 Future Work

We plan to improve the method's speed (FPS) by focusing on a sub-scene of interest, i.e. by creating pseudo-images of only the part of the point cloud which needs to be processed further for tracking the object in the next frame. We plan to investigate the best configuration of AABB prediction backbone and a 3D prediction head to improve tracking performance.

Table 24: Results on 3D Car Tracking on KITTI evaluation split. The models are evaluated on separate tracklets by computing mean the IoU of prediction and ground truth objects. AABB/Rotated indicates the approach used.

Method	FPS	mIoU (%)	mTracked (%)
Siamese PointPillars AABB	5.7	38.7	49.0
Siamese PointPillars Rotated	4.8	37.2	53.8

3.3 Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning

3.3.1 Introduction, objectives and summary of state of the art

The goal in many tracking applications is *control* [242], instead of tracking *per se*, leading to *passive tracking* approaches, where tracking is disconnected from the actual task at hand. On the other hand, the emergence of powerful Deep Reinforcement Learning (DRL) approaches [188, 79], enabled the development of *active tracking* approaches [126, 127]. Such approaches are capable of learning end-to-end control policies, directly from raw RGB input, without any intermediate pre-processing step. In this way, the developed algorithms can be directly optimized for the task at hand, without involving separate intermediate tasks. At the same time, this unified approach also holds the credentials for providing less complex and more robust systems [127].

Despite their apparent advantages, little work has been done so far for active tracking approaches [126, 127], since active tracking requires using advanced and realistic simulation environment for simulating the effects of various control commands instead of relying on static datasets. At the same time, active tracking typically relies on reward signals for the training process instead of ground truth bounding boxes that are usually used in passive tracking approaches. However, defining the appropriate reward functions for such tasks is not trivial, since there are many alternative ways to formulate the goal of the system [170]. This is in contrast with other DRL applications, such as games [139], where the reward function is intrinsic to the problem. Indeed, the way that the reward function is defined can have a significant effect on the behavior of the resulting DRL model, especially for complex control tasks [160]. At the same time, providing additional rewards, in the form of *reward shaping* [146, 206, 47], can often allow for further increasing the stability of the training process, along with its convergence speed.

The contribution of our work is two-fold. First, we develop and evaluate an active tracking simulation environment, demonstrating that active tracking methods can be indeed trained in an end-to-end fashion operating *directly* on raw RGB inputs. To this end, a realistic robot simulation environment was constructed using the Webots simulator [134], while the proposed method was evaluated using a real robot, the e-puck robot [141], and two different setups, involving different movement patterns of the robots. Furthermore, two different control scenarios were evaluated, corresponding to the two main approaches that are typically used for control: a) control using discrete actions/steps and b) control using continuous action spaces. Second, a simple, yet effective temporal difference-based reward function is introduced and evaluated for active tracking, improving the tracking error and allowing for directly performing control based on a tracking objective. Indeed, it is demonstrated, through extensive experiments, that the employed reward function can indeed lead to improvements in tracking accuracy, under a

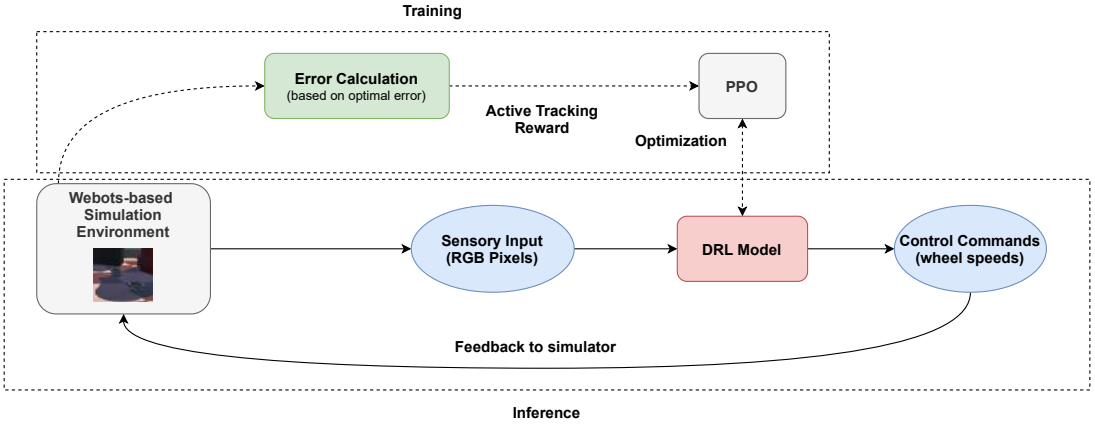


Figure 16: Overview of the proposed DRL-based active tracking setup. First, the agent (DRL Model) is trained using the developed Webots-based simulation environment. Then, the trained agent is evaluated on a number of different scenarios.

wide range of different setups.

A summary of this work is provided hereafter. The corresponding publication is listed below, and can be found in Appendix 7.9:

1. [210] P. Tiritiris, N. Passalis and A. Tefas, "*Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning*", International Conference on Emerging Techniques in Computational Intelligence, 2021

3.3.2 Description of work performed so far

The simulation environment was built using the Webots simulator [134], which is an open source highly realistic robot simulator. The developed simulation environment contains two GCtronic e-puck robots [141], with the first one being the tracked target and the second one carrying an RGB camera (64×64 pixels) and aiming to track the first robot. E-puck robot moves by appropriately setting the velocity of its two wheels. Two different kinds of agents were employed in this work: a) discrete action space agents and b) continuous action space agents. For agents that work with discrete actions, there are $2k+1$ possible velocity values within the range $[-maxSpeed, +maxSpeed]$, splited by $2k$ equally spaced partitions. Therefore, the total number of actions for these agent is $(2k+1)^2$, given that both wheels of the robot must be controlled. On the other hand, for continuous agents, two real numbers, one for each wheel, are used. The output of the agent is constrained between the minimum and the maximum speed supported by the robot, ensuring that all control actions will be within the hardware capabilities of the robots.

For training the agent, a state-of-the-art DRL optimization method, the Proximal Policy Optimization (PPO) [188], was employed. An actor-critic architecture was used to this end, where the actor is responsible for selecting the next action, while the critic was used for estimating the value of each state. PPO relies on the *advantage* for each performed action a when the agent is at state s:

$$A_\theta(s,a) = Q_\theta(s_t) - \sum_{t=0}^{T-1} \gamma^t r_t, \tag{81}$$

where $Q_\theta(s_t)$ denotes the critic's network estimated value, γ is the discount factor and r_t is the obtained reward at time t for a total number of T time-steps. To ensure the stability of the optimization process, PPO uses a probability ratio between new and old policy. Clipping this ratio, as proposed in [188], improves the behavior of the training process. Finally, note that in training phase PPO collects samples from the older policy using importance sampling.

The architecture used for both the actor and critic follows: Three 2D convolutional layers were used, followed by three fully connected layer. The ReLU activation function was used after each layer [75], while average pooling was employed instead of max pooling to avoid discarding potentially useful information. For discrete agents, the final layer of the actor has the same number of neurons as the number of possible actions, i.e., $(2k+1)^2$, while for continuous agents two output neurons are used (one for controlling the speed of each wheel). In the latter case, the output of the network is passed through the *tanh* activation, to ensure that it will range between -1 (maximum speed backward) and 1 (maximum speed forward). When the network outputs a value of 0, then the corresponding wheel stays stationary.

The RL agents must learn to follow the moving target as close as possible over a safe distance, while the camera looking at the center of the target. To this end, we define the distance error as:

$$e_d(t) = |d(t) - s|, \tag{82}$$

where $d(t)$ is the distance between the robots at time t and s is a predefined distance that must be kept from the robot (safe distance). Also, we define the angular error $e_a(t)$ at time t as the absolute value of the angle between the optimal look-at vector of the camera and the speed vector of the target robot. A separate reward is calculated for each of these two errors, i.e., $r_d(t)$ and $r_a(t)$ respectively, while the total reward is calculated as the sum between the corresponding rewards:

$$r(t) = r_d(t) + r_a(t) \tag{83}$$

For defining these two individual rewards, i.e., $r_d(t)$ and $r_a(t)$, that contribute to the final reward provided in (83) several methods can be considered. Perhaps the simplest one is to directly use the negative of error, i.e.,

$$r(t) = -error(t), \tag{84}$$

where $r(t)$ refers to either $r_d(t)$ and $r_a(t)$ (according to the value used in place of $error(t)$), aiming to directly minimize the control error. However, as we experimentally demonstrate in Section 3.3.3, this naive way of defining the reward leads to suboptimal results. Recent works in vision-based DRL control proposed to use hint-based rewards inspired by reward shaping methods [160], i.e.,

$$r(t) = \begin{cases} c_g, & error(t) < margin \\ c_m, & error(t) < error(t-1) \\ c_p, & \text{otherwise} \end{cases}, \tag{85}$$

where c_g corresponds to the reward for achieving the desired target, c_m to the reward for moving towards a direction that reduces the error , while c_p corresponds to the penalty the agent receives in any other case (its absolute value must be larger than c_g to avoid oscillations). We set these parameters to $c_g = 1$, $c_m = 0.5$ and $c_g = -1.2$ following the protocols proposed in the literature and experiments conducted to select the optimal parameters. However, setting these parameters can require a significant amount of effort, involving repeated experiments to determine the optimal values. Therefore, to overcome this limitation, in this work we propose using a simpler,

yet more effective approach for calculating the reward based on the *temporal difference* between two subsequent error values, i.e.,

$$r(t) = error(t-1) - error(t). \tag{86}$$

This approach does not require any kind of additional fine-tuning, while it leads to more accurate control. Note that the first and third reward functions are directly linked to the (unnormalized) values of the error, therefore min-max normalization is required after the end of each episode, to ensure the stability of the optimization process. This is not necessary for the hint-based reward, since the values of this reward are already bounded.

3.3.3 Performance evaluation

The different agents and reward functions were evaluated under different setups, as analytically reported in Appendix 7.1. The developed agents were evaluated in two different setups:

1. *random movement*, where the velocity of both wheels of the front robot are selected randomly (therefore the first robot moves on a non-straight trajectory). The velocities remain the same during the entire episode. This setup was used both for training and evaluation of the trained agents.

2. *random movement with velocity changing*, where the same setup as before is used, but the speed of the wheels changes every 200 steps. As a result, the trajectory of the first robot changes several times within the same episode, requiring the tracking robot to promptly adjust in order to avoid losing the target robot. This setup was used for evaluating the agents trained with the first setup.

Each episode consists of 1000 steps, while an episode ends early when the tracking robot loses its target. For evaluation we report two different metrics:

1. *average distance error* ("Distance error"), which measures whether the agents keeps the desired distance from the tracked robot, and

2. *average control steps per episode* ("Steps per episode"), which measures the ability of the agent to track the front robot (note that an episode ends when the tracking robot loses its target).

The proposed method was initially evaluated using the first setup (random movement evaluation). The evaluation results for the discrete agent are reported in Table 25, while for the continuous agent in Table 26. For all the evaluated reward functions, the agents are indeed successfully trained, since they solve almost all the test episodes perfectly (the average number of steps is larger than 950 out of a maximum of 1000). At the same time, the temporal difference reward seems to lead to the overall best distance error (both for the discrete and continuous agent). However, it lead to slightly worse behavior for the discrete agent, failing to correctly track the target robot in a few cases, since the average number of steps per episode is reduced from 1000 to 969. However, the opposite behavior is observed in the second evaluation setup reported in Table 27, where the proposed reward leads to both the best distance error, as well as leads to never losing the target robot for all the evaluated episodes. The same results are achieved for the hint-based reward, even though hint-based rewards seem to always lead to higher distance error. This can be explained, since hint-based rewards are disconnected from

Table 25: Evaluation Setup 1: Discrete agent

Reward	Distance error	Steps per episode
Negative of error	0.014	1000
Hint-based	0.054	969
Proposed	**0.009**	969

Table 26: Active Tracking Evaluation Setup 1: Continuous agent

Reward	Distance error	Steps per episode
Negative of error	0.02	954
Hint-based	0.03	966
Proposed	**0.01**	1000

the actual value of the error. Even though in previous works this has been shown to improve the training stability when combined with Q-learning based algorithms [160], this reward function does not seem to lead to similar improvements for the problem at hand.

4 Deep SLAM and 3D scene reconstruction

4.1 EfficientLPS: Efficient LiDAR Panoptic Segmentation

4.1.1 Introduction and objectives

Understanding the scene in which an autonomous robot operates is critical for its competent functioning. Such scene comprehension necessitates recognizing instances of traffic participants along with general scene semantics that can be effectively addressed by the panoptic segmentation task. In this section, we extend our work on image-based panoptic segmentation, namely EfficientPS, discussed in the OpenDR deliverable D4.1 to the LiDAR domain. We present the novel top-down Efficient LiDAR Panoptic Segmentation (EfficientLPS) architecture comprised of a shared backbone, scale-invariant semantic and instance segmentation heads, and a panoptic fusion module. We design the model architecture to addresses multiple challenges that occur in the LiDAR domain including distance-dependent sparsity, severe occlusion, large scale-variations, and reprojection errors. For further improvement, we formulate a regularized pseudo labelling framework to enable training on unlabelled data. Finally, we evaluate Effi-

Table 27: Active Tracking Evaluation Setup 2: Discrete agent

Reward	Distance error	Steps per episode
Negative of error	0.016	985
Hint-based	0.035	1000
Proposed	**0.007**	1000

cientLPS on two challenging datasets: 1) SemanticKITTI [8] and 2) nuScenes [22], for which we also release ground truth labels.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.10:

- [195] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada, "EfficientLPS: Efficient LiDAR Panoptic Segmentation", *arXiv preprint arXiv:2102.08009*, 2021. (To appear in: *IEEE Transactions on Robotics (T-RO)*)

4.1.2 Summary of state of the art

Panoptic segmentation is a scene understanding problem [97] that unifies the tasks of semantic segmentation and instance segmentation. While the majority of research in this domain focuses on RGB images, only a few methods address 3D LiDAR point clouds. Since most works are largely built upon advances made in semantic segmentation and instance segmentation, we first review recent approaches for these closely related subtasks, followed by state-of-the-art methods proposed for LiDAR-based panoptic segmentation. Most works in this domain are largely built upon advances made in semantic segmentation and instance segmentation, therefore we first review recent methods that have been proposed for these closely related sub-tasks, followed by state-of-the-art approaches that have been introduced for panoptic segmentation.

Semantic Segmentation: The challenges posed by the unordered and sparse nature of point clouds has hindered the progress in LiDAR semantic segmentation for autonomous driving. Considerably fewer techniques have been proposed to address this task using point clouds in comparison to methods in the visual domain. Dewan *et al.* [48] propose an approach to classify points into movable, non-movable, and dynamic classes, by combining deep learning-based semantic cues and rigid motion based motion cues in a Bayesian framework. Wu *et al.* [234] propose a projection-based approach that builds upon SqueezeNet and introduces the so-called fire module. The release of the SemanticKITTI [8] dataset motivated many works in semantic segmentation of LiDAR point clouds. Milioto *et al.* [136] propose a 2D CNN architecture that operates on spherically projected point clouds and employs a kNN-based post-processing step to account for the occlusions due to the projection. SalsaNext [42] uses spherical projection for segmentation and also performs uncertainty estimation. PolarNet [256] projects the point cloud in the birds-eye view and employs a ring convolutions on the radially defined grids. KPRNet [98] uses a 2D-3D hybrid approach combining 2D convolutions for semantic segmentation followed by KPConv-based [207] post-processing using point-wise convolutions.

Instance Segmentation: Similar to the image domain, instance segmentation of point clouds can be classified into two categories: proposal-based and proposal-free methods. Proposal-based methods perform 3D bounding box detection followed by point-wise mask generation for the points in each bounding box. 3D Bonet [239] follows this approach using two separate 3D bounding box proposal generation and mask generation branches. GSPN [243] generates proposals using a shape aware proposal generation for different instances. On the other hand, proposal-free methods directly predict the instances by detecting keypoints such as the centroid of the instance, or the similarity between points, which is followed by clustering [251]. SGPN [228] learns a similarity matrix between the points which is used to cluster points with higher similarity scores between them. PointGroup [88] extracts semantic information using 3D sparse convolutions to cluster points towards the instance centroid, followed by using the original points and the clustered points to obtain the final prediction. VoteNet [49] predicts an offset vector to the centroid of every point and then employs clustering.

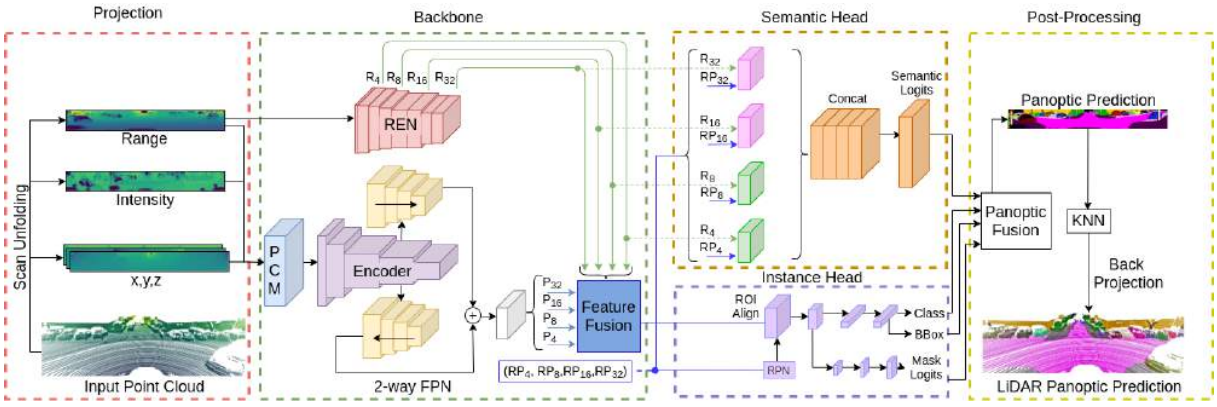


Figure 17: Illustration of the proposed EfficientLPS [195] architecture for LiDAR-based panoptic segmentation. A 2D projection of the point cloud is fed to the Proximity Convolution Module (PCM). Afterward, a shared backbone comprising the EfficientNet [205] encoder with a 2-way feature pyramid network (FPN) [140] and a range encoder network (REN). The fused output of these modules is fed to semantic and instance heads, whose logits are post-processed in the panoptic fusion module. The final output is projected back to 3D via a kNN algorithm.

Panoptic Segmentation: Panoptic segmentation methods can also be classified into proposal-free (bottom-up) or proposal-based (top-down) techniques. Bottom-up methods group points belonging to the same instances either by a voting scheme or based on pixel-pair affinity pyramid while simultaneously learning the semantic labels [65]. On the other hand, top-down approaches, e.g., our previous work MOPT [86], tackle the problem in two a stage manner with a dedicated instance segmentation branch for detecting and segmenting 'thing' classes and a semantic segmentation branch for segmenting 'stuff' classes. Miliotto *et al.* [135] adopt the bottom-up approach without instance proposals by using spherical projection of point clouds and predicting offsets to the centroids for aiding clustering. They also use 3D information available in the range images for trilinear upsampling in the decoder. Panoster [68] uses an instance head that directly provides the instance IDs of the points from learnable clustering without any explicit grouping requirement. In addition to the spherical projection-based method, Rangenet++ [136] and Panoster [68] also show the implementation of their clustering mechanism using the point-based method KPConv [207] for semantic segmentation.

4.1.3 Description of work performed so far

In this section, we first give a brief overview of our proposed EfficientLPS architecture followed by a summary of each of its constituting components. For a in-depth description of the approach we refer to the full paper [195], which is also included in the appendices. Our network follows the top-down layout as shown in Figure 17. The approach is composed of four modules: 1) projection of the point cloud into the 2D space comprising channels for range, intensity, and (x,y,z) coordinates; 2) a shared backbone consisting of the proposed Proximity Convolution Module (PCM), a modified EfficientNet-B5 [205] encoder with the 2-way FPN [140], the proposed Range Encoder Network (REN), and the multi-scale feature fusion module; 3) and 4) task-specific heads for semantic and instance segmentation; and 5) post-processing to fuse the logits from the separate heads yielding the panoptic prediction and finally back-projection to the 3D domain.

Projection: We employ scan unfolding [217] to project the point cloud into the 2D range

image format. Scan unfolding aims to mitigate the problems due to the alternate spherical projection method that suffers from point occlusions due to ego-motion correction. Scan unfolding yields a much denser representation than spherical projection. LiDAR sensors typically provide raw data in a range image-like format, with each pixel describing the range value at a particular row and column. Each column of this range image consists of measurements taken by individual modules stacked vertically within the sensor at a particular time and each row represents the consecutive measurements of one module taken during spinning of the sensor. However, most of the publicly available datasets provide LiDAR measurements as a list of 3D Cartesian coordinates, without any information about the column or row indices. Hence, we assign these indices to every point in the laser scan for recovering the range image-like representation.

Backbone: The backbone consists of our proposed Proximity Convolution Module (PCM), followed by an encoder and the novel Range-aware Feature Pyramid Network (RFPN). The core of the PCM is the proximity convolution operation, which is defined as

$$y(p) = \sum_{p_n \in N} w(p_n) \cdot x(p + p_n), \tag{87}$$

where N consists of offsets for the n nearest neighbors of pixel p. The weights w are learned in the same manner as in standard convolutions. PCM overcomes a drawback of the vanilla convolution by relaxing the fixed grid structure that limits the geometric transformation modeling capacity. In particular, PCM exploits range information to augment the spatial sampling locations for effectively improving the transformation modeling ability. We adopt the EfficientNet [205] topology for the main encoder as well as the Range Encoder Network (REN). We remove the Squeeze and Excitation (SE) connections to enable better localization of features and contextual elements. Similar to the proposed proximity convolution module, we replace the batch normalization layers with iABNsync and Leaky ReLU activation. Our proposed range-aware FPN (RFPN) reinforces the coherently aggregated fine and contextual features of Feature Pyramid Networks (FPNs) with distance awareness. This enables the network to better segregate adjacent objects with different range variations. We build upon the 2-way FPN [140] that enables bidirectional flow of information using two parallel branches that aggregate multi-scale from the main encoder in both a top-down and a bottom-up manner. To fuse the outputs of the 2-way FPN and the REN, we add a multi-scale fusion module. It enables the network to emphasize on the more informative features between the 2-way FPN features and the corresponding fused features, hence incorporating distance awareness selectively.

Semantic Head: The main component of our proposed distance-dependent semantic head is the novel range-guided depth-wise atrous separable convolution operation. We essentially encode the range using the REN module and compute the dilation factor to apply at each central pixel from the encoded features. We then employ the depth-wise atrous separable convolution operation with the computed dilation factor, thereby enabling the receptive field to be adaptable to the range data. We extend the semantic head proposed in [140] consisting of Dense Prediction Cells (DPC), Large Scale Feature Extractor (LSFE), and Mismatch Correction Module (MC) with bottom-up path augmentation connections. We effectively retain the MC module and the bottom-up path augmentation connections but redesign the DPC and LSFE modules by incorporating our range-guided depth-wise atrous separable convolutions. We refer to this new LSFE variant as Range-guided Large Scale Feature Extractor (RLSFE).

Instance Head: We adapt the Mask R-CNN [74] topology and make certain modifications, including replacing the batch normalization and ReLU activations with iABNsync and Leaky ReLU layers, respectively. In the first stage, the Region Proposal Network (RPN) employs a

fully convolutional network to generate object proposals and objectness scores for each output resolution of the RFPN module. In the subsequent stage, ROI align extracts features by directly pooling from the n^{th} channel of the FPN encodings with a 14×14 spatial resolution bounded within the object proposals obtained in the previous stage. These extracted features are then fed to specialized bounding box regression, object classification, and mask segmentation networks.

Post-processing: We fuse the outputs of the semantic and instance heads using the heuristic proposed in [140] to yield the panoptic predictions in the 2D domain. This heuristic enables us to adaptively fuse the predictions of both heads to alleviate the inherent overlap problem. During the projection to point clouds, different points may get assigned to the same pixel in the projected image resulting in the assignment of the same label to all overlapping points. Moreover, due to the downsampling operations in the network, the convolutions produce blurry outputs in the decoder leading to leaking of the labels at the boundaries of the instances during back-projection to the 3D domain. We use a k-nearest neighbor (kNN) based back-projection scheme [136] to mitigate these issues. For every point in the point cloud, the nearest k neighbors to the point vote for its semantic and instance labels. We obtain the labels of the selected neighbors from the corresponding pixels in the projected output prediction. To compute the nearest neighbors, we search for nearest neighbors within a pre-defined window around the pixel in the projected range image, out of which we select k nearest points based on the differences in their absolute range value.

4.1.4 Performance evaluation

We benchmark our proposed EfficientLPS for LiDAR panoptic segmentation on two challenging datasets, namely SemanticKITTI [8] and nuScenes [22]. We use PyTorch for implementing all our architectures and we trained our models on a system with an Intel Xenon (2.20GHz) processor and a single NVIDIA TITAN X GPU. For a detailed performance evaluation, we refer to the appended paper in Appendix 7.10. The implementation of our proposed EfficientPS model and a live demo on various datasets is publicly available at `http://lidar-panoptic.cs.uni-freiburg.de`. In Figure 18 we show some qualitative results of our approach and compare them to two baseline models.

Our proposed EfficientLPS sets a new state-of-the-art in LiDAR panoptic segmentation. In particular, at the time of submission of the corresponding paper, it was ranked first on the SemanticKITTI panoptic segmentation leaderboard.

4.1.5 Future Work

In our ongoing work, we are exploring extensions of our image and LiDAR-based panoptic segmentation methods to new frontiers in robotics, e.g., prediction of instance labels for occluded objects. Furthermore, we are planning to tightly couple panoptic segmentation with other important tasks such as depth estimation to supervise and improve the respective other method. Finally, these approaches will then be integrated in SLAM systems, leveraging images and/or point clouds, to exploit the use of semantic information in mapping and localization.

4.2 CMRNet++: Map and Camera Agnostic Monocular Visual Localization in LiDAR Maps

4.2.1 Introduction and objectives

Autonomous mobile robots, such as self-driving cars require accurate localization to safely navigate. Although Global Navigation Satellite System (GNSS) provides global positioning, its accuracy and reliability is not adequate for robot navigation. For example, in urban environments, buildings often block or reflect satellites signals, causing non-line-of-sight and multipath issues. In order to alleviate this problem, localization methods that exploit sensors on the robot

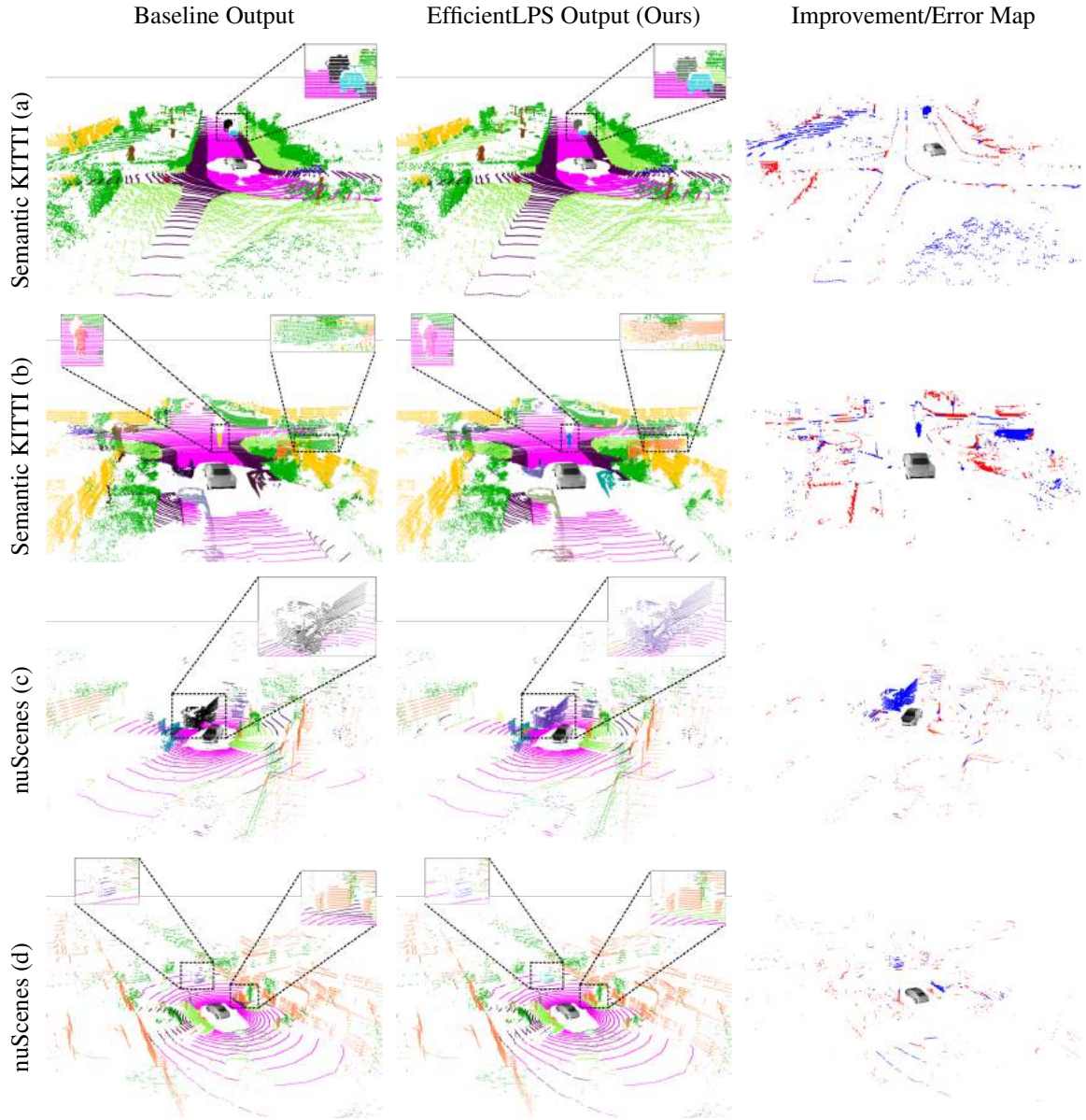


Figure 18: Qualitative comparison of our proposed EfficientLPS [195] against PanopticTrackNet on SemanticKITTI [8] and (KPConv + Mask R-CNN) on nuScenes [22] validation sets. The map in the right column shows the points that are misclassified by EfficientLPS in red and the points that are misclassified by the baseline but correctly predicted by EfficientLPS in blue.

are used to improve precision and robustness. Moreover, deep learning has found its way to the localization tasks. However, existing CNN-based pose regression methods fail to generalize well to unseen places. The previously proposed CMRNet [28] effectively addresses this limitation by enabling map independent monocular localization in LiDAR maps. A follow-up work enhances the method by introducing CMRNet++ [27], which is a significantly more robust model that not only generalizes to new places effectively but is also independent of the camera parameters. It is evaluated on KITTI [69], Agroverse [30], and Lyft5 [81] datasets. We work towards an extension of CMRNet++ demonstrating that our approach exceeds state-of-the-art methods. A summary of this work is provided hereafter.

4.2.2 Summary of state of the art

A wide range of methods have been proposed to tackle the localization task, using a variety of onboard sensors. While LiDAR-based approaches [104, 9] typically achieve sufficiently accurate localization, their adoption is primarily hindered due to the associated high cost. On the other hand, camera-based methods [185, 184, 17] are more promising for widespread adoption in autonomous vehicles as they are significantly less expensive. Although historically the performance of camera-based approaches has been subpar compared to LiDAR-based methods, recent advances in computer vision and machine learning have substantially narrowed this gap. Some of these methods employ CNNs [92, 175, 223, 20, 224] and random forests [192, 29] to directly regress the pose of the camera given a single image. Although these methods have achieved remarkable results in indoor environments, their performance has been significantly limited in large-scale outdoor environments [186]. Moreover, they can only be employed in locations where these models have been previously trained on.

In the last decade, map providers have been developing the next generation HD maps tailored for the automotive domain. These maps include accurate geometric reconstructions of road scenes in the form of point clouds, most often generated from LiDARs. This factor has motivated researchers to develop methods to localize a camera inside LiDAR-maps. Localization can typically be performed by reconstructing the three-dimensional geometry of the scene from a camera, and then matching this reconstruction with the map [26, 202], or by matching in the image plane [232, 145, 28].

Our work builds upon the previously proposed CMRNet [28] model, which was inspired by camera-to-LiDAR pose calibration techniques [187]. Unlike other state-of-the-art CNN-based approaches for pose regression [92, 20, 175], CMRNet does not learn the map, but to match images to a pre-existing map. Consequently, CMRNet can be used in any environment for which a LiDAR-map is available. However, since the output of CMRNet is metric (a 6-DoF rigid body transformation from an initial pose), the weights of the network are tied to the intrinsic parameters of the camera used for collecting the training data.

4.2.3 Description of work performed so far

CMRNet++ is an extension of CMRNet decoupling the localization into two steps: 1) pixel-to-3D-point matching and 2) pose regression. In the first step, the CNN only focuses on matching at the pixel-level instead of metric basis, which makes the network independent of the intrinsic parameters of the camera. These parameters are instead employed in the second step, where traditional computer vision methods are exploited to estimate the pose of the camera, given the matches from the first step. Consequently, after training, CMRNet++ can also be used with

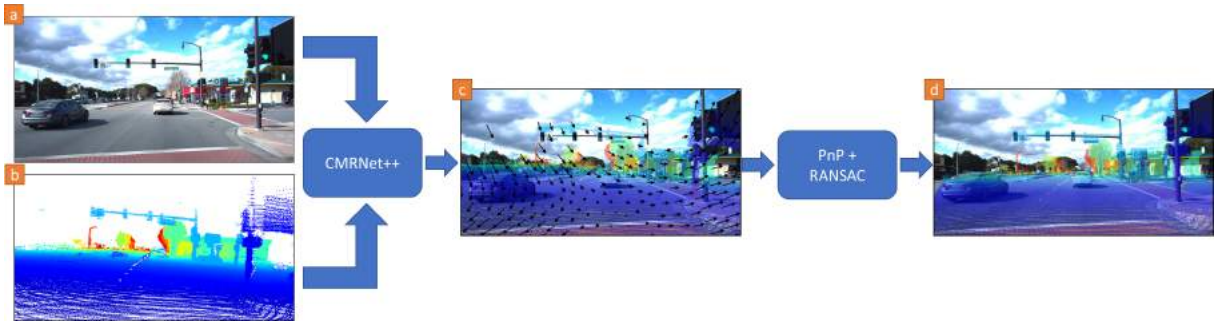


Figure 19: Outline of the approach proposed in [27]. (a) The input RGB image and (b) the LiDAR-image are fed to our CMRNet++, predicting (c) pixel displacements between the two inputs. (d) The predicted matches are used to localize the camera using a PnP+RANSAC scheme.

different cameras and maps from those used during training. An outline of our proposed method is depicted in Figure 19.

For the matching step, we generate a synthesized depth image, the LiDAR-image, by projecting the map into a virtual image plane using a rough pose estimate obtained from GNSS and the camera intrinsics. To deal with occlusions in point clouds, we employ a z-buffer technique followed by an occlusion estimation filter [28]. Once the inputs to the network (camera and LiDAR images) have been obtained, for every 3D point in the LiDAR-image, CMRNet++ estimates the pixel of the RGB image that represents the same world point.

The network architecture is based on PWC-Net [201], which was introduced for optical flow estimation between two consecutive RGB frames. Unlike PWC-Net, CMRNet++ does not share weights between the two feature pyramid extractors, as the inputs to CMRNet++ are inherently different (RGB and LiDAR images). Moreover, since the LiDAR-image has only a single channel, we change the number of input channels of the first convolutional layer in the feature extractor from three to one. The output of CMRNet++ is a dense feature map, which is $1/4$ of the input resolution and consists of two channels that represent the displacement of the pixel in the RGB image from the same world point.

Once CMRNet++ has been trained, we have the map, i.e., a set of 3D points with known coordinates, along with their projection in the LiDAR-image and a set of matching points in the RGB image that is predicted by the CNN. Estimating the pose of the camera given a set of 2D-3D correspondences and the camera intrinsics is known as the Perspective-n-Point Pose (PnP) problem. We use the EPnP algorithm [114] within a RANSAC scheme [60] to solve this.

Similar to CMRNet, we employ an iterative refinement technique where we train different instances of CMRNet++, each specialized in handling different initial error ranges. During inference, we feed the RGB and LiDAR-image to the network trained with the highest error range and generate a new LiDAR-image by projecting the map in the predicted pose. The latter is then fed to the second instance of CMRNet++ that is trained with a lower error range. This process can be repeated multiple times, iteratively improving the estimated localization pose.

4.2.4 Performance evaluation

We evaluate the localization performance and generalization ability of our CMRNet++ on three challenging datasets covering a diverse set of countries, sensor setups, and traffic conditions. In particular, we benchmark CMRNet++ on the KITTI dataset [69], Argoverse [30], and the Lyft Level 5 AV dataset [81].

Table 28: Median localization error of CMRNet++ on KITTI, Argoverse, and Lyft5 datasets.

	Training Max. Error Range		KITTI Localization Error			Argoverse Localization Error			Lyft5 Localization Error		
	Transl. [m]	Rot. [deg]	Transl. [m]	Rot. [deg]	Fail [%]	Transl. [m]	Rot. [deg]	Fail [%]	Transl. [m]	Rot. [deg]	Fail [%]
Initial pose	-	-	≈ 1.97	≈ 9.83	-	≈ 1.97	≈ 9.83	-	≈ 1.97	≈ 9.83	-
Iteration 1	[-2, +2]	$[-10, +10]$	0.55	1.46	2.18	0.80	1.55	6.24	1.32	2.13	10.56
Iteration 2	[-1, +1]	$[-2, +2]$	0.22	0.77	-	0.34	0.58	-	0.79	1.28	-
Iteration 3	[-0.6, +0.6]	$[-1, +1]$	**0.14**	**0.43**	-	**0.25**	**0.45**	-	**0.70**	**1.18**	-

We generate 3D LiDAR maps for each of the datasets by accumulating single scans using the ground truth position (Argoverse and Lyft5) or a SLAM system [104] (KITTI). Afterward, we downsample the maps to a resolution of 10cm and remove potentially dynamic objects, e.g., pedestrians or cars (except for KITTI due to the lack of ground truth object boxes).

We employ the presented iterative refinement process by training three instances of CMRNet++. To improve the generalization ability of our approach, we apply data augmentation: 1) color jittering of the images by randomly changing the contrast, saturation, and brightness; 2) random horizontally mirroring of the images plus modifying the camera calibration accordingly; 3) random rotation of the images in the range [-5°, 5°]; and 4) transformation of the LiDAR point cloud to reflect these changes before generating the LiDAR-image.

We summarize the localization errors for the three datasets in Table 28. For more results, we refer to our previous work [27]. A live demo on various datasets is publicly available at `http://vloc-in-lidar.cs.uni-freiburg.de/`.

4.2.5 Future Work

In our future work, we plan to investigate whether the differentiable RANSAC approach [20] could be employed to train CMRNet++ in and end-to-end fashion, while maintaining the camera parameters outside of the learning step. We further aim to enhance our map building method by leveraging semantic segmentation or object detection.

4.3 SLAM and Plant row guidance

The plant row guidance algorithm is the base technology to allow the field robot to be able to navigate based on its RGB cameras instead of the GPS. Navigating by the cameras is necessary to be able to respond accordingly to the changes in the plant rows. If the robot does not navigate based on the plant rows, then the robot and/or implement will damage the crop, reducing the farmer's profits.

The algorithm has been moved from python into C++ and ROS to increase speed and reduce GPU requirements. In addition, AGI decided to remove the 2nd front camera as seen in the hardware specifications for the agriculture use case in D2.1 and use only 1 camera with the fish eye lens. The algorithm has been calibrated to adjust for the fisheye lens and the new angle. In the settings file, figure 1, the size of the image within the actual image (warping area), the width of the actual crop and the width between the crop can be input. In addition, the number of horizontal slices the image can be split into can be adjusted. This is for developmental purposes, so the model can be tuned.

```
input_image_topic: /camera_front_node/image_raw

publish_debug_image: True

warping_area:
top_left: [870, 600]
top_right: [1180, 600]
bottom_right: [1500, 1000]
bottom_left: [600, 1000]

number_of_slices: 10
crop_window_width: 20 #[pix] This can be seen as the row width (crop size)
user_row_distance_mm: 750 #mm
```

Figure 1: Setting of the yaml file

AGI has been improving the plant row guidance algorithm to increase accuracy and better be able to handle situations where the algorithm will fail. To be able to handle situations where the plant row can become unclear, e.g. tire tracks running over the crop rows or low crop density in the rows, a confidence level has been added to the algorithm. The confidence level gives an indication how well the algorithm thinks it knows where the crop row is located. If the confidence level drops below a certain level, the robot navigation system can choose to follow path of the last 2 way points or choose to follow the GPS and navigation system.

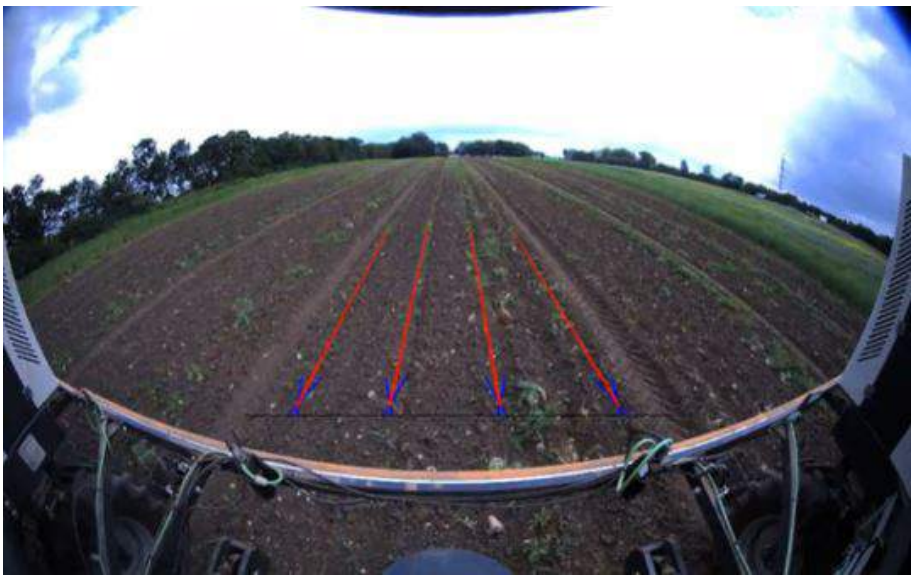


Figure 2: Plant row guidance being tested on Robotti

5 Sensor information fusion

5.1 Robustness and Adaptivity via Multimodal Feature Fusion Framework for Control

5.1.1 Introduction and objectives

The robot, while performing a specific task, often has access to multiple different data sources from its sensors. Such sensors include RGB cameras, Lidars, force feedback sensors, microphones, infrared sensors and more. As different modalities vary greatly in their properties, it is not trivial to meaningfully combine them, although the importance and benefits of fusion in environmental perception have been long established [16]. We specifically focus on robot manipulation tasks, as they both offer a significant challenge and are essential to the majority of the robotic applications [102], including the core use cases of OpenDR, such as industrial robotics and healthcare robotics.

Our ongoing contribution in OpenDR has produced a multimodal feature fusion framework, based on a generalization of approaches taken by Lee et al. [112, 113]. Our formulation takes advantage of representation learning to separate the training of the feature encoders and the fusion module from the training of the robotic RL-driven controller. This significantly increases the data efficiency of training the entire pipeline, particularly reducing the utilization of the robot itself. There are also indications that pretrained feature encoders can generalize between different manipulation tasks, which allows further savings on computation. Moreover, the use of self-supervised task-oriented objectives reduces or negates the need for expert-labeled training data, which could be a significant bottleneck otherwise. The framework is flexible with respect to the number and type of the input modalities, as long as appropriate feature extractors are specified. The structure of the framework has been described in more detail in D4.1.

We previously defined multiple objectives for the framework to fulfill, which are not inherent to its basic design. Specifically:

- robustness to damaged or incomplete inputs
- producing compact and lightweight representations
- compactness of the model architectures

To continue the development of the framework and to achieve these objectives, we now incorporate the following extensions to our approach:

- cross-modal compensation for robustness
- neural architecture search on fusion architectures for efficiency and compactness

The following sections elaborate on the details of these extensions, describe the existing related methods and our own ongoing work.

5.1.2 Summary of state of the art

Robustness to damaged or incomplete outputs is a highly desirable property for any practical multimodal application. Input corruption can happen as a result of sensor malfunction, damage

to the robot, harsh environmental conditions, etc. There are numerous works addressing this issue from different points of view, of which we consider some representative examples.

Cross-modality prediction is an approach where the information of some modalities can be inferred from the other modalities. For example, the tactile properties of an object can be estimated from its visual appearance, which would allow for the multimodal system to operate with limited or missing tactile input. Works by Li et al. [118] and Takahashi and Tan [204] follow this exact scenario, while Liu et al. [121] and Wu and Goodman [235] assume a more general formulation. However, the limitation of such approaches is that they assume knowledge of which modality is compromised, and utilize models that are trained between a given pair of modalities. This approach is not only ill-suited for larger number of modalities, it also does not allow for detection of which input is faulty, which is essential in practice.

Out-of-distribution detection (ODD) allows for the detection of input failures, which is highly relevant for many robot control tasks and particularly autonomous driving [91]. Some works, such as Filos et al. [59] and McAllister et al. [133], describe solutions that allow for the corrupted input to be partially compensated. However, most ODD works do not take advantage of multiple modalities, basing their compensation instead on the same signal or resorting to querying the human expert.

Overall, while there is a significant corpus of work on input robustness, these works are mainly limited to unimodal cases or fixed failure scenarios for specific modalities. As our framework is input-agnostic, we have to take advantage of the latent representation to make the input out-of-distribution detection and the subsequent compensation/reconstruction possible without advance knowledge of the failure case.

Neural architecture search (NAS) is a problem of discovering the most suitable neural architecture for a given task, given a set of criteria, such as task-specific performance, inference time, memory footprint, etc. NAS allows for the automated design of highly performing structures while relying significantly less on human expertise, which makes it a promising tool in exploring novel problem formulations and solution spaces. NAS is applicable across the entire range of deep learning systems and is seeing use in a plethora of fields. More details on NAS as a whole can be found in the following reviews: [54, 177]. We limit our consideration here to a few key works related to multimodal fusion.

Multimodal Fusion Architecture Search (MFAS) [165] considers two modalities that are originally independently processed by multilayer neural networks. The fusion is achieved by constructing a bimodal fusion network, every layer (or block) of which takes as inputs the intermediate activations from both unimodal networks (as well as the output of its own previous layer, if present). The search process optimizes the choice of points from which the unimodal features are taken, as well as the operations applied by the fusion network. During the search the unimodal networks' weights are kept frozen to save on computation, but the entire system is fine-tuned once the architecture is finalized. The search itself is performed by progressively unrolling the fusion network complexity, driven by temperature-based sampling (to achieve gradual transition from exploration to exploitation over the algorithm running time). MFAS is comparatively efficient in terms of computation due to restricted search space, but it relies on predefined (and individually powerful) unimodal feature extractors, which severely limits its generalization and flexibility

Multimodal Fusion Architecture Search (MUFASA) [238] is a more general method which optimizes simultaneously the unimodal and the fusion branches as part of a single model. It follows the widely used block-based approach to NAS, whereupon the networks are constructed from the set of predefined structural units (blocks). Every block belonging to the unimodal

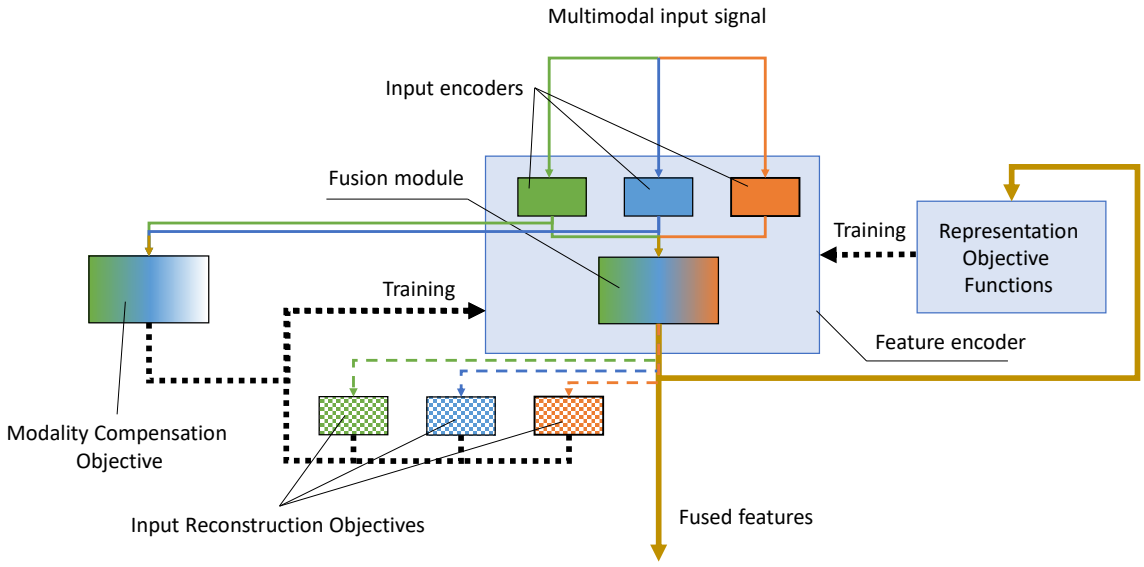


Figure 20: Multimodal fusion framework with robustness extensions.

branch of the model is only allowed to take states within the same branch as inputs, while the blocks of the fusion branch can use inputs from itself, but also any of the unimodal branches. This results in a highly flexible configuration, which is capable of reproducing early, late and hybrid fusion approaches and any mixture thereof. In fact, multiple fusion approaches per single modality are reported as the most performant. However, the flexibility of MUFASA naturally requires a more sophisticated search algorithm (evolutionary search in this case) and induces significantly higher computational expenses as a result.

As both approaches represent a trade-off between generality and complexity, we take inspiration from both as we design the search spaces for our own NAS implementation.

5.1.3 Description of work performed so far

To achieve robustness with respect to the corruption of inputs, we expand the list of objectives that drive the training of the feature encoder. In addition to previously described representation objectives, input reconstruction and modality compensation are added. The structure of the framework with incorporated robustness extensions is shown on Figure 20. Note that only the feature encoder is depicted, as the data flow between the RL controller and the robot remain unchanged.

Input reconstruction objectives measure how well the individual inputs may be reproduced from the fused features. These objectives may have limited utility when it comes to learning an efficient task-specific latent representation, but they do allow for a straightforward ODD to be performed. During inference the reproduced values can be compared against actual inputs, and a significant reconstruction error would then indicate the corruption of the corresponding modality. It is important to keep in mind that as the reconstruction is performed from the fused features, one damaged input is likely to cause the error increase in every other input. Comparing relative error values and carefully setting thresholds is therefore necessary to avoid mistakes.

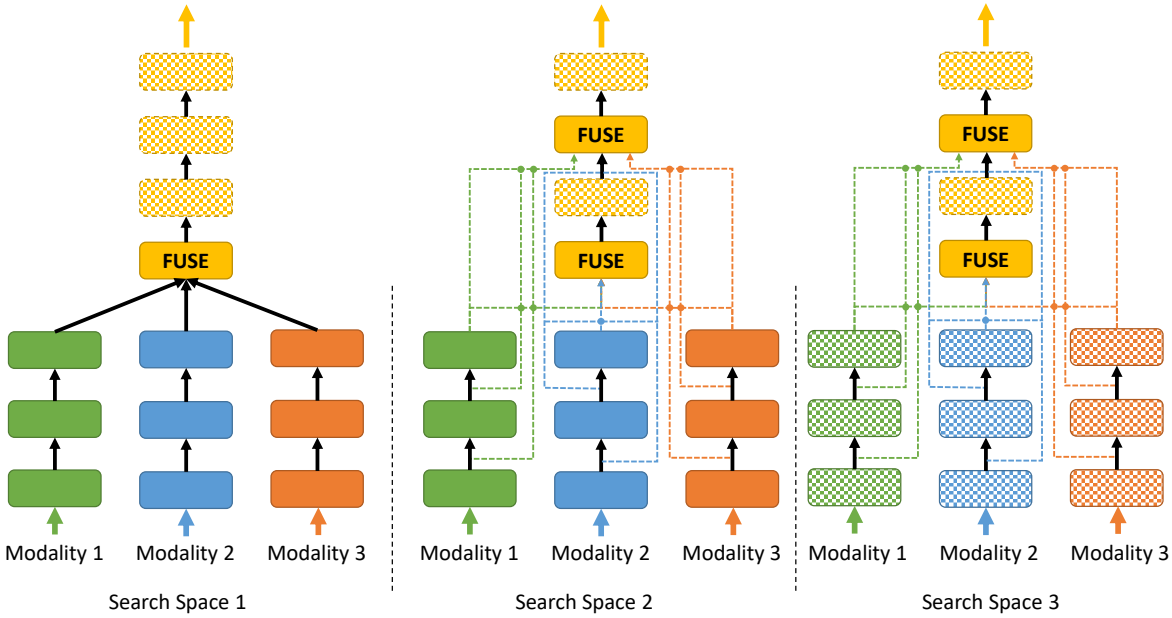


Figure 21: Neural architecture search spaces for multimodal feature fusion.

Modality compensation objective guides the latent representation to preserve its structure under the effect of corrupted inputs. On every step of the encoder training we can compute a representation where one of the input modalities is randomly dropped. We can then define an objective function by computing a difference (such as L2-norm) between the reduced and the original representations. By minimizing such a difference we gradually train the representation to compensate for the absence of individual inputs. Having thus trained the feature encoder, we now have an option to exclude the corrupted modality entirely once it's detected to be out-of-distribution, allowing the system to continue operation. This approach can be further extended to yield the relative importance of modalities, which would allow us e.g. to not query certain sensors unless absolutely necessary, reducing the overall redundancy and increasing efficiency. However, this development is left for future work.

The neural architecture search can be incorporated in the framework with or without robustness extensions. The goal of the search is to discover fusion architectures that would offer a better combination of accuracy of computational efficiency. We use the internally developed search algorithm called ASED (Architecture Search based on Estimation of Distributions) [144], as it was demonstrated to be capable of identifying small and efficient architectures. For the sake of brevity we only discuss the task-specific aspects here and we refer the reader to the original paper [144] for the more detailed description of the core method.

Figure 21 illustrates the search spaces designed for the task. Rectangular "Units" shown on this Figure can be either individual neural layers or complex blocks, as the ASED algorithm, while originally designed for layers, in fact supports both options. Solid colors denote fixed elements of the structure, while dashed lines and patterns show the elements optimized via the search. FUSE operation here can refer to one of the basic combination operations, such as concatenation, addition or multiplication. As the illustration shows, different search spaces offer trade-offs between the flexibility and complexity. Search space 1 keeps the unimodal branches of the model (individual feature extractors) fixed, and optimizes only the operations that follow

the fusion. Being the most restricted, this is the simplest of formulations and the least general one. Search space 2 is patterned after the work of Pérez-Rúa et al. [165], where modality concatenation is allowed before every layer of the fusion network, with the intermediate unimodal features also being accessible. Finally, search space 3 allows also unimodal branches to learn layer/block operations, leaving only some of the information flow directions to be fixed (sequential connections are never removed within each branch). This search space is naturally the largest and the most difficult to optimize, but the flexibility it offers can allow for very unconventional architectures to be discovered.

At the core of neural architecture search is the sampling and evaluation of the candidate models from the design space, with the results of this evaluation guiding the following iterations. The candidate ranking can be defined over both the performance with respect to the target objectives (defined on the framework level) and the computational complexity metrics (such as inference time, number of operations, memory footprint). This allows for the exploration over a range of architectures with differing properties. While the neural architecture search itself is a resource-intensive process, it is performed offline and on non-embedded hardware. The resultant models, however, can be consequently trained and deployed on the devices and environments targeted by OpenDR, where their advantages can be fully utilized. As multimodal solutions overall tend to be more computationally demanding compared to unimodal ones (for obvious reasons), any savings that can be achieved by architectural optimization offer significant practical benefits.

5.1.4 Training data and performance evaluation

The evaluation of the core framework and its performance indicators is ongoing. Appropriate simulation environments (in Webots) are in development, in collaboration with WP5 Deep Robot Action and Decision Making. Deployment and evaluation on the real robotic arm are planned to follow. These scenarios would allow for qualitative and quantitative evaluation of the framework for control, measuring success rates for different objects, sensor inputs, manipulation tasks, environments, etc. Generalization of trained feature encoders under different task parameters is also to be investigated. Additionally, in the recent months the multimodal representation learning benchmark was proposed [119], allowing the multimodal fusion algorithms to be evaluated in a standardized way. We intend to study the performance of our methods on this benchmark within the next reporting period.

The current extensions of the framework can be evaluated without significant modification to the environments or collection of the additional data. The robustness with respect to inputs is to be evaluated by reporting the success rates of manipulation experiments with different modalities being absent or corrupted. The neural architecture search is to be evaluated by the standard success rate criteria, while conducting the comparison between the discovered architectures of varying complexities.

5.2 Multi-modal Object Detection in Harsh Lighting Conditions

5.2.1 Introduction and objectives

Human visual object recognition is often rapid, effortless and largely viewpoint or object orientation independent [70]. However, with the advent of deep neural networks, computer vision algorithms have achieved unprecedented performance and even surpassed human capabilities

on tasks including image classification, face identification and object recognition [193]. Despite the accuracy of deep neural networks, their generalisation property across changes in the input distribution e.g., illumination changes and harsh conditions, is not established yet. In a behavioural comparison of humans and well-known deep neural architectures like ResNet-152 [76], classification performance of neural networks seems to decline rapidly with decreasing signal-to-noise ratio under image distortions [70, 71]. In the context of object detection, multi-sensor configurations are known to provide redundancy and often enhance performance of the detection algorithms. Moreover, efficient sensor fusion strategies minimize uncertainties, increase reliability and are crucial in achieving robustness against asymmetric sensor failures. Increasing number of sensors might enhance the performance of detection algorithms, however this comes with a considerable computational/energy cost. This is often not desirable in mobile robotic systems which typically have constraints in terms of computational power and battery consumption. In such cases, intelligent choice/combination of sensors is critical. In the following, we will summarize works on a multi-modal fusion strategy and a data augmentation method that are proposed for object detection in harsh lighting conditions:

- [131] Mazhar, O., Babuska, R., & Kober, J. (2021). GEM: Glare or Gloom, I Can Still See You - End-to-End Multi-Modal Object Detection. IEEE Robotics and Automation Letters, 6(4), 6321-6328. https://doi.org/10.1109/LRA.2021.3093871
- [132] Mazhar, O., & Kober, J. (2021). Random Shadows and Highlights: A new data augmentation method for extreme lighting conditions. arXiv preprint arXiv:2101.05361.

The corresponding papers can be found in Appendices 7.11 and 7.12, respectively.

5.2.2 Summary of state of the art

In this section, we first review deep learning based object detection strategies, followed by a discussion on existing methods for multi-modal fusion schemes in relevant tasks.

Deep Learning Based Object Detection
Detailed literature surveys for deep learning based object detectors have been published in [89, 123]. Here we briefly discuss some of the well-known object detection strategies. Typically, object detectors can be classified into two types namely, two-stage and single-stage object detectors.

Two-stage object detection Two-stage object detectors exploits a region proposal network (RPN) in their first stage with an input image being fed into the backbone of convolutional neural network. RPN ranks region boxes alias anchors, and proposes the ones that most likely contain objects as candidate boxes. In the second stage, the features are extracted by region-of-interest pooling (RoIPool) operation from each candidate box. These features are then utilized for bounding-box regression and classification task. The classification and localization accuracy of two-stage detectors has been higher as compared to well-known single stage detectors, however because of an extra region proposal network, the inference speed is low. Typically, models in the Region-based Convolutional Neural Networks (R-CNN) family fall into this class of object detectors. Fast R-CNN, Faster R-CNN and Region-based Fully Convolutional Network (RFCN) are among the popular methods adopting this approach.

Single-stage object detection

Single-stage detectors propose predicted boxes from input images in one forward pass directly without region proposal step. Thus, this type of object detectors are time efficient and can be utilized for real-time operations. You Only Look Once (YOLO) and Single-shot mutilbox detector (SSD) have been the prominent representatives of single-stage object detectors. YOLO and its subsequent versions, divide the input image into grid cells. For each object that is present in the image, one grid cell is responsible for predicting it. Each grid cell predicts several bounding boxes and class probabilities. Each bounding box prediction has a confidence score associated with it. Only the bounding boxes with the confidence score higher than the threshold are considered. To select a single entity out of several overlapping bounding boxes detected by the network, YOLO like most other object detectors, employs a hand-crafted strategy known as non-maximum suppression (NMS). NMS selects the bounding box with highest confidence score and removes the overlapping ones with intersection-over-union (IoU) values higher than the threshold. This step is often additionally required in generating proposals in two-stage detectors while assigning anchor boxes to the ground-truth labels. SSD uses a convolutional neural network as a backbone feature extractor to extract feature maps at multiple scales/layers. Each layer is responsible to produce proposals based on default bounding boxes associated to each feature map similar to anchor boxes in Faster-RCNN. The default boxes are pre-selected manually to cover a wide spectrum of real-world objects. This allows classification of objects of various sizes. For each default box, the network learns an offset for its position and a scaling factor for its size.

Lately, an end-to-end object detection strategy is proposed in [25] that eliminates the need for hand-crafted components like anchor boxes and non-maximum suppression. It employs transformers in an encoder-decoder fashion, combined with a set-based Hungarian loss which forces unique predictions for each ground truth bounding box by utilizing bipartite matching. Transformers consist of multi-headed self-attention modules and are permutation-invariant. Thus they are supplemented with fixed positional encodings which are added to the encoder input. The decoder transforms the learned positional encodings, which are unique for each object query, into an output embedding. These are subsequently decoded into bounding box coordinates and class labels by a feed forward neural network. Object detection strategy in [25] is among the pioneering works that have exploited transformers in the image domain. However, technically it is still a combination of CNN and self-attention. Lately, the authors in [50] directly employ a standard transformer in image domain. They interpret an image as a sequence of patches and process it by a transformer encoder for image recognition task. Promising results were obtained when trained on large scale datasets e.g., ImageNet, ImageNet-21k and JFT. The inherent non-sequential architecture of transformers allows parallelization of models. They are computationally efficient thus require substantially fewer computational resources to train. Therefore, we opted to build upon the methodology of [25] for our multi-modal object detector for harsh lighting conditions.

Sensor Fusion

Sensor fusion strategies can be roughly divided into three types according to the level of abstraction where fusion is performed or in which order transformations are applied compared to features combination namely low-level, medium level and high-level fusion [66]. In low-level or early fusion, raw information from each sensor is fused at pixel level e.g., disparity maps in stereo-vision cameras [229, 167]. Further examples of early fusion include strategies where multi-modal data e.g., point cloud from LiDAR and camera images are stacked in depth and

passed through a classifier to predict object classes, or they complement each other to generate region-of-interest patches for object detection [259]. In medium-level, a set of features is extracted for each modality in a pre-processing stage while different approaches [154] are exploited to fuse the extracted features. These include multiplication which computes element-wise product of feature maps, concatenation where feature maps are stacked without any blend, and convolution where stacked feature maps are convolved with 1×1 filter size. Other methods employ a combination of two fusing methods e.g., convolution of stacked feature maps followed by several fully connected layers with dropout regularization [18]. Such fusion methodologies are otherwise known as late-fusion. In high-level fusion or ensemble learning methods, predictions are obtained individually for each modality and the learnt scores or hypotheses are subsequently combined via different methods e.g., weighted majority votes [2, 147, 249]. Deep fusion or cross fusion [23] is another type of fusion strategy which repeatedly combines inputs, then transforms them individually. In each repetition, the transformation learns different features [13].

Multi-Modal Object Detection

Most of the efforts on multi-modal object detection in the literature are focused on pedestrian or vehicles detection in automotive context. In terms of sensors combination choice, fusion strategies are typically proposed for camera-LiDAR, camera-radar and camera-radar-LiDAR setups. Lately, entropy steered multi-modal deep fusion architecture has been proposed in [13] following the single-shot object detector framework for adverse weather conditions. Fusion is performed for multiple sensors including RGB camera, gated camera (NIR band), LiDAR and radar. For LiDAR, instead of employing BeV projection or point cloud representation, the authors encode depth, height and pulse-intensity on an image plane. Moreover, radar output is also projected onto an image plane parallel to the image horizontal dimension while considering radar output invariant along the vertical image axis, the data is replicated across horizontal image axis. A modified VGG architecture is utilized for feature extraction, while features are exchanged among all modalities driven by sensor entropy, after each pooling operation. Fused feature maps from the last 6 layers of the features extractors are passed to SSD bounding box regressor and classification layers. In [45], the authors propose a pseudo multi-modal object detector from thermal IR images in Faster-RCNN setting. They exploited I2I translation networks namely CycleGAN and UNIT to transform thermal images from FLIR ADAS and KAIST datasets to RGB domain. Inputs from each modality are passed through CNN feature extractors with ResNet blocks. The output feature volumes are then stacked and passed through 1×1 convolution for combined feature learning. Subsequently, the output of 1×1 convolution is directly passed to RPN, bounding box regressor and classification head.

Here we also discuss some fusion strategies which are relevant to our work but originally proposed for applications other than object detection. Two sensor fusion strategies are proposed in [34] for Visual-Inertial Odometry, namely soft fusion and hard fusion. Soft fusion method is implemented in a deterministic fashion that learns soft masks and assign weights to each element in the feature vector. Hard fusion exploits gumbel-softmax resampling strategy which assigns hard masks resampled from a discrete stochastic distribution, parameterized by a variable conditioned on feature vectors. This can be viewed as a switcher for each component in the feature map. The performance of the fusion strategies is studied through sensor data degradation e.g., vision degradation through occlusions, gaussian noise and missing images and similarly for IMU degradation and cross-sensor degradation. In [93], the authors propose a sensor fusion strategy for RGB and depth images to steer a self-driving vehicle. A semantic

segmentation network is trained by using RGB images by employing an encoder-decoder architecture without skip connections. Once it is trained, the latent semantic vector obtained after passing the RGB image through the encoder, is fused with the depth features obtained through a separate CNN features extractor. The fusion architecture proposed by [93] is similar to the gating mechanism driven by the learned scalar weights presented in [164]. To imitate sensor failure cases, the network is trained partially with corrupted data. More specifically, random RGB images are fed into the encoder to obtain irrelevant semantic features to mimic RGB camera failures while blank images are passed through depth features extractor module to simulate failure of depth sensor. In this report, we propose deterministic weighted fusion strategies that enable robust and sensor-aware object detection in harsh lighting conditions. We also propose a selective fusion scheme by exploiting the Gumbel-softmax trick similar to [34]. We perform extensive experiments and show that our methods have surpassed the state-of-the-art results on FLIR Thermal Dataset 4.

5.2.3 Description of work performed so far

Multi-Modal Object Detection
In the paper that can be found in Appendix 7.11, deterministic and stochastic sensor-aware feature fusion strategies are proposed. Deep neural networks designed for vision tasks are often prone to failure when they encounter environmental conditions not covered by the training data. Single-modal strategies are insufficient when the sensor fails to acquire information due to malfunction or its design limitations. Multi-sensor configurations are known to provide redundancy, increase reliability, and are crucial in achieving robustness against asymmetric sensor failures. To address the issue of changing lighting conditions and asymmetric sensor degradation in object detection, we develop a multi-modal 2D object detector, and propose deterministic and stochastic sensor-aware feature fusion strategies. The proposed fusion mechanisms are driven by the estimated sensor measurement reliability values/weights. Reliable object detection in harsh lighting conditions is essential for applications such as self-driving vehicles and human-robot interaction. We also propose a new "r-blended" hybrid depth modality for RGB-D sensors. Through extensive experimentation, we show that the proposed strategies outperform the existing state-of-the-art methods on the FLIR-Thermal dataset, and obtain promising results on the SUNRGB-D dataset. We additionally record a new RGB-Infra indoor dataset, namely L515-Indoors, and demonstrate that the proposed object detection methodologies are highly effective for a variety of lighting conditions.

Data augmentation technique for robustness against lighting perturbations
In the paper that can be found in Appendix 7.12, we propose a new data augmentation method, Random Shadows and Highlights (RSH) to acquire robustness against lighting perturbations. Our method creates random shadows and highlights on images, thus challenging the neural network during the learning process such that it acquires immunity against such input corruptions in real world applications. It is a parameter-learning free method which can be integrated into most vision related learning applications effortlessly. With extensive experimentation, we demonstrate that RSH not only increases the robustness of the models against lighting perturbations, but also reduces over-fitting significantly. Thus RSH should be considered essential for all vision related learning systems.

5.2.4 Performance evaluation

In this section, performance evaluation of the multi-modal object detector and the data augmentation method will be discussed briefly. For more extensive performance evaluation of the work on multi-modal object detection and on a data augmentation technique to improve robustness against lighting perturbations, we refer the reader to the results presented in Appendices 7.11 and 7.12, respectively.

Performance evaluation of the multi-modal object detector on the FLIR-Thermal dataset is shown in Table 1 in 7.11. We show Average Precision (AP) values at Intersection over Union(IoU) of 0.5 for each dominant class, while the mean Average Precision (mAP) is also estimated with and without lighting perturbations. These lighting corruptions are introduced by creating Random Shadows and Highlights (RSH) on the test RGB images. The evaluation with lighting perturbation is performed for 10 trials in all experiments, while the average of the obtained mAP is shown in the table. The results are compared with the single modality object detector, the multi-modal baseline fusion networks, and the existing state-of-the-art methods on this dataset. In the baselines, the features from the backbones are fused in two configurations: averaged and concatenated, without any weighing or re-sampling mechanism. Additionally, we compared the performance of simple averaging fusion as the baseline method (SSD-BL) and a weighted averaging fusion scheme (SSD-WA) on the FLIR-Thermal dataset. It is clear from the evaluation results, that our proposed methodologies, i.e., g_{sa} , g_{sc}, g_{ma}, g_{mc} and g_{sf}, outperform the previously reported results on this dataset.

The performances of the RSH data augmentation strategies on Tiny-ImageNet are evaluated at different activation probabilities p, ranging from 0.0 to 1.0 with a step size of 0.1. Table 1 in 7.12 presents the results of these experiments at $p = 0.5$ and $p = 1$ only. For comparison, models are trained with Random Gamma Correction (RGC), Random Color Jitter (RCJ) and Random Disk Illumination (RDI) as well. Apart from train and test error computation, performance is measured with lighting perturbations as well by applying RSH in the test set with $p = 1$. Moreover, the Train-Test Difference (TTD) is also computed to quantify over-fitting of the models. Negative TTD implies over-regularization which is not desired. The results show that for lighting perturbations at $p = 1$ on the test set, the performance of the model trained with RSH is increased by approximately 25% at both $p = 0.5$ and $p = 1$ when compared to RGC which performed best among other strategies. Furthermore, we estimate the Train-Test Difference (TTD) to measure over-fitting for cases with and without lighting corruption. In addition to the increased robustness, our RSH augmentation method proves to have reduced over-fitting significantly as well.

5.2.5 Future Work

The paper on multi-modal object detection in harsh lighting condition [131] is accepted for publication in IEEE Robotics and Automation Letters. After submitting *Random shadows and highlights: a new data augmentation method for extreme lighting conditions*, we received feedback from the reviewers that the method is promising, but that the evaluation was not yet convincing enough. Their main point of criticism was that the method was not yet evaluated on a dataset with harsh lighting conditions. Since such a dataset is not publicly available, we are considering to create an artificial dataset using 3D rendering software. Furthermore, we are also considering to extend the method to a more general class of shapes, i.e. non-self-intersecting polygons.

6 Conclusions

In the second year of the project, the consortium carried out a series of activities following the overall objectives of the project. In the context of work package 4, we focused on objective O1, providing a modular, open and non-proprietary toolkit for core robotic functionalities enabled by lightweight deep learning, and objective O2, leveraging AI and cognition in robotics to go from perception to action.

ALU-FR worked towards objective O2 by proposing two new methods for T4.3, namely Efficient LiDAR Panoptic Segmentation (EfficientLPS) [195] and an extension of CMRNet++ [27]. The EfficientLPS architecture combines a shared CNN-based backbone with two task-specific heads for semantic and instance segmentation, respectively, and a final panoptic fusion module. It operates on a 2D representation of LiDAR point clouds by projecting them to a multi-channel image comprising range, intensity, and (x,y,z) coordinates. The extended CMRNet++ advances the state-of-the-art of monocular vision-based localization in 3D LiDAR maps by formulating the task independent of the camera parameters. It further shows superior performance compared to other methods by effective generalization to unseen places, i.e., allowing for accurate localization without the need to retrain.

AUTH worked towards O1a by proposing an online distillation method for efficiently training lightweight models for robotics applications, as well as a self-distillation method for training fast, yet accurate models. Also, AUTH worked towards O1a by proposing ways for improving the performance of BiseNet semantic segmentation model both in terms of deployment speed and segmentation accuracy, considering binary segmentation problems. In addition, AUTH worked towards the same objective by developing a supervised hashing methodology that enables retrieving objects similar to a query image, minimizing inference and query time and providing a useful tool both for few-shot learning, as well as for mining training data when few training data exist. Furthermore, AUTH worked towards O1b by incorporating label embedding criteria into the learning objective of lightweight neural networks, capable of running at real-time on high-resolution inputs on embedded devices. Also, AUTH worked towards addressing O2a by proposing a pseudo-active sensory refinement method that works by applying a number of neural transformation layers on the sensor data, allowing for improving the perception accuracy of DL models without having to re-acquire sensor data. Also, AUTH worked towards developing end-to-end active tracking approaches.

AU worked towards objective O1 by proposing new methods for 3D object detection and tracking. A new method for 3D object detection uses spherical projection images for a natural ordering of point clouds and uses a heatmap-based approach to find central object points and regress a final 3D bounding box from features of this point. A single object 3D tracking method was developed that uses a Siamese PointPillars architecture to find an object in the generated pseudo-image space and then regress its 3D bounding box based on the features of the found region. Furthermore, AU introduced a new type of neural networks, called Variational Neural Networks, with an inherent ability to express uncertainty. This uncertainty has higher quality than other approaches, such as Monte Carlo Dropout [63] and Bayes By Backprop [14] methods, according to tests on Epistemic Neural Networks framework [152]. The uncertainty, estimated by this network, could be an indicator for the active perception method to change position for better predictions.

TUD worked towards objective O2c by developing sensor fusion strategies for multi-modal object detection. Each sensor modality is passed through an individual sensor-specific feature extractor backbone. The obtained feature volumes are subsequently utilized to learn respec-

tive weights by specialized FFNs in a self-supervised fashion. The feature volumes are subsequently counterbalanced individually, guided by the learned weights, before they are passed to the bounding box regressor and object classification head. This allows the network to intelligently adjusts its focus on the modality that contains the most relevant information for the task at hand. Furthermore, a selective feature fusion strategy driven by gumbel-softmax trick is also proposed. This allows to obtain dominant discrete sample from the feature maps of each sensor modality in an end-to-end fashion. TUD also proposed a lightweight learning free data augmentation method (RSH) which creates random highlights and shadows in the input images. The proposed fusion network(s) learns to handle asymmetric data corruption or sensor failures, guided by RSH which mimics harsh-lighting/sensor-failure conditions. The proposed multimodal object detector is build upon an end-to-end object detection methodology that exploits transformers in an encoder-decoder fashion. It converges fairly quickly due to transformers thus reducing training time and optimizing computational resources for large-scale datasets.

TAU has contributed to objective O1 by developing a knowledge distillation methodology [212]. The method performs distillation by means of sparse representation learning, i.e., by encoding each pixel of the feature maps in a sparse domain and using these sparse representations as the source of supervision rather than directly utilizing the intermediate feature maps or soft labels of the. This allows to perform distillation efficiently from strong teacher to a lightweight student as shown by experiments on image classification in a range of datasets. In addition, TAU has worked towards the given objectives by investigating the relationships between properties of learnt feature spaces in deep neural networks and performances of corresponding models in image classification tasks [40]. Discovered properties will enable development of methods in further work of the project. Finally, TAU expanded on their multimodal sensor fusion framework to incorporate robustness with respect to damaged or missing inputs. Optimization of the fusion models via neural architecture search offers further improvements in both task-specific capabilities and computational efficiency.

AGI worked towards objective O2c, to enable the field robot to be able to navigate based on the actual position of the plant rows and thus its environment.

The approaches and methods developed during the second year of the project are well aligned with the project goals, complement the work done in the first year of the project, and set new state of the art performance in individual areas. Most of these methodologies will be integrated to the OpenDR toolkit and complement the already integrated tools. Given the achieved progress we are confident to achieve the overall goals of the project in the targeted time frame and provide the robotics community with a useful toolkit for deep perception, cognition and decision making.

References

[1] S. Abe. On invariance of support vector machines. In *Proceedings of the 4th international conference on intelligent data engineering and automated learning*, 2003. 56

[2] M. Aeberhard, S. Paul, N. Kaempchen, and T. Bertram. Object existence probability fusion using dempster-shafer theory in a high-level sensor data fusion architecture. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 770–775. IEEE, 2011. 87

[3] M. Afifi and M. S. Brown. What else can fool deep learning? addressing color constancy errors on deep neural network performance. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 243–252, 2019. 14

[4] I. Alonso, L. Riazuelo, and A. C. Murillo. Mininet: An efficient semantic segmentation convnet for real-time robotic applications. *IEEE Transactions on Robotics*, 2020. 31

[5] R. Anil, G. Pereyra, A. T. Passos, R. Ormandi, G. Dahl, and G. Hinton. Large scale distributed neural network training through online distillation. 2018. 19

[6] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Proceedings of the Advances in Neural Information Processing Systems 27*, pages 2654–2662. 2014. 26

[7] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018. 14

[8] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 71, 74, 75

[9] J. Behley and C. Stachniss. Efficient surfel-based slam using 3d laser range data in urban environments. In *Robotics: Science and Systems*, volume 2018, 2018. 76

[10] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006. 22

[11] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. *arXiv:1606.09549*, 2016. 9, 63

[12] S. Bianco, C. Cusano, and R. Schettini. Color constancy using cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 81–89, 2015. 14

[13] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11682–11692, 2020. 87

[14] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv:1505.05424*, 2015. 8, 49, 90

[15] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. 2015. 47

[16] W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intelligenz*, 29(4):353–362, 2015. 80

[17] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard. Robot localization in floor plans using a room layout edge extraction network. *arXiv preprint arXiv:1903.01804*, 2019. 76

[18] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara. Poseidon: Face-from-depth for driver pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4661–4670, 2017. 87

[19] T. Bozinis, N. Passalis, and A. Tefas. Improving visual question answering using active perception on static images. In *Proceedings of the International Conference on Pattern Recognition*, pages 879–884, 2021. 14

[20] E. Brachmann and C. Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4654–4662, 2018. 76, 78

[21] W. Burgard, D. Fox, and S. Thrun. Probabilistic robotics. *The MIT Press*, 2005. 10

[22] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 61, 71, 74, 75

[23] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde. Lidar–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131, 2019. 87

[24] Y. Cao, M. Long, B. Liu, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018. 45

[25] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 86

[26] T. Caselitz, B. Steder, M. Ruhnke, and W. Burgard. Monocular camera localization in 3d lidar maps. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1926–1931. IEEE, 2016. 76

[27] D. Cattaneo, D. G. Sorrenti, and A. Valada. CMRNet++: Map and camera agnostic monocular visual localization in LiDAR maps. *IEEE International Conference on Robotics and Automation (ICRA) Workshop on Emerging Learning and Algorithmic Methods for Data Association in Robotics*, 2020. 10, 13, 76, 77, 78, 90

[28] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard. Cmrnet: Camera to lidar-map registration. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1283–1289. IEEE, 2019. 76, 77

[29] T. Cavallari, S. Golodetz, N. A. Lord, J. Valentin, L. Di Stefano, and P. H. Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4457–4466, 2017. 76

[30] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 76, 77

[31] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3552–3561, 2019. 32

[32] D. S. Chaplot, E. Parisotto, and R. Salakhutdinov. Active neural localization. In *International Conference on Learning Representations*, 2018. 10

[33] T. Charnock, L. Perreault-Levasseur, and F. Lanusse. Bayesian neural networks. *arXiv:2006.01490*, 2020. 46

[34] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni. Selective sensor fusion for neural visual-inertial odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10542–10551, 2019. 87, 88

[35] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen. Online knowledge distillation with diverse peers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3430–3437, 2020. 19, 25, 31

[36] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*, 2019. 32

[37] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 18, 25

[38] S.-T. Chiu. Bandwidth selection for kernel density estimation. *The Annals of Statistics*, pages 1883–1905, 1991. 23

[39] J. H. Cho and B. Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4794–4802, 2019. 55, 56

[40] K. Chumachenko, J. Laakom, Firas andand Raitoharju, A. Iosifidis, and M. Gabbouj. Rethinking compactness in deep neural networks. *ICML Workshops: Overparameterization, Pitfalls and Opportunities 2021*, 2021. 57, 58, 59, 91, 222

[41] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 33

[42] T. Cortinhal, G. Tzelepis, and E. E. Aksoy. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In *International Symposium on Visual Computing*, pages 207–222. Springer, 2020. 71

[43] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. Annual Symposium on Computational Geometry*, pages 253–262, 2004. 40

[44] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 50

[45] C. Devaguptapu, N. Akolekar, M. M Sharma, and V. N Balasubramanian. Borrow from anywhere: Pseudo multi-modal object detection in thermal imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 87

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 49, 50

[47] S. M. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440, 2012. 66

[48] A. Dewan, G. L. Oliveira, and W. Burgard. Deep semantic classification for 3d lidar data. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3544–3549. IEEE, 2017. 71

[49] Z. Ding, X. Han, and M. Niethammer. Votenet: A deep learning label fusion method for multi-atlas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 202–210. Springer, 2019. 71

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 86

[51] K. E. Dungan and L. C. Potter. Classifying vehicles in wide-angle radar using pyramid match hashing. *IEEE Journal of Selected Topics in Signal Processing*, 5(3):577–591, 2011. 40

[52] M. Ebner. *Color constancy*, volume 7. John Wiley & Sons, 2007. 14

[53] N. El Gayar, F. Schwenker, and G. Palm. A study of the robustness of knn classifiers trained using soft labels. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 67–80. Springer, 2006. 36

[54] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2019. 81

[55] T. Emara, H. E. Abd El Munim, and H. M. Abbas. Liteseg: A novel lightweight convnet for semantic segmentation. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7. IEEE, 2019. 32

[56] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 15

[57] Z. Fang, S. Zhou, Y. Cui, and S. Scherer. 3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud. *IEEE Sensors Journal*, 21(4), 2021. 63

[58] H. M. Fayek, M. Lech, and L. Cavedon. Modeling subjectiveness in emotion recognition with deep neural networks: Ensembles vs soft labels. In *2016 international joint conference on neural networks (IJCNN)*, pages 566–570. IEEE, 2016. 36

[59] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *International Conference on Machine Learning*, pages 3145–3153, 2020. 81

[60] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 77

[61] N. I. Fisher. *Statistical analysis of circular data*. Cambridge University Press, 1995. 43

[62] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar. Born again neural networks. In *ICML*, 2018. 25

[63] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2016. 8, 49, 90

[64] M. Gao, Y. Shen, Q. Li, J. Yan, L. Wan, D. Lin, C. C. Loy, and X. Tang. An embarrassingly simple approach for knowledge distillation. *arXiv preprint arXiv:1812.01819*, 2018. 51

[65] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 642–651, 2019. 72

[66] F. Garcia, D. Martin, A. De La Escalera, and J. M. Armingol. Sensor fusion methodology for vehicle detection. *IEEE Intelligent Transportation Systems Magazine*, 9(1):123–133, 2017. 86

[67] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017. 31

[68] S. Gasperini, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari. Panoster: End-to-end panoptic segmentation of lidar point clouds. *IEEE Robotics and Automation Letters*, 6(2):3216–3223, 2021. 72

[69] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 76, 77

[70] R. Geirhos, D. H. J. Janssen, H. H. Schütt, J. Rauber, M. Bethge, and F. A. Wichmann. Comparing deep neural networks against humans: object recognition when the signal gets weaker. *CoRR*, abs/1706.06969, 2017. 84, 85

[71] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann. Generalisation in humans and deep neural networks. *CoRR*, abs/1808.08750, 2018. 85

[72] S. Giancola, J. Zarzar, and B. Ghanem. Leveraging shape completion for 3d siamese tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1359–1368, 2019. 63

[73] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. 18, 24

[74] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 73

[75] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. 68

[76] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 17, 54, 85

[77] X. He, S. Bai, J. Chu, and X. Bai. An Improved Multi-View Convolutional Neural Network for 3D Object Retrieval. *IEEE Transactions on Image Processing*, 29:7917–7930, 2020. 59

[78] B. Heo, M. Lee, S. Yun, and J. Y. Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3779–3787, 2019. 51

[79] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings oft he AAAI Conference on Artificial Intelligence*, 2018. 66

[80] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 18, 25, 50, 54, 55

[81] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, L. Chen, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. *arXiv preprint arXiv:2006.14480*, 2020. 76, 77

[82] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 17, 50

[83] Y. Hu, B. Wang, and S. Lin. Fc4: Fully convolutional color constancy with confidence-weighted pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4085–4094, 2017. 14

[84] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017. 18, 36, 45, 50

[85] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017. 50

[86] J. V. Hurtado, R. Mohan, W. Burgard, and A. Valada. Mopt: Multi-object panoptic tracking. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Scalability in Autonomous Driving*, 2020. 72

[87] A. Iosifidis, A. Tefas, and I. Pitas. On the optiaml class representation in Linear Discriminant Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 24(9):1491–1497, 2013. 57

[88] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 71

[89] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019. 85

[90] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015. 10

[91] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017. 81

[92] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 76

[93] Q. Khan, T. Schön, and P. Wenzel. Towards self-supervised high level sensor fusion. *arXiv preprint arXiv:1902.04272*, 2019. 87, 88

[94] J. Kim, M. Hyun, I. Chung, and N. Kwak. Feature fusion for online mutual knowledge distillation. *CoRR*, abs/1904.09058, 2019. 19

[95] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114*, 2014. 47

[96] S. Kiranyaz, T. Ince, and M. Gabbouj. Real-time patient-specific ecg classification by 1-d convolutional neural networks. *IEEE Transactions on Biomedical Engineering*, 63(3):664–675, 2015. 49

[97] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 71

[98] D. Kochanov, F. K. Nejadasl, and O. Booij. Kprnet: Improving projection-based lidar semantic segmentation. *arXiv preprint arXiv:2007.12668*, 2020. 71

[99] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 54

[100] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009. 36, 54

[101] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 15, 17

[102] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *arXiv preprint arXiv:1907.03146*, 2020. 80

[103] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. 46

[104] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard. Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, 32(4):565–589, 2015. 76, 78

[105] F. Laakom, N. Passalis, J. Raitoharju, J. Nikkanen, A. Tefas, A. Iosifidis, and M. Gabbouj. Bag of color features for color constancy. *IEEE Transactions on Image Processing*, 29:7722–7734, 2020. 14

[106] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2015. 40

[107] x. lan, X. Zhu, and S. Gong. Knowledge distillation by on-the-fly native ensemble. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7517–7527. 2018. 19, 31

[108] X. Lan, X. Zhu, and S. Gong. Self-referenced deep learning. In *Asian Conference on Computer Vision*, pages 284–300. Springer, 2018. 19

[109] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*, 16(3):82–88, 2017. 40

[110] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 59, 63

[111] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 25, 50

[112] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May:8943–8950, 2018. 80

[113] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making Sense of Vision and Touch: Learning Multimodal Representations for Contact-Rich Tasks. *IEEE Transactions on Robotics*, 36:582–596, 2019. 80

[114] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An accurate o(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2):155, 2009. 77

[115] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. on Multimedia Computing, Communications, and Applications*, 2(1):1–19, 2006. 42

[116] B. Li, T. Zhang, and T. Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. In *Robotics: Science and Systems XII*, 2016. 59

[117] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proc. Twenty-Fifth Int. Joint Conf. on Artificial Intelligence*, pages 1711–1717, 2016. 44, 45

[118] Y. Li, J.-Y. Zhu, R. Tedrake, and A. Torralba. Connecting touch and vision via cross-modal prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10609–10618, 2019. 81

[119] P. P. Liang, Y. Lyu, X. Fan, Z. Wu, Y. Cheng, J. Wu, L. Y. Chen, P. Wu, M. A. Lee, Y. Zhu, R. Salakhutdinov, and L.-P. Morency. Multibench: Multiscale benchmarks for multimodal representation learning. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 84

[120] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela. Using synthetic data and deep networks to recognize primitive shapes for object grasping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10494–10501. IEEE, 2020. 26

[121] G.-H. Liu, A. Siravuru, S. Prabhakar, M. Veloso, and G. Kantor. Learning end-to-end multimodal sensor policies for autonomous navigation. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 249–261, 2017. 81

[122] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2064–2072, 2016. 40, 44, 45

[123] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020. 85

[124] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37, 2016. 17

[125] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai. TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. In *AAAI Conference on Artificial Intelligence*, 2020. 59, 63

[126] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang. End-to-end active object tracking via reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2018. 66

[127] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang. End-to-end active object tracking and its real-world deployment via reinforcement learning. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 66

[128] D. J. C. Mackay. Probable networks and plausible predictions — a review of practical bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505, 1995. 46

[129] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):791–804, 2011. 51

[130] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, and R. Schettini. Automated pruning for deep neural network compression. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 657–664. IEEE, 2018. 50

[131] O. Mazhar, R. Babuska, and J. Kober. Gem: Glare or gloom, i can still see you - end-to-end multi-modal object detection. *IEEE Robotics and Automation Letters*, 6(4):6321–6328, 2021. Accepted Author Manuscript. 11, 85, 89, 255

[132] O. Mazhar and J. Kober. Random shadows and highlights: A new data augmentation method for extreme lighting conditions. *CoRR*, abs/2101.05361, 2021. 11, 85, 264

[133] R. McAllister, G. Kahn, J. Clune, and S. Levine. Robustness to out-of-distribution inputs via task-aware generative uncertainty. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2083–2089, 2019. 81

[134] O. Michel. Cyberbotics ltd. webots™: professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004. 66, 67

[135] A. Milioto, J. Behley, C. McCool, and C. Stachniss. Lidar panoptic segmentation for autonomous driving. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8505–8512. IEEE, 2020. 72

[136] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. 71, 72, 74

[137] S. Mirzadeh, M. Farajtabar, A. Li, and H. Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *CoRR*, abs/1902.03393, 2019. 26

[138] A. Mitrokhin, P. Sutor, C. Fermüller, and Y. Aloimonos. Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception. *Science Robotics*, 4(30), 2019. 14

[139] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pages 1928–1937, 2016. 66

[140] R. Mohan and A. Valada. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision*, 129(5):1551–1579, 2021. 10, 72, 73, 74

[141] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65, 2009. 66, 67

[142] V. Monga, Y. Li, and Y. C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *arXiv preprint arXiv:1912.10557*, 2019. 51

[143] R. Müller, S. Kornblith, and G. Hinton. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019. 40

[144] A. Muravev, J. Raitoharju, and M. Gabbouj. Neural architecture search by estimation of network structure distributions. *IEEE Access*, 9:15304–15319, 2021. 83

[145] P. Neubert, S. Schubert, and P. Protzel. Sampling-based methods for visual navigation in 3d maps by synthesizing depth images. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2492–2498. IEEE, 2017. 76

[146] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, volume 99, pages 278–287, 1999. 66

[147] T. T. Nguyen, J. Spehr, S. Zug, and R. Kruse. Multisource fusion for robust road detection using online estimated reliabilities. *IEEE Transactions on Industrial Informatics*, 14(11):4927–4939, 2018. 87

[148] D. Ni. Dense feature fusion for online mutual knowledge distillation. In *Journal of Physics: Conference Series*, volume 1865, page 042084. IOP Publishing, 2021. 31

[149] S. I. Nikolenko. Synthetic data for deep learning. *arXiv preprint arXiv:1909.11512*, 2019. 26

[150] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 54

[151] P. Nousi and A. Tefas. Deep learning algorithms for discriminant autoencoding. *Neurocomputing*, 266:325–335, 2017. 39

[152] I. Osband, Z. Wen, M. Asghari, M. Ibrahimi, X. Lu, and B. V. Roy. Epistemic Neural Networks. 2021. 8, 48, 90

[153] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien. On-line multi-view video summarization for wireless video sensor network. *IEEE Journal of Selected Topics in Signal Processing*, 9(1):165–179, 2015. 40

[154] E. Park, X. Han, T. L. Berg, and A. C. Berg. Combining multiple sources of knowledge in deep cnns for action recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016. 87

[155] W. Park, D. Kim, Y. Lu, and M. Cho. Relational knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019. 53, 54

[156] N. Passalis and A. Tefas. Learning bag-of-features pooling for deep convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5755–5763, 2017. 15

[157] N. Passalis and A. Tefas. Neural bag-of-features learning. *Pattern Recognition*, 64:277–294, 2017. 53

[158] N. Passalis and A. Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284, 2018. 23, 51, 54

[159] N. Passalis and A. Tefas. Training lightweight deep convolutional neural networks using bag-of-features pooling. *IEEE Trans. on Neural Networks and Learning Systems*, 2018. 40

[160] N. Passalis and A. Tefas. Deep reinforcement learning for controlling frontal person close-up shooting. *Neurocomputing*, 335:37–47, 2019. 66, 68, 70

[161] N. Passalis and A. Tefas. Leveraging active perception for improving embedding-based deep face recognition. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 1–6, 2020. 14

[162] N. Passalis and A. Tefas. Deep supervised hashing using quadratic spherical mutual information for efficient image retrieval. *Signal Processing: Image Communication*, 93:116146, 2021. 41

[163] N. Passalis and A. Tefas. Pseudo-active vision for improving deep visual perception through neural sensory refinement. In *IEEE International Conference on Image Processing*, 2021. 15

[164] N. Patel, A. Choromanska, P. Krishnamurthy, and F. Khorrami. Sensor modality fusion with cnns for ugv autonomous driving in indoor environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1531–1536. IEEE, 2017. 88

[165] J.-M. Pérez-Rúa, V. Vielzeuf, S. Pateux, M. Baccouche, and F. Jurie. Mfas: Multimodal fusion architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6966–6975, 2019. 81, 84

[166] C. Pestana, N. Akhtar, W. Liu, D. Glance, and A. Mian. Adversarial perturbations prevail in the y-channel of the ycbcr color space. *arXiv preprint arXiv:2003.00883*, 2020. 14

[167] P. Pinggera12, T. Breckon, and H. Bischof. On cross-spectral stereo matching using dense gradient features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, volume 2, 2012. 86

[168] N. Pinto, Z. Stone, T. Zickler, and D. Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *CVPR 2011 WORKSHOPS*, pages 35–42. IEEE, 2011. 54

[169] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir. Heterogeneous and multitask wireless sensor networks - algorithms, applications, and challenges. *IEEE Journal of Selected Topics in Signal Processing*, 11(3):450–465, 2017. 40

[170] R. Pocius, D. Isele, M. Roberts, and D. W. Aha. Comparing reward shaping, visual hints, and curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 66

[171] R. P. Poudel, S. Liwicki, and R. Cipolla. Fast-scnn: Fast semantic segmentation network. *arXiv preprint arXiv:1902.04502*, 2019. 32

[172] J. C. Principe. *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010. 40, 41, 42

[173] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 413–420. IEEE, 2009. 54

[174] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019. 49, 50

[175] N. Radwan, A. Valada, and W. Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414, 2018. 76

[176] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 49

[177] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, X. Chen, and X. Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2021. 81

[178] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 49

[179] D. R. Rico Jonschkowski and O. Brock. Differentiable particle filters: End-to-end learning with algorithmic priors. In *Proceedings of Robotics: Science and Systems*, 2018. 10

[180] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 51, 53, 54

[181] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 55

[182] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou. How should we evaluate supervised hashing? In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pages 1732–1736, 2017. 41

[183] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermüller, and Y. Aloimonos. Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robotics and Automation Letters*, 3(4):2799–2806, 2018. 14

[184] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756, 2016. 76

[185] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8601–8610, 2018. 76

[186] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3302–3312, 2019. 76

[187] N. Schneider, F. Piewak, C. Stiller, and U. Franke. Regnet: Multimodal sensor registration using deep neural networks. In *2017 IEEE intelligent vehicles symposium (IV)*, pages 1803–1810. IEEE, 2017. 76

[188] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 66, 67, 68

[189] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons, 2015. 23

[190] J. Shan, S. Zhou, Z. Fang, and Y. Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. 2021. 63

[191] E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Transsactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017. 60

[192] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 76

[193] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 85

[194] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 25, 33

[195] K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada. EfficientLPS: Efficient LiDAR panoptic segmentation. *arXiv preprint arXiv:2102.08009*, 2021. 10, 13, 71, 72, 75, 90, 234

[196] R. S. Snell and M. A. Lemp. *Clinical anatomy of the eye*. John Wiley & Sons, 2013. 14

[197] G. Song and W. Chai. Collaborative learning for deep neural networks. *Advances in Neural Information Processing Systems*, 31:1832–1841, 2018. 25, 31

[198] P. Sprechmann, A. M. Bronstein, and G. Sapiro. Learning efficient sparse and low rank models. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1821–1833, 2015. 51

[199] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 54

[200] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *IEEE International Conference on Computer Vision*, 2015. 59

[201] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 77

[202] M. Sun, S. Yang, and H. Liu. Scale-aware camera localization in 3d lidar maps with a monocular visual odometry. *Computer Animation and Virtual Worlds*, 30(3-4):e1879, 2019. 76

[203] X. Sun, B. Wei, X. Ren, and S. Ma. Label embedding network: Learning label representation for soft training of deep networks. *arXiv preprint arXiv:1710.10393*, 2017. 40

[204] K. Takahashi and J. Tan. Deep visuo-tactile learning: Estimation of tactile properties from images. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8951–8957, 2019. 81

[205] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 72, 73

[206] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villasenor-Pineda. Dynamic reward shaping: training a robot by voice. In *Ibero-American conference on artificial intelligence*, pages 483–492. Springer, 2010. 66

[207] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 71, 72

[208] Y. Tian, D. Krishnan, and P. Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020. 53, 54, 55

[209] Tianwei Yin and Xingyi Zhou and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CoRR*, 2020. 63

[210] P. Tiritiris, N. Passalis, and A. Tefas. Temporal difference rewards for end-to-end vision-based active robot tracking using deep reinforcement learning. In *International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, pages 21–25, 2021. 67

[211] K. Torkkola. Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003. 20, 21, 23, 41

[212] D. T. Tran, M. Gabbouj, and A. Iosifidis. Knowledge distillation by sparse representation matching. *arXiv preprint arXiv:2103.17012*, 2021. 91, 212

[213] D. T. Tran, A. Iosifidis, and M. Gabbouj. Improving efficiency in convolutional neural networks with multilinear filters. *Neural Networks*, 105:328–339, 2018. 50

[214] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis. Heterogeneous multilayer generalized operational perceptron. *IEEE transactions on neural networks and learning systems*, 2019. 50

[215] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis. Pygop: A python library for generalized operational perceptron algorithms. *Knowledge-Based Systems*, 182:104801, 2019. 50

[216] D. T. Tran, N. Passalis, A. Tefas, M. Gabbouj, and A. Iosifidis. Attention-based neural bag-of-features learning for sequence data. *arXiv preprint arXiv:2005.12250*, 2020. 49

[217] L. T. Triess, D. Peter, C. B. Rist, and J. M. Zöllner. Scan-based semantic segmentation of lidar point clouds: An experimental study. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121. IEEE, 2020. 72

[218] F. Tung and G. Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018. 50

[219] M. Tzelepi, N. Passalis, and A. Tefas. Online subclass knowledge distillation. *Expert Systems with Applications*, 181:115132, 2021. 36

[220] M. Tzelepi and A. Tefas. Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization. *Pattern Recognition*, 106:107407, 2020. 26, 29, 32, 33

[221] M. Tzelepi and A. Tefas. Efficient training of lightweight neural networks using online self-acquired knowledge distillation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021. 25, 36

[222] M. Tzelepi and A. Tefas. Semantic scene segmentation for robotics applications. *arXiv:2108.11128*, 2021. 32

[223] A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018. 76

[224] A. Valada, N. Radwan, and W. Burgard. Incorporating semantic and geometric priors in deep pose regression. In *Workshop on Learning and Inference in Robotics: Integrating Structure, Priors and Models at Robotics: Science and Systems (RSS)*, 2018. 76

[225] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 36

[226] N. Vyas, S. Saxena, and T. Voice. Learning soft labels via meta learning. *arXiv preprint arXiv:2009.09496*, 2020. 36, 40

[227] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 54

[228] W. Wang, R. Yu, Q. Huang, and U. Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2569–2578, 2018. 71

[229] J. Weichselbaum, C. Zinner, O. Gebauer, and W. Pree. Accurate 3d-vision-based obstacle detection for an autonomous train. *Computers in Industry*, 64(9):1209–1220, 2013. 86

[230] X. Weng, J. Wang, D. Held, and K. Kitani. AB3DMOT: A baseline for 3d multi-object tracking and new evaluation metrics. *CoRR*, abs/2008.08063, 2020. 63

[231] A. G. Wilson and P. Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. 2020. 48

[232] R. W. Wolcott and R. M. Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. IEEE, 2014. 76

[233] B. Wu, A. Wan, X. Yue, and K. Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *IEEE International Conference on Robotics and Automation*, 2018. 60

[234] B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 71

[235] M. Wu and N. Goodman. Multimodal generative models for scalable weakly-supervised learning. *Advances in Neural Information Processing Systems*, 31, 2018. 81

[236] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 2156–2162, 2014. 40

[237] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 50, 54

[238] Z. Xu, D. R. So, and A. M. Dai. Mufasa: Multimodal fusion architecture search for electronic health records. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10532–10540, 2021. 81

[239] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *Advances in neural information processing systems*, 32:6740–6749, 2019. 71

[240] C. Yang, L. Xie, S. Qiao, and A. Yuille. Knowledge distillation in generations: More tolerant teachers educate better students. *arXiv preprint arXiv:1805.05551*, 2018. 55

[241] J. Ye. Least squares linear discriminant analysis. *International Conference on Machine Learning*, 1:1087–1093, 2007. 56, 57

[242] W. Ye, Z. Li, C. Yang, J. Sun, C.-Y. Su, and R. Lu. Vision-based human tracking control of a wheeled inverted pendulum robot. *IEEE Transactions on Cybernetics*, 46(11):2423–2434, 2015. 66

[243] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019. 71

[244] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *arXiv preprint arXiv:2004.02147*, 2020. 32

[245] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 32

[246] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020. 40

[247] W. Yuan, R. Fan, M. Y. Wang, and Q. Chen. Active perception with a monocular camera for multiscopic vision. *arXiv preprint arXiv:2001.08212*, 2020. 14

[248] S. Zagoruyko and N. Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016. 51, 54, 55, 56

[249] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, and G. Troster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *2007 3rd international conference on intelligent sensors, sensor networks and information*, pages 281–286. IEEE, 2007. 87

[250] C.-B. Zhang, P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, and M.-M. Cheng. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30:5984–5996, 2021. 40

[251] F. Zhang, C. Guan, J. Fang, S. Bai, R. Yang, P. H. Torr, and V. Prisacariu. Instance segmentation of lidar point clouds. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9448–9455. IEEE, 2020. 71

[252] J. Zhang, L. Tai, M. Liu, J. Boedecker, and W. Burgard. Neural slam: Learning to explore with external memory. *arXiv preprint arXiv:1706.09520*, 2017. 10

[253] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Trans. on Image Processing*, 24(12):4766–4779, 2015. 40

[254] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *arXiv e-prints*, 2020. 9, 60

[255] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu. Deep mutual learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 18, 19, 31

[256] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020. 71

[257] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang. A survey of sparse representation: algorithms and applications. *IEEE access*, 3:490–530, 2015. 51

[258] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1556–1564. IEEE, 2015. 40

[259] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu. Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications. *IEEE Sensors Journal*, 20(9):4901–4913, 2020. 87

[260] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou. Balanced quantization: An effective and efficient approach to quantized neural networks. *Journal of Computer Science and Technology*, 32(4):667–682, 2017. 50

[261] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 2415–2421, 2016. 40

[262] X. Zhu, S. Gong, et al. Knowledge distillation by on-the-fly native ensemble. In *Advances in neural information processing systems*, pages 7517–7527, 2018. 55

[263] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 50

7 Appendix

7.1 Pseudo-Active Vision for Improving Deep Visual Perception through Neural Sensory Refinement

The appended paper follows.

PSEUDO-ACTIVE VISION FOR IMPROVING DEEP VISUAL PERCEPTION THROUGH NEURAL SENSORY REFINEMENT

Nikolaos Passalis and Anastasios Tefas

Dept. of Informatics, Aristotle University of Thessaloniki, Greece
Email: {passalis, tefas}@csd.auth.gr

ABSTRACT

Active vision approaches hold the credentials for improving the accuracy of Deep Learning (DL) models for many challenging visual analysis tasks and varying environmental conditions. However, active vision approaches are typically closely tied to the underlying hardware, slowing down their adoption, while they typically increase the latency of perception systems, since sensory data must be recaptured. In this work, we propose a pseudo-active data refinement method that works by appropriately refining the sensory input, without having to reacquire the sensor data through traditional camera control approaches. The proposed method is fully differentiable and can be trained for the task at hand in an end-to-end fashion, while it can be directly deployed in a wide variety of systems, tasks and conditions. The effectiveness and robustness of the proposed method is demonstrated across a variety of tasks using two challenging datasets.

Index Terms— Active Perception, Active Vision, Visual Perception, Deep Learning, Robotic Perception

1. INTRODUCTION

Deep Learning (DL) led to remarkable performance in many challenging computer vision tasks [1]. However, despite its success in such tasks, employing DL methods in real world applications, that often have different requirements than simply training a model on a typical computer vision dataset, pose significant challenges. For example, robotics typically require visual perception algorithms that can handle temporal and spatial embodiment [2], while also providing active perception capabilities [3], none of which is currently fully addressed by existing DL approaches to a satisfactory degree.

This paper focuses on active vision that allows for appropriately controlling the camera of a perception system in order to improve the perception accuracy [3]. It is worth noting that a camera can be controlled both regarding its external parameters, e.g., pan and tilt, as well as some of its internal parameters, e.g., exposure and color profile, etc. Even through there is an increasing amount of literature for handling the former [4, 5, 6, 7, 8], less focus has been given to the latter (with respect to the performance of DL models). Indeed, most DL algorithms implicitly assume that the heavy pre-processing that is involved most digital camera sensors, e.g., color constancy algorithms [9, 10, 11, 12], will mitigate the effect of varying illumination conditions. As a result, most DL models do not explicitly deal with these effects. However, as we experimentally demonstrate in this paper, DL models are especially prone to both varying illumination conditions, as well as changes in contrast, brightness, and slight color shifts. This is not a surprising finding, since adversarial attacks, as well as studies of the effect of color shifts on the accuracy of DL models, have indeed demonstrated the vulnerability of DL models to such transformations [13, 14].

Apart from distribution shifts that arise from the environmental factor described above, using different hardware for the deployment can also reduce the visual perception accuracy, if the sensors are not adequately calibrated. The typical solution to this problem is to manually tune the hardware, as well as employ the appropriate software pre-processing modules in order to ensure that the models will perform as expected. However, it is obvious that this process requires a significant amount of effort, often slowing down the integration in many real robotics applications.

Active perception algorithms can be employed to tackle the aforementioned challenges, i.e., to appropriately control the internal camera parameters in order to maximize perception accuracy for a given DL model. This process has many similarities with the visual perception systems developed in many biological organisms, such as many mammals. In these cases, different mechanisms exist for adjusting various parameters that affect the signal reaching to sensor cells, well before propagating the information to the visual cortex. Perhaps the most well-known example of such mechanism is the adaptive response of the iris to different illuminations conditions, altering the amount of light eventually reaching the retina [15]. Despite their potential, active vision methods come with two important drawbacks. First, they typically lead to the (complete or partial) loss of (at least) one frame acquired by the camera, which can negatively affect the latency of the system. Second, such algorithms are not easily deployed, since the output of the active perception algorithm must be first adequately translated in order to match the underlying hardware of each system, i.e., to translate the

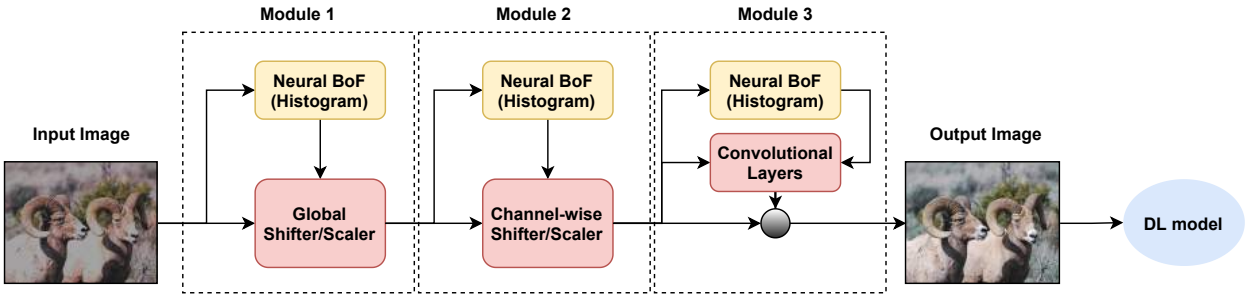


Fig. 1. The proposed method is composed of three separate neural modules. Each one is responsible for learning how to apply a different kind of image transformation in order to gradually refine the input image in order to maximize the accuracy of the subsequent DL model.

model's output to control signals.

To overcome these limitations in this work we propose a pseudo-active sensory refinement method that works by applying a number of neural transformation layers on the sensor data. This allows for refining the sensory input, without having to reacquire the sensor data. In contrast with traditional image processing operations, such as histogram equalization, contrast corrections, etc., the proposed method is *end-to-end* trainable and formulated as a series of neural layers. As a result, the proposed method can be fully integrated in DL end-to-end training pipelines. However, at the same time, it provides significant advantages, since a) it can be directly used with any DL model, without requiring any model-specific training or any platform-specific adjustments, b) it does not require support by the underlying hardware, and c) it allows for avoiding the need to reacquire a new frame for processing by the employed DL model. As a result, the proposed method provides a solid step towards developing practical and powerful tools that can be directly deployed in a wide variety of systems, tasks and conditions, increasing the perception accuracy. Indeed, we demonstrate the effectiveness of the proposed method using two different tasks and datasets, i.e., image recognition on ImageNet dataset (ILSVRC 2012) [16] and object detection on PASCAL VOC 2007 dataset [17].

The rest of the paper is structured as follows. First, we introduce the proposed method in Section 2. Then, we provide the experimental evaluation in Section 3. Finally, conclusions are drawn in Section 4.

2. PROPOSED METHOD

Let $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ be an input image, where W, H and C denote its width, height and number of channels respectively. Typically, the input image $\mathbf{x}$ is fed into a DL model $f(\mathbf{x})$ in order to perform visual information analysis, e.g., object recognition. The proposed method works by employing a series of neural transformation layers, as shown in Fig. 1, before feeding the input into the DL model. The proposed method is composed of three separate modules, each one performing a different *learnable* transformation on the input. Each of these modules operate on a dynamic differentiable color histogram compiled through the previous stage. Therefore, during the first stage, the image is globally shifted and scaled according to the input histogram. Then, the second stage performs the same transformation, but on channel level. These two modules combined aim at correcting global and per color channel brightness and contrast. Then, the third module is responsible for refining local regions of the input image, while also taking into account its global histogram, in order to recover information that is potentially lost during image acquisition, e.g., due to over-exposure.

More specifically, the proposed method works as follows. First, we compile a differentialable histogram with adaptive bins using a Neural Bag-of-Feature-based formulation [18]. To this end, we quantize the input image features $[\mathbf{x}]_{ij} \in \mathbb{R}^C$ using a codebook $\mathbf{V} = [\mathbf{v}_1; \dots; \mathbf{v}_{N_K}]^T \in \mathbb{R}^{N_K \times C}$ as:

$$\mathbf{h} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \mathbf{u}_{ij} \in \mathbb{R}^{N_K} \tag{1}$$

where N_K is the number of codewords, while the similarity vector $\mathbf{u}_{ij}$ for the codewords $\mathbf{v}_k$ and the input features $\mathbf{x}_{ij}$ is calculated as

$$[u_{ij}]_k = \text{sigm}(\mathbf{v}_k^T \mathbf{x}_{ij}), \tag{2}$$

where $\text{sigm}(\cdot) \in (0, 1)$ denotes the sigmoid function. Three different histograms are compiled, as shown in Fig. 1: a) one at the input ($\mathbf{h}_1$), b) one at the output of the first module ($\mathbf{h}_2$) and c) one at the end of the second module ($\mathbf{h}_3$). Note that the first histogram operate on the averaged color intensities, i.e., $C = 1$, instead of separate color channels, since its aim is to capture the global information regarding the brightness distribution in the image.

After compiling the first histogram, the proposed method employs two linear layers to appropriately shift and scale the

Table 1. Object recognition evaluation (recognition accuracy (%) is reported) on ImageNet dataset (ILSVRC 2012)

Method	Transformation	top-1	top-5	Method	Transformation	top-1	top-5
Baseline	No	69.76	89.08	Baseline	Brightness(-)	60.52	82.64
Baseline	Contrast (+)	62.18	83.94	Autocontrast	Brightness(-)	61.02	82.99
Hist. Equalization	Contrast (+)	54.69	77.74	Proposed	Brightness(-)	**61.67**	**83.58**
Autocontrast	Contrast (+)	62.18	83.93	Baseline	Brightness(–)	53.13	76.38
Proposed	Contrast (+)	**63.43**	**84.95**	Autocontrast	Brightness(–)	53.78	77.15
Baseline	Contrast (++)	57.49	80.23	Proposed	Brightness(–)	**54.78**	**78.01**
Hist. Equalization	Contrast (++)	50.00	73.48	Baseline	Hue (+)	62.11	84.38
Autocontrast	Contrast (++)	57.47	80.23	Autocontrast	Hue (+)	62.17	84.39
Proposed	Contrast (++)	**59.23**	**81.78**	Proposed	Hue (+)	**62.37**	**84.70**
Baseline	Brightness (+)	64.55	85.92	Baseline	Hue (-)	62.63	85.02
Autocontrast	Brightness (+)	66.81	87.26	Autocontrast	Hue (-)	62.69	85.05
Proposed	Brightness (+)	**67.03**	**87.35**	Proposed	Hue (-)	**63.41**	**85.46**
Baseline	Brightness(++)	60.09	82.48	Baseline	Combined	55.31	78.35
Autocontrast	Brightness(++)	64.10	85.14	Autocontrast	Combined	55.17	78.23
Proposed	Brightness(++)	**64.31**	**85.38**	Proposed	Combined	**55.60**	**78.80**

input:

$$\mathbf{x}' = (\mathbf{x} - \mathbf{h}_1^T \mathbf{W}_{11})/(1 + \mathbf{h}_1^T \mathbf{W}_{12}), \quad (3)$$

where $\mathbf{W}_{11} \in \mathbb{R}^{N_K \times 1}$ denotes the weights of the shifting sub-layer and $\mathbf{W}_{12} \in R^{N_K \times 1}$ denotes the weights of the scaling sub-layer. Then, this process is repeated in the second module, but the shifting and scaling is now applied per-channel:

$$[x'']_{ijk} = ([x']_{ijk} - [\mathbf{h}_2^T \mathbf{W}_{21}]_k)/(1 + [\mathbf{h}_2^T \mathbf{W}_{22}]_k), \quad (4)$$

where $\mathbf{W}_{21} \in \mathbb{R}^{N_K \times C}$ denotes the weights of the shifting sub-layer and $\mathbf{W}_{22} \in \mathbb{R}^{N_K \times C}$ denotes the weights of the scaling sub-layer.

Finally, the third module works by first projecting the third histogram into a lower dimensional space:

$$\mathbf{h}_x = \tanh(\mathbf{h}_3^T \mathbf{W}_p) \in \mathbb{R}^{N_P} \quad (5)$$

where $\mathbf{W}_p \in \mathbb{R}^{N_K \times N_P}$ and $\tanh(\cdot)$ denotes the hyperbolic tangent function. Then, the vector $\mathbf{h}_x$ is upsampled into a $W \times H \times N_P$ tensor and concatenated (channel-wise) with the output of the previous layer to form the $\mathbf{x}_p \in \mathbb{R}^{W \times H \times (C+N_P)}$ tensor. The output of the final layer is then calculated as:

$$\mathbf{x}''' = \mathbf{x}'' \odot (1 + \tanh(g(\mathbf{x}_p))), \quad (6)$$

where $g(\cdot) \in \mathbb{R}^{W \times H \times (C)}$ is a series of convolution and deconvolution layers and $\odot$ denotes the Hadamard (element-wise) product between two tensors. In this work we use two 5×5 convolution layers with 16 and 8 filters, followed by two symmetric deconvolution layers (the last layer has C filters). The $\tanh(\cdot)$ non-linearity is used for all the layers. Note that we assumed that the input is already normalized into a specific (limited) range. If this assumption does not hold, then the output of the last layer must be also scaled using a trainable parameter.

All the parameters of the proposed method are learned by first employing an image transformation on the original image $\mathbf{x}$, e.g., brightness, contrast or hue adjustment, leading to a *corrupted* image $\tilde{\mathbf{x}}$. Then, all the layers are trained using the back-propagation algorithm in order to recover the original image to the output of the proposed sequence of neural layers, denoted by $h(\cdot)$, by minimizing the following loss function:

$$\mathcal{L} = \frac{1}{WHC} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{C} ([\mathbf{x}''']_{ijk} - [\mathbf{x}]_{ijk})^2, \quad (7)$$

where $\mathbf{x}''' = h(\tilde{\mathbf{x}})$.

3. EXPERIMENTAL EVALUATION

The proposed method was first extensively evaluated on the ImageNet dataset (ILSVRC 2012) [16] using an image recognition setup and a ResNet-18 architecture [19]. The proposed method was trained on the ImageNet dataset using the Adam optimizer [20]. The proposed model was trained for 10,000 iterations with a learning rate of 0.001, followed by 2,000 iterations with a reduced learning of 0.0001. The batch size was set to 32, while the number of codewords was set to $N_K = 100$ and the number of projection dimensions for the last layer to $N_P = 8$. The proposed method was also compared to a) directly using the trained ResNet-18 architecture (denoted as "Baseline") with the default normalization scheme (global standardization), b) employing a histogram equalization approach (denoted as "Hist. Equalization") and c) maximizing the contrast (denoted as "Autocontrast").

We also used different approaches to deteriorate the original images: a) increasing the contrast by 30% (+) and 40% (++), b) increasing (+/++) or decreasing (-/–) the brightness by 30% and 40% respectively, c) increasing (+) or decreasing (-) the hue by 10%. Also, we evaluated the methods in a more challenging combined setup where the contrast was increased by 20%, brightness was decreased by 20% and hue was increased by 5%. The proposed model was trained by randomly

Table 2. Ablation study on the ImageNet dataset using the contrast (++) transformation (%)

Method	top-1	top-5
Baseline	57.49	80.23
Module 1	58.22	80.87
Module 1 + 2	58.42	80.99
Module 1 + 2 + 3	**59.23**	**81.78**

selecting one transformation and then recovering the original image.

The experimental results for the ImageNet dataset are reported in Table 1. Several interesting conclusions can be drawn from the reported results. First, note that all the applied transformations, i.e., increasing the contrast, increasing/reducing the brightness, increasing/decreasing the hue, and/or combining multiple transformations on the input image, lead to a significant reduction in recognition accuracy. For example, even a mild shift in the hue by 10% leads to a reduction in top-1 accuracy from 69.76% to 62.11%. At the same time, employing non-linear histogram equalization does not improve the recognition accuracy. In contrast, this type of image enhancement actually reduces the recognition accuracy, possibly by introducing a catastrophic distribution shift. Similar results were observed for the rest of transformations as well, but omitted from Table 1, due to lack of space. On the other hand, maximizing the contrast in the input image can lead to improved recognition accuracy in virtually all the evaluated cases. Employing the proposed method can further improve the recognition results, by being able to learn how to appropriately refine the original images in order to compile new transformed images that would be appropriate for the DL model at hand. It is worth noting that for some cases, e.g., for contrast transformations, using the proposed method can improve the top-1 recognition by over 3% (relative improvement). Furthermore, in order to evaluate the effect of the three different modules that are used by the proposed method, we also performed an ablation study, where each of these modules was separately evaluated. The experimental results are reported in Table 2, where we can observe that each of the employed modules further increases the recognition accuracy.

To further demonstrate the generality of the proposed method, we conducted additional experiments for one other task, i.e., object detection. To this end, we employed the Single Shot MultiBox Detector (SSD) [21], with a MobileNet backbone [22] for object detection. The proposed method was not trained on the corresponding detection dataset. Instead, we directly employed the model trained on the ImageNet dataset for a recognition task in order to evaluate how it generalizes on this related task. The experimental results are reported in Table 3. Indeed, using the proposed method again improves the perception accuracy over the evaluated baselines. Quite interestingly maximizing the contrast does not work so well for this task, while the proposed method still leads to impressive improvements (e.g., the mean Average Precision (mAP) increases by over 1.5% in some cases). These results highlight the ability of the proposed method to be directly employed and combined with virtually any DL model and further increase its robustness by providing an efficient pseudo-active vision approach.

Table 3. Evaluation on the VOC2007 (object detection) dataset (%)

Method	Transform.	VOC2007 mAP
Baseline	-	75.51
Baseline	Contrast (+)	62.24
Autocontrast	Contrast (+)	62.13
Proposed	Contrast (+)	**63.19**
Baseline	Contrast (++)	66.95
Autocontrast	Contrast (++)	66.86
Proposed	Contrast (++)	**67.44**
Baseline	Brightness (++)	70.30
Autocontrast	Brightness (++)	71.36
Proposed	Brightness (++)	**71.45**
Baseline	Brightness (–)	56.84
Autocontrast	Brightness (–)	55.81
Proposed	Brightness (–)	**56.85**
Baseline	Combined	56.64
Autocontrast	Combined	56.39
Proposed	Combined	**57.29**

4. CONCLUSIONS

In this work we presented a pseudo-active sensory refinement method that works by applying a number of neural transformation layers on the input, allowing for efficiently refining the sensory input, without having to re-acquire the sensor data. The proposed method is fully differentiable and can be trained for the task at hand in an end-to-end fashion. Furthermore, as demonstrated through the conducted experiments, the proposed method can perform well across a variety of tasks. In this way, it provides a solid step towards providing powerful tools that can be directly deployed in a wide variety of systems, tasks and conditions, increasing the perception accuracy, paving the way for developing more advanced active vision approaches for manipulating the camera parameters.

Acknowledgments: This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

5. REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al., "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[3] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

[4] Nitin J Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermüller, and Yiannis Aloimonos, "Gapflyt: Active vision based minimalist structure-less gap detection for quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2799–2806, 2018.

[5] Anton Mitrokhin, P Sutor, Cornelia Fermüller, and Yiannis Aloimonos, "Learning sensorimotor control with neuromorphic sensors: Toward hyperdimensional active perception," *Science Robotics*, vol. 4, no. 30, 2019.

[6] Nikolaos Passalis and Anastasios Tefas, "Leveraging active perception for improving embedding-based deep face recognition," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, 2020, pp. 1–6.

[7] Theodoros Bozinis, Nikolaos Passalis, and Anastasios Tefas, "Improving visual question answering using active perception on static images," in *Proceedings of the International Conference on Pattern Recognition*, 2021, pp. 879–884.

[8] Weihao Yuan, Rui Fan, Michael Yu Wang, and Qifeng Chen, "Active perception with a monocular camera for multiscopic vision," *arXiv preprint arXiv:2001.08212*, 2020.

[9] Marc Ebner, *Color constancy*, vol. 7, John Wiley & Sons, 2007.

[10] Simone Bianco, Claudio Cusano, and Raimondo Schettini, "Color constancy using cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 81–89.

[11] Yuanming Hu, Baoyuan Wang, and Stephen Lin, "Fc4: Fully convolutional color constancy with confidence-weighted pooling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4085–4094.

[12] Firas Laakom, Nikolaos Passalis, Jenni Raitoharju, Jarno Nikkanen, Anastasios Tefas, Alexandros Iosifidis, and Moncef Gabbouj, "Bag of color features for color constancy," *IEEE Transactions on Image Processing*, vol. 29, pp. 7722–7734, 2020.

[13] Mahmoud Afifi and Michael S Brown, "What else can fool deep learning? addressing color constancy errors on deep neural network performance," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 243–252.

[14] Camilo Pestana, Naveed Akhtar, Wei Liu, David Glance, and Ajmal Mian, "Adversarial perturbations prevail in the y-channel of the ycbcr color space," *arXiv preprint arXiv:2003.00883*, 2020.

[15] Richard S Snell and Michael A Lemp, *Clinical anatomy of the eye*, John Wiley & Sons, 2013.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[18] Nikolaos Passalis and Anastasios Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 21–37.

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

7.2 Probabilistic Online Self-Distillation

The appended paper follows.

Probabilistic Online Self-Distillation

Maria Tzelepi, Nikolaos Passalis, Anastasios Tefas

Aristotle University of Thessaloniki, Department of Informatics

Abstract

Deploying state-of-the-art deep learning models on devices with limited computational power imposes certain computation and storage restrictions. *Knowledge Distillation*, i.e. training compact models by transferring knowledge from more powerful models, constitutes a promising route to address this issue that has been followed during the recent years. A limitation of conventional knowledge distillation is that it is a long-lasting, computationally and memory demanding process, since it requires multiple stages of training process. To this end, a novel online probabilistic self-distillation method, namely *Probabilistic Online Self-Distillation* (POSD), aiming to improve the performance of any deep neural model in an online manner, is proposed in this paper. We argue that considering a classification problem, apart from the explicit concepts expressed with the hard labels, there are also implicit concepts expressed with the so-called *latent labels*. These implicit concepts reflect similarities among data, regardless of the classes. Then, our goal is to maximize the Mutual Information between the data samples and the latent labels. In this way, we are able to derive additional knowledge from the model itself, without the need of building multiple identical models or using multiple models to teach each other, like existing online distillation methods, rendering the POSD method more efficient. The experimental evaluation on six datasets validates that the proposed method improves the classification performance.

Keywords: Knowledge Distillation, Probabilistic Online Self-Distillation, Quadratic Mutual Information.

Email addresses: mtzelepi@csd.auth.gr (Maria Tzelepi), passalis@csd.auth.gr (Nikolaos Passalis), tefas@csd.auth.gr (Anastasios Tefas)

1. Introduction

Over the recent years deep learning (DL) models [1], have eclipsed previous solutions in a wide range of challenging computer vision tasks, [2, 3, 4, 5, 6, 7]. State-of-the-art DL models are generally parameter-heavy, containing millions or even billions of parameters [8, 9, 10, 11], assisted to some extent by the availability of increasingly powerful GPUs. That is, the outstanding performance of DL models stems from their depth and complexity. For instance, amongst others, ResNets [12], introducing the concept of residual learning, achieve state-of-the-art performance by increasing the depth of the networks, while Wide-ResNets [13] are designed to be wider, by reducing the depth. Therefore, applying these models in real-time terms, and/or on devices with restricted computational resources such as mobile phones and embedded systems, is obstructed by their enhanced capacity.

Thus, an apparent need for developing compact yet effective models, diminishing the storage requirements and the computational cost, has been arisen. Substantial research work has been performed over the recent few years to achieve this goal [14]. This work epigrammatically includes developing compact and effective models by design, such as [15, 16, 17, 18, 19, 20], parameter pruning, where the redundancy in the parameters of the model is investigated and the complexity of the model is reduced by removing the redundant parameters [21, 22], network quantization where in a similar way the required bits for the parameter representation are removed in order to compress the model [23, 19], and finally, *Knowledge Transfer* (KT), [24, 25, 26, 27, 28, 29]. KT has been emerged as a highly promising approach to address this issue proposing to transfer the knowledge from one, usually larger, model to a more compact model. *Knowledge Distillation* (KD) [30, 31, 32, 33] constitutes the most prominent offshoot of KT.

KD, in its typical version, refers to the procedure where the knowledge of a heavyweight and powerful model, known as teacher, which achieves high performance, is transferred to a lighter and faster model, known as student. The latter is trained to match the so-called *soft labels* generated by the teacher model, by

raising the temperature of the softmax activation function on the output layer of the network. The above procedure is also known as *softening* the output distribution. The underlying rationale behind this practice is that these soft labels relay more information about the way that the model learns to generalize, comparing to the hard labels. Apart from the aforementioned strategy, where the knowledge is transferred from a complex model to a weaker one, there have also been proposed strategies where the knowledge is transferred from a weaker to a more complex model [26] or the knowledge is transferred from teachers to students of identical capacity [34, 35, 36]. The latter process is known as *self-distillation*. Amongst the self-distillation works, we distinguish [36] where the knowledge is discovered from the model itself.

KD methods fall into two categories: *online* and *offline* KD. The multi-stage procedure of initially training a powerful teacher model and then distilling the knowledge to a weaker student model, stands for *offline* KD. However, offline distillation is accompanied by some flaws. That is, it is a long-lasting, complex, and computationally and memory demanding process, since it requires to train first a powerful and heavyweight model, and after the convergence to transfer the acquired knowledge to a faster model. Thus, in the recent literature several online KD methods have been proposed, in order to circumvent the aforementioned flaws of offline KD . Online KD describes the procedure where the teacher and the student models are trained concurrently, that is without the stage of pre-training the teacher model. Online KD includes methods proposing to train multiple (student) models mutually from each other [37, 38], where each model acts as a teacher to other model, as well as methods proposing to train k copies of a target (student) model in parallel by adding a distillation term to the loss function of the i-th model to match the average prediction of the other models [37]. Existing online distillation methods are thoroughly discussed in the Related Work Section.

In this work, a novel probabilistic single-stage self-distillation method, called *Probabilistic Online Self-Distillation* (POSD), is proposed. The proposed method is model-agnostic, that is, it improves the performance of any deep neural model considering object classification tasks (i.e., lightweight and common heavyweight

models, as it is demonstrated through the conducted experiments) in an online manner. Thus, as compared to conventional offline KD methods, reduces the computational and memory requirements. It should be emphasized that the proposed method derives the additional knowledge from the model itself -that is without any powerful teacher model- as in [36], but also in an online manner.

More specifically, we argue that in each classification problem there are *explicit concepts*, expressed with the *hard labels*, but there are also *implicit concepts*, expressed with so-called *latent labels*. These implicit concepts reflect potential sub-clusters formed by data samples that share specific attributes, or similarities among data samples regardless of the class labels. That is, they are cross-label concepts, which convey useful information about the relationships between data samples.

Thus, while conventional KD argues that it is useful to maintain the similarities of the samples with the classes (that express the explicit concepts) during the training process, we argue that there are samples that also share other semantic attributes, extending beyond the explicit classes. Therefore, we aim to discover these similarities during the training process and then appropriately employ them to transfer additional knowledge. To achieve this goal, we use a well-established metric in Information Theory, that is *mutual information*, which allows us to quantify the information regarding these implicit concepts. Since mutual information is in general impractical and inefficient to be estimated in high dimensional spaces, we utilize a variant of mutual information, that is *Quadratic Mutual Information* [39], which allows us to efficiently optimize the network toward the aforementioned objective. That is, our goal is to maximize the Mutual Information (MI) between the distribution of the data samples and the latent labels. In this way, the model is not only able to discriminate between different classes, but also to better encode the geometric relationships between them.

The overarching motivation of the proposed work is to mine further knowledge beyond the hard labels from the model itself and also in an online manner, so as to overcome the limitations of the conventional knowledge distillation (i.e., time consuming, complex, and computationally and memory demanding process).

Thus, taking also into consideration the observations that useful information can be obtained even by transferring knowledge from a model of identical capacity to the student [35], and also that small models usually have the same representation capacity as their heavier counterparts but they are harder to train [33], we propose a single stage self-distillation methodology for developing fast-to-execute yet effective models for various applications with computational and memory restrictions.

To the best of our knowledge, the proposed POSD method, is the first method proposing an online self-distillation methodology in a single stage training pipeline, without affecting the model architecture or employing multiple models, and also without the need of tuning the temperature hyper-parameter like the most distillation methods. At the same time, the proposed method constitutes a powerful and efficient approach for exploiting the manifold structure of the feature space formed by the various layers of neural networks. In this way, the proposed method can significantly reduce over-fitting and, as a result, improve the classification accuracy of the models, as experimentally demonstrated in this paper. Finally, it should be highlighted that the proposed distillation method can be combined with any other method for developing effective and faster models, e.g. [16, 17].

The main contributions of this work can be summarized as follows:

- We propose a novel *Probabilistic Online Self-Distillation* method.
- The POSD method acquires the soft labels from the model itself in an online manner, without requiring the utilization of multiple models or instances / copies of the model like the most online KD methods, rendering it more efficient.
- The proposed method is model-agnostic, that is, it is applicable to several neural network architectures.
- The experimental evaluation indicates that the POSD method can improve the classification performance of any deep neural model.

The rest of the manuscript is structured as follows. Section 2 contains a discussion on relevant works on knowledge distillation. Section 3 presents the proposed method. Subsequently, in Section 4 we provide the experimental evaluation of the proposed method, including the utilized CNN model and the datasets, the implementation details, as well as the experimental results. Finally, the conclusions are drawn in Section 5.

2. Related Work

In this section recent works in the general area of KT, as well as on online KD, which is more relevant to our work, are presented.

Knowledge Transfer, for transferring the knowledge from one neural network to another, has been extensively studied during the recent few years with a wide range of applications [40, 41, 42, 43]. Firstly in [32] and then in [30] the idea of distilling the knowledge from a powerful teacher to a weaker student by encouraging the latter to regress the soft labels produced by the teacher by appropriately raising the temperature of the softmax activation function on the output layer of the network, is proposed. Subsequently, a new pre-training approach is proposed in [44], utilizing soft labels. The knowledge transfer procedure is employed for domain adaptation in combination with limited labeled data, in [45], whilst similarly knowledge is transferred from a recurrent neural network (RNN) model to a small CNN model, in [25]. From a different perspective in the sense that the teacher is assumed to be weaker than the student, knowledge from conventional deep neural networks is used to train a RNN model in [26].

Subsequently, the idea of KD [30] is expanded to allow for thinner and deeper students, by using not only soft labels but also hints from the teacher's intermediate layers in order to guide the training of the student model, in [31]. A KD method where the student model is encouraged to mimic the attention map of the teacher model is proposed in [46], whilst an approach where the parameters of the student model are initialized according to the parameters of the teacher

model is proposed in [24]. Subsequently, a method where the student model is trained to maintain the same amount of mutual information between the learned representation and a set of labels as the teacher model is proposed in [28], while a method that uses similarity-induced embedding to transfer the knowledge between two layers of neural networks, is proposed in [27]. Additionally, under the information-theoretic perspective, knowledge transfer is formulated as maximizing the mutual information between the student and the teacher networks in [47]. A multi-step KD approach where an intermediate-sized network is utilized to bridge the gap between the student model and the teacher model is proposed in [48], since as it is stated the performance degrades when the gap between the teacher model and student model is large. Subsequently, an effective KD method even when there is a distribution mismatch between teacher's and student's training data is proposed in [49]. That is, the method first learns a distribution based on student's training data from which images well-classified by the teacher are sampled. In this way, the data space where the teacher has good knowledge to transfer is discovered. In addition, a new loss function is proposed for training the student network.

Surveying the recent literature, several works has been emerged, proposing self-distillation approaches. Self-distillation as we have already mentioned refers to the kind of distillation where distillation is applied from one model to another of identical architecture. For instance, KD is applied from a teacher to a student of identical architecture where the student accomplishes better performance while it is also optimized faster, in [34]. The flow of solution procedure matrix is utilized in this approach instead of the previously mentioned hints for transferring the knowledge between the intermediate layers. A self-distillation approach where a teacher model is initially trained, and then after its convergence an identical student model is trained with both the goals of the hard labels and matching the output of the teacher model is proposed in [35], however without softening the logits (i.e. the inputs to the final softmax activation function) by raising the temperature. Similarly, a target model is trained with a conventional supervised loss, the self-discovered knowledge is extracted, and in the second training stage,

the model is trained both with the supervised and distillation losses, in [36]. Finally, a framework, named Self-Supervised Knowledge Distillation, proposes to employ self-supervised tasks in order to acquire richer knowledge from the teacher model to the student model in [50]. Moreover, the impact of various self-supervised pretext tasks and the effect of noisy self-supervised predictions to the distillation performance are investigated.

In the recent literature, several works proposing online distillation have also been emerged. A method namely co-distillation improves the accuracy by proposing to train k copies of a target model in parallel by adding a distillation term to the loss function of the i-th model to match the average prediction of the other models [37]. A quite similar approach, where multiple students teach each other throughout the training process, is proposed in [38]. That is, each student is trained with a conventional supervised learning loss, and a distillation loss that matches each student's class posterior probabilities with the class probabilities of other students. In this way, each model acts as a teacher of the other models. In this approach, as opposed to the aforementioned co-distillation method [37], different model architectures can be utilized for the mutual training.

Subsequently, an online distillation approach proposes to build a multi-branch version of the network by adding identical branches, each of which constitutes an independent classification model with shared low level layers, and to create a strong teacher model utilizing a gated logit ensemble of the multiple branches in [51]. Each branch is trained with the conventional classification loss and the distillation loss which regresses the teacher's output distributions. Next, a framework of collaborative learning which trains several classifier heads of the same network at the same time, on the same training data, is proposed in [52]. More specifically, the framework generates a population of classifier heads during the training process, where each head learns, apart from the hard labels, from the soft labels produced by the whole population. Furthermore, the method involves an intermediate-level representation sharing with backpropagation rescaling that aggregates the gradient flows for all the heads.

Next, a recent work [53], combines the previous works [38] and [51], by propos-

ing an online mutual knowledge distillation method for enhancing both the performance of the fusion module and the sub-networks. That is, when different sub-networks are used, the sub-networks are trained similar to [38], while when identical sub-networks are used, the low level layers are shared, and a multi-branch architecture similar to [51] is used. The architecture consists of an ensemble classifier using the ensemble logit produced from the sub-networks and a fused classifier, using the fused feature map. The model distills knowledge from the ensemble classifier to the fused classifier, and simultaneously from the fused classifier to each sub-network classifier.

Subsequently, a two-level distillation methodology, named Online Knowledge Distillation with Diverse peers (OKDDip), where two types of students are involved, i.e., multiple auxiliary peers and one group leader, is proposed in [54]. Distillation is conducted among auxiliary peers with a mechanics for preserving diversity, and then an ensemble of predictions of these peers is further distilled to the group leader. Finally, based on [53], a method named Dense Feature Fusion for Online Mutual Knowledge Distillation (DFL), where the mid-level features from the subnetworks are also fused, is proposed in [55].

In this paper, we propose an efficient online self-distillation method which uses soft labels to reveal the implicit relationships between data samples. Furthermore, a key attribute of the proposed method is that the knowledge is distilled within the same model online, without requiring multiple training stages that typically increase the computational cost, which renders the proposed method more efficient compared to existing online KD methods.

3. Proposed Method

In each classification problem there are *explicit concepts*, expressed with the *hard labels*, but there are also *implicit concepts*, associated with the *latent labels*. Deep neural models transform the probability distribution of the data, layer by layer, learning increasingly complex layer representations. As the distribution of the data is being transformed through the layers during the training procedure,

we aim to derive useful information about the relationships among the data samples which is ultimately ignored, as the samples are forced by the conventional supervised loss to suppress the implicit concepts. To this aim, we propose to introduce an auxiliary objective aiming to encode the useful implicit concepts, that reflect similarities among the data, from the model itself. These implicit concepts are unknown, and they can be discovered through several ways. For example, they could be pre-calculated using clustering, or discovered by exploiting the local manifold structure of the space, etc. In this work, without loss of generality, we consider the degenerated case where the each sample defines a different implicit concept.

More specifically, we consider a C-class classification problem, and the labeled data $\{\boldsymbol{x}_i, \boldsymbol{l}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathfrak{R}^D$ an input vector and D its dimensionality, while $\boldsymbol{l}_i \in \mathcal{Z}^C$ corresponds to its C-dimensional one-hot class label vector (hard label). For an input space $\mathcal{X} \subseteq \mathfrak{R}^D$ and an output space $\mathcal{F} \subseteq \mathfrak{R}^C$, we consider as $\phi(\cdot\,;\mathcal{W}) : \mathcal{X} \rightarrow \mathcal{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of parameters $\mathcal{W} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{N_L}\}$, where $\boldsymbol{W}_L$ are the weights of a specific layer L, which transforms its input vector to a C-dimensional probability vector. That is, $\phi(\boldsymbol{x}_i\,;\mathcal{W}) \in \mathcal{F}$ corresponds to the output vector of $\boldsymbol{x}_i \in \mathcal{X}$ given by the network ϕ with parameters $\mathcal{W}$.

Thus, considering the typical classification problem, we seek for the parameters $\mathcal{W}^*$ that minimize the cross entropy loss, $\mathcal{J}_{ce}$, between the predicted and hard label distributions:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{i=1}^{N} \mathcal{J}_{ce}(\boldsymbol{l}_i, \phi(\boldsymbol{x}_i\,;\mathcal{W})), \tag{1}$$

The cross entropy loss for a set of N samples is formulated as:

$$\mathcal{J}_{ce} = -\sum_{i=1}^{N}\sum_{m=1}^{C} l_i^m log(z_i^m), \tag{2}$$

where l_i^m is the m-th element of $\boldsymbol{l}_i$ one-hot label vector, and z_i^m is defined as the output of the softmax operation on the C-dimensional network's output:

$$z_i^m = \frac{\exp(\phi(\boldsymbol{x}_i\,;\mathcal{W})^m)}{\sum_{j=1}^{C}\exp(\phi(\boldsymbol{x}_i\,;\mathcal{W})^j)}, \tag{3}$$

where the notation $\phi(\boldsymbol{x}_i\,;\mathcal{W})^m$ is used to refer to the m-th value of the output vector of the network.

The cross entropy loss generally suppresses the aforementioned implicit concepts, and thus our goal is to circumvent this by enforcing the data samples to maintain the MI with these concepts.

For simplicity, we consider Y to be a random variable representing the image representations of the feature space generated by a specific deep neural layer L, that is $\boldsymbol{y}_i = \phi(\boldsymbol{x}_i\,;\mathcal{W}_L)$. We also consider a discrete-value variable C that represents the latent labels. Each feature representation $\boldsymbol{y}$ defines an implicit concept and it is associated with a latent label c.

MI measures dependence between random variables, that is, it measures how much the uncertainty for the latent label c is reduced by observing the feature vector $\boldsymbol{y}$. Let $p(c)$ be the probability of observing the latent label c, and $p(\boldsymbol{y}, c)$ the probability density function of the corresponding joint distribution. The MI between the two random variables is defined as:

$$MI(Y,C) = \sum_{c=1}^{N}\int_{\boldsymbol{y}} p(\boldsymbol{y},c)\log\frac{p(\boldsymbol{y},c)}{p(\boldsymbol{y})P(c)}d\boldsymbol{y}, \tag{4}$$

where $P(c) = \int_{\boldsymbol{y}} p(\boldsymbol{y},c)d\boldsymbol{y}$. Instead of utilizing MI, we use Quadratic Mutual Information [39], since directly calculating MI is not computationally tractable. That is, MI can be interpreted as a Kullback-Leibler divergence between the joint probability density $p(\boldsymbol{y},c)$ and the product of marginal probabilities $p(\boldsymbol{y})$ and $P(c)$. Thus, QMI is derived by replacing the Kullback-Leibler divergence by the quadratic divergence measure [39]. That is:

$$QMI(Y,C) = \sum_{c=1}^{N}\int_{\boldsymbol{y}} (p(\boldsymbol{y},c) - p(\boldsymbol{y})P(c))^2 d\boldsymbol{y}. \tag{5}$$

And thus, by expanding eq. (5) we arrive at the following equation:

$$QMI(Y,C) = \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c)^2 d\boldsymbol{y} + \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y})^2 P(c)^2 d\boldsymbol{y} \\ -2 \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c) p(\boldsymbol{y}) P(c) d\boldsymbol{y}. \tag{6}$$

The quantities appearing in eq. (6), are called *information potentials* and they are defined as follows: $V_{IN} = \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c)^2 d\boldsymbol{y}$, $V_{ALL} = \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y})^2 P(c)^2 d\boldsymbol{y}$, $V_{BTW} = \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c) p(\boldsymbol{y}) P(c) d\boldsymbol{y}$, and thus, the QMI between the data samples and the corresponding latent labels can be expressed as follows utilizing the information potentials:

$$QMI = V_{IN} + V_{ALL} - 2V_{BTW}. \tag{7}$$

As we have previously mentioned, we consider that each sample defines an implicit concept, expressed with a latent label, and thus there are N different latent labels. Under the manifold assumption [56], we consider that the implicit concepts are expressed in the feature space as the geometric proximity between the data samples. That is, nearest neighbors, in terms of Euclidean distance, in the feature space for each sample defining an implicit concept, share the same concept. Note, that the nearest neighbors are updated at each iteration, and thus throughout the network's training we obtain more useful nearest neighbors, since the procedure is driven by the supervised loss. It is also noteworthy that manifold assumption allows us to infer only similarities, not dissimilarities. Therefore, we can infer that two neighbors are similar because they are close to each other, however, being far apart does not guarantee that they do not share the same implicit concept. We should also highlight that as the network is optimized to discriminate between the different classes (hard labels), the intermediate representations and the corresponding manifolds are distorted in a way that facilitates the task at hand. Therefore, some implicit concepts that are not relevant to the task at hand are suppressed, while other, relevant to the task at hand, are possibly reinforced.

Thus, each of N different latent labels consists of k_p samples (i.e. a certain number of nearest samples and the sample itself), and the class prior probability for the c_p latent label is given as: $P(c_p) = \frac{k_p}{N}$, where N corresponds to the total number of samples. KDE can be used to estimate the joint density probability: $p(\boldsymbol{y}, c_p) = \frac{1}{N}\sum_{j=1}^{k_p} K(\boldsymbol{y}, \boldsymbol{y}_j^p; \sigma^2)$, for a symmetric kernel K, with width σ, where we use the notation $\boldsymbol{y}_j^p$ to refer to the j-th sample of the p-th latent label, that is the j-th nearest neighbor, as well as the probability density of Y as $p(\boldsymbol{y}) = \sum_{p=1}^{k_p}(\boldsymbol{y}, c_p) = \frac{1}{N}\sum_{j=1}^{N} K(\boldsymbol{y}, \boldsymbol{y}_j; \sigma^2)$.

Therefore, the information potentials that appear in (7) can be efficiently calculated as:

$$V_{IN} = \frac{1}{N^2}\sum_{p=1}^{N}\sum_{k=1}^{k_p}\sum_{l=1}^{k_p} K(\boldsymbol{y}_k^p, \boldsymbol{y}_l^p; 2\sigma^2), \tag{8}$$

$$V_{ALL} = \frac{1}{N^2}\Big(\sum_{p=1}^{N}(\frac{k_p}{N})^2\Big)\sum_{k=1}^{N}\sum_{l=1}^{N} K(\boldsymbol{y}_k, \boldsymbol{y}_l; 2\sigma^2), \tag{9}$$

$$V_{BTW} = \frac{1}{N^2}\sum_{p=1}^{N}\frac{k_p}{N}\sum_{j=1}^{k_p}\sum_{k=1}^{N} K(\boldsymbol{y}_j^p, \boldsymbol{y}_k; 2\sigma^2). \tag{10}$$

The pairwise interactions described above between the samples can be interpreted as follows:

- V_{IN} expresses the interactions between pairs of samples sharing each implicit concept
- V_{ALL} expresses the interactions between all pairs of samples, regardless of the latent label
- V_{BTW} expresses the interactions between samples of each implicit concept against all other samples

The kernel function $K(\boldsymbol{y}_i, \boldsymbol{y}_j; \sigma^2)$ expresses the similarity between two samples i and j. There are several choices for the kernel function, [57]. For example, in [39] the Gaussian kernel is used, while in [27] the authors utilize a cosine similarity

based kernel to avoid defining the width, in order to ensure that a meaningful probability estimation is obtained, since fine-tuning the width of the kernel is not a straightforward task, [58]. In this work, we use the power kernel. Power kernel, K_P , also known as unrectified triangular kernel is defined as follows:

$$K_P = \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^d \tag{11}$$

Our goal is to maximize the QMI between the distribution of the data samples sharing implicit concepts with the distribution of the latent labels. This process can be accelerated by observing that V_{BTW} term includes only the distant neighbors to each implicit concept and, as a result, it typically has a negligible contribution to the optimization compared to V_{IN}. Furthermore, the V_{ALL} term is just a contractive term that tends to shrink all the samples in the feature space, contributing equally to all data samples, regardless their label. Given that our goal is not to accurately estimate the exact value of MI, but to enhance the implicit concepts that appear in the data, we propose accelerating the optimization by using V_{IN} as a proxy to QMI. This approximation allows for accelerating the training process, while having only a negligible effect on the optimization.

Thus, the overall loss, J can be formulated as:

$$J = J_{CE} - \lambda J_{POSD}, \tag{12}$$

where $J_{POSD} = V_{IN}$, and λ balances the importance between the hard and the latent labels.

Simple SGD is utilized to train the model:

$$\Delta \mathcal{W} = -\eta \frac{\vartheta J}{\vartheta \mathcal{W}} \quad (14). \tag{14}$$

In this way, the model is trained synchronously both with the conventional supervised loss (hard labels) so as to discriminate between different classes, and distillation loss so as maximize the QMI of the samples with the latent labels. We should finally highlight that in the early stages of training, the POSD methodology may bring less informative knowledge. That is, the nearest neighbors share by definition some concepts, and hence they are near in the feature space, but

these shared concepts may be less informative for the classification performance. However, as it also verified through the figures that illustrate the test accuracy throughout the training process in the subsequent Section, as the training progresses, we are assured that more useful nearest neighbors are brought, since the process is driven by the supervised loss.

4. Experimental Evaluation

Six datasets were used to validate the performance of the proposed method. In the following subsections, the descriptions of the datasets and the utilized models' architectures are provided. Three sets of experiments were conducted for three different batch sizes considering also four different number of nearest samples in each case. Throughout this work, test accuracy (i.e. Top-1 accuracy) were used for evaluating the proposed method. Each experiment was repeated five times and the mean value and the standard deviation are reported, considering the maximum value of test accuracy for each experiment. The curves of mean test accuracy are also provided. Finally, we use the sum of floating point operations (FLOPs) to evaluate the complexity of the proposed POSD method.

4.1. Datasets

In order to evaluate the performance of the proposed online self-distillation method extensive experiments were conducted on three datasets (Cifar-10, SVHN, and Fashion MNIST). Additionally, since the input dimensions of all the utilized datasets are 32×32, we have also performed representative experiments on Crowd-drone and Tiny-ImageNet datasets. Finally for comparisons against state-of-the-art, we also perform experiments on Cifar-100.

4.1.1. Cifar-10

The *Cifar-10* dataset, [59], consists of 60,000 images of size 32×32 divided into 10 classes with 6,000 images per class. 50,000 images are used as the train set and 10,000 images as the test set. Sample images of the *Cifar-10* dataset are provided in Fig. 1.

Figure 1: Sample images of the Cifar-10 dataset.

4.1.2. Cifar-100

The *Cifar-100* dataset, [59], consists of 60,000 images of size 32×32 divided into 100 classes with 600 images per class. 50,000 images are used as the train set and 10,000 images as the test set.

4.1.3. Street View House Numbers

The Street View House Numbers (SVHN) dataset, [60], obtained from house numbers in Google Street View images. It contains 73,257 train images and 26,032 test images, divided into 10 classes, 1 for each digit from 0 to 9. Input images are of size 32×32 and sample images are provided in Fig. 2.

Figure 2: Sample images of the SVHN dataset.

4.1.4. Fashion MNIST

The Fashion MNIST dataset, [61] comprises of 28×28 gray-scale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. Sample images are presented in Fig. 3.

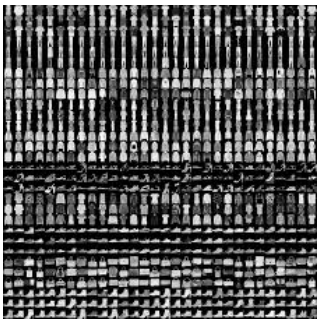


Figure 3: Sample images of the Fashion MNIST dataset.

4.1.5. Tiny-ImageNet

The Tiny-ImageNet dataset contains a training set of 200 classes, each of them containing 500 images, and a validation set consisting of 50 images per class. Input image are of size 64×64. Sample images are provided in Fig. 4.

Figure 4: Sample images of the Tiny-ImageNet dataset.

4.1.6. Crowd-drone

The Crowd-drone dataset is an augmented version the dataset presented in [3]. Crowd-drone is a binary dataset that contains drone-captured images that depict human crowds, and non-crowded scenes. The training set of the dataset consists of 7,000 crowd images, and 7,000 non-crowd images, while the test set consists of 3,000 crowd images, and 3,000 non-crowd images. Images are of size 128×128. Sample images are presented in Fig. 5.

Figure 5: Sample images of the Crowd-drone dataset.

4.2. CNN Models

The main focus of this work is to evaluate the effect of the proposed KD method on training lightweight model that can be effectively deployed on embedded and mobile devices. Therefore, three lightweight CNN architectures are employed for the conducted experiments. In the case of the first two datasets (i.e., Cifar-10, and SVHN-10) we utilize a simple CNN model consisting of five layers; two convolutional layers with 6 filters of size 5×5 and 16 filters of size 5×5 respectively, followed by a Rectified Linear Unit (ReLU) [62] activation, and three fully connected layers ($128 \times 64 \times 10$). The convolutional layers are followed by a 2×2 max-pooling layer with a stride of 2. In the first two fully connected layers the activation function is the ReLU, while the last output layer is a 10-way softmax layer which produces a distribution over the 10 class labels of the utilized datasets. In the case of the Fashion MNIST dataset we also utilize a simple architecture consisting of two convolutional with 20 filters of size 5×5 and 50 filters of size 5×5 followed by a ReLU activation, and two fully connected layers (64×10). The convolutional layers are followed by 2×2 max-pooling layer with a stride of 2. In the first fully connected layer a ReLU activation is applied, while the last output layer is a 10-way softmax layer.

In the case of the Crowd-drone dataset, we use a lightweight fully convolutional network consisting of four convolutional layers followed by a ReLu activation. The first convolutional layer is followed by a max-pooling layer. In the case of Tiny-ImageNet, we use the ResNet-50 model, since it is a challenging dataset. Finally, for comparison purposes against previous online KD approaches, we also

utilize Wide ResNet 16-2 (abbreviated as WRN-16-2) and Wide ResNet 28-10 (abbreviated as WRN-28-10) [13], Wide ResNet 20-08 (abbreviated as WRN-20-08), and ResNet-32 [12] to perform experiments on Cifar-10 and Cifar-100 datasets.

4.3. Implementation Details

All the experiments were performed using the PyTorch framework. We use the mini-batch gradient descent for the networks' training. That is, an update is performed for every mini-batch of N_b training samples. Experiments conducted for $N_b = 32, 64, 128$ samples. The learning rate is set to 10^{-3}, and the momentum is 0.9. The models are trained on an NVIDIA GeForce GTX 1080 with 8GB of GPU memory for 100 epochs. The parameter λ in eq. (12) for controlling the balance between the contributing losses is set to 0.0001 for all the utilized datasets, however we also utilize different values of λ, in order to investigate its impact to the performance of the POSD method. That is, for fixed number of 4 nearest neighbors, we perform experiments for various values of λ on Cifar-10 dataset, for mini-batch of 64 samples. Evaluation results are provided in Table 1 and Fig. 6. As it is shown, the POSD method improves the baseline performance for any considered λ.

Method	**Cifar-10**
W/o Distillation	64.734% ± 0.654%
$\lambda = 0.1$	66.042% ± 0.575%
$\lambda = 0.01$	65.595% ± 0.467%
$\lambda = 0.001$	65.482% ± 0.533%
$\lambda = 0.0001$	**66.362% ± 0.726%**
$\lambda = 0.00001$	65.354% ± 0.330%

Table 1: Test Accuracy - Batch Size: 64, 4NN

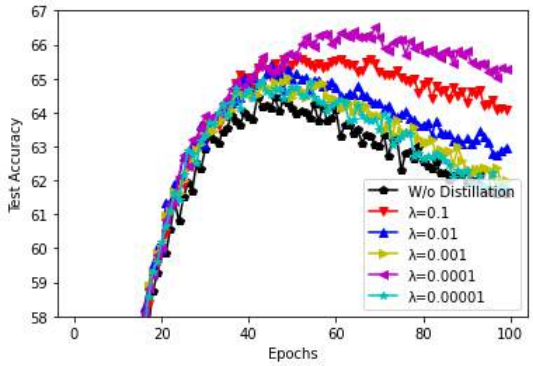


Figure 6: On the parameter λ in eq. (12).

.

4.4. Experimental Results

The first set of experiments setting the mini-batch size to 32 for the first three utilized datasets (i.e., Cifar-10, SVHN-10, and Fashion MNIST) is presented in Table 2. Four different values of nearest neighbors (NN), that is 2NN, 4NN, 8NN and 10NN were used and their performance was compared with the baseline model, that is without applying knowledge distillation. Best results are printed in bold. As we can observe, the proposed online distillation method improves the baseline performance for all the values of nearest neighbors on all the utilized datasets. We can also see that better performance for mini-batch size equal to 32 is accomplished utilizing the maximum considered number of nearest neighbors, that is 10NN.

The corresponding results setting the mini-batch size to 64 for all the utilized datasets are presented in Table 3. Similar remarks are drawn in this set of experiments. The proposed method achieves improved performance over the baseline in any considered case. In this case, we can observe that better results are obtained for fewer nearest neighbors as compared to the previous case, that is considering 4NN and 8NN.

Finally, the evaluation results considering 2NN, 4NN, 8NN and 10NN for mini-batch size set to 128 are provided in Table 4. As we can notice the enhanced

performance of the proposed distillation approach against the baseline is also confirmed in this case. It also shown that as the mini-batch size increases, better performance is achieved utilizing fewer nearest neighbors.

Method	**Cifar-10**	**SVHN-10**	**Fashion MNIST**
W/o Distillation	64.826% ± 0.573%	88.822% ± 0.217%	91.278% ± 0.141%
POSD-2NN	65.508% ± 0.729%	89.223% ± 0.404%	91.770% ± 0.160%
POSD-4NN	65.674% ± 0.526%	89.483% ± 0.279%	91.810% ± 0.115%
POSD-8NN	65.622% ± 0.595%	89.478% ± 0.154%	91.822% ± 0.127%
POSD-10NN	**66.280% ± 1.190%**	**89.512% ± 0.379%**	**91.882% ± 0.108%**

Table 2: Test Accuracy - Batch Size: 32

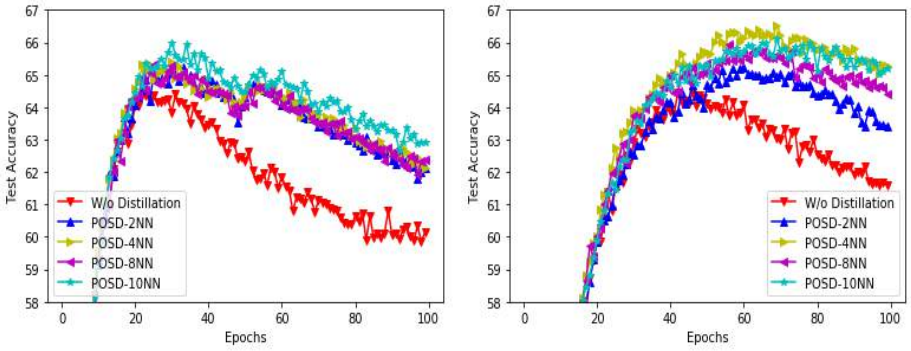
Method	**Cifar-10**	**SVHN-10**	**Fashion MNIST**
W/o Distillation	64.734% ± 0.654%	88.706% ± 0.306%	91.214% ± 0.141%
POSD-2NN	65.472% ± 0.914%	89.625% ± 0.307%	91.624% ± 0.111%
POSD-4NN	**66.782% ± 0.691%**	89.534% ± 0.500%	91.650% ± 0.138%
POSD-8NN	66.140% ± 1.013%	**89.912% ± 0.250%**	**91.730% ± 0.157%**
POSD-10NN	66.336% ± 0.699%	89.522% ± 0.260%	91.548% ± 0.172%

Table 3: Test Accuracy - Batch Size: 64

Method	**Cifar-10**	**SVHN-10**	**Fashion MNIST**
W/o Distillation	65.048% ± 0.620%	88.013% ± 0.083%	91.058% ± 0.130%
POSD-2NN	66.248% ± 0.592%	89.387% ± 0.433%	**91.728% ± 0.175%**
POSD-4NN	66.362% ± 0.726%	**89.946% ± 0.433%**	91.636% ± 0.154%
POSD-8NN	**66.760% ± 0.680%**	89.491% ± 0.363%	91.724% ± 0.121%
POSD-10NN	66.304% ± 0.919%	89.655% ± 0.334%	91.592% ± 0.151%

Table 4: Test Accuracy - Batch Size: 128

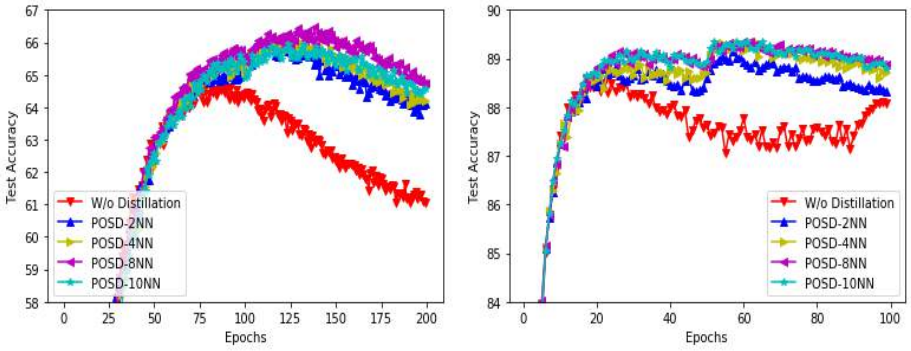
Figs 7-11 illustrate the comparisons of the mean test accuracy over the epochs of training of the proposed method considering 2NN, 4NN, 8NN and 10NN, for the three considered mini-batch sizes. The enhanced performance of the proposed



(a) Cifar-10: Batch Size of 32 samples.

(b) Cifar-10: Batch Size of 64 samples.

Figure 7: Evaluating the test accuracy during the training process for different methods.



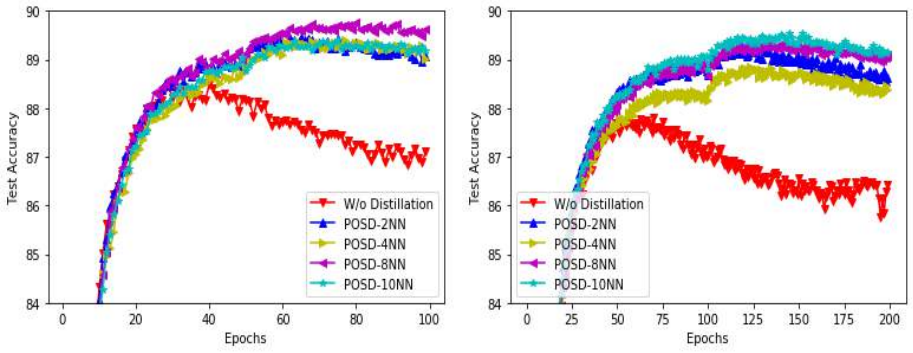
(a) Cifar-10: Batch Size of 128 samples.

(b) SVHN-10: Batch Size of 32 samples.

Figure 8: Evaluating the test accuracy during the training process for different methods

method is depicted.

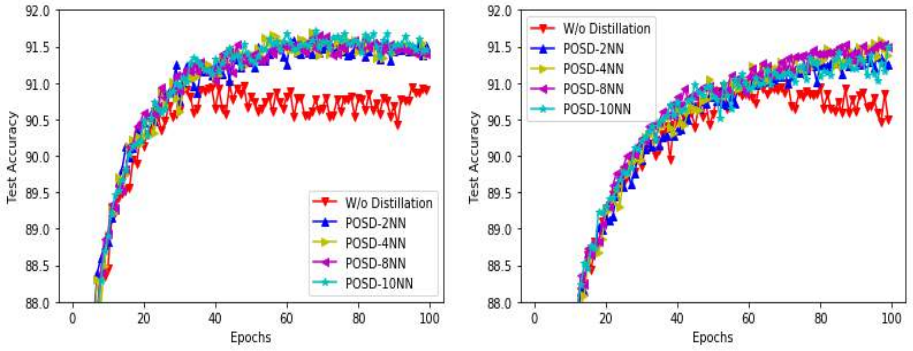
Subsequently, we have conducted representative experiments on datasets consisting of larger images (i.e., 64×64, and 128×128), that is Tiny-ImageNet and Crowd-drone. The experimental results for the Tiny-ImageNet case are provided in Table 5 for different numbers of NNs, while Figure 12 illustrates test accuracy throughout the training process for the baseline and training with the POSD methods with the optimal number of NNs (i.e., 2NNs). Correspondigly, for the Crowd-drone case, the experimental results are provided in Table 6 for batch size of 32 samples, and for various numbers of NNs, while Figure 13 illus-



(a) SVHN-10: Batch Size of 64 samples.

(b) SVHN-10: Batch Size of 128 samples.

Figure 9: Evaluating the test accuracy during the training process for different methods



(a) Fashion MNIST: Batch Size of 32 samples.

(b) Fashion MNIST: Batch Size of 64 samples.

Figure 10: Evaluating the test accuracy during the training process for different methods

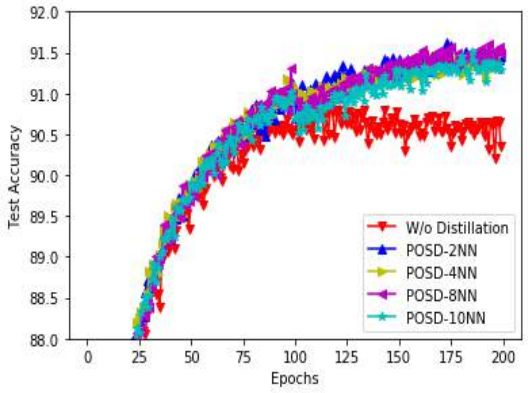


Figure 11: Fashion MNIST: Evaluating the test accuracy during the training process for different methods, batch Size of 128 samples.

Method	**Test Accuracy**
W/o Distillation	29.700% ± 0.510%
POSD - 2NN	**30.639% ± 0.409%**
POSD - 4NN	30.406% ± 0.750%
POSD - 8NN	30.620% ± 0.697%
POSD - 10NN	30.000% ±0.508%

Table 5: Tiny-ImageNet dataset.

trates test accuracy throughout the training process for the baseline and training with the POSD methods with the optimal number of NNs (i.e., 2NNs). From the demonstrated results, it is evident that the POSD method improves the baseline classification performance on datasets of 64×64 and 128×128 input dimensions, too.

Next, we have performed experiments in order to compare the proposed methods with state-of-the-art online distillation methods. More specifically, first we utilize a common architecture, that is WRN-16-2 [13], we apply the proposed online distillation method on Cifar-10 dataset, and compare the performance with the competitive online distillation methods, [51, 53]. In order to ensure a fair comparison, we follow the same training setup as in [53, 13]. That is, we use

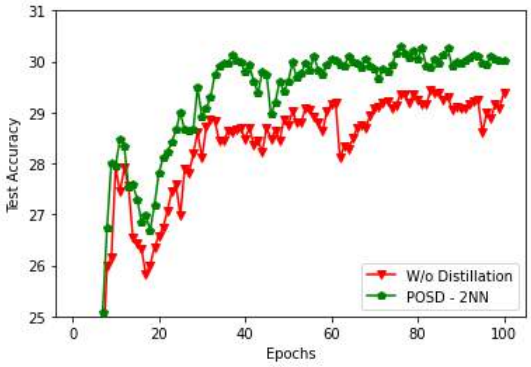


Figure 12: Tiny-ImageNet: Evaluating the test accuracy during the training process for the baseline and training with the POSD methods with the optimal number of NNs.

Method	**Test Accuracy**
W/o Distillation	96.576% ± 0.811%
POSD - 2NN	**97.672% ± 0.269%**
POSD - 4NN	96.743% ± 0.563%
POSD - 8NN	96.823 % ± 0.364%
POSD - 10NN	97.103% ± 0.360%

Table 6: Crowd-drone dataset.

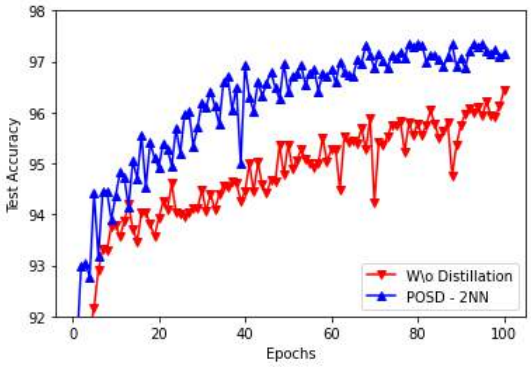


Figure 13: Crowd-drone dataset: Evaluating the test accuracy during the training process for the baseline and training with the POSD methods with the optimal number of NNs.

the SGD with Nesterov momentum and set the momentum to 0.9. The initial learning rate is set to 0.1 and drops by 0.2 at 60, 120 and 160 epochs. Models are trained for 200 epochs using mini-batch of 128 samples.

We compare the proposed method with ONE [51] and FFL [53]. It should be emphasized that the proposed POSD method is a single branch method, that is, it does not utilize multiple copies/branches of the network. Thus, for as much as possible fair comparisons, we use only two sub-networks in all the competitive approaches, similar to [53]. Note that the manuscript in [51] reports the experimental results utilizing three copies of the network. Therefore, we compare the POSD method with ONE distillation method, considering the average performance of the two branches, and correspondingly with the FFL-S distillation method considering the average performance of the two sub-networks. We should note that the number of parameters in both FFL-S and ONE cases is identical to the POSD case, since the additional branches in both cases as well as the fusion module in FFL-S are discarded in the test phase. Furthermore, even we do not follow an ensembling methodology, we also compare the performance of the proposed POSD method with the ensembling methods, that is ONE-E and FFL. We should note that the number of parameters in ONE-E is 1.24M, and 1.29M in FFL, while the number of parameters of POSD is 0.70M, considering WRN 16-2. Evaluation results are presented in Table 7. As it can be observed from the demonstrated results, the proposed online distillation method achieves superior performance over competitive online distillation methods. Best performance is achieved utilizing 2NN. Furthermore, as it is demonstrated that the proposed method can even outperform ensembling methods.

Subsequently, we perform experiments utilizing the ResNet-32 [12] and WRN-28-10 [13] models, in order to compare the performance of the POSD method against online distillation methods (i.e., DML [38]) and also offline self-distillation methods (i.e., SRDL [36]) on Cifar-10 and Cifar-100 datasets. Besides, we provide comparisons against the common offline KD method [30].

Regarding the ResNet-32 model, we use the same experimental setup as in [38] in order to ensure a fair comparison. That is, we use SGD with Nesterov

Method	Test Accuracy
WRN 16-2	93.55% ± 0.11%
ONE [51]	93.76%± 0.16%
FFL-S [53]	93.79% ± 0.12%
ONE-E [51]	93.84%± 0.20%
FFL [53]	93.86% ± 0.11%
POSD	**94.39% ± 0.17%**

Table 7: Comparisons against online distillation methods on Cifar-10 utilizing the WRN 16-2 architecture.

momentum and set the initial learning rate to 0.1, momentum to 0.9 and mini-batch size is set to 64. The learning rate dropped by 0.1 every 60 epochs and we train for 200 epochs. Evaluation results are presented in Table 8 for the Cifar-10 dataset, and in Table 9 for the Cifar-100 dataset. In the case of DML, we report both the average accuracy of the two networks, as well as the accuracy of each one separately. As it is demonstrated, regarding the Cifar-10 dataset, the proposed method achieves superior performance as compared to the competitive ones. Note also that KD [30], using as teacher model the powerful ResNet-110 model achieves accuracy 92.75%, which is also inferior as compared to the proposed POSD method's performance. Regarding the Cifar-100 dataset, the proposed method achieves superior performance as compared to the DML method, however is slightly inferior as compared to the SRDL method. Furthermore, the KD [30] method utilizing as teacher achieves accuracy 71.17%, which is also inferior compared to the proposed method.

Subsequently, we compare the performance against DML [38] and SRDL [36] on Cifar-10 and Cifar-100 datasets, using the WRN-28-10 model [13]. To ensure a fair comparison we use the same experimental setup as in [13]. That is, we use SGD with Nesterov momentum and set the initial learning rate to 0.1, momentum to 0.9 and mini-batch size is set to 128. The learning rate dropped by 0.1 every 60 epochs and we train for 200 epochs. Experimental results are presented in Table

Method	Test Accuracy
ResNet-32	92.47%
DML [38]	92.74% (Net1: 92.68% Net2: 92.80%)
SRDL [36]	93.12%
POSD	**93.16% ± 0.14%**

Table 8: Comparisons against online and offline distillation methods on Cifar-10 utilizing the ResNet-32 architecture.

10 for the Cifar-10 dataset and in Table 11 for the Cifar-100 dataset. As it can be observed the proposed method achieves superior performance against all the compared methods.

Next, we use the WRN-20-08 model to perform experiments on the Cifar-10 and Cifar-100 datasets, and compare the performance of the proposed method with the OKDDip [54], DFL [55], DML [38], ONE [51], and CL-ILR [52] methods. In order to ensure fair comparisons, we use the same setup as described in [54]. That is, the models are trained for 300 epochs, the mini batch size is set to 128, and the learning rate is set to 0.1 and is divided by 10 at 150 and 225 epochs. The experimental results are provided in Tables 12 and 13 for the Cifar-10 and Cifar-100 datasets respectively. As it can be observed, the proposed POSD method accomplishes superior performance over the state-of-the-art online distillation works.

Consequently, the proposed method achieves in general (except one case) superior performance against all the compared online and offline distillation methods, utilizing different baseline models, as well as on different datasets, validating the effectiveness of the proposed method.

Subsequently, we evaluate the complexity of the proposed online distillation method using the sum of floating point operations (FLOPs) in one forward pass on a fixed input size. FLOPs are estimated according to `https://github.com/1adrianb/pytorch-estimate-flops`. Model size, represented by the model's parameters, is also reported for each of the utilized models. We utilize the

Method	Test Accuracy
ResNet-32	68.99%
DML [38]	70.97% (Net1: 71.19% Net2: 70.75%)
SRDL [36]	**71.63%**
POSD	71.31% ± 0.28%

Table 9: Comparisons against online and offline distillation methods on Cifar-100 utilizing the ResNet-32 architecture.

Method	Test Accuracy
WRN-28-10*	95.83%
DML [38]	95.65% (Net1: 95.66% Net2: 95.63%)
SRDL [36]	95.41%
POSD	**96.03% ± 0.11%**

Table 10: Comparisons against online and offline distillation methods on Cifar-10 utilizing the WRN-28-10 architecture. *As reported in [13]

Method	Test Accuracy
WRN-28-10*	79.50%
DML [38]	80.18% (Net1: 80.28% Net2: 80.08%)
SRDL [36]	79.38%
POSD	**80.28% ± 0.28%**

Table 11: Comparisons against online and offline distillation methods on Cifar-100 utilizing the WRN-28-10 architecture. *As reported in [13]

Method	Test Accuracy
WRN-20-8	94.73% ± 0.06%
DML [38]	94.96% ± 0.08%
ONE [51]	94.73% ± 0.02%
CL-ILR [52]	94.88% ± 0.16%
OKDDip [54]	95.16% ± 0.07%
POSD	**96.14%± 0.16%**

Table 12: Comparisons against online distillation methods on Cifar-10 utilizing the WRN-20-8 architecture.

Method	Test Accuracy
WRN-20-8	77.50% ± 0.44%
DML [38]	79.79% ± 0.11%
ONE [51]	78.81% ± 0.12%
CL-ILR [52]	79.56% ± 0.13%
OKDDip [54]	80.37% ± 0.07%
DFL [55]	80.51% ± 0.49%
POSD	**80.80%± 0.10%**

Table 13: Comparisons against online distillation methods on Cifar-100 utilizing the WRN-20-8 architecture.

Method	Teacher	Student	Complexity
KD [30]	ResNet-110 (1.7M)	ResNet-32 (0.5M)	0.33 GFLOPs
POSD	-	ResNet-32 (0.5M)	0.07 GFLOPs
KD [30]	WRN-40-2 (2.26M)	WRN-16-2 (0.7M)	0.43 GFLOPs
POSD	-	WRN-16-2 (0.7M)	0.10 GFLOPs

Table 14: Complexity of the proposed POSD and KD [30] methods using the sum of floating point operations (FLOPs) in one forward pass on a fixed input size utilizing the Cifar-10 dataset. Model size, represented by the model's parameters, is also reported inside parentheses for each of the utilized models.

ResNet-32 and WRN-16-2 models on the Cifar-10 dataset. In order to validate the efficiency of the proposed method, we compare the complexity with the most famous offline KD method, [30]. In this case, for the ResNet-32 student model, we use as teacher the stronger ResNet-110 model. Correspondingly, for the WRN-16-2 student model, we use as teacher the stronger Wide ResNet 40-2 model (abbreviated as WRN-40-2).

Evaluation results are presented in Table 14. As it can be observed from the demonstrated results, the proposed POSD method is significantly more efficient compared to the conventional offline methodology. Furthermore, it should be noted that competitive online distillation methods that utilize multiple branches or copies of a given network, require at least two times more FLOPs than the proposed one. That is, the proposed online distillation method is also more efficient as compared to competitive online methods, too.

Furthermore, on the evaluation of the efficiency of the POSD methodology, we highlight that apart from the common strong models, used mainly for comparison purposes against state-of-the-art online distillation works, we use fast and lightweight models. The proposed models, trained with the POSD methodology are extremely low-memory demanding, considering the memory required for training the models using the proposed training pipeline. More specifically, the required memory to train the lightweight models is 918 MiB in the case

Method	Required Memory
KD (ResNet 110/ResNet32)	3196 MiB
POSD (ResNet 32)	1222 MiB
KD (WRN-40-2/WRN-16-2)	2950 MiB
POSD (WRN-16-2)	1256 MiB

Table 15: Required memory for training common models with the POSD method against conventional KD.

of CIfar-10 and SVHN datasets, and 920 MiB in the case of Fashion MNIST dataset. To gain some more insights on the efficiency with respect to the memory requirements for training with the proposed online methodology, we can compare the performance of the POSD methodology with the conventional offline KD[30] using the ResNet-32 and WRN-16-2 models. The POSD method requires 1222 MiB, while the conventional KD requires 1974 MiB for training first the powerful ResNet-110 model and the 1222 MiB to train the ResNet-32 transferring the knowledge acquired from ResNet-110. Correspondingly, 1256 MiB are required for training with the POSD method using the WRN-16-2 model, while for training first the powerful WRN-40-2 model and transferring the knowledge to the WRN-16-2 model with the offline KD methodology are required 2950 MiB. The overall results are presented in Table 15.

Finally, as it is shown in Table 16, the proposed method requires extremely low memory for storing the trained models weights. Again, to gain some more insights on the efficiency, we can compare the performance of the proposed methodology using e.g., the ResNet-32 model, or WRN-16-2, compared to an offline KD methodology that needs to store a strong teacher to mine additional knowledge, e.g., ResNet-110 or WRN-40-2. The required memory for each of these cases is provided in Table 17. As it is shown, the POSD methodology would provide exceptional performance (as it validated through the experiments with respect to test accuracy) requiring only 3.6MiB, while offline KD would also require 13.4MiB, considering the ResNet case.

Lightweight Model	Required Memory
Cifar-10	248 KiB
SVHN	248 KiB
Fashion MNIST	303.9 KiB

Table 16: Required memory for storing the weights of the lightweight models trained with the proposed POSD methodology.

Model	Required Memory
ResNet-32	3.6 MiB
ResNet-110	13.4 MiB
WRN-16-2	5.4 MiB
WRN-40-2	17.2 MiB

Table 17: Required memory for storing common models trained with the proposed POSD methodology.

5. Conclusions

In this paper, we proposed a novel single-stage online self-distillation approach, namely Probabilistic Online Self-Distillation. The proposed method considers that there are implicit concepts in each classification task expressed with latent labels. These concepts convey useful information about the relationships between data samples. Our goal is to maximize the QMI between data samples and the latent labels. We are able, in this way, to derive additional knowledge directly from the data, without affecting the model architecture by adding multiple branches or employing multiple models, and at the same time in a single stage training pipeline. The experimental evaluation indicates the effectiveness of the proposed method to improve the classification performance.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR).

This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] L. Deng, A tutorial survey of architectures, algorithms, and applications for deep learning, APSIPA Transactions on Signal and Information Processing 3 (2014) e2.

[2] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M. S. Lew, Deep learning for visual understanding: A review, Neurocomputing 187 (2016) 27–48.

[3] M. Tzelepi, A. Tefas, Graph embedded convolutional neural networks in human crowd detection for drone flight safety, IEEE Transactions on Emerging Topics in Computational Intelligence.

[4] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, arXiv preprint arXiv:1612.08242.

[5] M. Tzelepi, A. Tefas, Deep convolutional learning for content based image retrieval, Neurocomputing 275 (2018) 2467–2478.

[6] A. Graves, A. Mohamed, G. E. Hinton, Speech recognition with deep recurrent neural networks, CoRR abs/1303.5778.

[7] N. Passalis, A. Tefas, Deep reinforcement learning for controlling frontal person close-up shooting, Neurocomputing 335 (2019) 37–47.

[8] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in neural information processing systems, 2015, pp. 2377–2385.

[9] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[10] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.

[11] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5927–5935.

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CoRR abs/1512.03385.

[13] S. Zagoruyko, N. Komodakis, Wide residual networks, arXiv preprint arXiv:1605.07146.

[14] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, arXiv preprint arXiv:1710.09282.

[15] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861.

[16] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[18] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size, arXiv preprint arXiv:1602.07360.

[19] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding, in: ICLR, 2016.

[20] G. Huang, S. Liu, L. Van der Maaten, K. Q. Weinberger, Condensenet: An efficient densenet using learned group convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2752–2761.

[21] S. Srinivas, R. V. Babu, Data-free parameter pruning for deep neural networks, arXiv preprint arXiv:1507.06149.

[22] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, arXiv preprint arXiv:1611.06440.

[23] J. Wu, C. Leng, Y. Wang, Q. Hu, J. Cheng, Quantized convolutional neural networks for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4820–4828.

[24] T. Chen, I. Goodfellow, J. Shlens, Net2net: Accelerating learning via knowledge transfer, arXiv preprint arXiv:1511.05641.

[25] W. Chan, N. R. Ke, I. Lane, Transferring knowledge from a RNN to a DNN, CoRR abs/1504.01483.
URL http://arxiv.org/abs/1504.01483

[26] Z. Tang, D. Wang, Z. Zhang, Recurrent neural network training with dark knowledge transfer, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016, pp. 5900–5904.

[27] N. Passalis, A. Tefas, Learning deep representations with probabilistic knowledge transfer, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 268–284.

[28] N. Passalis, A. Tefas, Unsupervised knowledge transfer using similarity embeddings, IEEE Transactions on Neural Networks and Learning Systems 30 (3) (2019) 946–950.

[29] J. Kim, S. Park, N. Kwak, Paraphrasing complex network: Network compression via factor transfer, in: Advances in Neural Information Processing Systems, 2018, pp. 2760–2769.

[30] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, arXiv preprint arXiv:1503.02531.

[31] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, Y. Bengio, Fitnets: Hints for thin deep nets, arXiv preprint arXiv:1412.6550.

[32] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil, Model compression, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, 2006.

[33] J. Ba, R. Caruana, Do deep nets really need to be deep?, in: Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 27, 2014, pp. 2654–2662.

[34] J. Yim, D. Joo, J. Bae, J. Kim, A gift from knowledge distillation: Fast optimization, network minimization and transfer learning, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[35] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, A. Anandkumar, Born again neural networks, in: ICML, 2018.

[36] X. Lan, X. Zhu, S. Gong, Self-referenced deep learning, in: Asian Conference on Computer Vision, Springer, 2018, pp. 284–300.

[37] R. Anil, G. Pereyra, A. T. Passos, R. Ormandi, G. Dahl, G. Hinton, Large scale distributed neural network training through online distillation, 2018. URL https://openreview.net/pdf?id=rkr1UDeC-

[38] Y. Zhang, T. Xiang, T. M. Hospedales, H. Lu, Deep mutual learning, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[39] K. Torkkola, Feature extraction by non-parametric mutual information maximization, Journal of machine learning research 3 (Mar) (2003) 1415–1438.

[40] B. Pan, Y. Yang, H. Li, Z. Zhao, Y. Zhuang, D. Cai, X. He, Macnet: Transferring knowledge from machine comprehension to sequence-to-sequence models, in: Advances in Neural Information Processing Systems, 2018, pp. 6092–6102.

[41] Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, J. Wang, Structured knowledge distillation for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2604–2613.

[42] J. Mun, K. Lee, J. Shin, B. Han, Learning to specialize with knowledge distillation for visual question answering, in: Advances in Neural Information Processing Systems, 2018, pp. 8081–8091.

[43] X. Wang, R. Zhang, Y. Sun, J. Qi, Kdgan: knowledge distillation with generative adversarial networks, in: Advances in Neural Information Processing Systems, 2018, pp. 775–786.

[44] Z. Tang, D. Wang, Y. Pan, Z. Zhang, Knowledge transfer pre-training, CoRR abs/1506.02256.

[45] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, CoRR abs/1510.02192.

[46] S. Zagoruyko, N. Komodakis, Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, CoRR abs/1612.03928.

[47] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, Z. Dai, Variational information distillation for knowledge transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9163–9171.

[48] S. Mirzadeh, M. Farajtabar, A. Li, H. Ghasemzadeh, Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher, CoRR abs/1902.03393.

[49] D. Nguyen, S. Gupta, T. Nguyen, S. Rana, P. Nguyen, T. Tran, K. Le, S. Ryan, S. Venkatesh, Knowledge distillation with distribution mismatch, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2021, pp. 250–265.

[50] G. Xu, Z. Liu, X. Li, C. C. Loy, Knowledge distillation meets self-supervision, in: European Conference on Computer Vision, Springer, 2020, pp. 588–604.

[51] x. lan, X. Zhu, S. Gong, Knowledge distillation by on-the-fly native ensemble, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31, 2018, pp. 7517–7527.

[52] G. Song, W. Chai, Collaborative learning for deep neural networks, Advances in Neural Information Processing Systems 31 (2018) 1832–1841.

[53] J. Kim, M. Hyun, I. Chung, N. Kwak, Feature fusion for online mutual knowledge distillation, CoRR abs/1904.09058. arXiv:1904.09058.
URL http://arxiv.org/abs/1904.09058

[54] D. Chen, J.-P. Mei, C. Wang, Y. Feng, C. Chen, Online knowledge distillation with diverse peers, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 3430–3437.

[55] D. Ni, Dense feature fusion for online mutual knowledge distillation, in: Journal of Physics: Conference Series, Vol. 1865, IOP Publishing, 2021, p. 042084.

[56] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, Journal of machine learning research 7 (Nov) (2006) 2399–2434.

[57] D. W. Scott, Multivariate density estimation: theory, practice, and visualization, John Wiley & Sons, 2015.

[58] S.-T. Chiu, Bandwidth selection for kernel density estimation, The Annals of Statistics (1991) 1883–1905.

[59] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep., Citeseer (2009).

[60] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning.

[61] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747.

[62] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th international conference on machine learning (ICML-10), 2010, pp. 807–814.

7.3 Efficient Multilayer Online Self-Acquired Knowledge Distillation for Image Classification

The appended paper follows.

Efficient Multilayer Online Self-Acquired Knowledge Distillation for Image Classification

Maria Tzelepi, Charalampos Symeonidis, Nikos Nikolaidis *Member, IEEE* and Anastasios Tefas *Member, IEEE*

***Abstract*—Online Knowledge Distillation (KD) has been proposed as a promising approach for circumventing the flaws of the conventional offline KD (that is, time consuming, complex and memory and computationally demanding process). In this paper, a novel method, namely *Multilayer Online Self-Acquired Knowledge Distillation* (MOSAKD) is proposed, aiming to ameliorate the performance of any deep neural model in an online fashion. To accomplish this goal, we use k-nn non-parametric density estimation in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer (either intermediate layers or the output later). This enables us to directly estimate the posterior class probabilities of the data samples and use them as soft labels that explicitly express the similarities of the data with the classes, with negligible computational cost. The experimental validation on six datasets, including a dataset of synthetic images, indicates the effectiveness of the MOSAKD method.**

***Impact Statement*—Knowledge Distillation (KD) is a promising solution for developing both fast and accurate models for various robotics applications. However, conventional KD requires multiple stages of training (training first a powerful model and distilling then the knowledge to a faster) that can be prohibitive for some practical applications. The single-stage, online self-distillation method that we propose in this paper overcomes these limitations. The proposed method improves the baseline on Human Synthetic dataset from 97.048% to 98.592% utilizing a model that runs in real-time on a Jetson TX-2, while it also improves the previous state-of-the-art on Cifar10 from 93.86% to 94.28%, utilizing the WRN 16-2. The proposed method allows for developing effective and fast-to-execute models that could be used on applications with computation and memory restrictions where the accuracy is as important as speed, e.g., human detection towards human avoidance in autonomous unmanned aerial vehicles.**

***Index Terms*—Knowledge Distillation, Online Knowledge Distillation, Self-Distillation, Multilayer, Intermediate Layers**

I. Introduction

OVER the recent years, deep learning (DL) models [1] have been successfully utilized in order to address a wide range of computer vision problems, gradually displacing previous solutions [2], [3]. Following the evolution of DL, it is evident that DL models are becoming increasingly heavier (e.g., the older LeNet-5 model [4] has approximately 60k parameters, while the more recent VGG-16 model [5] has approximately 138M parameters). The increasing complexity of DL models is accompanied by specific computational and memory requirements, obstructing their application for robotics applications. These limitations have fueled the need for developing lightweight, fast, and effective models. Several solutions have been proposed to accomplish this goal [6], ranging from applying pruning techniques [7] to developing lightweight models by design [8]. Amongst them *Knowledge Distillation* (KD) [9] (also known as *Knowledge Transfer*) has emerged as a very effective way to tackle this challenge.

The seminal KD approach refers to the process where a high-capacity and powerful model, known as teacher, which achieves considerable performance, transfers its knowledge to a more compact and fast model, known as student. Thus, apart from the regular supervised loss, the student model is concurrently trained to mimic the so-called *soft labels* produced by the teacher model. To this aim, the parameter of temperature is introduced at the softmax activation function on the output layer of the network. The rationale behind this practice is that, compared to the hard labels, the soft labels relay more information about the way the model learns to generalize, trying implicitly to uncover similarities over the data. Later approaches include the transfer of knowledge from a weaker to a more complex model [10] or from teachers to students of identical capacity [11], [12], [13]. The latter process is known as *self-distillation*.

KD methods can be classified into two broad categories: *offline* and *online* KD. Offline KD refers to the multi-stage process of training first a complex and powerful teacher model and then transferring the knowledge to a simpler and faster student model, which has already been described. Contrariwise, online KD refers to the single-stage process of training the teacher and the student models concurrently, that is by omitting the stage of pretraining the teacher model. In the recent literature, we can find methods that propose to create a powerful teacher from ensembles of multiple copies of a target model and then distill the acquired knowledge to the target model [14], or methods where mutliple models train each other mutually, that is each model serve as teacher model for the rest models [15].

In this work, we propose a single-stage online self-acquired knowledge distillation method, namely *Multilayer Online Self-Acquired Knowledge Distillation* (MOSAKD), for ameliorating the performance of any deep neural model

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

The authors are with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, email: {mtzelepi, charsyme, nnik, tefas}@csd.auth.gr.

for classification tasks. The overarching motivation of this work is to effectively train fast and lightweight models with an extra supervision, apart from the regular supervised loss, that reveals further knowledge beyond the hard labels, from the model itself and also in an online manner, so as to overcome the inherent limitations of offline KD caused by the multi-step training pipeline (i.e., time consuming, complex, and memory and computationally demanding process).

The proposed method is established on the following remarks. First, considering a regular classification task, a probability distribution over the classes is produced by the softmax activation function at the output of neural network. Then, the softmax classifier inherently suppresses all the activations apart from the highest one. Hence, the conventional (offline) KD methodology manifests that the class probability distribution of a powerful teacher carries useful information on the similarities of the data with all the classes, and thus it is advantageous to retain these similarities during the training process, instead of merely training the model with the hard labels [9]. Furthermore, it has been shown that useful information can also be obtained even by transferring the knowledge from a teacher of identical capacity with the student [12]. Finally, it has been demonstrated that compact models usually have the same representation capacity as their heavier counterparts but they are harder to train [16].

Thus, our goal is to obtain additional knowledge about the similarities of the samples with the classes from the model itself and also in an online manner, in order to train fast and lightweight yet effective DL models. To accomplish this goal, we propose to use the k-nn non-parametric density estimation [17] in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either by intermediate layers or by the output layer. In this fashion, as it will be shown, we are capable of directly estimating the posterior class probabilities of the data samples, based on the neighborhood of each sample, and use them as soft labels. These soft labels explicitly express the similarity of each sample with all the classes. It should be emphasized that the proposed method is able to acquire additional knowledge from any layer, and, as shown, useful information can be distilled even from the shallower layers.

Furthermore, it should be noted that, considering a general classification task, when estimating the posterior class probabilities some errors lead to the accumulation of errors, and the goal is to minimize them. On the other hand, in our case, we propose to estimate the posterior class probabilities using the k-nn density estimation, in order to use them as soft labels, i.e., as an auxiliary task to the regular supervised loss. Therefore, our target is not to directly minimize the estimation error for them, but rather to continuously estimating and regarding them during the training procedure, in order to aid the main classification objective. Finally, we anticipate that increasingly meaningful and reliable soft labels are learnt as the training progresses, since the procedure is driven by the supervised loss. This argument is also experimentally confirmed.

The experimental evaluation on six datasets validates the effectiveness of the MOSAKD method. Amongst them, a synthetic dataset for discriminating between humans and non humans has been created. Synthetic data has become an increasingly useful tool in training DL models, accompanied by a series of benefits [18], with a wide range of robotics applications, e.g., [19], [20], [21]. A few of those benefits are that synthetic data 1) provide detailed annotations, since these are automatically generated without containing errors that usually occur in the manual annotation procedure; 2) are usually large in scale, since they are procedurally generated; 3) minimize the risk of DL methods deployed in simulation environments in robotics to exhibit unstable behaviours or complete failures, due to not having been adapted to the visual differences between the virtual and the real world data. In this work, since we are focusing on robotics applications, we use a fully convolutional model which is capable of operating in real-time ($\sim$ 25 Frames Per Second - FPS) utilizing a low-power GPU [22] for high resolution input. The target is to provide semantic heatmaps of human presence on real data. That is, we train the real-time model using the proposed online distillation method on the synthetic data (only 100 real human images were used), and we test the model on unseen images that contain real humans, providing semantic heatmaps, as explained in [22].

The main contributions of this paper can be summarized as follows:

- We propose a novel *Multilayer Online Self-Acquired Knowledge Distillation* method.
- The MOSAKD method mines additional knowledge in an online manner from the model itself, rendering it more efficient. That is, the MOSAKD method does not require using multiple models to teach each other or copies of a given model like the existing online KD methods that leads to multiple times more computationally expensive training processes.
- The MOSAKD method is capable of mining additional information about the similarities of the samples with the classes employing intermediate and the output layers.
- The proposed method can be applied to various architectures varying from lightweight models to more complex ones.
- The self-nature of the MOSAKD method ensures that advantageous knowledge will be extracted which is a crucial issue, since the compatibility between the teacher and student models can significantly affect the effectiveness of the distillation procedure, as it is demonstrated in [23].
- The experimental evaluation on six dataset, including a dataset of synthetic data, validates the MOSAKD method can ameliorate the classification performance of simple and deeper models, while comparison results confirm the superiority of the method over the state-of-the-art methods.

This article is an extended version of our previous work presented in [24]. We further extended our previous work by: 1) extending the proposed methodology on various (intermediate) layers rather than the output layer of the model; 2) providing additional experiments on a new synthetic dataset, constructed for discriminating between humans and non humans; 3) performing additional extensive experiments, utilizing various layers and setups (number of nearest neighbors for computing the soft labels); 4) performing a post-hoc Bonferroni test for evaluating the statistical significance of the obtained results; 5) performing additional experiments utilizing different models (i.e., Wide ResNet 16-2, Wide ResNet 20-08) and an additional dataset (i.e., Cifar-100) for comparing the performance of the proposed MOSAKD method with state-of-the-art online distillation methods (OKDDip [25] , DFL [26], CL-ILR [27], DML[15], ONE[14], FFL[28]); 6) evaluating the efficiency of the proposed online method against conventional offline methodology under different aspects; 7) providing qualitative results utilizing the proposed model trained on synthetic human data; 8) thoroughly discussing the relevant literature.

The rest of the paper is organized as follows. Section II discusses relevant online and offline distillation works. Section III presents in detail the proposed MOSAKD method. Subsequently, in Section IV the experiments performed to evaluate the proposed method are provided, and finally the conclusions are drawn in Section V.

II. Relevant Work

In this Section, we first summarize recent works in the general area of KD, subsequently recent self-distillation methods are presented, and finally recent online KD works, are discussed.

KD has been extensively researched in recent years [29], [30]. The idea of transferring the knowledge from a powerful teacher model to a weaker student model by forcing the latter to mimic the soft labels produced by the teacher by appropriately raising the temperature of the softmax activation function on the output layer of the network, was initially introduced in [31] and subsequently expanded in [9]. Subsequently, the idea of KD [9] was extended so as to allow for compressing wide and deep models to thinner and deeper ones, by using not only soft labels but also intermediate-level hints from the teacher's hidden layers so as to guide the training of the student model, in [32]. A method where the parameters of the student model are initialized as specified by the parameters of the teacher model is proposed in [33], while a method that forces the student model to mimic the teacher's model attention map is proposed in [34].

In addition, [35] formulates the knowledge transfer as maximizing the mutual information between the student and the teacher models, while a multistep approach which uses an intermediate-sized model in order to eliminate the gap between the student and teacher models is proposed in [23], since, as it is demonstrated, when the gap between the teacher and student models is large, the performance degrades. Next, a KD method that models the information flow through various layers of the teacher model and overcomes several limitations of previous approaches especially when utilized for training very lightweight students that significantly differ from teacher, is presented in [36]. Finally, an effective KD method even when there is a distribution mismatch between teacher's and student's training data is proposed in [37]. That is, the method first learns a distribution based on student's training data from which images well-classified by the teacher are sampled. In this way, the data space where the teacher has good knowledge to transfer is discovered. In addition, a new loss function is proposed for training the student network.

Surveying the relevant literature, we also come across several self-distillation approaches. For example, a self-distillation approach where a teacher model is initially trained, and when it convergences, an identical student model is trained both with the targets of the hard labels and matching the output of the teacher model, is proposed in [12]. Similarly, a given model is trained with a conventional supervised loss, the self-discovered knowledge is extracted, and then in the second stage of training, the model is trained both with the supervised loss and the distillation loss, in [13]. Finally, a framework, named Self-Supervised Knowledge Distillation, proposes to employ self-supervised tasks in order to acquire richer knowledge from the teacher model to the student model in [38]. Moreover, the impact of various self-supervised pretext tasks and the effect of noisy self-supervised predictions to the distillation performance are investigated.

In the recent literature, several works proposing online distillation have also been emerged. First, co-distillation ameliorates the accuracy by training k copies of a target model in parallel, by introducing a distillation term to the loss function of the i-th model in order to mimic the average prediction of the rest models [39]. In a quite similar approach in [15], multiple student models teach each other during the training procedure. More specifically, every student, apart from the regular supervised loss, is trained with a distillation loss that matches each student's posterior class probabilities with the class probabilities of the rest students. In this manner, each model operates as a teacher of the rest models.

Subsequently, in [14] a multi-branch version of a given network is created by introducing identical branches, each of them being an independent classification model with shared low level layers. In this way a strong teacher model is created using a gated logit ensemble of the multiple branches. Each branch is trained with the regular supervised loss and the distillation loss which matches the teacher's output distributions. Next, a framework of collaborative learning which trains several classifier heads of the same network at the same time, on the same training data, is proposed in [27]. More specifically, the framework generates a population of classifier heads during the training process, where each head learns, apart from the hard labels, from the soft labels produced by the whole population. Furthermore,

TABLE I: SUMMARY OF RELATED KD METHODS

Type	Methods
Online Distillation	[15], [27], [14], [28], [39], [25], [26]
Offline Distillation	[9], [33], [40], [41], [12], [42], [36], [38], [32], [34], [43], [44], [45], [46]
Self Distillation	[12], [39], [14], [35], [13], [11], [38]
Knowledge Source	**Methods**
Output Layer	[12], [9], [40], [41], [15], [14], [28], [47], [39], [48], [49]
Intermediate Layers	[32], [34], [43], [44], [45], [46], [42], [35], [36], [26]

the method involves an intermediate-level representation sharing with backpropagation rescaling that aggregates the gradient flows for all the heads.

Next, an online distillation method that combines the previous works [15] and [14] is proposed in [28]. More specifically, the method proposes an online mutual knowledge distillation method for improving the performance of both the fusion module and the sub-networks. More specifically, when identical sub-networks are used, the low level layers are shared, and a multi-branch architecture similar to [14] is used, while when different sub-networks are used, the sub-networks are trained similar to [15]. The architecture consists of an ensemble classifier using the ensemble logit produced from the sub-networks and a fused classifier, using the fused feature map. The model mines knowledge from the ensemble classifier to the fused classifier, and at the same time from the fused classifier to each sub-network classifier.

Subsequently, a two-level distillation methodology, named Online Knowledge Distillation with Diverse peers, where two types of students are involved, i.e., multiple auxiliary peers and one group leader, is proposed in [25]. Distillation is conducted among auxiliary peers with a mechanics for preserving diversity, and then an ensemble of predictions of these peers is further distilled to the group leader. Finally, based on [28], a method named Dense Feature Fusion for Online Mutual Knowledge Distillation, where the mid-level features from the subnetworks are also fused, is proposed in [26].

Finally, a summary of indicative previous KD methods along with information on their key attributes, is presented in Table I. The Table discriminates between online and offline KD methods, while it also provides information on self-distillation methods. It also discriminates previous KD methods with regard to the carrier of the knowledge (i.e., output and intermediate layers).

III. PROPOSED MOSAKD METHOD

In this paper, a novel self-distillation method is proposed that allows for developing fast-to-execute yet effective models that can comply with applications (e.g., robotics applications) with memory and computational restrictions. The proposed method allows for acquiring additional knowledge about the similarities of the samples with the classes from the model itself and also in an online manner.

More specifically, a C-class classification problem, and the labeled data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \Re^D$ is an input vector and D its dimensionality, while $\boldsymbol{y}_i \in \mathcal{Z}^C$ corresponds to its C-dimensional one-hot class label vector are considered. We also consider, for an input space $\mathcal{X} \subseteq \Re^D$ and an output space $\mathcal{F} \subseteq \Re^C$, as $\phi(\cdot;\mathcal{W}) : \mathcal{X} \rightarrow \mathcal{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of parameters $\mathcal{W} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{N_L}\}$, where $\boldsymbol{W}_L$ are the weights of a specific layer L, which transforms its input vector to a C-dimensional probability vector. That is, for an input vector $\boldsymbol{x}_i \in \mathcal{X}$, $\phi(\boldsymbol{x}_i;\mathcal{W}) \in \mathcal{F}$ is the output vector given by the network ϕ with parameters $\mathcal{W}$.

Thus, considering the regular classification task, the goal is to find the parameters $\mathcal{W}^*$ that minimize the cross entropy loss, ℓ_{ce}, between the output vector $\phi(\boldsymbol{x}_i;\mathcal{W})$ and the one-hot class label vector $\boldsymbol{y}_i$:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{i=1}^{N} \ell_{ce}(\boldsymbol{y}_i, \phi(\boldsymbol{x}_i;\mathcal{W})), \tag{1}$$

The cross entropy loss for a sample i is defined as:

$$\ell_{ce}(\boldsymbol{y}_i, \boldsymbol{z}_i) = \sum_{m=1}^{C} y_i^m \log(z_i^m), \tag{2}$$

where y_i^m is the m-th element of $\boldsymbol{y}_i$ one-hot label vector, and z_i^m is defined as the output of the softmax operation on the C-dimensional network's output:

$$z_i^m = \frac{\exp(\phi(\boldsymbol{x}_i;\mathcal{W})^m)}{\sum_{j=1}^{C}\exp(\phi(\boldsymbol{x}_i;\mathcal{W})^j)}. \tag{3}$$

In this paper, our goal is to mine additional knowledge from the model itself in an online fashion. To achieve this goal, we propose to use k-nn non-parametric density estimation [17] in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either any intermediate layer or the output layer. The idea of non-parametric density estimation is based, in general, on the fact that the probability P that a vector $\boldsymbol{x}$ will fall in a region R is given by:

$$P = \int_R p(\boldsymbol{x}')d\boldsymbol{x}'. \tag{4}$$

Thus, P is a smoothed version of the density function $p(\boldsymbol{x})$ and this smoothed value of p can be estimated by estimating the probability P. If we consider that N samples $\{\boldsymbol{x}_i\}_{i=1}^N$ are drawn independently and identically distributed according to $p(\boldsymbol{x})$, then the probability that k of these N lie inside in R is given by the binomial law:

$$P_k = \binom{N}{k} P^k (1-P)^{N-k}, \tag{5}$$

and the expected value for k is given by:

$$\mathbb{E}[k] = NP. \tag{6}$$

Furthermore, this binomial distribution for k peaks very sharply about the mean, thus we expect that the ratio $\frac{k}{N}$ will be a good estimate for the probability P, and hence for the smoothed density function. The estimate becomes more accurate as N increases. Making also the assumption that

$p(\boldsymbol{x})$ is continuous and the region R is so small that p does not significantly vary within it, we arrive at the equation:

$$\int_R p(\boldsymbol{x}')d\boldsymbol{x}' \simeq p(\boldsymbol{x})V, \tag{7}$$

where $\boldsymbol{x}$ is a point within R and V is the volume enclosed by R. By combining Eq. (4), (6), and (7) we arrive at the following estimate for $p(\boldsymbol{x})$:

$$p(\boldsymbol{x}) = \frac{\frac{k}{N}}{V}. \tag{8}$$

This equation can be, in general, exploited in two different manners: either to define the number of nearest neighbors, k, and to adjust the volume V from the data, which stands for the k-nn density estimation, or to define the volume V and to observe how many points k fall into the region, which gives rise to the parzen windows. The essential advantage of the first one is that it allows for directly estimating the posterior probabilities $P(c_m|\boldsymbol{x})$ of the class being c_m, $m = \{1, \cdots, C\}$, from a set of N labeled data by using the samples to estimate the densities involved. To do this, we apply the k-nn density estimation to each class separately and then we make use of the Bayes rule. More specifically, assuming that we have a dataset consisting of N_m samples that belong to class c_m, with N samples in total, so that $\sum_m N_m = N$, and we place a sphere of volume V around $\boldsymbol{x}$ and capture k samples, k_m of which belong to the class c_m. Then, the estimate for the probability $p(\boldsymbol{x}|c_m)$ is:

$$p(\boldsymbol{x}|c_m) = \frac{k_m}{N_m V}, \tag{9}$$

and similarly the unconditional density is given by:

$$p(\boldsymbol{x}) = \frac{k}{NV}, \tag{10}$$

while the priors can be approximated by:

$$P(c_m) = \frac{N_m}{N}. \tag{11}$$

Thus, we the Bayes rule to compute the posterior class probabilities:

$$P(c_m|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|c_m)P(c_m)}{p(\boldsymbol{x})} = \frac{\frac{k_m}{N_m V}\frac{N_m}{N}}{\frac{k}{NV}} = \frac{k_m}{k}. \tag{12}$$

Thus, considering the output of a specific layer, L, and for a specific number of nearest neighbors, k, at the feature space generated by the layer L, the posterior probabilities can be estimated and used as soft labels. These soft labels explicitly encode information about the similarities of the samples with the classes. That is, the soft label for a sample i is given by:

$$\boldsymbol{s}_i^L = [\frac{k_1^L}{k}, \frac{k_2^L}{k}, \ldots, \frac{k_C^L}{k}], \tag{13}$$

where k_m^L, $m = \{1, \cdots, C\}$ is the number of nearest neighbors that belong to the class c_m, $m = \{1, \cdots, C\}$ at the layer L.

Therefore, in the MOSAKD training procedure, the goal is to find the parameters $\mathcal{W}^*$ that minimize the overall loss of cross entropy, ℓ_{ce} and self-distillation, ℓ_{sd}:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{i=1}^{N} [\ell_{ce}(\boldsymbol{y}_i, \phi(\boldsymbol{x}_i; \mathcal{W})) + \lambda \ell_{sd}(\boldsymbol{s}_i^L, \phi(\boldsymbol{x}_i; \mathcal{W}))], \tag{14}$$

where λ controls the relative importance between the two losses.

In this paper, we use Mean Squared Error (MSE) as self-distillation loss, however Kullback-Leibler (KL) divergence could also be used. The optimization process is summarized in Algorithm 1.

Consequently, in the MOSAKD training process the input images are fed to the network, and for each sample the predictions for belonging to each of the classes are produced. Concurrently, the soft labels are computed based on the neighborhood of each sample, considering the feature space generated by a specific neural layer, according to the procedure described above. Then, the network is trained using the cross entropy loss with the hard labels, and concurrently using the distillation loss to regress the produced soft labels, encouraging it to regard the similarity of each sample with the classes. The MOSAKD training procedure is also illustrated in Fig. 1.

Algorithm 1: MOSAKD Training

Input: Labeled data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$; Training epochs T;
Parameters: Learning rate η; Parameter λ; Number of nearest neighbors k;
Output: Trained model ϕ

Initialization: t=1; Randomly initialize model ϕ parameters;
while $t \leqslant T$ **do**
Compute the prediction of the network ϕ using Eq. (3);
Compute the soft labels using Eq. (13);
Compute the overall loss using Eq. (14);
Update the model ϕ parameters by the mini-batch gradient descent algorithm;
end

IV. Experiments

A. Datasets

Six datasets are used in order to validate the effectiveness of the proposed MOSAKD method: *Cifar-10*, *Cifar-100* [50], Street View House Numbers (SVHN) [51], Fashion MNIST [52], Tiny ImageNet [1], and Synthetic Human datasets.

The synthetic human dataset consists of real background images populated with 3D human models in various poses. The human models were generated using PIFu [53], a state-of-the-art deep learning method for generating realistic 3D human models from single-view images. Overall, the dataset contains 133 human models, generated using full-body images of people from the Clothing Co-Parsing (CCP)

[1] https://tiny-imagenet.herokuapp.com/

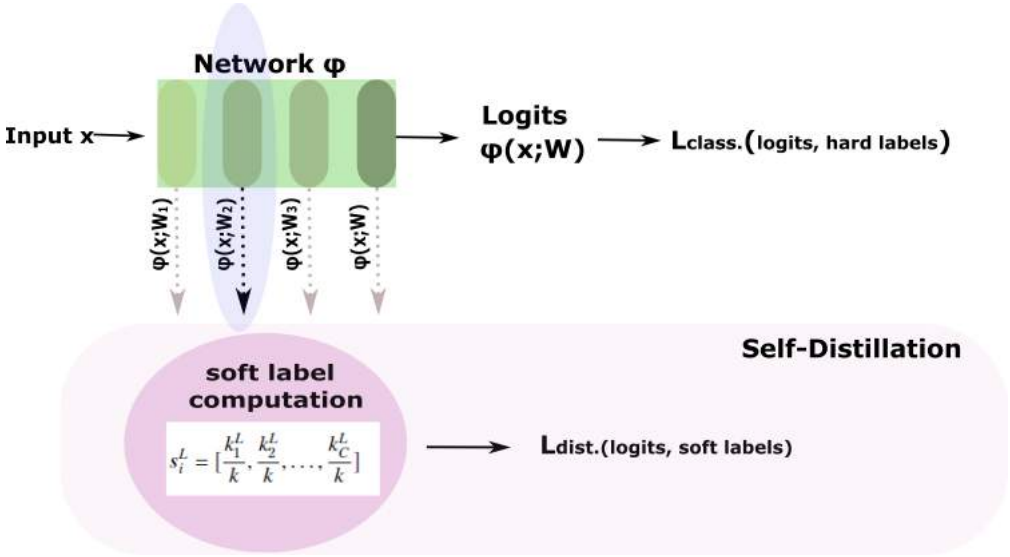


Fig. 1: MOSAKD Training Procedure: The input images are fed to the network, and for each sample the predictions for belonging to each of the classes are produced. Concurrently, considering a specific neural layer, the soft labels are computed based on the neighborhood of each sample in the feature space generated by the specific layer. Then, the network is trained at the same time using the cross entropy loss with the hard labels and using the distillation loss with the produced soft labels.

[54] dataset as PIFu's input. The background images were taken from the Cityscapes [55] dataset which is composed of video sequences depicting street scenes in various cities. To achieve a higher level of realism, the 3D human models are placed on potential 2D image locations (e.g., pavements, roads), based on coarse annotations for semantic image segmentation provided by Cityscapes.

Since, as mentioned previously, the target is to train models that can run in real-time on high-resolution input for producing heatmaps of human presence [22], we crop the generated images, and we construct a train set of 19,900 synthetic cropped images containing humans and 100 real images depicting humans, which are derived from the CUHK Person Re-identification datasets [56], [57]. Furthermore, the train set contains 20,000 non human images, cropped from images of the Cityscapes dataset. The test set consists of 5,000 real images containing humans and 5,000 real images without humans, cropped from video frames that were gathered by querying YouTube[2] video search engine with random keywords. The cropped images are of size 64×64. Sample images of the constructed dataset are provided in Fig. 2.

[2]http://www.youtube.com/

Fig. 2: Sample images of Synthetic Human dataset.

B. CNN Models and Utilized Layers

The target of this work is to evaluate the effect of the proposed multilayer online self-distillation method on training lightweight models that can be effectively deployed on embedded devices. To this end, in the first set of experiments we utilize lightweight models, apart from the case of Tiny ImageNet dataset, which is more challenging. More specifically, in the case of the first two datasets, a lightweight five-layer CNN model is used, comprising of 63K parameters. The first two layers are convolutional with 6 filters of size 5×5 and 16 filters of size 5×5 respectively, while the last three layers are fully connected ($128 \times 64 \times 10$). A 2×2 max-pooling layer with a stride of 2 follows the convolutional layers. Each convolutional and fully connected layer, apart from the last one, is followed by a ReLU activation, while the last fully connected layer is a 10-way softmax layer. In the conducted experiments we use all the layers for computing

the soft labels. The first convolutional layer is denoted as *Layer 1* in the experimental results, and correspondingly the last fully connected layer is denoted as *Layer 5*.

In the case of the Fashion MNIST dataset a lightweight four-layer model is utilized, comprising of 77K parameters. More specifically, the model consists of two convolutional layers with 20 filters of size 5×5 and 50 filters of size 5×5, and two fully connected layers (64×10). A 2×2 max-pooling layer with a stride of 2 follows the convolutional layers. A ReLU activation is applied on all the layers apart from the output layer, which is a 10-way softmax layer. As previously, we use all the layers for the soft label computation. The first convolutional layer is denoted as *Layer 1* in the experimental results, and correspondingly the last fully connected layer is denoted as *Layer 4*.

In the case of Synthetic Human dataset, we use the fully convolutional lightweight VGG-1080p model [22] comprising of 11K parameters. The model, which consists of five convolutional layers, runs in real-time for input 1080p on a low-power Jetson TX-2. We use all the convolutional layers for the soft labels computation, where the first convolutional layer is denoted as *Layer 1* in the experimental results, and correspondingly the last convolutional layer is denoted as *Layer 5*.

In the case of Tiny ImageNet dataset, we use the powerful ResNet-50 [58] architecture (without utilizing any pre-trained model), since it is a challenging dataset. This also validates our claim that the proposed MOSAKD method is model agnostic.In our experiments, we use the last fully connected layer denoted as *Layer 5*, and the output of the four convolutional blocks before the aforementioned last fully connected layer, where the last convolutional block is denoted as *Layer 4*, and so on.

Finally, we use the Wide ResNet 16-2 (WRN-16-2) [59], and Wide ResNet 20-8 (WRN-20-8) models [59] in order to compare the proposed method against state-of-the-art online KD approaches on Cifar-10 and Cifar-100 datasets. In the conducted experiments for both the WRN-16-2 and WRN-20-08 models, we utilize the last fully connected layer denoted as *Layer 4*, and the output of the four convolutional blocks before the last fully connected layer, where the last convolutional block is denoted as *Layer 3*, and so on.

C. Experimental Setup and Evaluation Metrics

Four set of experiments are performed. First, we evaluate the performance of the MOSAKD method using different layers of the utilized models, and different numbers of nearest neighbors for computing the soft labels, using mainly lightweight models (apart from the case of Tiny ImageNet, where ResNet-50 was used, as previously explained). Classification accuracy (test accuracy) is used as evaluation metric (each experiment is repeated five times, and the mean value of classification accuracy and the standard deviation are reported).

Subsequently, in the second set of experiments, we compare the performance of the MOSAKD method with state-of-the-art online distillation methods, using more complex models. Next, in the third set of experiments, we evaluate the efficiency of the proposed method using different metrics. More specifically, we evaluate the complexity (training cost) utilizing the sum of floating point operations (FLOPs), the model sizes using the model parameters, and the memory requirements for training. Finally, in the fourth set of experiments, we provide some qualitative results using the real-time model, trained mainly on synthetic human images (only 100 real human images are used). The trained model is used to generate heatmaps of human presence on real high-resolution test images.

D. Implementation Details

In this work, we train our networks using the mini-batch gradient descent with mini-batch of 64 samples, and we set the momentum to 0.9 and the learning rate to 10^{-3}. In addition, we set the parameter λ in eq. (14) for controlling the relative importance between the classification and the distillation losses to 0.1 for all the datasets except for the Tiny ImageNet where it is set to 0.01. The method is implemented in Pytorch, and the models are trained for 100 epochs on an NVIDIA 2080 Ti with 11GB of GPU memory.

E. Experimental Results

In the following subsections the four performed sets of experiments are presented.

1) Evaluation on lightweight models: The experimental results for evaluating the performance of the MOSAKD method using the lightweight models on the Cifar-10, SVHN, Fashion MNIST, Tiny ImageNet, and Synthetic Human datasets are presented in Tables II-VI respectively. We utilize five different numbers of nearest neighbors for the soft label computation, i.e., 4NN, 8NN, 12NN, 16NN, and 20NN. Best results, considering the different utilized layers, for each specific number of nearest neighbors are printed in bold. Furthermore, the best performance in each dataset is also underlined. In Table VII we provide the best performance achieved using the MOSAKD method against the baseline training without distillation for the five utilized datasets.

From the demonstrated results, it is obvious that the proposed online distillation method notably ameliorates the baseline performance of training without distillation on all the utilized datasets, using different layers and numbers of nearest neighbors to compute the soft labels. Interestingly, considerably good performance can be achieved utilizing the first layers for the soft label computation, apart from the output layer which is mainly used in the KD methodologies, while in some cases, e.g., in the case of Fashion dataset using 20NN, the best performance is obtained using the first layer. However, we can not draw any conclusion on the optimal layer for computing the soft labels, since it is evident that, depending on the dataset, both first and output layers convey useful information that allow for producing reliable soft labels.

Furthermore, it can be observed that the proposed method considerably ameliorates the performance on all utilized

TABLE II: Cifar-10: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: 64.734% ± 0.654%.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	66.204% ± 0.768%	66.076% ± 0.241%	65.818% ± 0.560%	65.994% ± 0.659%	66.022% ± 0.469%
Layer 2	**66.307% ± 0.687%**	65.660% ± 0.710%	**66.756% ± 0.953%**	66.423% ± 1.100%	**66.584% ± 0.742%**
Layer 3	66.150% ± 0.593%	65.752% ± 0.534%	65.318% ± 1.035%	65.856% ± 0.638%	66.304% ± 0.883%
Layer 4	65.210% ± 0.764%	**66.176% ± 0.979%**	66.561% ± 0.635%	66.468% ± 0.587%	65.332% ± 0.511%
Layer 5	66.056% ± 0.808%	65.729% ± 0.424%	65.962% ± 0.841%	**66.570% ± 0.563%**	66.472% ± 0.260%

TABLE III: SVHN: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: 88.706% ± 0.306%.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	89.339% ± 0.257%	89.393% ± 0.243%	89.175% ± 0.552%	89.283% ± 0.303%	**89.583% ± 0.126%**
Layer 2	89.334% ± 0.503%	**89.619% ± 0.329%**	89.275% ± 0.219%	89.568% ± 0.460%	89.311% ± 0.245%
Layer 3	89.599% ± 0.249%	89.606% ± 0.104%	**89.644% ± 0.203%**	**89.736% ± 0.187%**	89.449% ± 0.573%
Layer 4	89.540% ± 0.256%	89.256% ± 0.388%	89.612% ± 0.234%	89.708% ± 0.279%	89.376% ± 0.192%
Layer 5	**89.717% ± 0.210%**	89.318% ± 0.204%	89.560% ± 0.198%	89.422% ± 0.330%	89.036% ± 0.257%

TABLE IV: Fashion MNIST: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: 91.214% ± 0.141%.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	91.256% ± 0.167%	91.249% ± 0.230%	91.324% ± 0.159%	91.250% ± 0.128%	**91.530% ± 0.117%**
Layer 2	**91.332% ± 0.150%**	91.122% ± 0.070%	**91.384% ± 0.159%**	91.302% ± 0.143%	91.297% ± 0.082%
Layer 3	91.330% ± 0.149%	91.226% ± 0.096%	91.258% ± 0.168%	**91.390% ± 0.111%**	91.342% ± 0.180%
Layer 4	91.316% ± 0.089%	**91.300% ± 0.097%**	91.280% ± 0.120%	91.250% ± 0.103%	91.338% ± 0.146%

TABLE V: Tiny ImageNet: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: 31.050% ± 1.550%.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	**32.196% ± 0.831%**	31.131% ± 0.876%	31.270% ± 1.022%	31.695% ± 0.724%	31.651% ± 0.366%
Layer 2	31.883% ± 0.650%	31.101% ± 1.120%	31.236% ± 0.551%	**32.245% ± 0.527%**	31.320% ± 0.662%
Layer 3	32.166% ± 0.579%	31.215% ± 1.020%	31.800% ± 0.811%	31.381% ± 0.741%	31.588% ± 1.201%
Layer 4	31.412% ± 0.672%	**32.070% ± 0.600%**	**31.895% ± 0.635%**	31.588% ± 0.788%	**31.854% ± 0.564%**
Layer 5	31.401% ± 1.106%	31.631% ± 0.519%	31.357% ± 1.045%	31.556% ± 0.643%	31.380% ± 0.753%

TABLE VI: Synthetic Human: Classification Accuracy of the Proposed MOSAKD Method Utilizing Different Layers and Number of Nearest Neighbors. Baseline of Training Without Distillation: 97.048% ± 0.476%.

Layer	4NN	8NN	12NN	16NN	20NN
Layer 1	97.052% ± 0.716%	**98.275% ± 0.550%**	98.192% ± 0.801%	98.223% ± 1.288%	**98.592% ± 0.744%**
Layer 2	97.866% ± 0.684%	98.096% ± 1.129%	**98.322% ± 0.738%**	97.914% ± 0.641%	97.982% ± 0.672%
Layer 3	97.515% ± 0.636%	96.814% ± 1.916%	97.084% ± 1.113%	98.046% ± 0.908%	97.842% ± 1.009%
Layer 4	**97.982% ± 0.882%**	97.264% ± 1.482%	98.192% ± 1.082%	97.404% ± 1.129%	98.426% ± 0.981%
Layer 5	96.962% ± 0.997%	97.946% ± 0.899%	98.268% ± 0.556%	**98.376% ± 0.611%**	97.960% ± 0.711%

TABLE VII: Classification Accuracy of the Best Performance Achieved Using the Proposed MOSAKD Method Against the Baseline Training Without Distillation for All the Utilized Datasets.

Dataset	Baseline (W/o Distillation)	MOSAKD
Cifar-10	64.734% ± 0.654%	**66.756% ± 0.953%**
SVHN	88.706% ± 0.306%	**89.736% ± 0.187%**
Fashion MNIST	91.214% ± 0.141%	**91.530% ± 0.117%**
Tiny ImageNet	31.050% ± 1.550%	**32.245% ± 0.527%**
Synthetic Human	97.048% ± 0.476%	**98.592% ± 0.744%**

datasets regardless of the number of classes. That is, it improves the performance of 2-class, 10-class, and 200-class problems. In the case of the latter ones, inherently rich information about the similarities with the classes can be conveyed, leading to improved performance. However, it is evident that the proposed method can acquire useful information in the case of binary problems too, considerably improving the performance. This can be attributed to the contextual information that the soft label reveals. Additionally, it can be observed that more reliable soft labels, that convey more useful information about the similarities of the samples with the classes, can be achieved using larger numbers of nearest neighbors for the soft label computation, that is 16NN and 20NN in four out of five datasets.

Subsequently, the proposed method can also be applied on two (or more) different layers at the same time. In this case, two distillation losses are added to the overall loss. This

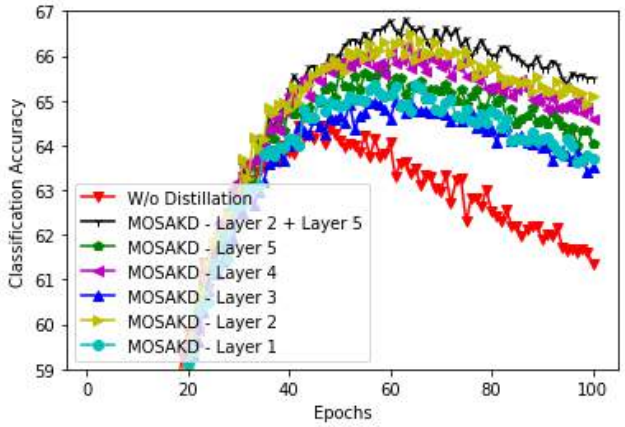


Fig. 3: Cifar-10 Dataset: Evaluating the classification accuracy throughout the training epochs utilizing 12NN, and different layers for computing the soft labels.

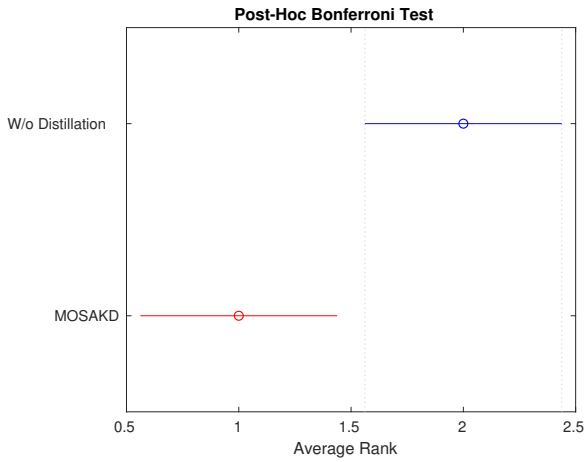


Fig. 4: Post-Hoc Bonferroni Test

TABLE VIII: Cifar-10 Dataset: Classification Accuracy for different values of parameter λ in eq. (14) using 12NN for the soft label computation. Baseline of Training Without Distillation: 64.734% ± 0.654%.

Layer	$\lambda = 0.1$	$\lambda = 0.01$
1	65.818% ± 0.560%	65.797 ± 0.495
2	66.756% ± 0.953%	65.202 ± 0.833
3	65.318% ± 1.035%	65.789 ± 0.632
4	66.561% ± 0.635%	65.520 ± 1.041
5	65.962% ± 0.841%	65.840 ± 0.745

strategy can in some cases further improve the performance achieved by utilizing each of the layers separately. For example, in the case of the Cifar-10 dataset, we can achieve classification accuracy 67.118±0.251 by combining the second and fifth layer, using 12NN, which is the best performance on Cifar-10 dataset. In Fig. 3 the curves of mean classification accuracy, utilizing 12NN, and various layers for computing the soft labels are illustrated.

Furthermore, as it has been previously mentioned, the parameter λ in eq. (14), for controlling the balance between the classification and the distillation losses is set to 0.1, since it generally performs well. However, it should be noted that different setups can be utilized, since the MOSAKD method improves the baseline performance for different values of the parameter, too. For example, in Table VIII we provide the experimental results utilizing different values of the parameter λ on Cifar-10 dataset, utilizing 12NN. As it is shown, the proposed online distillation method improves the classification accuracy in all the considered cases. Finally, as it is mentioned MSE is used for the distillation loss, however KL divergence can also be utilized. For example, utilizing the KL divergence we achieve classification accuracy 65.578±0.170, utilizing 8NN and the *Layer 4* (baseline w/o distillation: 64.734±0.654), while utilizing the MSE we achieve classification accuracy 66.176±0.979.

Finally, we perform a post-hoc Bonferroni test [60] aiming to rank the MOSAKD method against training W/o Distillation, and assess the statistical significance of the acquired results. More specifically, when the average ranks over the utilized datasets of two compared methods differ by at least a Critical Difference (CD), then the performance of these methods is significantly different. The CD is defined as follows:

$$CD = q_a \sqrt{\frac{v(v+1)}{6D}}, \tag{15}$$

where the critical values q_α can be found in [60], v corresponds to the number of the compared methods, while D corresponds to the number of datasets. In the conducted comparison, α is set to 0.05, the number compared methods, v, is two, while the number of datasets, D, is five. Fig. 4 illustrates the ranking results. The two compared methods are depicted on the vertical axis, while the performance ranking is depicted on the horizontal axis. The circles denote the average rank, and the intervals around the circles denote the confidence interval, which is defined by the CD value. Two methods in comparison are significantly different if the corresponding intervals are non-overlapping, while there is no statistically significant difference between them when the corresponding intervals are overlapping. As can be observed, the proposed MOSAKD method is significantly different from training w/o distillation.

2) Comparisons against state-of-the-art: In the second set of experiments, we compare the proposed online distillation method with state-of-the-art online distillation methods. More specifically, we first utilize the WRN-16-2 model, apply the proposed MOSAKD method on Cifar-10 dataset, and compare the performance with the competitive online distillation methods, [28], [14]. In the conducted experiments we use the same training setup as in [59], [28] is followed. More specifically, the SGD with Nesterov momentum is used and the momentum is set to 0.9. The initial learning rate is set to 0.1 and drops by 0.2 at 60, 120 and 160 epochs. A mini-batch of 128 samples is used and the models are trained for 200 epochs.

It should be noted, that even though we do not follow an ensemble methodology, apart from the compared online methods, FFL-S [28] and ONE [14], we also compare the

TABLE IX: COMPARISONS AGAINST ONLINE DISTILLATION METHODS ON CIFAR-10 UTILIZING THE WRN 16-2 ARCHITECTURE.

Method	Classification Accuracy
WRN 16-2	93.55% ± 0.11%
ONE [14]	93.76%± 0.16%
FFL-S [28]	93.91% ± 0.09%
ONE-E [14]	93.79%± 0.12%
FFL [28]	93.86% ± 0.11%
MOSAKD - Layer 1	**94.286% ± 0.188%**
MOSAKD - Layer 2	**94.260% ± 0.088%**
MOSAKD - Layer 3	**94.272% ± 0.178%**
MOSAKD - Layer 4	**94.268% ± 0.060%**

TABLE X: COMPARISONS AGAINST ONLINE DISTILLATION METHODS ON CIFAR-10 UTILIZING THE WRN-20-8 ARCHITECTURE.

Method	**Classification Accuracy**
WRN-20-8	94.73% ± 0.06%
DML [15]	94.96 ± 0.08%
ONE [14]	94.73 ± 0.02%
CL-ILR [27]	94.88 ± 0.16%
OKDDip [25]	95.16 ± 0.07%
MOSAKD - Layer 1	**96.00% ± 0.06%**
MOSAKD - Layer 2	**96.19% ± 0.10%**
MOSAKD - Layer 3	**96.09% ± 0.07%**
MOSAKD - Layer 4	**96.01% ±0.11%**

performance of the proposed online distillation method, with the ensemble and FFL [28] and ONE-E [14] approaches. It is emphasized that the number of parameters in FFL is 1.29M, in ONE-E is 1.24M, while in the proposed MOSAKD method is 0.70M. The evaluation results are provided in Table IX. As it can be observed, the MOSAKD method accomplishes superior performance against the compared online distillation methods, for all the utilized layers. The best performance is achieved using the first convolutional block. Furthermore, as it is demonstrated the proposed method can even outperform ensemble approaches.

Finally, we use the WRN-20-08 model to compare the performance of the MOSAKD method with the recent OKDDip [25] and DFL [26] methods, as well as with DML [15], ONE [14], and CL-ILR [27]. We perform experiments on Cifar-10 and Cifar-100 datasets. To ensure the fairness of comparisons, we use the same experimental settings as in [25]. That is, stochastic gradient descent is used with Nesterov momentum. The initial learning rate is set to 0.1 and is divided by 10 at 150 and 225 epochs, while the networks are trained for 300 epochs. Finally, we consider batch sizes of 128 samples. The experimental results are provided in Tables X and XI for the Cifar-10 and Cifar-100 datasets, respectively. Note that the DFL method [26] provides results only on Cifar-100 dataset. As it evident from the provided results, the MOSAKD method significantly improves the baseline, and achieves superior performance over the competitive online distillation methods using the WRN-20-08 model on both the Cifar-10 and Cifar-100 datasets.

3) Efficiency evaluation: In the third set of experiments, the efficiency of the MOSAKD method is evaluated. First, we evaluate the complexity (training cost) utilizing as

TABLE XI: COMPARISONS AGAINST ONLINE DISTILLATION METHODS ON CIFAR-100 UTILIZING THE WRN-20-8 ARCHITECTURE.

Method	**Classification Accuracy**
WRN-20-8	77.50% ± 0.44%
DML [15]	79.79 ± 0.11%
ONE [14]	78.81 ± 0.12%
CL-ILR [27]	79.56 ± 0.13%
OKDDip [25]	80.37 ± 0.07%
DFL [26]	80.51% ± 0.49%
MOSAKD - Layer 1	**80.79% ± 0.20%**
MOSAKD - Layer 2	**80.76% ± 0.21%**
MOSAKD - Layer 3	**80.72% ± 0.23%**
MOSAKD - Layer 4	**81.01% ± 0.21%**

evaluation metric the FLOPS considering one forward pass. FLOPs are estimated according to https://github.com/1adrianb/pytorch-estimate-flops. The experiments are conducted on the Cifar-10 dataset, using the WRN-16-2 model. To gain further insights on the efficiency of the proposed method, we compare the complexity with the most well-known offline KD methodology, [9]. For the offline KD, the powerful Wide ResNet 40-2 model (WRN-40-2) is used as teacher to mine further knowledge. Furthermore, we use the model parameters to represent the model sizes for both cases.

The experimental results are presented in Table XII, where the effectiveness of the proposed online distillation method against the offline methodology is clearly validated. Apart from the expected superiority over the offline methodology, due to the fact the proposed method does not require utilizing a separate teacher model for realizing the distillation procedure, it should be emphasized that the MOSAKD method is also more cost-effective compared to the existing online distillation methods. That is, the existing online distillation methods require at least two times more FLOPS than the MOSAKD method, since they use at least two networks or copies of a network to mine additional knowledge. Therefore, the MOSAKD method which acquires the knowledge for the same model is more efficient.

Finally, for further evaluating the efficiency of the proposed online distillation methodology, we highlight that apart from the common strong models which were used mainly for comparison purposes against state-of-the-art, we use fast and lightweight models. These models are extremely low-memory demanding, considering the memory required for training the models using the proposed training pipeline. More specifically, the required memory to train a lightweight model with the proposed method for example on CIfar-10 dataset is 920 MiB. To gain some more insights on the efficiency with respect to the memory requirements for training with the proposed online methodology, we can compare the performance of the MOSAKD methodology again with the conventional offline KD using the WRN-16-2 model. The MOSAKD method requires 1260 MiB, while for training first the more powerful WRN-40-2 model and transferring the knowledge to the WRN-16-2 model with the offline KD methodology are required 2984 MiB.

4) Qualitative Results: Finally, in the fourth set of experiments, we use the proposed trained model on the syn-

TABLE XII: Complexity of the Proposed MOSAKD and KD [9] Methods.

Method	Student	Teacher	Complexity
KD [9]	WRN-16-2 (0.7M)	WRN-40-2 (2.26M)	0.43 GFLOPS
MOSAKD	WRN-16-2 (0.7M)	-	0.10 GFLOPS

thetic human dataset to generate heatmaps on unseen high-resolution images that contain real humans. That is, unseen images of size 1920×1080 are fed to the network, and for every window 64×64 we compute the output of the network at the output layer. Indicative evaluation results are illustrated in Figs. 5 and 6. As it shown, the model which is trained on synthetic images (only 100 real images depicting humans are used), can successfully detect humans on real images.

V. Conclusions

In this paper, a novel online self-distillation approach was proposed, namely Multilayer Online Self-Acquired Knowledge Distillation. The MOSAKD method employs the k-nn non-parametric density estimation for estimating the unknown density distributions of the data samples in the feature space generated by any layer of a deep neural model. Hence, the posterior class probabilities can be directly estimated and be used as soft labels that explicitly reveal the similarities of the data samples with each class. In this fashion, we are able to mine further knowledge directly from the data, without modifying the model, e.g., by introducing multiple copies of the model, and simultaneously in an one-stage training pipeline. Interestingly, it is evident that apart from the output layer of the model, which is mainly used in distillation, intermediate layers can also provide useful information. The effectiveness of the proposed method to ameliorate the classification accuracy of any model, regardless of their complexity, is experimentally validated on six datasets, including a synthetic dataset. The superiority of the proposed MOSAKD method against compared online distillation methods is, finally, validated through the comparison experiments.

References

[1] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[2] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27 – 48, 2016.

[3] S. Sengupta, S. Basak, P. Saikia, S. Paul, V. Tsalavoutis, F. Atiah, V. Ravi, and A. Peters, "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowledge-Based Systems*, vol. 194, p. 105596, 2020.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017.

[7] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[10] Z. Tang, D. Wang, and Z. Zhang, "Recurrent neural network training with dark knowledge transfer," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 5900–5904.

[11] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[12] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.

[13] X. Lan, X. Zhu, and S. Gong, "Self-referenced deep learning," in *Proceedings of the Asian Conference on Computer Vision*, 2018, pp. 284–300.

[14] x. lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Proceedings of the Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7517–7527.

[15] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[16] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proceedings of the Advances in Neural Information Processing Systems 27*, 2014, pp. 2654–2662.

[17] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[18] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[19] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, 2018.

[20] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 211–220.

[21] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 494–10 501.

[22] M. Tzelepi and A. Tefas, "Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization," *Pattern Recognition*, vol. 106, p. 107407, 2020.

[23] S. Mirzadeh, M. Farajtabar, A. Li, and H. Ghasemzadeh, "Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher," *CoRR*, vol. abs/1902.03393, 2019.

[24] M. Tzelepi and A. Tefas, "Efficient training of lightweight neural networks using online self-acquired knowledge distillation," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2021, pp. 1–6.

[25] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3430–3437.

[26] D. Ni, "Dense feature fusion for online mutual knowledge distillation," in *Journal of Physics: Conference Series*, vol. 1865, no. 4. IOP Publishing, 2021, p. 042084.

[27] G. Song and W. Chai, "Collaborative learning for deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1832–1841, 2018.

[28] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," *arXiv preprint arXiv:1904.09058v1*, 2019.

[29] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[30] L. Wang and K.-J. Yoon, "Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[31] C. Bucila, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, 2006.

Fig. 5: Heatmap on real-image containing humans utilizing the human detection model trained on synthetic humans.

Fig. 6: Heatmap on real-image containing humans utilizing the human detection model trained on synthetic humans.

[32] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6550

[33] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," 2015.

[34] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[35] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9163–9171.

[36] N. Passalis, M. Tzelepi, and A. Tefas, "Heterogeneous knowledge distillation using information flow modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2339–2348.

[37] D. Nguyen, S. Gupta, T. Nguyen, S. Rana, P. Nguyen, T. Tran, K. Le, S. Ryan, and S. Venkatesh, "Knowledge distillation with distribution mismatch," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2021, pp. 250–265.

[38] G. Xu, Z. Liu, X. Li, and C. C. Loy, "Knowledge distillation meets self-supervision," in *European Conference on Computer Vision*. Springer, 2020, pp. 588–604.

[39] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," 2018.

[40] F. Zhang, X. Zhu, and M. Ye, "Fast human pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3517–3526.

[41] Z. Meng, J. Li, Y. Zhao, and Y. Gong, "Conditional teacher-student learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 6445–6449.

[42] X. Jin, B. Peng, Y. Wu, Y. Liu, J. Liu, D. Liang, J. Yan, and X. Hu, "Knowledge distillation via route constrained optimization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1345–1354.

[43] J. Kim, S. Park, and N. Kwak, "Paraphrasing complex network: Network compression via factor transfer," in *Proceedings of the Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 2760–2769.

[44] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 268–284.

[45] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3779–3787.

[46] N. Passalis and A. Tefas, "Unsupervised knowledge transfer using similarity embeddings," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 3, pp. 946–950, 2019.

[47] S. Kim and H. Kim, "Transferring knowledge to smaller network with class-distance loss," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*, 2017.

[48] R. Muller, S. Kornblith, and G. E. Hinton, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019, pp. 4696–4705.

[49] Q. Ding, S. Wu, H. Sun, J. Guo, and S.-T. Xia, "Adaptive regularization of labels," *arXiv preprint arXiv:1908.05474*, 2019.

[50] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[51] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[52] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[53] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[54] W. Yang, P. Luo, and L. Lin, "Clothing co-parsing by joint image segmentation and labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
[55] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *PProceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
[56] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *ACCV*, 2012.
[57] W. Li and X. Wang, "Locally aligned feature transforms across views," in *CVPR*, 2013.
[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[59] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, 2016, pp. 87.1–87.12.
[60] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

PLACE PHOTO HERE

Maria Tzelepi received the B.Sc. degree in Informatics, the M.Sc. degree in Digital Media - Computational Intelligence, and the Ph.D. degree in Informatics from the Aristotle University of Thessaloniki, Greece, in 2013, 2016, and 2021, respectively. She is currently a post-doctoral researcher in the Artificial Intelligence & Information Analysis Laboratory in the Department of Informatics at the Aristotle University of Thessaloniki. She has co-authored 10 journal papers and 15 papers in international conferences. From November 2017 to February 2020 her doctoral studies were supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI). Her research interests include deep learning, pattern recognition, computer vision, and image analysis and retrieval.

PLACE PHOTO HERE

Charalampos Symeonidis received his B.Sc. degree in Informatics from Aristotle University of Thessaloniki, Greece, in 2017. Presently, he is a Ph.D candidate at the Artificial Intelligence and Information Analysis Laboratory of the same Department. His current research interests include deep learning, computer vision and computer graphics.

PLACE PHOTO HERE

Nikos Nikolaidis received the Diploma of Electrical Engineering and the Ph.D. degree in Electrical Engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1991 and 1997, respectively. He is currently Associate Professor at the Department of Informatics, Aristotle University of Thessaloniki. He has co-authored 1 book, 16 book chapters, 61 journal papers and 184 conference papers and co-edited one book and two special issues in journals. Moreover he has co-organized 6 special sessions in international conferences. The number of citations to his work by third authors exceeds 6150 (h-index 33). He has participated into 26 research projects funded by the EU and national funds. His current research interests lie in the areas of computer vision and computer graphics. Dr. Nikolaidis is currently serving as associate/area editor for Signal Processing: Image Communication, EURASIP Journal on Image and Video Processing and IET Image Processing. He served as Technical Program chair of IEEE IVMSP 2013 workshop, and Publicity co-chair of EUSIPCO 2015 and IEEE ICIP 2018. Dr Nikolaidis is a Senior Member of IEEE.

PLACE PHOTO HERE

Anastasios Tefas received the B.Sc. in Informatics in 1997 and the Ph.D. degree in Informatics in 2002, both from the Aristotle University of Thessaloniki, Greece. Since 2017 he has been an Associate Professor at the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2017, he was a Lecturer, Assistant Professor at the same University. From 2006 to 2008, he was an Assistant Professor at the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. Dr. Tefas participated in 20 research projects financed by national and European funds. He is the Coordinator of the H2020 project OpenDR, \Open Deep Learning Toolkit for Robotics". He is Area Editor in Signal Processing: Image Communications journal. He has co-authored 132 journal papers, 242 papers in international conferences and contributed 8 chapters to edited books in his area of expertise. Over 6500 citations have been recorded to his publications and his H-index is 40 according to Google scholar. His current research interests include computational intelligence, deep learning, pattern recognition, statistical machine learning, digital signal and image analysis and retrieval and computer vision

7.4 Improving Binary Semantic Scene Segmentation for Robotics Applications

The appended paper follows.

IMPROVING BINARY SEMANTIC SCENE SEGMENTATION FOR ROBOTICS APPLICATIONS

Maria Tzelepi, Nikolaos Tragkas and Anastasios Tefas

Aristotle University of Thessaloniki

ABSTRACT

Robotics applications are accompanied by particular computational restrictions, i.e., operation at sufficient speed, on embedded low power GPUs for high-resolution input. Semantic scene segmentation performs an important role in a broad spectrum of robotics applications, e.g., autonomous driving. In this paper, we focus on binary segmentation problems, considering the specific requirements of the robotics applications. To this aim, we utilize the BiseNet model, which achieves significant performance considering the speed-segmentation accuracy trade-off. The target of this work is two-fold. First, we propose a lightweight version of BiseNet model, providing significant speed improvements. Second, we explore different losses for enhancing the segmentation accuracy of the proposed lightweight version of BiseNet on binary segmentation problems. Experiments conducted on various high and low power GPUs, utilizing two binary segmentation datasets.

***Index Terms*—** Semantic segmentation, Binary, Bisenet, Robotics, Low power GPUs

1. INTRODUCTION

Semantic scene segmentation refers to the task of assigning a class label to each pixel of an image, and hence it is also known as pixel-level classification. Semantic scene segmentation is a challenging task involved in numerous robotics applications, such as autonomous driving [1, 2, 3]. Robotics applications are accompanied by particular computational requirements. That is, the utilized models should be able to effectively operate at sufficient speed, on embedded low power GPUs, while also considering high-resolution input.

Recent advances in Deep Learning (DL) have provided effective models for addressing the general problem of semantic scene segmentation [4]. The seminal approach introduced fully convolutional neural networks [5]. Subsequently, considerable research has been conducted, focusing on improving the segmentation accuracy [6, 7], however, without considering the issue of deployment (inference) speed. That is, most of the existing state-of-the-art DL segmentation models are computationally heavy, and hence ill-suited for robotics applications.

Thus, in the recent literature there have been works that also focus on the deployment speed, providing real-time segmentation models, considering mainly high power GPUs [8, 9, 10, 11, 12, 13]. A comparative study of current semantic segmentation models considering the inherent computational restrictions in the context of robotics applications is provided in [14]. More specifically, extensive experiments have been conducted on different embedded platforms (e.g., AGX Xavier, NVIDIA TX-2), and also for various input resolutions, ranging from lower to higher ones. From the conducted experiments, it is evident that the Bilateral Segmentation Network (BiseNet) [13] model achieves considerable performance considering the segmentation accuracy-speed trade-off. Towards this end, in this work we employ BiseNet model and we address the problem of binary semantic segmentation considering robotics applications.

The target of this work is to explore ways of improving the performance of BiseNet model both in term of deployment speed and segmentation accuracy, considering binary segmentation problems. To this aim, we first propose a lightweight version of the model. That is, we propose a lightweight network instead of ResNet-18 [15] that is used in the so-called *context path*. Subsequently, we exploit the available losses. Specifically, apart from the widely used cross entropy (softmax) loss, which is also used in the initial version of BiseNet, we apply hinge loss, since as it is shown in the recent literature, it provides improved accuracy considering binary classification problems [16].

The remainder of the manuscript is structured as follows. Section 2 provides the description of the proposed ways of improving binary segmentation, that is the proposed lightweight version of BiseNet and the investigation on loss functions. Next, Section 3 provides the experimental evaluation, and finally conclusions are drawn in Section 4.

2. PROPOSED METHOD

The BiseNet model consists of two paths, that is *Spatial Path* and *Context Path*. The spatial path is used in order to preserve the spatial information and generate high resolution features, and the context path with a fast downsampling strategy is used in order to to obtain sufficient receptive field. Furthermore, the model includes two modules, that is a Feature

Fusion Module and an Attention Refinement Module, in order to further improve the accuracy with acceptable cost. Finally, apart from the principal cross entropy loss which supervises the output of the BiseNet model, two auxiliary losses are utilized to supervise the output of the context path.

In this work, we propose to replace the ResNet-18 model used in the context path, with a more lightweight model. The proposed model, which is based on the VGG model [17], consists of five pairs of convolutional layers followed by batch normalization and ReLU activation. A max-pooling layer follows each convolutional block. The proposed model architecture is illustrated in Fig. 1. As it will be presented the modified BiseNet model utilizing the proposed lightweight model in the context path achieves considerable speed improvements.

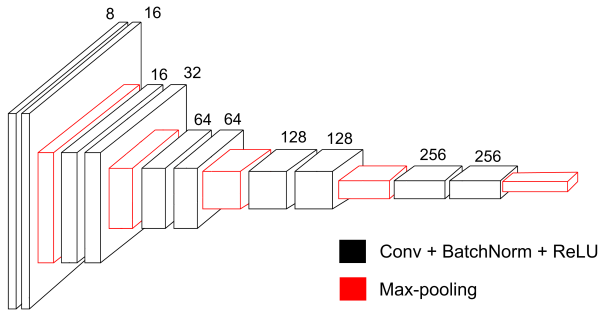


Fig. 1: Proposed lightweight model in the context-path: Red boxes represent the max-pooling layers, while the black boxes represent the convolutional layers, followed by batch normalization ReLU activation. The numbers of output channels of each convolutional layer are also depicted.

Subsequently, since lightweight models usually have inferior performance as compared to the heavyweight counterparts, we explore ways to improve their performance. To achieve this goal, we focus on the loss functions for training the segmentation model. More specifically, even though cross entropy (softmax) loss, is a widely used loss function in DL, in a recent work [16] it has been demonstrated that considering binary classification problems, hinge loss can achieve improved classification performance. Motivated by the aforementioned observation, in this work, we extend this investigation on binary segmentation problems. Thus, we utilize hinge loss so as to supervise the output of the whole model. Hinge loss per pixel is defined as:

$$\ell_h = \sum_{j=1}^{N_c} max(0, 1 - \zeta\{c = j\} y_j^{last}) \tag{1}$$

where $c \in [1, \cdots, N_c]$ indicates the correct class among the N_c classes, y_j^{last} indicates the score with respect to the j-th class, and

$$\zeta\{condition\} = \begin{cases} 1 & \text{, if condition} \\ -1 & \text{, otherwise} \end{cases}$$

In our case, $N_c = 2$, since we deal with binary segmentation problems.

3. EXPERIMENTS

In this work, we first evaluate the deployment speed of the proposed modified BiseNet model, since a principal target of this work is to provide a faster model for binary semantic segmentation. We evaluate the deployment speed in term of Frames Per Second (FPS), on various high power and low power GPUs, as well as for various input sizes. Subsequently, we evaluate the performance of the modified model utilizing hinge loss against cross entropy loss as principal supervised loss, utilizing mean Intersection Over Union (mIOU) as evaluation metric.

3.1. Datasets

In this work, we utilize the CityScapes [18] dataset, exploiting only the Human and Vehicle classes, in order to build the two binary segmentation datasets, i.e., *Human Vs Non-Human* and *Vehicle Vs Non-Vehicle*, respectively. The first one consists of 11,900 train images and 2,000 test images, while the second one consists of 2,975 train images and 500 test images.

3.2. Implementation Details

All the experiments conducted using the Pytorch framework. Mini-batch gradient descent is used for the networks training, where an update is performed for every mini-batch of 8 samples. Momentum is set to 0.9, while the learning rate policy of the initial work was followed. All the models are trained on an NVIDIA 2080 Ti, and the deployment speed was tested on various low power GPUs.

3.3. Experimental Results

In the first set of experiments we evaluate the deployment speed of the proposed modified lightweight BiseNet model against the initial version which uses the ResNet-18 model. We have conducted experiments on a high power NVIDIA 2080 Ti, a high power NVIDIA 2070, a low power NVIDIA Jetson TX-2, and a low power NVIDIA Jetson AGX Xavier. Furthermore, we use various input dimensions ranging from 400×400 to 1024×1024. Experimental results are illustrated in Tables 1-4. Best results are printed in bold. As it is shown, the proposed model runs significantly faster as compared to the initial model, in any considered case. It can also be observed increased discrepancy for lower input sizes.

Table 1: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA 2080 Ti.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	66.51	**75.23**
1280×720	70.69	**84.06**
800×800	98	**119.76**
600×600	163.91	**215.61**
640×360	216.37	**305.89**
400×400	269.58	**353.59**

Table 2: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA 2070.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	50.40	**59.15**
1280×720	56.17	**66.31**
800×800	77.46	**93.85**
600×600	125.04	**166.26**
640×360	184.59	**251.26**
400×400	237.61	**315.24**

Table 3: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA Jetson TX2.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	3.98	**5.02**
1280×720	4.32	**5.58**
800×800	5.71	**7.69**
600×600	9.7	**13.36**
640×360	14.71	**21.05**
400×400	18.82	**26.42**

Table 4: Evaluation of speed in terms of FPS utilizing the proposed lightweight model in the context path, against the ResNet-18 on an NVIDIA Jetson AGX Xavier.

Input Size	BiseNet - ResNet18	BiseNet - Proposed
1024×1024	11.58	**15.32**
1280×720	12.23	**16.96**
800×800	16.38	**23.40**
600×600	26.78	**40.12**
640×360	40.13	**60.66**
400×400	52.42	**77.83**

Subsequently, the experimental results for evaluating the hinge loss against cross entropy loss, utilizing the proposed lightweight BiseNet model, on the two binary segmentation datasets are presented in Table 5. Best results are printed in bold. As it is demonstrated, hinge loss accomplishes superior performance as compared to the cross entropy loss, considering binary segmentation problems.

Table 5: Evaluation on segmentation performance in terms of mIOU (%) utilizing the modified lightweight BiseNet model, for evaluating hinge loss against cross entropy loss.

Dataset	Cross Entropy Loss	Hinge Loss
Human Vs Non-Human	87.82	**89.23**
Vehicle Vs Non-Vehicle	94.82	**95.03**

Finally, some qualitative results are presented utilizing the proposed modified model trained for human segmentation and vehicle segmentation in Fig. 2.

4. CONCLUSIONS

In this paper, we dealt with binary segmentation problems, utilizing the BiseNet model, which achieves significant performance considering the speed-segmentation accuracy trade-off. First, we proposed a lightweight version of BiseNet model, in order to improve the deployment speed. Subsequently, we explored different losses in order to enhance the segmentation accuracy of the proposed lightweight version of BiseNet on binary segmentation problems. The experiments conducted on various high and low power GPUs, utilizing two binary segmentation datasets, validated the effectiveness of proposed lightweight version of BiseNet in terms of deployments speed, as well as that hinge loss provides improved performance considering binary segmentation problems.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

5. REFERENCES

[1] Inigo Alonso, Luis Riazuelo, and Ana C Murillo, "Mininet: An efficient semantic segmentation convnet for real-time robotic applications," *IEEE Transactions on Robotics*, 2020.

[2] Andres Milioto, Philipp Lottes, and Cyrill Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background

Fig. 2: Predictions of the modified lightweight BiSeNet model trained on Human Vs Non-Human and Vehicle Vs Non-Vehicle datasets.

knowledge in cnns," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2229–2235.

[3] Yuxiao Zhang, Haiqiang Chen, Yiran He, Mao Ye, Xi Cai, and Dan Zhang, "Road segmentation for all-day outdoor robot navigation," *Neurocomputing*, vol. 314, pp. 316–325, 2018.

[4] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *arXiv preprint arXiv:1704.06857*, 2017.

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[7] Fahad Lateef and Yassine Ruichek, "Survey on semantic segmentation using deep learning techniques," *Neurocomputing*, vol. 338, pp. 321–348, 2019.

[8] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang, "Fasterseg: Searching for faster real-time semantic segmentation," *arXiv preprint arXiv:1912.10917*, 2019.

[9] Rudra PK Poudel, Stephan Liwicki, and Roberto Cipolla, "Fast-scnn: Fast semantic segmentation network," *arXiv preprint arXiv:1902.04502*, 2019.

[10] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin, "Hardnet: A low memory traffic network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3552–3561.

[11] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang, "Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation," *arXiv preprint arXiv:2004.02147*, 2020.

[12] Taha Emara, Hossam E Abd El Munim, and Hazem M Abbas, "Liteseg: A novel lightweight convnet for semantic segmentation," in *2019 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2019, pp. 1–7.

[13] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang, "Bisenet: Bilateral segmentation network for real-time semantic segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 325–341.

[14] Maria Tzelepi and Anastasios Tefas, "Semantic scene segmentation for robotics applications," 2021.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] Maria Tzelepi and Anastasios Tefas, "Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization," *Pattern Recognition*, p. 107407, 2020.

[17] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.

7.5 Self-Learned Label Embeddings for General Class and Instance Specific Similarity Modeling in Classification

The appended paper follows.

Self-Learned Label Embeddings for General Class and Instance Specific Similarity Modeling in Classification

Paraskevi Nousi[a], Anastasios Tefas[a]

[a]*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*

Abstract

The one-hot 0/1 encoding method is the most popularized encoding method of class labels for classification tasks. Despite its simplicity and popularity, it comes with limitations and weaknesses, like failing to capture the inherent uncertainty in data labels, and making classifiers more prone to overfitting. In this paper, we tackle these shortcomings with a framework for learning soft label embeddings. Two variants are proposed: first, a learnable general-class embedding which aims to capture information regarding inter-class similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly classified ones. Our experimental study on five challenging classification benchmarks, using neural networks of varying depth and width, show that the proposed method leads to better classification accuracy, highlighting its ability to generalize to unseen samples.

Keywords: label embedding, soft labels, class similarities, instance similarities

1. Introduction

The one-hot 0/1 encoding method is the most popularized encoding method of class labels for classification tasks. Most existing neural network based classification methods use this encoding along with some variant of the cross entropy loss function [1]. Despite its simplicity and popularity, it comes with limitations and weaknesses which we investigate in this work.

One-hot encoding fails to capture the *inherent uncertainty* in data labeling processes. There are multiple aspects to this issue. First, a dataset

typically consists of data samples belonging to classes of the same category, e.g., MNIST depicts handwritten digits. In general, a class of digits may resemble another class more or less than the remaining classes. For example, *fours* tend to resemble - and be confused as - *nines* more often than any other digit. Class similarities like these are present in most classification datasets, even more generic ones. Second, some instances of a class may heavily resemble instances of other classes, much like a small dog may resemble a cat or a fox, or a flying bird may resemble a plane. These similarities are specific to each instance of the dataset and may be considered as outliers. Finally, there is always the potential of human error, either related to the aforementioned uncertainties or simply by mistake.

Figure 1 illustrates this point using the CIFAR10 dataset [2]. Two dimensional representations are obtained using t-SNE [3] and a DenseNet [4] classifier, using the representations of the penultimate layer. The general class similarities, like those between the *dog* and *cat* classes, or the *automobile* and *truck* classes, reflect the actual similarity between these real categories. Instance specific similarities exist as well, as shown in the marked misclassified samples: (a) a dog misclassified as a cat, (b) a truck misclassified as a car, (c) a deer misclassified as a horse, (d) a bird misclassified as a ship, (e) a bird misclassified as a ship, (f) a deer misclassified as a horse, and (g) a dog misclassified as a bird.

Furthermore, one-hot encoding can lead to overfitting more easily, as neural networks have the capacity to bend their representation space in ways such that they perfectly capture the training samples, despite their outlying status. As an example, consider a binary classification example, where a difficult sample may be correctly classified with a predicted probability of 0.8. Despite the prediction being correct, most recent approaches to this problem would further modify the network weights such that the predicted probability lies closer to 1. In order to do so, the separating hyperplane may bend in a way that excludes unseen samples even if they lie close to the groups of training samples.

In this paper, we tackle these shortcomings with a framework for learning soft label embeddings. Two variants are proposed: first, a learnable general-class embedding which aims to capture information regarding inter-class similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly

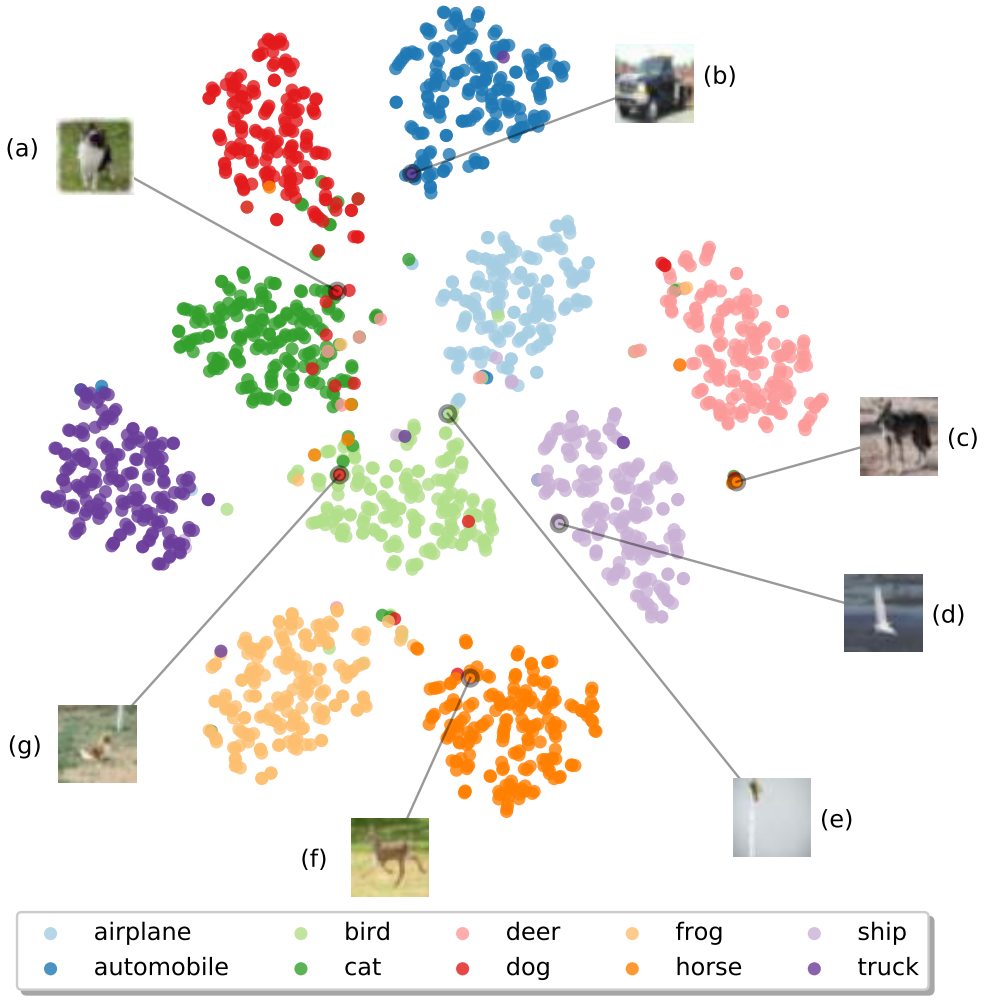


Figure 1: Visualization of feature vectors from the CIFAR10 dataset using t-SNE. General class similarities exist in this dataset as shown by the proximity of the *dog* and *cat* classes, as well as that between the *automobile* and *truck* classes. In contrast, the *deer* and *frog* classes are less similar and less likely to be confused for one another. Instance specific similarities exist as well, as shown in the marked misclassified samples: (a) a dog misclassified as a cat, (b) a truck misclassified as a car, (c) a deer misclassified as a horse, (d) a bird misclassified as a ship, (e) an airplane misclassified as a bird, (f) a deer misclassified as a horse, and (g) a dog misclassified as a bird.

classified ones. The concept of this framework is similar to real-life learning procedures, where uncertainty rules over many subjects. Students are encouraged to eventually become teachers themselves, in the broader sense of these words, and to achieve that they must be allowed some liberties when learning a new subject which presents uncertainties. During the learning process, students may draw from their own experiences and go on to even enhance the subject with their knowledge.

The rest of this paper is structured as follows. Section 3 introduces the notation used in the proposed method as well as the general concept. Sections 3.1 and 3.2 describe the two variants of the proposed method, aiming to capture general-class and instance-specific similarities in the soft embedding. The method for combining of the two embeddings is presented in Section 3.3. Finally, Section 4 summarizes our experimental results and our conclusions are drawn in Section 5.

2. Related Work

Soft labels, label embedding and label smoothing are all problems similar to the one tackled in this work, and they have been studied to some extent in recent literature.

Image annotation is a tedious and uninspiring task, prone to errors, in both objective and subjective criteria. Subjective errors in particular are almost impossible to overcome, forcing neural networks to either learn these mistakes or to find ways to overcome these *noisy* labels. In fact, labelling errors are evident in the large number of works revolving around learning from noisy labels. In [5], two types of label noise are considered: label flips, where images have been assigned a wrong class, and outliers, where the image does not depict any of the classes in the dataset but has been erroneously assigned to one. To mitigate these errors, a linear layer was added at the final softmax layer, to learn the noise distribution without supervision, i.e., prior modeling of this distribution. More recently, in [6], a single layer network was used to learn to generate soft labels from noisy labels.

Soft labels have been studied in the past, in the context of simpler classifiers like the k-Nearest Neighbor one [7], and recently scientific research in this task has resurfaced. In [8], a method for modeling subjectivity in emotion recognition was proposed. An ensemble method, where different networks are trained with different annotators, was compared against a single network

trained on soft labels, generated by averaging the labels from different annotators. In [9], a soft label method was introduced for ordinal regression, i.e., classification problems were classes are not independent, but follow some sort of order. Maintaining inter-class relationships is crucial in such tasks, and we further argue that a natural order between classes can be found in most classification datasets, especially as the number of classes increases. In [10], soft labels were learned in a meta learning fashion, by treating them as learnable parameters, modeling both class-level and instance-level similarities.

Soft labels have also recently been linked to knowledge distillation methods [11, 12]. In [13], soft labels were explored in the context of relation extraction with neural networks. A teacher network was used to learn well-informed soft labels and its knowledge was distilled and transferred to a student network. In [11], posterior class probabilities were estimated using the data samples, which were used as soft labels, encoding information about the similarities between the data samples. Further, in [12], a distillation method aiming to reveal subclass similarities was introduced. In [14], a teacher-free knowledge distillation method was proposed, using label smoothing regularization. It was also proven that knowledge distillation is a type of learned label smoothing.

The significance of label smoothing has been gaining interest in recent years. In [15], it was shown that label smoothing does not hurt the generalization of a model's predictions, but using a simple label smoothing method on a network makes it a less effective teacher in a knowledge-distillation setting. In [16], it was shown that label smoothing can indeed be useful to teacher networks in the presence of noisy labels. In [17], an investigation was conducted into the effect of label smoothing regularization on the convergence of stochastic gradient descent methods. It was shown that label smoothing can help speed up the convergence. In [18], a theoretical framework was introduced, to explain how label smoothing controls the generalization loss. In [19], an online label smoothing strategy was proposed, to generate soft labels based on the statistics of the model's predictions.

Label embedding is closely related to label smoothing. In [20], a label embedding method was proposed for text classification, in a multi-task learning context. Also in text classification, in [21], a model-based label embedding method was coupled with a self-interaction attention mechanism. In [22], a label embedding network was introduced for soft training of deep neural networks. The embedding network learns from the predictions of the classifier and the two networks are trained in unison. Label embedding is also useful

in multi-label classification tasks, as recently showcased in [23].

We are interested in modelling relationships both at a class-level and an instance-level into the proposed label embedding methods. Inter-class similarities were also studied in [24], using a label smoothing technique to instill general class similarity information into the learning task. We achieve the same goal using a simple linear network, which learns to map the one-hot encodings into soft versions of themselves, using the network's predictions. Instance-specific labels were recently studied in [25], in a self-distillation mechanism. Instead of self-learning the instance-specific labels, we use an external observer, in the form of an Autoencoder (AE) [26], to uncover the instance level similarities in its learned latent space.

3. Proposed Methodology

Let $\mathbf{x}_i \in \mathbb{R}^D$ denote an input sample and $\mathbf{y}_i \in \{0,1\}^K$ such that $\sum_{k=1}^{K} y_{i,k} = 1$ be its one-hot encoding, in a set of $i = 1, \ldots, N$ training samples spanning over K distinct classes. A standard classifier $f(\mathbf{x})$ is trained to map the input samples $\mathbf{x}_i$ to their corresponding one-hot labels and to make predictions on unseen samples. Typically, the Cross Entropy loss function is used for this purpose:

$$CE(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^{K} y_k \cdot \log(p_k) \tag{1}$$

where $\mathbf{p}$ are the probabilities predicted by $f(\mathbf{x})$, obtained as the output of a softmax function on the output, or logits, of the classifier. We propose the use of soft labels $\hat{\mathbf{y}}_i \in \mathbb{R}^K$ such that $\sum_{k=1}^{K} \hat{y}_{i,k} = 1$ for all samples in the dataset, such that they capture not only the groundtruth label of each sample but also inter-class similarities as well as similarities between each sample and other samples present in the dataset. Our hypothesis is that by using soft labels, a neural classifier can learn to generalize better, as the sum of errors from correctly-classified but difficult samples will decrease. In terms of the representation learned, less effort will be consumed towards separating such samples from their similar neighbors.

The new learning task is formulated as:

$$CE(\hat{\mathbf{y}}, \mathbf{p}) = -\sum_{k=1}^{K} \hat{y}_k \cdot \log(p_k) \tag{2}$$

where $\hat{\mathbf{y}}$, i.e., the soft labels, are given by a differentiable function $g(\cdot)$, the parameters of which can be learned in conjunction with the parameters of the classifier during training.

In the following Sections we define two methods to generate such soft labels. In both cases, the label embedding learned stems from the representations learned by the classifier itself, hence the title of self-learned embedding. The first method aims to capture resemblances between the classes present in the dataset and is described in Section 3.1. The second method aims to capture resemblances between each sample and any and all instances in the dataset, regardless of their corresponding classes. Section 3.2 describes this method in detail, while Section 3.3 presents our proposed method of combining the aforementioned methods in a single architecture.

3.1. General Class Soft-Label Embedding

We formulate the function $g(\cdot)$ for the case of general class similarities as a simple embedding of the form $g(\mathbf{y}) = \hat{\mathbf{y}} = \mathbf{W} \cdot \mathbf{y}$ where $\mathbf{W} \in \mathbb{R}^{K \times K}$ is a learnable weights matrix. This can be viewed as a neural network with a single linear hidden layer. Despite its simplicity, this embedding can model extreme cases where:

$$\mathbf{W} = \mathbf{I}_K \tag{3}$$

corresponding to the 0/1 one-hot encoding, or:

$$\mathbf{W} = \frac{1}{K}\mathbf{J}_K \tag{4}$$

where the notation $\mathbf{J}_K$ denotes an all-ones matrix of size $K \times K$, corresponding to the case where all class probabilities are equal.

We are, however, interested in the more generic case where each class label is given as a linear combination of all classes, including itself which should intuitively hold a larger weight:

$$\hat{\mathbf{Y}} = g(\mathbf{Y}) = \mathbf{W} \cdot \mathbf{Y} = \mathbf{W}. \tag{5}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{K \times K}$ is the soft label matrix, i.e., each row $\hat{\mathbf{y}}_k$ corresponds to the soft labels of the k-th class, and $\mathbf{Y} = \mathbf{I}_K$ holds the one-hot encodings of all classes. That is, the new label vector for the k-th class is given by:

$$\hat{\mathbf{y}}_k = g(\mathbf{y}_k) = \mathbf{W} \cdot \mathbf{y}_k = [W_{k1}, W_{k2}, \ldots, W_{kk}, \ldots, W_{kK}] \tag{6}$$

i.e., practically the k-th row of the weight matrix $\mathbf{W}$, as $\mathbf{y}_k$ is the one-hot encoding of the k-th class. As mentioned, the weights matrix should optimally conform to:

$$W_{kk} \geq \sum_{l \neq k}^{K} W_{kl}, \tag{7}$$

so as to retain the groundtruth information. A softmax function is applied on $\hat{\mathbf{y}}_k$ to ensure that $\sum_{l=1}^{K} \hat{y}_{kl} = 1$.

Figure 2 illustrates the use of the proposed general-class network alongside a generic neural classifier. The two networks are trained in parallel using a single learning objective, optimizing the cross entropy given by Eq. (2) for all input samples. If not for the constraint imposed by Eq. (7), it is evident that this architecture is in direct danger of collapsing to the case presented in Eq. (4), where all class labels are equal and both networks learn random weights. There are multiple straightforward ways to avoid this scenario. One is to add a regularization term to the loss to directly enforce Eq. (7). However, this entails the threat of collapsing to the case Eq. (3), i.e., the labels remain binary. Instead, we enforce this constraint by first initializing the weights matrix using an identity matrix. Some noise is added to these weights to avoid harsh 0/1 numbers. Furthermore, a different learning rate is used for the label embedding network which forces the network weights to update slowly in comparison to the weights of the classification network, for which a larger learning rate is used. Another way to circumvent this issue is to set the new classification targets to be a weighted combination of the one-hot encoding and their linear combinations given by $g(\mathbf{y})$, i.e.:

$$\hat{\mathbf{y}}_k = \alpha \cdot g(\mathbf{y}_k) + (1 - \alpha) \cdot \mathbf{y}_k \tag{8}$$

where $\alpha \in (0, 1)$ controls the softness of the labels. This approach is used in this work, for its simplicity.

3.2. Instance Specific Soft-Label Embedding

In the instance-specific case, the soft labels for each instance $\mathbf{x}_i$ are a function of the instance itself, that is:

$$\hat{\mathbf{y}}_i = h(\mathbf{x}_i). \tag{9}$$

The criterion from Eq. (2) still applies in this case and the function $h(\cdot)$ can take the form of a network which takes $\mathbf{x}_i$ as input and outputs $\hat{\mathbf{y}}_i$.

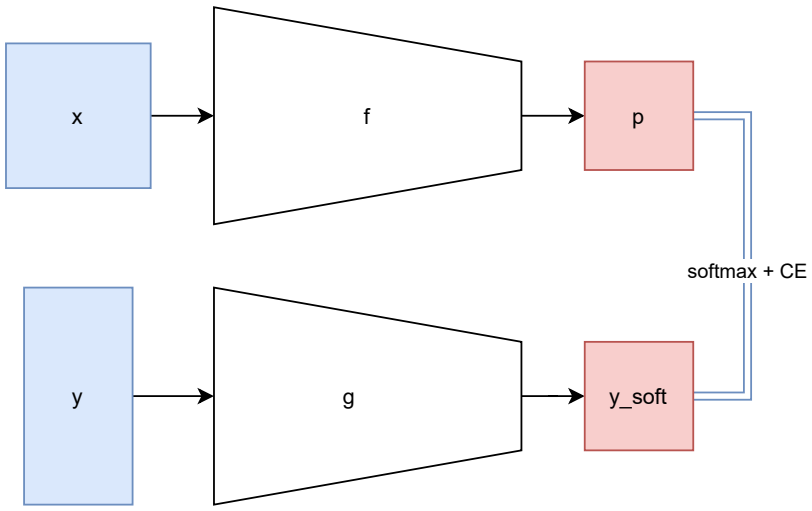


Figure 2: Schematic representation of the learning procedure of a classification network attached with the proposed general-class label embedding network.

An *external* representation of $\mathbf{x}_i$ can be extracted using an Autencoder, where the intermediate representation $\mathbf{z}_i$ is set to be K-dimensional. An AE can be formally defined by its two parts, the encoder and decoder networks, as a composite function:

$$\hat{\mathbf{x}_i} = f(g(\mathbf{x_i})), \tag{10}$$

where $g(\cdot)$, $f(\cdot)$ are the encoding and decoding functions respectively, and $\hat{\mathbf{x}}_i \in \mathbb{R}^D$ is the network's output, which is trained to approximate the input. The intermediate representation $\mathbf{z}_i = g(\mathbf{x}_i)$ is given by the encoder. To ensure a proper mapping between the AEs latent dimensions and the one-hot encoding, a cross entropy criterion is added to the objective. In this setting, in an extreme case of overfitting, the intermediate representation can take the form of a one-hot encoding. Furthermore, it is also possible that the AE objective will collapse to mapping every sample to the same representation, which is another tedious solution of the problem. Both of these scenarios are mitigated by the addition of the reconstruction loss of the AE, in terms of MSE between its input and predicted output $\tilde{\mathbf{x}}$:

$$MSE(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2. \tag{11}$$

As stated, in order to ensure an identity mapping between $\mathbf{z}_i$ and $\mathbf{y}_i$, a cross entropy loss is added to the objective, between the intermediate repre-

sentations of the AE and the groundtruth labels. The final learning objective is:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} CE(\mathbf{y}_i, \sigma(\mathbf{z}_i)) + CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \lambda \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{12}$$

where λ is a constant introduced to weigh the classification and reconstruction losses. Figure 3 summarizes the proposed architecture for this case. The input is fed into both the classifier and an AE. The intermediate representation of the AE is trained to match the groundtruth labels using a softmax function $\sigma(\cdot)$, and the cross entropy criterion. The AE is also trained using the MSE between the input and its prediction. Finally, the classifier's predictions are matched to the soft targets, which are given by the intermediate representation. In practice, an equation similar to Eq. 8 is used to generate the soft targets, so as to maintain the groundtruth information:

$$\hat{\mathbf{y}}_k = \beta \cdot \sigma(\mathbf{z}_k/T) + (1-\beta) \cdot \mathbf{y}_k \tag{13}$$

where T denotes the softmax temperature.

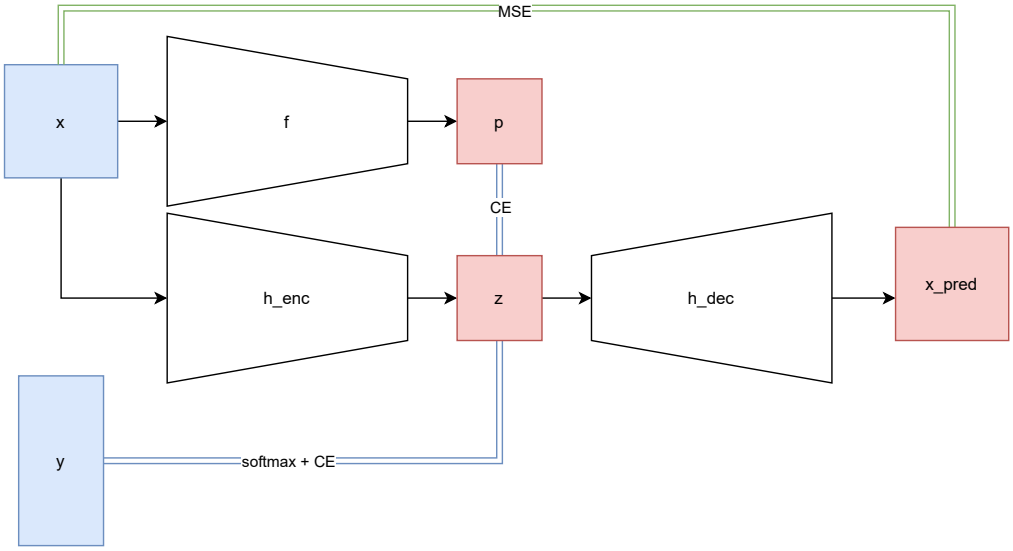


Figure 3: Schematic representation of the learning procedure of a classification network attached with the proposed instance-specific label embedding network.

3.3. Combined General-Class and Instance-Specific Label Embedding

The two methods described in the previous Sections can easily be combined into a single architecture, at the cost of increased training time. The combination is straightforward and a graphical description is given by Figure 4. The main difference to the instance-specific case is the addition of the general-class label embedding network $g(\mathbf{y})$, and the learning objective is modified accordingly:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} CE(f(\mathbf{y}_i), \mathbf{z}_i) + \beta \cdot CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \gamma \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{14}$$

where β and γ weigh the classification and reconstruction losses. In this case, it is crucial to properly initialize the general-class embedding network so as to output labels as close as possible to the groundtruth one-hot encoding. In practice, the soft labels used to train the classifier are a linear combination of the one-hot groundtruth labels, the general-class embeddings and the instance-specific embeddings, weighted by two hyperparameters to control the effect of each embedding on the final targets. The same practice can be applied to the individual cases of general-class and instance-specific embeddings.

4. Experimental Study

We conducted experiments on CIFAR10, CIFAR100 [2], Fashion MNIST [27], STL10 [28] and SVHN [29] datasets. The CIFAR-10 dataset consists of 60000 RGB images of size 32×32, spanning over 10 classes, i.e., with 6000 images per class. The training set contains 50000 and the remaining 10000 images constitue the test set. The CIFAR100 dataset is similar, except it contains 100 classes with 600 images each, 500 of which are used for training and the remaining 100 are used for testing. The STL10 dataset is similar to the CIFAR10 dataset, but each class has fewer labeled training examples. There are 10 classes in this dataset, the images are of size 96×96 and colour, and there are 500 training images and 800 test images per class. The SVHN dataset is a real-world image dataset for digit recognition. There are 10 classes in the dataset, the digits are cropped and resized to 32×32, and in total there are 73257 digits for training, and 26032 digits for testing. Finally, the FashionMNIST dataset consists of a training set of 60000 samples and

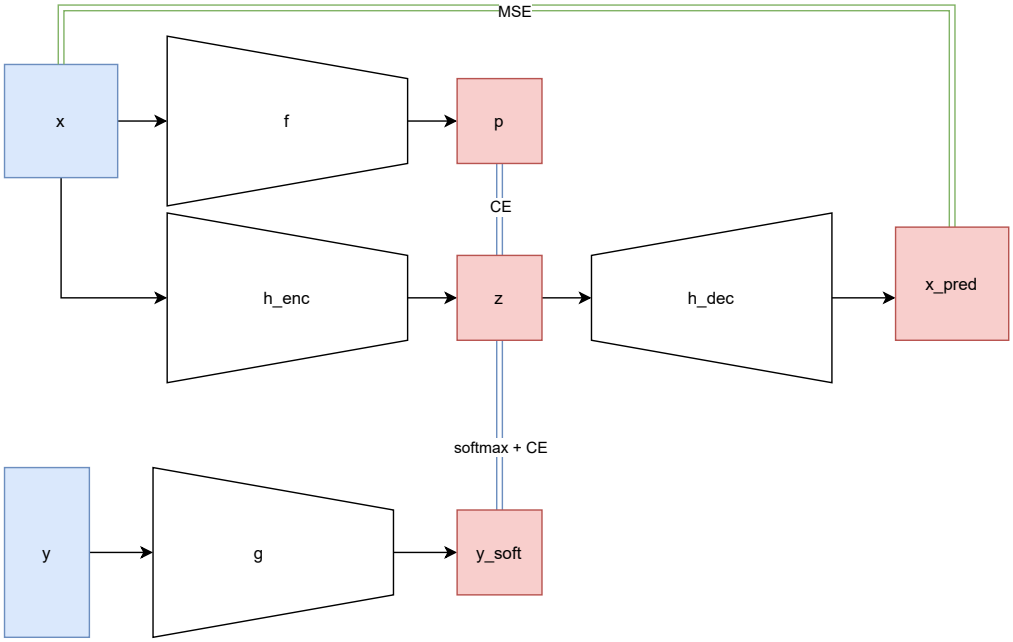


Figure 4: Schematic representation of the learning procedure of a classification network attached with the combination of the proposed general-class and instance-specific label embedding networks.

a test set of 10000 samples. Each sample is a 28×28 grayscale image and there are 10 classes in this dataset, corresponding to clothing categories.

Four types of networks are used based on the ResNet [30] architecture: a ResNet-8, a ResNet-18, a ResNet-6 model, and a version of the ResNet-6 net with fewer channels, specifically using only a quarter of the learnable filters (ResNet-6l0.25). The models run at various speeds, and the two most lightweight ones were chosen so that they can run at real-time on embedded devices for HD inputs.

The results of our experiments are summarized in Tables 1,2,3 and 4 for the ResNet6-l0.25, ResNet-6, ResNet-8 and ResNet-18 networks and all datasets. The general-class method is denoted as GC, the instance-specific case as IS and a combination of the two is also investigated and denoted as GC+IS. The proposed variants lead to increases in accuracy in all cases over the baseline models. Note that, despite the increased training cost, the cost during deployments remains the same as that of the baseline models.

For the lighweight ResNet-6l0.25 model, both the GC and IS methods

Dataset	Baseline	GC	IS	GC+IS
CIFAR10	86.81	87.37	87.50	87.63
CIFAR100	62.02	62.59	62.37	62.00
STL10	59.78	60.78	63.29	62.26
SVHN	95.09	95.54	95.44	95.17
FashionMNIST	92.73	93.38	93.20	93.41

Table 1: Res-Net-6l0.25 results on all datasets.

outperform the baseline separately and in combination. However, the combined GC+IS method only offers a small improvement over the GC variant in the CIFAR10 dataset, and otherwise performs the same or slightly worse than the independent variants. The combined GC+IS is a more difficult scheme for this somewhat weak classifier, in terms of its learning capability.

The results are similar for the ResNet-6 model. The proposed GC, IS and GC+IS outperform the baseline training for all datasets. The combined variant exceeds the performance of each of the GC, IS methods on the STL10 and FashionMNIST datasets in this case. It also does not perform significantly worse for the rest of the datasets than the GC and IS methods. This classifier contains four times as many learnable parameters as the ResNet-6l0.25 model, which is visible in the overall better results, with the exception of the STL10 dataset.

Dataset	Baseline	GC	IS	GC+IS
CIFAR10	90.22	91.78	91.42	91.49
CIFAR100	69.05	71.37	69.85	71.13
STL10	45.01	45.94	45.35	46.91
SVHN	95.21	96.09	95.66	95.92
FashionMNIST	93.10	93.55	93.44	93.71

Table 2: ResNet6 results on all datasets.

Moving on to the ResNet-8 model, the performance of the baseline model is significantly better than the previous, more lightweight models. Furthermore, all proposed methods improve upon that baseline. In three out of the five datasets, the combined GC+IS performs better than the corresponding GC and IS methods.

Dataset	Baseline	GC	IS	GC+IS
CIFAR10	93.71	94.08	93.93	93.95
CIFAR100	73.95	74.99	74.41	74.91
STL10	73.75	74.66	75.19	75.55
SVHN	95.46	96.10	95.54	96.14
FashionMNIST	93.88	94.19	94.06	94.45

Table 3: ResNet8 results on all datasets.

Finally, the deepest network, namely ResNet-18, achieves overall better baseline results than the previous models. All of the proposed variants improve the performance of the model on all datasets. Notice that the combined GC+IS method performs better than the GC and IS methods for four out of five datasets for this model, indicating the existence of a correlation between the network depth, and capacity learning and the effectiveness of the combined GC+IS method.

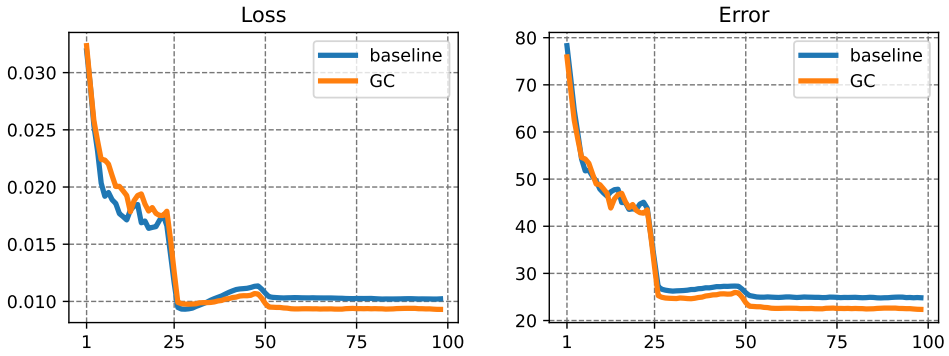
Dataset	Baseline	GC	IS	GC+IS
CIFAR10	93.92	94.41	94.43	94.79
CIFAR100	75.81	77.40	77.11	77.27
STL10	71.88	72.58	73.53	76.23
SVHN	95.95	96.48	96.17	96.63
FashionMNIST	93.88	94.35	94.21	94.53

Table 4: ResNet18 results on all datasets.

As mentioned, the combination of the two methods is riskier than each of the methods used separately, and it seems to work better when the classifier used has a higher learning capacity. This can be attributed to two factors. First, to the better performance of the deeper baseline networks, corresponding to a deeper knowledge of the datasets. Second, the networks with more parameters have more freedom to learn.

Figures 5a and 5b show the loss and accuracy error progression during training of the ResNet-18 model on the CIFAR100 dataset, for the baseline and GC methods.

We finally compare the performance of the proposed method with various state-of-the-art soft label methods, in the CIFAR100 dataset. The results are



(a) (b)

Figure 5: (a) Loss curves and (b) accuracy error progression, for baseline and GC methods, using ResNet-18 on CIFAR100.

presented in Table 5, in terms of baseline accuracy, accuracy using each compared method, and relative improvement in accuracy, to account for different training hyperparameters for each method. The proposed method provides a significant relative improvement, comparable to similar methods.

Network	Method	Acc	Acc w/ Method	Rel. Impr.
ResNet-18	Ours, GC	75.81	77.40	2.09
	Ours, IS		77.11	1.71
	Ours, GC+IS		77.27	1.92
	Tf-KD [14]	75.87	77.10	1.62
	TSLA [17], best	76.87	78.55	2.13
	DynLS [10], Class	77.1	78.8	2.20
	DynLS [10], Instance	77.1	79.2	2.72
	LabelEmb [22]	72.65	76.03	4.65
ResNet-56	OLS [19]	74.73	76.09	1.81
	LS [15]	72.1	72.7	0.83

Table 5: Relative improvement in accuracy for the CIFAR100 dataset.

5. Conclusions

In this paper, soft label embedding methods were studied and two variants were proposed, with an aim to capture both general-class resemblances as well as instance-specific similarities, and to incorporate these into the training process of neural classifiers in order to ease the training process. For the general-class a simple linear network was used to generate labels which are a linear combination of the groundtruth one-hot encodings. As the training process progresses, the network learns to capture general correlations between the classes, allowing the network to achieve lower errors for well-classified but difficult samples and so to focus on incorrectly classified samples. In the instance-specific case, the addition of a decoder network and corresponding reconstruction error leads the logits learned by the classifier to retain information necessary for the reconstruction, which acts as a regularizer upon the class labels. The proposed methods were shown experimentally to lead to increased performance in various networks and datasets, both in the general-class and instance-specific cases, as well as in a combination of the two.

Acknowledgements

This work was supported by the European Union's Horizon2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[2] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[3] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).

[4] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[5] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, R. Fergus, Training convolutional networks with noisy labels, arXiv preprint arXiv:1406.2080 (2014).

[6] G. Algan, I. Ulusoy, Metalabelnet: Learning to generate soft-labels from noisy-labels, arXiv preprint arXiv:2103.10869 (2021).

[7] N. El Gayar, F. Schwenker, G. Palm, A study of the robustness of knn classifiers trained using soft labels, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer, 2006, pp. 67–80.

[8] H. M. Fayek, M. Lech, L. Cavedon, Modeling subjectiveness in emotion recognition with deep neural networks: Ensembles vs soft labels, in: 2016 international joint conference on neural networks (IJCNN), IEEE, 2016, pp. 566–570.

[9] R. Diaz, A. Marathe, Soft labels for ordinal regression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4738–4747.

[10] N. Vyas, S. Saxena, T. Voice, Learning soft labels via meta learning, arXiv preprint arXiv:2009.09496 (2020).

[11] M. Tzelepi, A. Tefas, Efficient training of lightweight neural networks using online self-acquired knowledge distillation, in: 2021 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2021, pp. 1–6.

[12] M. Tzelepi, N. Passalis, A. Tefas, Online subclass knowledge distillation, Expert Systems with Applications 181 (2021) 115132.

[13] Z. Zhang, X. Shu, B. Yu, T. Liu, J. Zhao, Q. Li, L. Guo, Distilling knowledge from well-informed soft labels for neural relation extraction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 9620–9627.

[14] L. Yuan, F. E. Tay, G. Li, T. Wang, J. Feng, Revisiting knowledge distillation via label smoothing regularization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3903–3911.

[15] R. Müller, S. Kornblith, G. Hinton, When does label smoothing help?, arXiv preprint arXiv:1906.02629 (2019).

[16] M. Lukasik, S. Bhojanapalli, A. Menon, S. Kumar, Does label smoothing mitigate label noise?, in: International Conference on Machine Learning, PMLR, 2020, pp. 6448–6458.

[17] Y. Xu, Y. Xu, Q. Qian, H. Li, R. Jin, Towards understanding label smoothing, arXiv preprint arXiv:2006.11653 (2020).

[18] B. Chen, L. Ziyin, Z. Wang, P. P. Liang, An investigation of how label smoothing affects generalization, arXiv preprint arXiv:2010.12648 (2020).

[19] C.-B. Zhang, P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, M.-M. Cheng, Delving deep into label smoothing, IEEE Transactions on Image Processing 30 (2021) 5984–5996.

[20] H. Zhang, L. Xiao, W. Chen, Y. Wang, Y. Jin, Multi-task label embedding for text classification, arXiv preprint arXiv:1710.07210 (2017).

[21] Y. Dong, P. Liu, Z. Zhu, Q. Wang, Q. Zhang, A fusion model-based label embedding and self-interaction attention for text classification, IEEE Access 8 (2019) 30548–30559.

[22] X. Sun, B. Wei, X. Ren, S. Ma, Label embedding network: Learning label representation for soft training of deep networks, arXiv preprint arXiv:1710.10393 (2017).

[23] C. Chen, H. Wang, W. Liu, X. Zhao, T. Hu, G. Chen, Two-stage label embedding via neural factorization machine for multi-label classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3304–3311.

[24] C. Liu, J. JaJa, Class-similarity based label smoothing for generalized confidence calibration, arXiv preprint arXiv:2006.14028 (2020).

[25] Z. Zhang, M. R. Sabuncu, Self-distillation as instance-specific label smoothing, arXiv preprint arXiv:2006.05065 (2020).

[26] P. Nousi, A. Tefas, Deep learning algorithms for discriminant autoencoding, Neurocomputing 266 (2017) 325–335.

[27] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017). arXiv:cs.LG/1708.07747.

[28] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning (2011).

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

7.6 Quadratic Mutual Information Modeling for Efficient Hashing

The appended paper follows.

Deep Supervised Hashing using Quadratic Spherical Mutual Information for Efficient Image Retrieval

Nikolaos Passalis, Anastasios Tefas

Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki 54124, Greece Tel,Fax: +30-2310996304

Abstract

Several deep supervised hashing techniques have been proposed to allow for extracting compact and efficient neural network representations for various tasks. However, many deep supervised hashing techniques ignore several information-theoretic aspects of the process of information retrieval, often leading to sub-optimal results. In this paper, we propose an efficient deep supervised hashing algorithm that optimizes the learned compact codes using an information-theoretic measure, the Quadratic Mutual Information (QMI). The proposed method is adapted to the needs of efficient image hashing and information retrieval leading to a novel information-theoretic measure, the Quadratic Spherical Mutual Information (QSMI). Apart from demonstrating the effectiveness of the proposed method under different scenarios and outperforming existing state-of-the-art image hashing techniques, this paper provides a structured way to model the process of information retrieval and develop novel methods adapted to the needs of different applications.

Keywords: Deep Supervised Hashing, Compact Representations, Quadratic Mutual Information, Image Hashing

1. Introduction

The vast amount of data available nowadays, combined with the need to efficiently and promptly provide answers to users' queries led to the development of several *hashing* techniques [1, 2]. Hashing provides a way to represent objects using compact codes, that allow for performing fast and efficient queries in large object databases, lowering the computational requirements and accelerating many information retrieval-based applications. The increasing need to perform inference on the edge using embedded devices with limited processing power [3, 4], along with the vast amount of multimedia data collected from various sensors [5, 6], further stress the need for developing efficient hashing methods.

Early hashing methods, e.g., Locality Sensitive Hashing (LSH) [7], focused on extracting generic codes that could, in principle, describe every possible object and information need. However, it was later established that *supervised hashing*, that learns compact hash codes that are tailored to the task at hand, can significantly improve the retrieval precision, as well as reduce the size of the extracted codes. In this way, it is possible to learn even smaller hashing codes, since the extracted code must only encode the information needs for which the users are actually interested in. However, note that the extracted hash codes must also encode part of the semantic relationships between the encoded objects, to allow for providing a meaningful ranking of the retrieved results. Many supervised and semi-supervised hashing methods have been proposed [8, 9, 10]. However, many deep supervised hashing techniques ignore several information-theoretic aspects of the process of information retrieval, often leading to sub-optimal results. For example, many methods employ the pairwise distances between the images [8, 9, 11], or are based on sampling *triplets* that must satisfy specific relationships according to the given ground truth [10, 12]. On the other hand, *information-theoretic* measures, such as mutual information [13], have been proven to provide robust solutions to many machine learning problems, e.g., classification [13]. However, very few steps towards using these measures for supervised hashing tasks have been made so far.

In this paper, we provide a connection between an information-theoretic measure, the Mutual Information (MI) [13], and the process of information retrieval. More specifically, we argue that mutual information can naturally model the process of information retrieval, providing a solid framework to develop efficient retrieval-oriented supervised hashing techniques. Even though MI provides a well-defined theoretical formulation for the problem of information retrieval, applying it in real scenarios is usually intractable, since there is no efficient way to calculate the actual probability densities, that are involved in the calculation of MI. The great amount of data as well as their high dimensionality further complicate the practical application of such measures.

The main contribution of this paper is the proposal of an

Email addresses: passalis@csd.auth.gr (Nikolaos Passalis), tefas@csd.auth.gr (Anastasios Tefas)

efficient deep supervised hashing algorithm that is capable of extracting short and efficient codes using a novel extension of an information-theoretic measure, the Quadratic Mutual Information (QMI) [14]. The architecture of the proposed method is shown in Fig. 1. To derive a practical algorithm that can efficiently scale to large datasets:

1. We adapt QMI to the needs of supervised hashing by employing a similarity measure that is close to the actual distance used for the retrieval process, i.e., the Hamming distance. This gives rise to the proposed *Quadratic Spherical Mutual Information* (QSMI). It is also experimentally demonstrated that the proposed QSMI is more robust compared to the classical Gaussian-based Kernel Density Estimation used in QMI [14], while it does not require careful tuning of any hyper-parameters.
2. We propose using a more smooth optimization objective employing a novel square clamping approach. This allows for significantly improving the stability of the optimization, while reducing the risk of converging to bad local minima.
3. We adapt the proposed approach to work in batch-based setting by employing a method that dynamically estimates the prior probabilities, as they are observed within each batch. In this way, the proposed method can efficiently scale to larger datasets.
4. We demonstrate that the proposed method can be readily extended to efficiently handle different scenarios, e.g., retrieval of unseen classes [15].

The proposed method is extensively evaluated using five image datasets, including two standard datasets used for evaluating supervised hashing methods, the CIFAR10 [16] and NUS-WIDE [17] datasets, and it is demonstrated that it outperforms several existing approaches. Following the suggestions of [15], we also evaluate the proposed method in a different evaluation setup, where the learned hash codes are evaluated using unseen information needs.

The rest of the paper is structured as follows. The related work is discussed in Section 2. The proposed method is presented in detail in Section 3, while the experimental evaluation is provided in Section 4. Finally, Section 5 concludes the paper.

2. Related Work

The increasing interest for learning compact hash codes, together with the great learning capacity of recent deep learning models, led to the development of several deep supervised hashing techniques [11, 18], along with semi-supervised approaches [19, 20] and sophisticated unsupervised ones [21, 22]. Deep supervised hashing techniques involve: a) a deep neural network, that is used to extract a representation from the data, b) a (semi)-supervised loss function, that is used to train the network, and c) a hashing mechanism, e.g., an appropriate non-linearity [18] or regularizer [11], that ensures that the output of the network can be readily transformed into a compact hash code. Most of the proposed methods fall into one of the following two categories according to the loss function employed for learning the supervised codes: a) pairwise-based hashing methods [2, 8, 9, 11, 18, 23, 24, 25] and b) triplet-based hashing methods [10, 12, 26].

Pairwise-based methods work by learning hash codes that minimize / maximize the pairwise distance / log-likelihood between similar / dissimilar pairs, e.g., Convolutional Neural Network (CNN)-based hashing [9], network in network hashing [8], deep hashing network [11], deep pairwise-supervised hashing [24], deep hashing network [11], and deep supervised discrete hashing [18]. More advanced pairwise methods employ margins that allow for learning more regularized representations, e.g., deep supervised hashing [2], use asymmetric hashing schemes, e.g., deep asymmetric pairwise hashing [25], asymmetric deep supervised hashing [23], discriminative deep metric learning for asymmetric discrete hashing [27], or use more advanced techniques to obtain the binary codes, e.g., hashing by continuation [28], or focus on cross-modal retrieval [29].

Triplet-based methods work by sampling an anchor point along with a positive and a negative example [10, 12, 26, 30]. Then, they learn codes that increase the similarity between the anchor and the positive example, while reducing the similarity between the anchor and the negative example. However, triplet-based methods are significantly more computationally expensive than pairwise-based methods, requiring a huge number of triplets to be generated (many of which convey no information, since they are already satisfied by the code learned by the network), limiting their practical application. Also note that many non-deep supervised hashing methods have also been proposed, e.g., [31, 32, 33], but an extensive review of them is out of the scope of this paper. The interested reader is referred to [34] for an extensive literature review on hashing.

More recent works on deep supervised hashing employ objectives based on class-wise loss [35], semantic cluster-based unary loss [36], multi task-based loss [37], list-wise loss [38], or using anchor graphs for defining the loss function and further improving the hashing performance [39]. Furthermore, an end-to-end supervised product quantization approach for information retrieval was proposed in [40], while deep discrete hashing approaches [18, 41], incremental hashing methods [42] and correlation filtering-based fine-grained hashing approaches [43] have also been utilized to the same end.

The use of MI has also been investigated to aid various aspects of the retrieval process. In [44, 45] MI is employed to provide relevance feedback, in [46, 47], MI is used to provide a powerful deep hashing formulation, while in [48] MI is employed for performing locality sensitive hashing. The Shannon's definition for MI is used in [46] and [47], leading to employing a Monte Carlo sampling scheme to approximate MI, together with a differentiable histogram

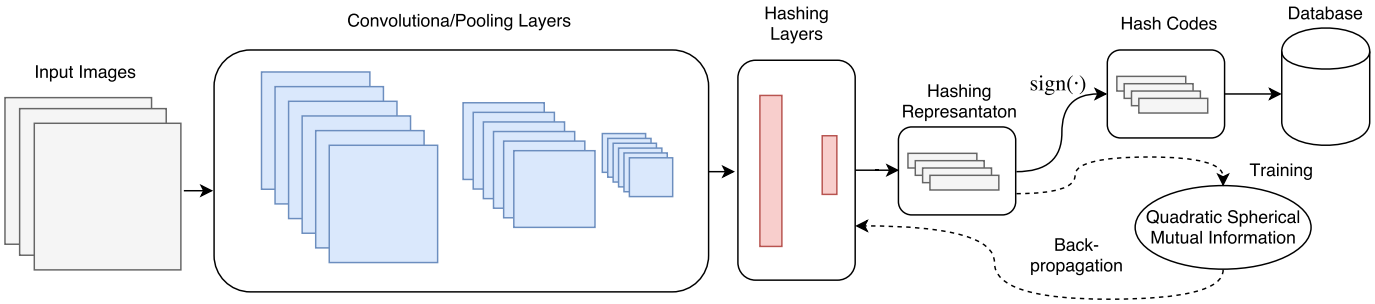


Figure 1: Pipeline of the proposed method: A deep convolutional neural network (CNN) is used to extract a representation that can be used to directly obtain a compact binary hash code. The network is optimized using the proposed Quadratic Spherical Mutual Information loss that is adapted towards the needs of efficient image hashing.

binning technique. It should be noted that our approach is vastly different, since instead of approximating MI through random sampling, we analytically derive computationally tractable solutions for calculating MI through a closed-form solution obtained through a Quadratic Mutual Information formulation. Therefore, even though both approaches begin with the same objective, i.e., maximizing mutual information between the hash codes and the information needs, a significantly different approach is employed for tackling the intractable problem of mutual information estimation and optimization in high dimensional spaces.

To the best of our knowledge, this is one of the first works that employs a quadratic spherical mutual information loss fully adapted to the needs of deep supervised hashing. Apart from deriving a practical algorithm and demonstrating its ability to outperform existing state-of-the-art methods, the proposed method provides a complete framework that can be used to model the process of information retrieval. This formulation is fully differentiable allowing for the end-to-end optimization of deep neural networks for any retrieval-related task, ranging from learning retrieval-oriented representations and compact hash codes to fine-tuning the extracted representations using relevance feedback.

3. Proposed Method

The proposed method is presented in detail in this Section. First, the links between mutual information and information retrieval are provided. Then, the quadratic mutual information is introduced, the proposed quadratic spherical mutual information is derived and several aspects of the proposed method are discussed.

3.1. Information Retrieval and Mutual Information

Let $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ be a collection of N images, where $\mathbf{y}_i \in \mathbb{R}^n$ is the representation of the i-th image extracted using an appropriate feature extractor, e.g., a deep neural network. Each image $\mathbf{y}_i$ fulfills a set of information needs. For example, an image that depicts a "red car near a beach" fulfills at least the following information needs: "car", "red car", "beach", "car near beach". Note that the information needs that an image actually fulfills depend on both its content and the needs of the users, since, depending on the actual application, the interests of the users are usually focused on a specific area. For example, an image of a man entering a bank represents different information needs for a forensics database used by the police to identify suspects and for a generic web search engine. The problem of information retrieval can be then defined as follows: *Given an information need q retrieve the images of the collection $\mathcal{Y}$ that fulfill this information need and rank them according to their relevance to the given information need.* This work focuses on *content-based* information retrieval [49], where the information need q is expressed through a *query image* $\mathbf{q} \in \mathbb{R}^n$, that is usually not part of the collection $\mathcal{Y}$.

To be able to measure how well an information retrieval system works, a ground truth set that contains a set of information needs and the corresponding images that fulfill these information needs is usually employed. Let M be the number of information needs $\mathcal{Q} = \{q_1, q_2, ..., q_M\}$. Then, for each information need q_i, a set of images $\mathcal{Q}_i = \{\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, ..., \mathbf{y}_{N_i}^{(i)}\}$, where $\mathbf{y}_j^{(i)} \in \mathbb{R}^n$ is the representation of the j-th image that fulfills the i-th information need, is given. Note that $|\mathcal{Q}_i| = N_i$. Since all these images fulfill the same information need, they can be all used as queries to express this information need. However, there are also other images, which are usually not known beforehand, that also express the same information need and they can be also used to query the database. The distribution of the images that fulfill the i-th information need can be modeled using the *conditional probability density* function $p(\mathbf{y}|q_i)$.

Let Y be a random vector that represents the images and Q be a random variable that represents the information needs. The Shannon's entropy of the information needs, that expresses the uncertainty regarding the information need that a randomly sampled image fulfills, is

defined as [13]:

$$H(Q) = -\sum_q P(q)log(P(q)), \quad (1)$$

where $P(q)$ is the prior probability of the information need q, i.e., the probability that a random image of the collection fulfills the information need q. Note that above definition implicitly assumes that the information needs are mutually exclusive, i.e., $\sum_q P(q) = 1$, or equivalently, that each image satisfies only one information need. This is without loss of generality, since it is straightforward to extend this definition to the general case, where each image can satisfy multiple information needs, simply by measuring the entropy of each information need separately: $H(Q) = -\sum_q \Big(P(q)log(P(q)) + (1-P(q))log(1-P(q))\Big)$.

To simplify the presentation of the proposed method, we assume that the information needs are mutually exclusive. Nonetheless, the proposed approach can be still used with minimal modifications, as we also experimentally demonstrate in Section 4, even when this assumption does not hold. When the query vector is known, then the uncertainty of the information need that it fulfills can be expressed by the conditional entropy:

$$H(Q|Y) = -\int_{\mathbf{y}} p(\mathbf{y}) \left(\sum_q p(q|\mathbf{y})log(p(q|\mathbf{y}))\right) d\mathbf{y}. \quad (2)$$

Mutual information is defined as the amount by which the uncertainty for the information needs is reduced after observing the query vector:

$$\begin{aligned} I(Q, Y) &= H(Q) - H(Q|Y) \\ &= \sum_q \int_{\mathbf{y}} p(q, \mathbf{y})log\left(\frac{p(q, \mathbf{y})}{P(q)p(\mathbf{y})}\right) d\mathbf{y}. \end{aligned} \quad (3)$$

It is easy to see that MI can be interpreted as the Kullback-Leibler divergence between $p(q, \mathbf{y})$ and $P(q)p(\mathbf{y})$. It is desired to maximize the MI between the representation of the images Y and the information needs Q, since this ensures that the uncertainty regarding the information need, that a query image expresses, is minimized. Also, note that MI models the intrinsic uncertainty regarding the query vectors, since it employs the conditional probability density between the information needs and the images. On the other hand, it is usually intractable to directly calculate the required probability density $p(\mathbf{y}|q_i)$ and the corresponding integral in (3), limiting the practical applications of MI. However, as it is demonstrated later, it is possible to efficiently estimate the aforementioned probability density and derive a practical algorithm that maximizes the MI between a representation and a set of information needs.

3.2. Quadratic Mutual Information

When the aim is not to calculate the exact value of MI, but to optimize a distribution that maximizes the MI, then a quadric divergence metric, instead of the Kullback-Leibler divergence, can be used. In this way, the *Quadratic Mutual Information* (QMI) is defined as [14]:

$$I_T(Q, Y) = \sum_q \int_{\mathbf{y}} \left(p(q, \mathbf{y}) - P(q)p(\mathbf{y})\right)^2 d\mathbf{y}. \quad (4)$$

By expanding (4), QMI can be expressed as the sum of three *information potentials* as $I_T(Q, Y) = V_{IN}(Q, Y) + V_{ALL}(Q, Y) - 2V_{BTW}(Q, Y)$, where: $V_{IN}(Q, Y) = \sum_q \int_{\mathbf{y}} p(q, \mathbf{y})^2 d\mathbf{y}$, $V_{ALL}(Q, Y) = \sum_q \int_{\mathbf{y}} P(q)^2 p(\mathbf{y})^2 d\mathbf{y}$, and $V_{BTW}(Q, Y) = \sum_q \int_{\mathbf{y}} p(q, \mathbf{y})P(q)p(\mathbf{y})d\mathbf{y}$.

To calculate these quantities, the probability $P(q)$ and the densities $p(\mathbf{y})$ and $p(q, \mathbf{y})$ must be estimated. The prior probabilities depend only on the distribution of the information needs in the collection of images. Therefore, for the i-th information need: $P(q_i) = \frac{N_i}{N}$, where N_i is the number of images that fulfill the i-th information need. The conditional density of the images that fulfill the i-th information need can be estimated using the Parzen window estimation method [50]:

$$p(\mathbf{y}|q_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} K(\mathbf{y} - \mathbf{y}_j^{(i)}; \sigma^2), \quad (5)$$

where $K(\mathbf{y}; \sigma^2)$ is a Gaussian kernel (in an n-dimensional space) with width σ defined as:

$$K(\mathbf{y}; \sigma) = \frac{1}{(2\pi)^{n/2}\sqrt{\sigma}} exp(-\frac{\mathbf{y}^T\mathbf{y}}{2\sigma}). \quad (6)$$

Then, the joint probability density can be estimated as:

$$p(q_i, \mathbf{y}) = p(\mathbf{y}|q_i)p(q_i) = \frac{1}{N} \sum_{j=1}^{N_i} K(\mathbf{y} - \mathbf{y}_j^{(i)}; \sigma^2), \quad (7)$$

while the density of all the images as:

$$p(\mathbf{y}) = \frac{1}{N} \sum_{j=1}^{N} K(\mathbf{y} - \mathbf{y}_j^{(i)}; \sigma^2). \quad (8)$$

By substituting these estimations into the definitions of the information potentials, the following quantities are obtained:

$$V_{IN}(Q, Y) = \frac{1}{N^2} \sum_{k=1}^{M} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} K(\mathbf{y}_i^{(k)} - \mathbf{y}_j^{(k)}, 2\sigma^2), \quad (9)$$

$$V_{ALL}(Q, Y) = \frac{1}{N^2} \left(\sum_{k=1}^{M} \left(\frac{N_k}{N}\right)^2\right) \sum_{i=1}^{N} \sum_{j=1}^{N} K(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2), \quad (10)$$

and

$$V_{BTW}(Q, Y) = \frac{1}{N^2} \sum_{k=1}^{M} \left(\left(\frac{N_k}{N}\right) \sum_{i=1}^{N_k} \sum_{j=1}^{N} K(\mathbf{y}_i^{(k)} - \mathbf{y}_j, 2\sigma^2)\right), \quad (11)$$

where the following property regarding the convolution between two Gaussian kernels was used: $\int_{\mathbf{y}} K(\mathbf{y} - \mathbf{y}_i; \sigma^2)K(\mathbf{y} - \mathbf{y}_j; \sigma^2)d\mathbf{y} = K(\mathbf{y}_i - \mathbf{y}_j, 2\sigma^2)$. The information potential V_{IN} expresses the interactions between the images that fulfill the same information need, the information potential V_{ALL} the interactions between all the images

of the collection, while the potential V_{BTW} models the interactions of the images that fulfill a specific information need against all the other images. Therefore, the QMI formulation allows for the efficient calculation of MI, since the MI is expressed as a weighted sum over the pairwise interactions of the images of the collection.

Using Parzen window estimation with a Gaussian kernel for estimating the probability density leads to the implicit assumption that the similarity between two images is expressed through their Euclidean distance. Thus, the images that fulfill an information need expressed by a query vector $\mathbf{q}$ can be retrieved simply using nearest-neighbor search.

3.3. Quadratic Spherical Mutual Information Optimization

Even though QMI allows for more efficient optimization of distributions, it suffers from several limitations: a) QMI involves the calculation of the pairwise similarity matrix between all the images of a collection. This quickly becomes intractable as the size of the collection increases. b) Selecting the appropriate width for the Gaussian kernels is not always straightforward, as a non-optimal choice can distort the feature space and slow down the optimization. c) The discrepancy between the distance metric used for QMI (Euclidean distance) and the distance used for the actual retrieval of the hashed images (Hamming distance) can negatively affect the retrieval accuracy. Finally, d) it was experimentally observed that directly optimizing the QMI is prone to bad local minima, due to the linear behavior of the loss function that fails to distinguish between the pairs of images that cause high error and those which have a smaller overall effect on the learned representation (more details are given later in this Section).

To overcome the limitations (b) and (c), we propose the *Quadratic Spherical Mutual Information* (QSMI). The proposed QSMI method replaces the Gaussian kernel in (6), used for calculating the similarity between two images in the information potentials in (9), (10), and (11), with the cosine similarity:

$$S_{cos}(\mathbf{y_1}, \mathbf{y_2}) = \frac{1}{2}\left(\frac{\mathbf{y}_1^T\mathbf{y}_2}{\|\mathbf{y}_1\|_2\|\mathbf{y}_2\|_2} + 1\right), \tag{12}$$

where $\|\cdot\|_2$ is the l^2 norm of a vector. In this way, we maintain the computationally efficient QMI formulation and avoid the need for manually tuning the width parameter of the Gaussian kernel, while adopting a formulation that is close to the Hamming distance that is actually used for the retrieval process [33].

Therefore, QSMI is defined as:

$$I_T^{cos}(Q, Y) = V_{IN}^{cos}(Q, Y) + V_{ALL}^{cos}(Q, Y) - 2V_{BTW}^{cos}(Q, Y), \tag{13}$$

where

$$V_{IN}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{k=1}^{M}\sum_{i=1}^{N_k}\sum_{j=1}^{N_k} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j^{(k)}), \tag{14}$$

$$V_{ALL}^{cos}(Q, Y) = \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{N_k}{N}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j), \tag{15}$$

and

$$V_{BTW}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{k=1}^{M}\left(\left(\frac{N_k}{N}\right)\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j)\right). \tag{16}$$

Note that when the information needs are equiprobable, i.e., $P(q) = \frac{1}{M}$, then QSMI can be simplified as $I_T^{cos}(Q, Y) = V_{IN}^{cos}(Q, Y) - V_{BTW}^{cos}(Q, Y)$. Therefore, when this assumption holds, QSMI can be easily implemented just by defining the similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$, where $[\mathbf{S}]_{ij} = S_{cos}(\mathbf{y}_i, \mathbf{y}_j)$ and the notation $[\mathbf{S}]_{ij}$ is used to refer to the i-th row and j-th column of matrix $\mathbf{S}$. Then, QSMI can be calculated as:

$$I_T^{cos} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta}\odot\mathbf{S} - \frac{1}{M}\mathbf{S}\right)\mathbf{1}_N, \tag{17}$$

where the indicator matrix is defined as:

$$[\mathbf{\Delta}]_{ij} = \begin{cases} 1, & \text{if the } i\text{-th and the } j\text{-th documents are similar} \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

The notation $\mathbf{1}_N \in \mathbb{R}^N$ is used to refer an N-dimensional vector of 1s, while the operator $\odot$ denotes the Hadamard product between two matrices. Please refer to the Appendix A for a more detailed derivation. This formulation also allows for directly handling information needs that are not mutually exclusive. In this case, the values of the indicator matrix are appropriately set to 1, if two images share at least one information need.

Instead of directly optimizing the QSMI, we propose using a "square clamp" around the similarity matrix $\mathbf{S}$, smoothing the optimization surface. Therefore, given that the values of $\mathbf{S}$ range in the unit interval, the loss function is re-derived as:

$$\mathcal{L}_{QSMI} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta}\odot(\mathbf{S}-1)\odot(\mathbf{S}-1) + \frac{1}{M}(\mathbf{S}\odot\mathbf{S})\right)\mathbf{1}_N. \tag{19}$$

As shown in Figure 2a this formulation penalizes the pairs with larger error more heavily than those with smaller error, allowing for discovering more robust solutions. This modification effectively addresses the limitation (d), as we also experimentally demonstrate in the ablation study given in Section 4.

The complexity for calculating QSMI is quadratic, since calculating V_{IN}^{cos}, V_{ALL}^{cos} and V_{BTW}^{cos} require a quadratic number of similarity calculations, i.e., $O(N^2)$. To allow for scaling to larger datasets, batch-based optimization is used. This allows for reducing the complexity of QMI from $O(N^2)$ to just $O(N_B^2)$ for one optimization step, where N_B is the used batch size that typically ranges from 64 to 256. Therefore, the total complexity for completing one training epoch is reduced from $O(N^2)$ to $O(NN_B)$. Note that during inference, the proposed method does not require any method-specific components, and, as a result, the complexity only depends on the used network architecture and the length of the hash codes. However, also

note that this implies that each batch will contain images only from a subsample of the available information needs. This in turn means that the observed in-batch prior probability $P(q)$ will not match the collection-level prior, leading to underestimating the influence of the potential V_{ALL} to the optimization. To account for this discrepancy, we propose a simple heuristic to estimate the in-batch prior, i.e., the value of M in (28): M is estimated as $M = N_B^2/(\mathbf{1}_{N_B}^T \boldsymbol{\Delta} \mathbf{1}_{N_B})$, where N_B is the batch size. To understand the motivation behind this, consider that if the whole collection was used for the optimization, then the number of 1s in $\boldsymbol{\Delta}$ would be: $M(\frac{N}{M})^2 = \mathbf{1}_N^T \boldsymbol{\Delta} \mathbf{1}_N$. Solving this equation for M yields the value used for approximating M. Note that the value of M is not constant and depends on the distribution of the samples in each batch. It was experimentally verified that this approach indeed improves the performance of the proposed method over using a constant value for M.

3.4. Deep Supervised Hashing using QSMI

The proposed QSMI is used to train a deep neural network to extract short binary hash codes, as shown in Fig. 1. Let $\mathbf{x}$ be the raw representation of an image (e.g., the pixels of an image) and let $\mathbf{y} = f_{\mathbf{W}}(\mathbf{x}) \in \mathbb{R}^n$ be the output of a neural network $f_{\mathbf{W}}(\cdot)$, where $\mathbf{W}$ denotes the matrix of the parameters of the network and n is the length of the hash code. Apart from learning a representation that minimizes the J_{QSMI} loss, the network must generate an output that can be easily translated into a binary hash code. Several techniques have been proposed to this end, e.g., using the *tanh* function [34]. In this work, the output of the network is required to be close to two possible values, either 1 or -1. Therefore, the used hashing regularizer is defined, following the recent deep supervised hashing approaches [2], as:

$$\mathcal{L}_{hash} = \sum_{i=1}^{N} \| |\mathbf{y}_i| - \mathbf{1}_n \|_1, \tag{20}$$

where $|\cdot|$ denotes the absolute value operator and $||\cdot||_1$ denotes the l^1 norm. The final loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{QSMI} + \alpha \mathcal{L}_{hash}, \tag{21}$$

where α is the weight of the hashing regularizer. The network $f_{\mathbf{W}}(\cdot)$ can be then trained using gradient descent, i.e., $\Delta\mathbf{W} = -\eta \frac{\partial J}{\partial \mathbf{W}}$, where η is the used learning rate. Please refer to Appendix A on details regarding the derivation of $\frac{\partial J}{\partial \mathbf{W}}$. After training the network, the hash codes can be readily obtained using the $sign(\mathbf{y})$ function.

Even though learning highly discriminative hash codes is desired for retrieving data that belong to the training domain, it can negatively affect the retrieval for previously unseen information needs [15]. The proposed method can be also easily modified to optimize the hash codes toward other *unsupervised* information needs. Even though any source of information can be used, in this work the information needs are discovered by clustering the training data. This allows for discovering information needs dictated by the structure of the data. Let $\mathcal{L}_{QSMI+U}$ denote the loss induced by applying the QMSI loss function on these information needs. Then, the final loss function for this semi-supervised variant is defined as: $\mathcal{L} = \mathcal{L}_{QSMI} + \alpha \mathcal{L}_{hash} + \beta \mathcal{L}_{QSMI+U}$. This variant is used for the experiments conducted in Section 4.4.

4. Experimental Evaluation

The proposed method is extensively evaluated in this Section, using both an ablation study and comparing it to other state-of-the-art methods. First, the used datasets and the employed evaluation setup are briefly described. The hyper-parameters and the network architectures used for the evaluation are provided in Appendix B. Finally, an ablation study is provided and the proposed method is evaluated using five different datasets.

4.1. Datasets and Evaluation Metrics

Five image datasets are used to evaluate the proposed method in this paper: The Fashion MNIST dataset, the CIFAR10 dataset, the NUS-WIDE dataset, the MS COCO dataset, as well as the ILSVRC dataset.

The Fashion MNIST dataset is composed of 60,000 training images and 10,000 test images [51]. The size of each image is 28×28 pixels (gray-scale images) and there is a total of 10 different classes (each one expresses a different information need). The whole training set was used to train the networks and build the database, while the test set was used to query the database and evaluate the performance of the methods. The CIFAR10 dataset is composed of 50,000 training images and 10,000 test images [16]. The size of each image is 32×32 pixels (color images) and there is a total of 10 different classes (information needs). The NUS-WIDE is a large-scale dataset that contains 269,648 images that belong to 81 different concepts [17]. The images were resized to 224×224 pixels before feeding them to the network. Following [24], only images that belong to the 21 most frequent concepts, i.e., 195,834 images, were used for training/evaluating the methods. Each image might belong to multiple different concepts, i.e., the information needs are not mutually exclusive. For evaluating the methods, two images were considered relevant if they share at least one common concept, which is the standard protocol used for this dataset [24]. Similarly to the other two datasets, the whole training set (193,734 randomly sampled images) was used to train the networks and build the database, while 2,100 randomly sampled queries (100 from each category) were employed to evaluate the methods. The MS COCO dataset [52] is another large-scale multi-label dataset that contain 80 different categories. For this dataset we followed a similar standardized setup, as reported in [53]. The ILSVRC dataset [54] is a large-scale image dataset that contains more than one million images that belong to 1,000 different categories. All training images were used for optimizing the models and building the

database, while the validation images were used for querying the database and evaluating the quality of the learned hash codes. Note that a 20-way classification setup is employed, i.e., for each experiment 20 different classes were randomly selected to build the database. All experiments were repeated 20 times and the mean and standard deviation is reported for all the conducted experiments.

To evaluate the proposed method, the following four metrics were used: precision, recall, mean average precision (mAP), and precision within hamming radius of 2. Nearest neighbor search using the Hamming distance was used to retrieve the relevant documents [55]. Following [55], precision is defined as $Pr(q,k) = \frac{rel(q,k)}{k}$, where k is the number of retrieved objects and $rel(q,k)$ is the number of retrieved objects that fulfill the same information need as the query q, while recall is defined as $Rec(q,k) = \frac{rel(q,k)}{ntotal(q)}$, where $ntotal(q)$ is the total number of database objects that fulfill the same information as q. Precision within hamming radius of 2 is defined as: $Pr_{H2}(q) = \frac{rel_{H2}(q)}{total_{H2}(q)}$, where $rel_{H2}(q)$ is the number of relevant documents within hamming distance 2 from the query, while $total_{H2}(q)$ is the total number of documents within hamming distance 2 from the query. Furthermore, we use the notation (AP_α) to refer to the average precision over all queries at the α-th recall level. For all the experiments conduncted in this work we report the mean Average Precision at eleven equally spaced recall points (0, 0.1, ..., 0.9, 1) calculated as:

$$mAP = \frac{1}{11} \sum_{\alpha=0,\ 0.1,\ ...,\ 0.9,\ 1} AP_\alpha. \tag{22}$$

For multi-label datasets, such as the NUS-WIDE dataset, we calculated precision based on the agreement on the labels of the query, e.g., retrieving objects that carry only half of the labels of the query would lead to a 50% precision for each of them, while retrieving an object that is annotated with all the labels of the query (and possibly more) would lead to a precision of 100% for the specific object. Then, using this definition, mean Average Precision can be similarly calculated for multi-label datasets.

Finally, note that the proposed method was implemented using the PyTorch framework [56] (version 1.0), while the experimental evaluation was conducted on an 8-core workstation that was equipped with an RTX 2060 Graphics Processing Unit (GPU). An open-source implementation of the proposed method is available at `https://github.com/passalis/qsmi`.

4.2. Ablation Study

First, the Fashion MNIST dataset [51], is used to perform an ablation study. The effect of various design choices, i.e., using the proposed clamped loss and spherical formulation, is evaluated in Table 1. The mean Average Precision (mAP) is averaged over 5 runs, while the code length was set to 48 bits for these experiments. Several conclusions can be drawn from the results reported in Table 1. First, employing the proposed clamped loss, instead of directly optimizing the QMI ($\sigma = 10$), improves the hashing precision, confirming our hypothesis regarding the benefits of using the proposed clamped loss (as also described in the previous Section and shown in Fig. 2a). This is also confirmed in the learning curve shown in Fig. 2b, where both the proposed clamped loss and MI are monitored during the optimization. Optimizing the proposed clamped loss is directly correlated with the QMI and the proposed QSMI, both of which steadily increase during the 50 training epochs. When the spherical formulation is used (QSMI method), then the mAP further increase to 86.1% from 72.7% (standard QMI formulation). The effect of the regularization parameter α for three datasets (Fashion MNIST, CIFAR-10 and NUS-WIDE) is evaluated in Fig. 3. The proposed method is quite stable and consistently achieve the best performance for $\alpha = 0.01$. However, note that this parameter can have a significant effect on the learned hash codes, since a sub-optimal choice can significantly reduce the retrieval precision, as demonstrated in Fig. 3, especially for the NUS-WIDE dataset.

Table 1: Ablation study using the Fashion MNIST dataset (the mAP is reported)

Clamped	**Spherical**	**mAP**	**precision (< 2bits)**
No	No	0.727 ± 0.008	0.674 ± 0.021
Yes	No	0.816 ± 0.009	0.864 ± 0.010
Yes	Yes	$\mathbf{0.861 \pm 0.004}$	$\mathbf{0.876 \pm 0.004}$

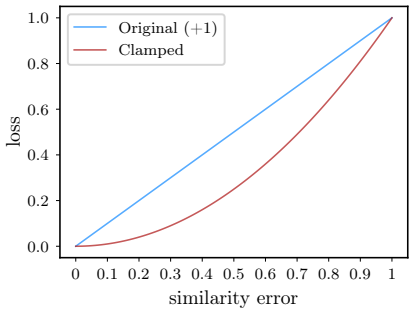
Table 2: Fashion MNIST Evaluation (the mAP for different hash code lengths is reported)

Method	**12 bits**	**24 bits**	**36 bits**	**48 bits**
DSH	0.761 ± 0.02	0.792 ± 0.01	0.809 ± 0.01	0.819 ± 0.01
DPSH	0.767 ± 0.02	0.773 ± 0.01	0.774 ± 0.01	0.759 ± 0.01
Proposed	$\mathbf{0.842 \pm 0.01}$	$\mathbf{0.857 \pm 0.01}$	$\mathbf{0.858 \pm 0.01}$	$\mathbf{0.861 \pm 0.01}$

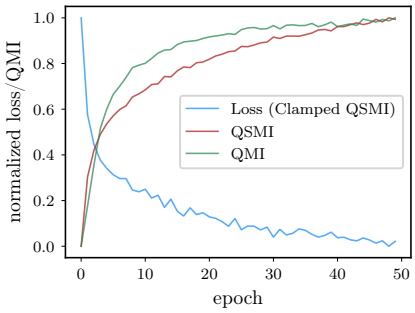
The proposed method was compared to two other state-of-the-art techniques, the Deep Supervised Hashing (DSH) method [2] and the Deep Pairwise Supervised Hashing (DPSH) method [24]. We carefully implemented these methods in a batch-based setting and we tuned their hyper-parameters to obtain the best performance (please refer to Appendix B). The evaluation results are shown in Table 2. The proposed method is abbreviated as "QSMIH" and significantly outperforms the other two competitive pairwise hashing techniques. Recall that a deep CNN, that was trained from scratch, was employed for the conducted experiments. Again, the proposed method outperforms all the other methods for all the evaluated hash code lengths.

4.3. Supervised Hashing Evaluation

The evaluation results for the CIFAR10 dataset are reported in Table 3. The proposed method outperforms all the other techniques by a large margin for small code

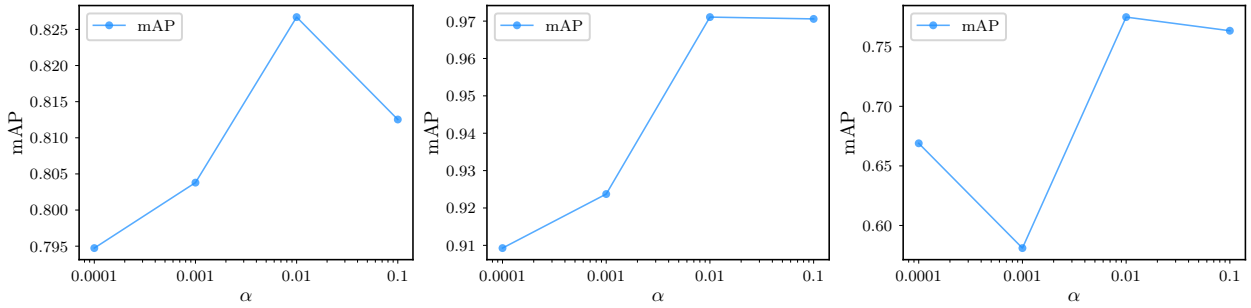


(a) Comparing the behavior of different loss functions for various errors (the original QMI loss is shifted by 1 to allow for easily comparing the plots)



(b) The proposed Clamped QSMI loss and the QSMI and QMI measures during the optimization (the plots have been normalized to the unit interval)

Figure 2: Ablation Study: Studying the differences between the original loss and the proposed hamming loss (Fig. 2a), and the effect of the optimization on the QMI and QSMI measures (Fig. 2b).



(a) Fashion MNIST (b) CIFAR-10 (c) NUS-WIDE

Figure 3: Sensitivity Analysis: Studying effect of the regularization parameter α

Table 3: CIFAR10 Evaluation (the mAP for different hash code lengths is reported)

Method	8 bits	12 bits	24 bits	36 bits	48 bits
DSH	0.936	0.958	0.967	0.970	0.970
DPSH	0.776	0.933	**0.971**	**0.971**	**0.971**
QSMIH	**0.962**	**0.970**	**0.971**	**0.971**	**0.971**

Table 4: Training and inference time evaluation (time per batch is reported)

Method	Feed-forward (inference)	Back-propagation (training)
DSH	2.9 ms	0.7 ms
DPSH	4.1 ms	0.7ms
Proposed	3.4 ms	0.7ms

lengths, i.e., 8 and 12 bits. For larger hash codes, the proposed method performs equally well with the DSH and DPSH methods. However, the proposed method is capable of achieving almost the same performance as the DSH and DPSH methods using less than half of the bits, highlighting the expressive power of the proposed technique.

The time required for training and performing inference using three different methods is reported in Table 4. More specifically, the time needed for performing one weight update in the last layer, i.e., calculating the loss and back-propagating the gradients in the last layer, is 3.4 ms for the proposed method, 2.9 ms for the DSH method and 4.1 ms for the DPSH method. Note that the actual overhead of the proposed approach during training is small (less than 20% compared to DSH method), while there is no overhead during inference, since the same architecture is used (0.7 ms is required per batch for feed-forwarding through the hashing layer of the network). For all these cases, the batch size was set to 128. Furthermore, the hash codes learned using the proposed method were plotted using the t-SNE algorithm [57] in Fig. 4. First, note the generalization abilities of the learned hash codes, since the structure of the space remains the same for both the database and query codes, while the representation of different classes remains quite disentangled, both for the train and test sets. There are some entanglements between different classes on the query/test split, but this only concerns a relatively small number of samples. Also, note that the intra-class similarities seem to be maintained, since the samples that belong to the same class do not collapse into a single point, but instead scatter around each class, both for the train and test sets, leveraging the additional bits to encode these similarities.

The proposed method was also compared to other competitive deep supervised methods in Table 5. The proposed method is compared to DNNH [58], DSH [2], DPSH [24], HashNet [59], MIHash [46, 47], HashGAN [60] and PGDH [61] methods using the same evaluation protocol, i.e., 1,000 test images are sampled, 5,000 images

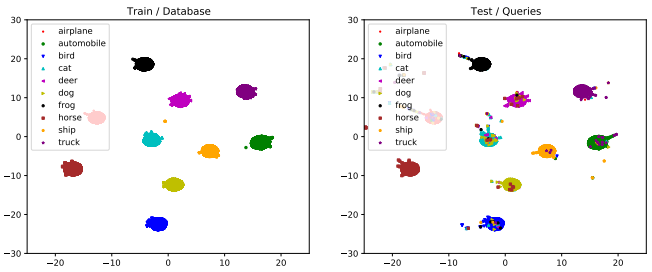


Figure 4: Visualization of the hash codes (48 bits) learned using the proposed method on the CIFAR-10 dataset (30,000 training / database samples are plotted on the left plot, while 10,000 testing samples / queries are plotted on the right plot). The t-SNE [57] algorithm was used for the visualization, while the perplexity of the algorithm was set to 2.

Table 5: CIFAR10 Evaluation: Comparison with other state-of-the-art approaches (the mAP for different hash code lengths is reported)

Method	16 bits	32 bits	64 bits
DNNH [58]	0.555	0.558	0.623
DSH [2]	0.689	0.691	0.716
DPSH [24]	0.646	0.661	0.686
HashNet [59]	0.703	0.711	0.739
HashGAN [60]	0.668	0.731	0.749
PGDH [61]	0.736	0.741	0.762
MIHash [46, 47]	0.760	**0.776**	0.761
Proposed	**0.762**	**0.776**	**0.780**

For all the evaluated methods, apart from the proposed one, the results are as reported in the corresponding literature, i.e., [61], [60] and [62]. The same setup and neural network architecture are used for the evaluated methods.

are used for training the models and the rest of them are used to form the database. Note that the results for the competitive methods are as reported in the corresponding literature [60, 61, 62], while we also used the same neural network architecture (AlexNet [54]) for the experiments conducted using the proposed method. The proposed method significantly outperforms most of the evaluated methods, including the recently proposed HashGAN [60] and PGDH [61] methods. It also achieves comparable performance with the MIHash approach for 16 and 32 bits. However, for longer hash codes (64 bits), it further increases the mAP to 0.78 from 0.76 (next best performing method).

The proposed QSMIH method was also evaluated using the larger-scale NUS-WIDE dataset that contains 269,648 images that belong to 81 different concepts. Following [24], we used the images that belong to the 21 most frequent concepts, i.e., 195,834 images. Note that an image might belong to more than one concept, i.e., fulfill multiple information needs. Furthermore, instead of using a subsample of the training set, we used the whole training set (193,734 randomly sampled images) to learn the hash codes (all the methods were used in a batch setting), and a test set of 2,100 randomly sampled queries was employed to evaluate the methods. Since there are many differences in the evaluation protocol used by different papers for this dataset, we compared the proposed method to the DSH and DPSH

Table 6: NUS-WIDE Evaluation (the mAP for different hash code lengths is reported)

Method	8 bits	12 bits	24 bits	36 bits	48 bits
DSH	0.660	0.659	0.671	0.689	0.694
DPSH	0.735	0.748	0.759	0.758	0.755
Proposed	**0.746**	**0.753**	**0.766**	**0.764**	**0.763**

Table 7: ILSVRC Evaluation (20-way experiments, the mAP for different hash code lengths is reported)

Method	16 bits	32 bits	48 bits
DSH	0.922 ± 0.006	0.941 ± 0.004	0.945 ± 0.003
DPSH	0.782 ± 0.036	0.909 ± 0.049	0.939 ± 0.020
DCH	0.944 ± 0.012	0.911 ± 0.016	0.902 ± 0.005
Proposed	$\mathbf{0.945 \pm 0.004}$	$\mathbf{0.951 \pm 0.004}$	$\mathbf{0.953 \pm 0.003}$

methods using the same network and evaluation setup. The evaluation results are shown in Table 6. Again, the proposed method outperforms the rest of the evaluated methods for any code length.

The proposed method was also evaluated using the ILSVRC dataset. A DenseNet backbone pre-trained on the same dataset was used [63], while the same hashing architecture and experimental setup as the one used for the NUS-WIDE dataset were employed. Note that a 20-way retrieval setup, that was repeated 20 times, was used for the evaluation. The experimental results are reported in Table 7. Again, the proposed method outperforms the other evaluated method, i.e., DSH [2], DPSH [24] and DCH [64], leading to both larger mAP, as well as smaller standard deviation among the different evaluation runs.

Finally, we also evaluated the proposed method with a recent state-of-the-art approach for deep quantization, the Central Similarity Quantization (CSQ) approach [53], as well as against several other hashing approaches, demonstrating that further improvements that can be obtained when the proposed method is employed. We followed the same setup as in [53] and evaluated the proposed method on both the NUS-WIDE and MS COCO datasets using a ResNet-50 architecture. The evaluation results are provided in Table 8. Note that again the proposed method led to the overall best results for all the evaluated datasets and hash code lengths.

Table 8: MS COCO and NUS-WIDE Evaluation using a ResNet-50 architecture (the mAP for the first 5000 results for different hash code lengths is reported). The results for the competitive methods are reported from [53].

Method	MS COCO		NUS-WIDE	
	32 bits	64 bits	32 bits	64 bits
CNNH [9]	0.617	0.620	0.659	0.647
DNNH [58]	0.651	0.647	0.738	0.754
DHN [11]	0.731	0.745	0.759	0.771
HashNet [59]	0.773	0.788	0.775	0.790
DCH [64]	0.801	0.825	0.795	0.818
CSQ [53]	0.838	0.861	0.825	0.839
Proposed	**0.854**	**0.883**	**0.830**	**0.842**

Table 9: CIFAR10 Evaluation - Retrieval of Unseen Information Needs (the mAP for different hash code lengths is reported)

Method	12 bits	24 bits	36 bits
DSH	0.615 ± 0.065	0.689 ± 0.063	0.674 ± 0.055
DPSH	0.568 ± 0.082	0.635 ± 0.078	0.658 ± 0.048
Proposed	$\mathbf{0.691 \pm 0.093}$	$\mathbf{0.795 \pm 0.061}$	$\mathbf{0.682 \pm 0.144}$

4.4. Retrieval of Unseen Information Needs

Finally, the proposed method was evaluated using the evaluation setup proposed in [15], i.e., the 75% of the classes were used to train the models and the rest 25% were used to evaluate the models. The process was repeated 5 times using different class/information needs splits and the mean and standard deviation are reported. The evaluation results are shown in Table 9. The proposed method also employed 5 unsupervised information needs (discovered using the k-means algorithm on the 75% of the training data). The weight of the unsupervised loss in the optimization was set to $\beta = 0.5$. The proposed variant, denoted by "QSMIH+U" leads to significantly more regularized representations, that do not collapse outside the training domain, increasing the mAP for unseen classes from 0.689 to 0.795, demonstrating the flexibility of the proposed approach as well as its effectiveness in this setup.

5. Conclusions

A deep supervised hashing algorithm, adapted to the needs of efficient image retrieval, that optimizes the learned codes using an novel information-theoretic measure, the Quadratic Spherical Mutual Information, was proposed. The proposed method was evaluated using five different datasets and evaluation setups and compared to other state-of-the-art supervised hashing techniques. The proposed method outperformed all the other evaluated methods regardless the size of the used dataset and training setup, exhibiting a significantly more stable behavior than the rest of the evaluated methods. More specifically, when used with a randomly initialized network, the proposed QSMIH method managed to outperform the rest of the methods by a large margin. On the other hand, when combined with powerful pre-trained networks, again it yielded the best results regardless the length of the used hash code. Also, the proposed method provides theoretical justification for several existing deep supervised hashing techniques, while also paves the way for developing more advanced representation learning techniques for information retrieval using the proposed information-theoretic formulation, e.g., handling cross-modal retrieval tasks [65].

Appendix A - Implementation Details

To simplify the implementation of the proposed method, we assume that all the information needs are equiprobable: $P(q) = \frac{1}{M}$. Then, the information potentials $V_{ALL}^{cos}(Q, Y)$ and $V_{BTW}^{cos}(Q, Y)$ can be calculated as:

$$\begin{aligned} V_{ALL}^{cos}(Q, Y) &= \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{N_k}{N}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= \frac{1}{N^2}\left(\sum_{k=1}^{M}\left(\frac{1}{M}\right)^2\right)\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j), \end{aligned} \tag{23}$$

and

$$\begin{aligned} V_{BTW}^{cos}(Q, Y) &= \frac{1}{N^2}\sum_{k=1}^{M}\left(\left(\frac{N_k}{N}\right)\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j)\right) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{N_k}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i^{(k)}, \mathbf{y}_j) \\ &= \frac{1}{N^2}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \\ &= V_{ALL}^{cos}(Q, Y), \end{aligned} \tag{24}$$

where we assumed that $N_k = \frac{N}{M}$, since $P(q) = \frac{1}{M}$. Also, the $V_{IN}^{cos}(Q, Y)$ information potential can be expressed using the indicator matrix $\mathbf{\Delta}$:

$$V_{IN}^{cos}(Q, Y) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N_k}[\mathbf{\Delta}]_{ij} S_{cos}(\mathbf{y}_i, \mathbf{y}_j). \tag{25}$$

Therefore, the QSMI can be simplified as:

$$\begin{aligned} I_T^{cos}(Q, Y) &= V_{IN}^{cos}(Q, Y) - V_{BTW}^{cos}(Q, Y) \\ &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left([\mathbf{\Delta}]_{ij} S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - \frac{1}{M} S_{cos}(\mathbf{y}_i, \mathbf{y}_j)\right) \end{aligned} \tag{26}$$

Finally, using the proposed clamping method, the final loss function is obtained as:

$$\begin{aligned} \mathcal{L}_{QSMI} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\Bigg([\mathbf{\Delta}]_{ij}\left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1\right)^2 \\ + \frac{1}{M}\left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j)\right)^2\Bigg) \end{aligned} \tag{27}$$

since we aim to maximize (26). Also, note that (27) can be equivalently expressed as:

$$\mathcal{L}_{QSMI} = \frac{1}{N^2}\mathbf{1}_N^T\left(\mathbf{\Delta} \odot (\mathbf{S} - 1) \odot (\mathbf{S} - 1) + \frac{1}{M}(\mathbf{S} \odot \mathbf{S})\right)\mathbf{1}_N \tag{28}$$

allowing for efficiently implementing the proposed method.

To implement the gradient descent algorithm, the derivative $\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{W}}$, where $\mathbf{W}$ are the parameters of the employed neural network, must be calculated. This derivative is calculated as:

$$\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{W}} = \sum_{i=1}^{N}\left(\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{y}_i}\right)^T \frac{\partial \mathbf{y}_i}{\partial \mathbf{W}}, \tag{29}$$

Table 10: Parameters used for the conducted experiments

Param.	Method	F. MNIST	CIFAR10	NUS-WIDE
Learn. rate	all	0.001	0.001	0.001
Batch size	all	128	128	128
Epochs	all	50	5	50
α	DSH	10^{-5}	10^{-5}	10^{-5}
α	QSMIH	10^{-2}	10^{-2}	10^{-1}
η_{DPSH}	DPSH	5/3* (*36-48 bits)	3	5

Table 11: Network architecture used for the Fashion MNIST dataset

Layer	Kernel	Filters / Neurons	Activation
Convolution	5×5	32	ReLU
Max Pooling	2×2	-	-
Convolution	5×5	64	ReLU
Max Pooling	2×2	-	-
Dense	-	# bits	-

where $\mathbf{y}_i$ is the output of the used neural network for the i-th document. The derivative $\frac{\partial \mathbf{y}_i}{\partial \mathbf{W}}$ depends on the employed architecture, while the derivative of the proposed loss with respect to the hash code $\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{y}_i}$ can be calculated as:

$$\frac{\partial \mathcal{L}_{QSMI}}{\partial \mathbf{y}_i} = \frac{1}{N^2} \sum_{j=1, i \neq j}^{N} \left([\mathbf{\Delta}]_{ij} \frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right)^2}{\partial \mathbf{y}_i} - \frac{1}{M} \frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \right)^2}{\partial \mathbf{y}_i} \right), \tag{30}$$

where

$$\frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right)^2}{\partial \mathbf{y}_i} = 2 \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) - 1 \right) \frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial \mathbf{y}_i}, \tag{31}$$

and

$$\frac{\partial \left(S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \right)^2}{\partial \mathbf{y}_i} = 2 S_{cos}(\mathbf{y}_i, \mathbf{y}_j) \frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial \mathbf{y}_i}. \tag{32}$$

Finally, the derivative of the cosine similarity, needed for calculating (31) and (32), can be computed as:

$$\frac{\partial S_{cos}(\mathbf{y}_i, \mathbf{y}_j)}{\partial [\mathbf{y}_i]_l} = \frac{1}{2} \left(\frac{[\mathbf{y}_j]_l}{\|\mathbf{y}_i\|_2 \|\mathbf{y}_j\|_2} - \frac{\mathbf{y}_i^T \mathbf{y}_j}{\|\mathbf{y}_i\|_2 \|\mathbf{y}_j\|_2} \frac{[\mathbf{y}_i]_l}{\|\mathbf{y}_i\|_2^2} \right). \tag{33}$$

The loss and the corresponding derivatives can be similarly calculated when the information needs are not equiprobable.

Appendix B - Hyper-parameters and Network Architectures

The selected hyper-parameters are shown in Table 10. We selected the best parameters for the two other evaluated methods, i.e., DSH and DPSH, by performing line search for each parameter. The Adam optimizer [66], with the default hyper-parameters, was used for the optimization. The experiments were repeated 5 times and the mean value of each of the evaluated metrics is reported, except otherwise stated.

For the experiments conducted on the Fashion MNIST dataset, a relatively simple Convolutional Neural Network (CNN) architecture was employed, as shown in Table 11. The network was initialized using the default PyTorch initialization scheme [56], and it was trained from scratch for all the conducted experiments.

For the CIFAR10 dataset, a DenseNet-BC-190 (growth rate 40 and compression rate 2) [63], that was pretrained on the CIFAR dataset, was used. For the NUS-WIDE dataset, a DenseNet-201 (growth rate 32 and compression rate 2), that was pretrained on the Imagenet dataset [67], was also employed. The feature representation was extracted from the last average pooling layers of the networks. Then, two fully connected layers were used: one with N_H neurons and rectifier activation functions, and one with as many neurons as the desired code length (no activation function was used for the output layer). The size of hidden layer was set to $N_H = 64$ for the CIFAR10 dataset and to $N_H = 2048$ for the NUS-WIDE dataset. To speedup the training process, we back-propagated the gradients only to the last two layers of the network, which were trained to perform supervised hashing. For the MS COCO and NUS-WIDE evaluation using the CSQ approach (reported in Table 8), we used an Imagenet-pretrained ResNet-50 architecture, while the batch size was set to 32. We also combined the CSQ loss ($\mathcal{L}_{hash}$) with the proposed one, after weighting the proposed one with $a_{csq} = 0.001$ to ensure that the gradients back-propagated from the different losses are on the same magnitude. The optimization ran for 20 epochs for the MS COCO dataset and 10 epochs for the NUS-WIDE dataset.

Acknowledgement

This work was supported by the European Union's Horizon2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] K. E. Dungan and L. C. Potter, "Classifying vehicles in wide-angle radar using pyramid match hashing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 3, pp. 577–591, 2011.

[2] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.

[3] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Computing*, vol. 16, no. 3, pp. 82–88, 2017.

[4] N. Passalis and A. Tefas, "Training lightweight deep convolutional neural networks using bag-of-features pooling," *IEEE Trans. on Neural Networks and Learning Systems*, 2018.

[5] J. Plata-Chaves, A. Bertrand, M. Moonen, S. Theodoridis, and A. M. Zoubir, "Heterogeneous and multitask wireless sensor networks - algorithms, applications, and challenges," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 3, pp. 450–465, 2017.

[6] S.-H. Ou, C.-H. Lee, V. S. Somayazulu, Y.-K. Chen, and S.-Y. Chien, "On-line multi-view video summarization for wireless video sensor network," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 165–179, 2015.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[8] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2015.

[9] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. on Artificial Intelligence*, 2014, pp. 2156–2162.

[10] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.

[11] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 2016, pp. 2415–2421.

[12] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 1556–1564.

[13] J. C. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science & Business Media, 2010.

[14] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *Journal of Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[15] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?" in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2017, pp. 1732–1736.

[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[17] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "NUS-WIDE: A real-world web image database from national university of singapore," in *Proc. of ACM Conf. on Image and Video Retrieval*, 2009.

[18] Q. Li, Z. Sun, R. He, and T. Tan, "Deep supervised discrete hashing," in *Advances in neural information processing systems*, 2017, pp. 2482–2491.

[19] T. Song, J. Cai, T. Zhang, C. Gao, F. Meng, and Q. Wu, "Semi-supervised manifold-embedded hashing with joint feature representation and classifier learning," *Pattern Recognition*, vol. 68, pp. 99–110, 2017.

[20] C. Zhang and W.-S. Zheng, "Semi-supervised multi-view discrete hashing for fast image search," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 2604–2617, 2017.

[21] L. Ma, H. Li, F. Meng, Q. Wu, and L. Xu, "Manifold-ranking embedded order preserving hashing for image semantic retrieval," *Journal of Visual Communication and Image Representation*, vol. 44, pp. 29–39, 2017.

[22] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Learning efficient binary codes from high-level feature representations for multilabel image retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2545–2560, 2017.

[23] Q.-Y. Jiang and W.-J. Li, "Asymmetric deep supervised hashing," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 2018.

[24] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. Twenty-Fifth Int. Joint Conf. on Artificial Intelligence*, 2016, pp. 1711–1717.

[25] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *Proc. ACM on Multimedia Conf.*, 2017, pp. 1522–1530.

[26] X. Wang, Y. Shi, and K. M. Kitani, "Deep supervised hashing with triplet labels," in *Proc. Asian Conf. on Computer Vision*, 2016, pp. 70–84.

[27] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Discriminative deep metric learning for asymmetric discrete hashing," *Neurocomputing*, vol. 380, pp. 115–124, 2020.

[28] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. on Computer Vision*, 2017.

[29] L. Ma, H. Li, F. Meng, Q. Wu, and K. N. Ngan, "Global and local semantics-preserving based deep hashing for cross-modal retrieval," *Neurocomputing*, vol. 312, pp. 49–62, 2018.

[30] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Trans. on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[31] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI Conf. on Artificial Intelligence*, 2016, pp. 1230–1236.

[32] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 1971–1978.

[33] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[34] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, "A survey on learning to hash," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2018.

[35] X. Zhe, S. Chen, and H. Yan, "Deep class-wise hashing: Semantics-preserving hashing via class-wise loss," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1681–1695, 2019.

[36] S. Zhang, J. Li, and B. Zhang, "Semantic cluster unary loss for efficient deep hashing," *IEEE Transactions on Image Processing*, vol. 28, no. 6, pp. 2908–2920, 2019.

[37] L. Ma, H. Li, Q. Wu, C. Shang, and K. Ngan, "Multi-task learning for deep semantic hashing," in *Proc. IEEE Visual Communications and Image Processing*, 2018, pp. 1–4.

[38] J. Revaud, J. Almazán, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *Proc. IEEE International Conference on Computer Vision*, 2019, pp. 5107–5116.

[39] Y. Chen, Z. Lai, Y. Ding, K. Lin, and W. K. Wong, "Deep supervised hashing with anchor graph," in *Proc. of the IEEE International Conference on Computer Vision*, 2019, pp. 9796–9804.

[40] B. Klein and L. Wolf, "End-to-end supervised product quantization for image search and retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5041–5050.

[41] Q.-Y. Jiang, X. Cui, and W.-J. Li, "Deep discrete supervised hashing," *IEEE Transactions on Image Processing*, vol. 27, no. 12, pp. 5996–6009, 2018.

[42] D. Wu, Q. Dai, J. Liu, B. Li, and W. Wang, "Deep incremental hashing network for efficient image retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9069–9077.

[43] L. Ma, X. Li, Y. Shi, J. Wu, and Y. Zhang, "Correlation filtering-based hashing for fine-grained image retrieval," *IEEE Signal Processing Letters*, 2020.

[44] M. Almasri, C. Berrut, and J.-P. Chevallet, "A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information," in *Proc. European Conf. on Information Retrieval*, 2016, pp. 709–715.

[45] J. Hu, W. Deng, and J. Guo, "Improving retrieval performance by global analysis," in *Proc. Int. Conf. on Pattern Recognition*, vol. 2, 2006, pp. 703–706.

[46] F. Cakir, K. He, S. Adel Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *Proc. IEEE Int.*

Conf. on Computer Vision, 2017.

[47] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Hashing with mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2424–2437, 2019.

[48] L. Chen, H. Esfandiari, G. Fu, and V. Mirrokni, "Locality-sensitive hashing for f-divergences: Mutual information loss and beyond," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 044–10 054.

[49] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-based multimedia information retrieval: State of the art and challenges," *ACM Trans. on Multimedia Computing, Communications, and Applications*, vol. 2, no. 1, pp. 1–19, 2006.

[50] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[51] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755.

[53] L. Yuan, T. Wang, X. Zhang, F. E. Tay, Z. Jie, W. Liu, and J. Feng, "Central similarity quantization for efficient image and video retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3083–3092.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[55] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge University Press, 2008, vol. 1, no. 1.

[56] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[57] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[58] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.

[59] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. on Computer Vision*, 2017, pp. 5608–5617.

[60] Y. Cao, B. Liu, M. Long, and J. Wang, "Hashgan: Deep learning to hash with pair conditional wasserstein gan," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 1287–1296.

[61] X. Yuan, L. Ren, J. Lu, and J. Zhou, "Relaxation-free deep hashing via policy gradient," in *Proc. European Conf. on Computer Vision*, 2018, pp. 134–150.

[62] Y. Shen, J. Qin, J. Chen, L. Liu, and F. Zhu, "Embarrassingly simple binary representation learning," in *Proc. Int. Conf. in Computer Vision - Compact and Efficient Feature Representation and Learning in Computer Vision*, 2019.

[63] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[64] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep cauchy hashing for hamming space retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1229–1237.

[65] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li, "Learning discriminative binary codes for large-scale cross-modal retrieval," *IEEE Trans. on Image Processing*, vol. 26, no. 5, pp. 2494–2507, 2017.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2015.

[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int. Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

7.7 Knowledge Distillation by Sparse Representation Matching

The appended paper [212] follows.

Knowledge Distillation By Sparse Representation Matching

Dat Thanh Tran*, Moncef Gabbouj*, Alexandros Iosifidis†
*Department of Computing Sciences, Tampere University, Tampere, Finland
†Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark
Email:{thanh.tran,moncef.gabbouj}@tuni.fi, ai@ece.au.dk

***Abstract*—Knowledge Distillation refers to a class of methods that transfers the knowledge from a teacher network to a student network. In this paper, we propose Sparse Representation Matching (SRM), a method to transfer intermediate knowledge obtained from one Convolutional Neural Network (CNN) to another by utilizing sparse representation learning. SRM first extracts sparse representations of the hidden features of the teacher CNN, which are then used to generate both pixel-level and image-level labels for training intermediate feature maps of the student network. We formulate SRM as a neural processing block, which can be efficiently optimized using stochastic gradient descent and integrated into any CNN in a plug-and-play manner. Our experiments demonstrate that SRM is robust to architectural differences between the teacher and student networks, and outperforms other KD techniques across several datasets.**

I. Introduction

Over the past decade, deep neural networks have become the primary tools to tackle learning problems in several domains, ranging from machine vision [1], [2], natural language processing [3], [4] to biomedical analysis [5], [6] or financial forecasting [7], [8], [9]. Of those important developments, Convolutional Neural Networks have evolved as a de facto choice for high-dimensional signals, either as a feature extraction block or the main workhorse in a learning system. Initially developed in the 1990s for handwritten character recognition using only two convolutional layers [10], state-of-the-art CNN topologies nowadays consist of hundreds of layers, having millions of parameters [11], [12]. In fact, not only in computer vision but also in other domains, state-of-the-art solutions are mainly driven by very large networks [3], [4], which limits their deployment in practice due to the high computational complexity.

The promising results obtained from maximally attainable computational power has encouraged a lot of research on developing smaller and light-weight models while achieving similar performances. This includes efforts on designing more efficient neural network families (both automatic and handcrafted) [13], [14], [15], [16], [17], [18], [19], compressing pretrained networks through weight pruning [20], [21], quantization [22], [23], approximation [24], [25], or designing compressive data acquisition and analysis systems [26], [27], [28], as well as transferring knowledge from one network to another via knowledge distillation [29]. Of these developments, Knowledge Distillation (KD) [29] is a simple and widely used technique that has been shown to be effective in improving the performance of a network, given the access to one or many pretrained networks. KD and its variants work by utilizing the knowledge acquired in one or many models (the teacher(s)) as supervisory signals to train another model (the student) along with the labeled data. Thus, there are two central questions in KD:

- How to represent the knowledge encoded in a teacher network?
- How to efficiently transfer such knowledge to other networks, especially when there are architectural differences between the teacher and the student networks?

In the original formulation [29], soft probabilities produced by the teacher represent its knowledge and the student network is trained to mimic this soft prediction. Besides the final predictions, other works have proposed to utilize intermediate feature maps of the teacher as additional knowledge [30], [31], [32], [33], [34]. Intuitively, intermediate feature maps contain certain clues on how the input is progressively transformed through layers of a CNN, thus can act as a good source of knowledge. However, we argue that the intermediate feature maps by themselves are not a good representation of the knowledge encoded in the teacher to teach the students. To address the question of representing the knowledge of the teacher CNN, instead of directly utilizing the intermediate feature maps of the teacher as supervisory signals, we propose to encode each pixel (of the feature maps) in a sparse domain and use the sparse representation as the source of supervision.

Prior to the era of deep learning, sparse representations attracted a great amount of interest in computer vision community and is a basis of many important works [35]. Sparse representation learning aims at representing the input signal in a domain where the coefficients are sparsest. This is achieved by using an overcomplete dictionary and decomposing the signal as a sparse linear combination of the atoms in the dictionary. While the dictionary can be prespecified, it is often desirable to optimize the dictionary together with the sparse decomposition using example signals. Since hidden feature maps in CNN are often smooth with high correlations between neighboring pixels, they are compressible, e.g., in Fourier domain. Thus, sparse representation serves as a good choice for representing information encoded in the hidden feature

maps.

Sparse representation learning is a well-established topic in which several algorithms have been proposed [35]. However, to the best of our knowledge, existing formulations are computationally intensive to fit a large amount of data. Although learning task-specific sparse representations have been proposed in prior works [36], [37], [38], we have not seen its utilization for knowledge transfer using deep neural networks and stochastic optimization. In this work, we formulate sparse representation learning as a computation block that can be incorporated into any CNN and be efficiently optimized using mini-batch update from stochastic gradient-descent based algorithms. Our formulation allows us to take advantage of not only modern stochastic optimization techniques but also data augmentation to generate target sparsity on-the-fly.

Given the sparse representations obtained from the teacher network, we derive the target pixel-level and image-level sparse representation for the student network. Transferring knowledge from the teacher to the student is then conducted by optimizing the student with its own dictionaries to induce the target sparsity. Thus, our knowledge distillation method is dubbed as Sparse Representation Matching (SRM). Extensive experiments presented in Section IV show that SRM significantly outperforms other recent KD methods, especially in transfer learning tasks by large margins. In addition, empirical results also indicate that SRM exhibits robustness to architectural mismatch between the teacher and the student. The implementation of the proposed method is publicly accessible through the following repository https://github.com/viebboy/SRM

II. Related Work

The idea of transferring knowledge from one model to another has existed for a long time. This idea was first introduced in [39] in which the authors proposed to grow decision trees to mimic the output of a complex predictor. Later, similar ideas were proposed for training neural networks [40], [41], [29], mainly for the purpose of model compression. Variants of the knowledge transfer idea differ in the methods of representing and transferring knowledge [42], [33], [30], [31], [34], [43], [44], as well as the types of data being used [45], [46], [47], [48].

In [41], the final predictions of an ensemble on unlabeled data are used to train a single neural network. In [40], the authors proposed to use the logits produced by the source network as the representation of knowledge, which is transferred to a target network by minimizing the Mean Squared Error (MSE) between the logits. The term Knowledge Distillation was introduced in [29] in which the student network is trained to simultaneously minimize the cross-entropy measured on the labeled data and the Kullback-Leibler (KL) divergence between its predicted probabilities and the soft probabilities produced by the teacher network. Since its introduction, this formulation has been widely adopted.

In addition to the soft probabilities of the teacher, later works have been proposed to utilize intermediate features of the teacher as additional sources of knowledge. For example, in FitNet [30], the authors referred to intermediate feature maps of the teacher as *hints* and the student is first pretrained by regressing its intermediate features to the teacher's hints, then later optimized with the standard KD approach. In other works, activation maps [31] as well as feature distributions [34] computed from intermediate layers have been proposed. In recent works [43], [44], instead of transferring knowledge about each individual sample, the authors proposed to transfer relational knowledge between pairs of samples.

Our SRM method bears some resemblances to previous works in the sense that SRM also uses intermediate feature maps as additional sources of knowledge. However, there are many differences between SRM and existing works. For example, in FitNet [30], the student network learns to regress from its intermediate features to the teacher's; however, the regressed features are not actually used in the student network. Thus, the hints in FitNet only *indirectly* influence the student's features. Since the sparse representation is another representation (equivalent) of the feature maps, SRM *directly* influences the student's features. In addition, by manipulating the sparse representation rather than the hidden features themselves, SRM is less prone to feature value range mismatch between the teacher and the student. This is because by construction, the sparse coefficients generated by SRM only have values in the range $[0, 1]$ as we will see in Section III. Attention-based KD method [31] overcomes this problem by normalizing (using l_2 norm) the attention maps of the teacher and the student. This normalization step, however, might suffer from numerical instability (when l_2 norm is very small) when attention maps are calculated from activation layers such as ReLU.

In [49], the authors employed the idea of sparse coding, however, to represent the network's parameters rather than the intermediate feature maps as in our work. In [50], the authors used the idea of feature quantization and k-means clustering using intermediate features of the teacher to train the student with additional convolutional modules to predict the cluster labels. Our pixel-level label bears some resemblances to this method. However, we explicitly represent the intermediate features by sparse representation (by minimizing reconstruction error) and use the same process to transfer both local (pixel-level) and global (image-level) information.

III. Knowledge Distillation by Sparse Representation Matching

A. Knowledge Representation

Given the n-th input image $\mathcal{X}_n$, let us denote by $\mathcal{T}_n^{(l)} \in \mathbb{R}^{H_l \times W_l \times C_l}$ the output of the l-th layer of the teacher CNN, with H_l and W_l being the spatial dimensions and C_l is the number of channels. In the following, we used the subscript $\mathcal{T}$ and $\mathcal{S}$ to denote a variable that is related to the teacher and student networks, respectively. In addition, we also denote by $\mathbf{t}_{n,i,j}^{(l)} = \mathcal{T}_n^{(l)}(i, j, :) \in \mathbb{R}^{C_l}$, which is the pixel at position (i, j) of $\mathcal{T}_n^{(l)}$. The first objective in SRM

is to represent each pixel $\mathbf{t}^{(l)}_{n,i,j}$ in a sparse domain. To do so, SRM learns an overcomplete dictionary of M_l atoms: $\mathbf{D}^{(l)}_{\mathcal{T}} = [\mathbf{d}^{(l)}_{\mathcal{T},1}, \ldots, \mathbf{d}^{(l)}_{\mathcal{T},M_l}] \in \mathbb{R}^{C_l \times M_l}$ ($M_l > C_l$), which is used to express each pixel $\mathbf{t}^{(l)}_{n,i,j}$ as a linear combination of $\mathbf{d}^{(l)}_{\mathcal{T},m}$ as follows:

$$\mathbf{t}^{(l)}_{n,i,j} = \sum_{m=1}^{M_l} \psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m}) \cdot \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m}) \cdot \mathbf{d}^{(l)}_{\mathcal{T},m} \quad (1)$$

where

- $\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m})$ denotes a function that measures the similarity between $\mathbf{t}^{(l)}_{n,i,j}$ and atom $\mathbf{d}^{(l)}_{\mathcal{T},m}$. We further denote by $\mathbf{k}^{(l)}_{\mathcal{T},n,i,j} = [\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},1}), \ldots, \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},M_l})]$ the vector that contains similarities between $\mathbf{t}^{(l)}_{n,i,j}$ and all atoms in the dictionary $\mathbf{D}^{(l)}_{\mathcal{T}}$.
- $\psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{m})$ denotes the indicator function that returns a value of 1 if $\kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m})$ belongs to the set of top-k values in $\mathbf{k}^{(l)}_{\mathcal{T},n,i,j}$, and a value of 0 otherwise.

The decomposition in Eq. (1) basically means that $\mathbf{t}^{(l)}_{n,i,j}$ is expressed as the linear combination of k most similar atoms in $\mathbf{D}^{(l)}_{\mathcal{T}}$, with the coefficients being the corresponding similarity values. Let $\lambda^{(l)}_{n,i,j,m} = \psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m}) \cdot \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m})$, then the sparse representation of $\mathbf{t}^{(l)}_{n,i,j}$ is then defined as:

$$\tilde{\mathbf{t}}^{(l)}_{n,i,j} = [\lambda^{(l)}_{n,i,j,1}, \ldots, \lambda^{(l)}_{n,i,j,M_l}] \in \mathbb{R}^{M_l} \quad (2)$$

By construction, there are only k non-zero elements in $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, and k defines the degree of sparsity, which is a hyper-parameter of SRM. In order to find $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, we simply minimize the reconstruction error in Eq. (1) as follows:

$$\begin{aligned} \arg\min_{\mathbf{D}^{(l)}_{\mathcal{T}}} \sum_{n,i,j} \Big\| & \mathbf{t}^{(l)}_{n,i,j} \\ & - \sum_{m=1}^{M_l} \psi_k(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m}) \cdot \kappa(\mathbf{t}^{(l)}_{n,i,j}, \mathbf{d}^{(l)}_{\mathcal{T},m}) \cdot \mathbf{d}^{(l)}_{\mathcal{T},m} \Big\|_2^2 \end{aligned} \quad (3)$$

There are many choices for the similarity function κ such as linear kernel, RBF kernel, sigmoid kernel and so on. In our work, we used the sigmoid kernel $\kappa(\mathbf{x}, \mathbf{y}) = sigmoid(\mathbf{x}^T\mathbf{y} + c)$ since the dot-product makes it computationally efficient and the gradients in the backward pass are stable. Although the RBF kernel is popular in many works, we empirically found that RBF kernel is sensitive to the learning rate, which easily leads to numerical issues.

B. Transferring Knowledge

Let us denote by $\mathcal{S}^{(p)}_n \in \mathbb{R}^{H_p \times W_p \times C_p}$ the output of the p-th layer of the student network given input image is $\mathcal{X}_n$. In addition, $\mathbf{s}^{(p)}_{n,i,j} \in \mathbb{R}^{C_p}$ denotes the pixel at position (i, j) of $\mathcal{S}^{(p)}_n$. We consider the task of transferring knowledge from the l-th layer of the teacher to the p-th layer of the student. To do so, we require that the spatial dimensions of both networks match ($H_p = H_l$ and $W_p = W_l$) while the channel dimensions might differ.

Given the sparse representation of the teacher in Eq. (2), a straightforward way is to train the student network to produce hidden features at spatial position (i, j), having the same sparse coefficients as its teacher. However, trying to learn exact sparse representations as produced by the teacher is a too restrictive task since this enforces learning the absolute value of every point in a high-dimensional space. Instead of enforcing an absolute constraint on how each pixel of every sample should be represented, to transfer knowledge, we only enforce a relative constraints between them in the sparse domain. Specifically, we propose to train the student to only approximate sparse structures of the teacher network by solving a classification problem with two types of labels extracted from the sparse representation $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ of the teacher: pixel-level and image-level labels.

Pixel-level labeling: for each spatial position (i, j), we assign a class label, which is the index of the largest element of $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$, i.e., the index of the closest (most similar) atom in $\mathbf{D}^{(l)}_{\mathcal{T}}$. This basically means that we partition all pixels into M_l disjoint sets using dictionary $\mathbf{D}^{(l)}_{\mathcal{T}}$, and the student network is trained to learn the same partitioning using its own dictionary $\mathbf{D}^{(p)}_{\mathcal{S}} = [\mathbf{d}^{(p)}_{\mathcal{S},1}, \ldots, \mathbf{d}^{(p)}_{\mathcal{S},M_l}] \in \mathbb{R}^{C_p \times M_l}$. Let $\mathbf{k}^{(p)}_{\mathcal{S},n,i,j} = [\kappa(\mathbf{s}^{(p)}_{n,i,j}, \mathbf{d}^{(p)}_{\mathcal{S},1}), \ldots, \kappa(\mathbf{s}^{(p)}_{n,i,j}, \mathbf{d}^{(p)}_{\mathcal{S},M_l})]$ denote the vector that contains similarities between pixel $\mathbf{s}^{(p)}_{n,i,j}$ and M_l atoms in $\mathbf{D}^{(p)}_{\mathcal{S}}$. The first knowledge transfer objective in our method using pixel-level label is defined as follows:

$$\arg\min_{\mathbf{\Theta}_{\mathcal{S}}, \mathbf{D}^{(p)}_{\mathcal{S}}} \sum_{n,i,j} \mathcal{L}_{CE}(c_{n,i,j}, \mathbf{k}^{(p)}_{\mathcal{S},n,i,j}) \quad (4)$$

where $\mathbf{\Theta}_{\mathcal{S}}$ denotes parameters of the student network. $\mathcal{L}_{CE}$ denotes the cross-entropy loss function, and $c_{n,i,j} = \arg\max(\tilde{\mathbf{t}}^{(l)}_{n,i,j})$. Here we should note that the idea of transferring the structure instead of the absolute representation is not new. For example, in [43] and [44], the authors proposed to transfer the relative distance between the embeddings of samples. In our case, the pixel-level labels provide supervisory information on how the pixels in the student network should be represented in the sparse domain so that their partition using the nearest atom is the same.

Image-level labeling: given the sparse representation $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ of the teacher, we generate an image-level label by averaging $\tilde{\mathbf{t}}^{(l)}_{n,i,j}$ over the spatial dimensions. While pixel-level labels provide local supervisory information encoding the spatial information, image-level label provides global supervisory information, promoting the shift-invariance property. Image-level label bears some resemblances to the Bag-of-Feature model [51], which aggregates the histograms of image patches to generate an image-level feature. The second knowledge transfer objective in our method using image-level labels is defined as follows:

TABLE I
PERFORMANCE ON CIFAR100

Model	Test Accuracy (%)
DenseNet121 (teacher)	75.09 ± 00.29
AllCNN	67.64 ± 01.87
AllCNN-KD	73.27 ± 00.20
AllCNN-FitNet	72.03 ± 00.27
AllCNN-AT	70.88 ± 0.29
AllCNN-PKT	72.22 ± 0.35
AllCNN-RKD	70.39 ± 0.17
AllCNN-CRD	72.70 ± 0.18
AllCNN-SRM (our)	$\mathbf{74.73} \pm 00.26$

$$\underset{\mathbf{\Theta}_{\mathcal{S}},\mathbf{D}_{\mathcal{S}}^{(p)}}{\arg\min} \sum_{n} \mathcal{L}_{BCE}\left(\frac{\sum_{i,j} \tilde{\mathbf{t}}_{n,i,j}^{(l)}}{H_l \cdot W_l}, \frac{\sum_{i,j} \mathbf{k}_{\mathcal{S},n,i,j}^{(p)}}{H_l \cdot W_l}\right) \quad (5)$$

where $\mathcal{L}_{BCE}$ denotes the binary cross-entropy loss. Here we should note that since most kernel functions output a similarity score in $[0, 1]$, elements of $\tilde{\mathbf{t}}_{n,i,j}^{(l)}$ and $\mathbf{k}_{\mathcal{S},n,i,j}^{(p)}$ are also in this range, making the two inputs to $\mathcal{L}_{BCE}$ in Eq. (5) valid.

To summarize, the procedures of our SRM method is similar to FitNet [30], which consists of the following steps:

- **Step 1**: Given the source layers (with indices l) in the teacher network $\mathcal{T}$, find the sparse representation by solving Eq. (3).
- **Step 2**: Given the target layers (with indices p), optimize the student network $\mathcal{S}$ to predict pixel-level and image-level labels by solving Eq. (4), (5).
- **Step 3**: Given the student network obtained in Step 2, optimize it using the original KD algorithm.

All optimization objectives in our algorithm are solved by stochastic gradient descent.

IV. EXPERIMENTS

The first set of experiments was conducted on CIFAR100 dataset [52] to compare our SRM method with other KD methods: KD [29], FitNet [30], AT [31], PKT [34], RKD [43] and CRD [44]. In the second set of experiments, we tested the algorithms under transfer learning setting. In the final set of experiments, we evaluated SRM on the large-scale ImageNet dataset.

For every experiment configuration, we ran 3 times and reported the mean and standard deviation. Regarding the source and the target layers for transferring intermediate knowledge, we used the outputs of the downsampling layers of the teacher and student networks. Detailed experimental setup and our analysis are provided in the following sections. In addition, our implementation is publicly accessible through the following repository https://github.com/viebboy/SRM

A. Experiments on CIFAR100

Since CIFAR100 has no validation set, we randomly selected 5K samples from the training set for validation purpose, reducing the training set size to 45K. Our setup is different from the conventional practice of validating and reporting the result on the test set of CIAR100. In this set of experiments, we used DenseNet121 [11] as the teacher network (7.1M parameters and 900M FLOPs) and a non-residual architecture (a variant of AllCNN network [53]) as the student network (2.2M parameters and 161M FLOPs).

Regarding the training setup, we used ADAM optimizer and trained all networks for 200 epochs with the initial learning rate of 0.001, which is reduced by 0.1 at epochs 31 and 131. For those methods that have pre-training phase, the number of epochs for pre-training was set to 160. For regularization, we used weight decay with coefficient 0.0001. For data augmentation, we followed the conventional protocol, which randomly performs horizontal flipping, random horizontal and/or vertical shifting by four pixels. For SRM, KD and FitNet, we used our own implementation, for other methods (AT, PKT, RKD, CRD), we used the code provided by the authors of CRD method [44].

TABLE II
COMPARISON (TEST ACCURACY %) BETWEEN FITNET AND SRM IN TERMS OF QUALITY OF INTERMEDIATE KNOWLEDGE TRANSFER ON CIFAR100 USING ALLCNN ARCHITECTURE

	Linear Probing	Whole Network Update
FitNet	$\mathbf{69.95} \pm 00.20$	68.06 ± 00.10
SRM (our)	69.10 ± 00.31	$\mathbf{71.99} \pm 00.08$

For KD, FitNet and SRM, we validated the temperature τ (used to soften teacher's probability) and the weight α (used to balance between classification and distillation loss) from the following set: $\tau \in \{2.0, 4.0, 8.0\}$, and $\alpha \in \{0.25, 0.5, 0.75\}$. For other methods, we used the provided values used in CRD paper [44]. In addition, there are two hyperparameters of SRM: the degree of sparsity ($\lambda = k/M_l$) and overcompleteness of dictionaries ($\mu = M_l/C_l$). Lower values of λ indicate sparser representations while higher values of μ indicate larger dictionaries. For CIFAR100, we performed validation with $\lambda \in \{0.01, 0.02, 0.03\}$ and $\mu = \{1.5, 2.0, 3.0\}$.

Overall comparison (Table I): It is clear that the student network significantly benefits from all knowledge transfer methods. The proposed SRM method clearly outperforms other competing methods, establishing more than 1% margin compared to the second best method (KD). In fact, the student network trained with SRM achieves very close performance with its teacher (74.73% versus 75.09%), despite having a non-residual architecture and $3\times$ less parameters. In addition, recent methods, such as PKT and CRD, perform better than FitNet, however, inferior to the original KD method.

Quality of intermediate knowledge transfer: Since FitNet and SRM have a pretraining phase to transfer intermediate knowledge, we compared the quality of intermediate knowledge transferred by FitNet and SRM by conducting two types of experiments after transferring intermediate knowledge: (1) the student network is optimized for 60 epochs using only the training data (without teacher's soft probabilities), given all

TABLE III
SRM TEST ACCURACY (%) ON CIFAR100 WITH DIFFERENT DICTIONARY SIZES (PARAMETERIZED BY μ) AND DEGREE OF SPARSITY (PARAMETERIZED BY λ)

λ \ μ	1.5	2.0	3.0
0.01	74.00 ± 00.15	74.12 ± 00.35	74.09 ± 00.08
0.02	74.34 ± 00.07	74.73 ± 00.26	74.20 ± 00.27
0.03	73.77 ± 00.05	73.83 ± 00.12	73.71 ± 00.51

TABLE IV
EFFECTS OF PIXEL-LEVEL LABEL AND IMAGE-LEVEL LABEL IN SRM ON CIFAR100

Pixel-level label	Image-level label	Test accuracy %
✓		73.16 ± 00.39
	✓	53.50 ± 09.96
✓	✓	$\mathbf{74.73} \pm 00.26$

parameters are fixed, except the last linear layer for classification. This experiment is called *Linear Probing*; (2) the student network is optimized with all parameters for 200 epochs using only the training data (without teacher's soft probabilities). This experiment is named *Whole Network Update*. The results are shown in Table II.

Firstly, both experiments show that the student networks outperform their baseline, thus, benefit from intermediate knowledge transfer by both methods, even without the final KD phase. In the first experiment when we only updated the output layer, the student pretrained by FitNet achieves slightly better performance than by SRM (69.95% versus 69.10%). However, when we optimized with respect to all parameters, the student pretrained by SRM performs significantly better than the one pretrained by FitNet (71.99% versus 68.06%). While full parameter update led the student pretrained by SRM to better local optima, it undermines the student pretrained by FitNet. This result suggests that the process of intermediate knowledge transfer in SRM can initialize the student network at better positions in the parameter space compared to FitNet.

Effects of sparsity (λ) and dictionary size (μ): in Table III, we show the performance of SRM with different degrees of sparsity (parameterized by $\lambda = k/M_l$, lower λ indicates higher sparsity) and dictionary sizes (parameterized by $\mu = M_l/C_l$, higher μ indicates higher overcompleness). As can be seen from Table III, SRM is not sensitive to λ and μ. In fact, the worst configuration still performs slightly better than KD (73.71% versus 73.27%), and much better than other AT, PKT, RKD and CRD.

Pixel-level label and image-level label: finally, to show the importance of combining both pixel-level and image-level label, we experimented with two other variants of SRM on CIFAR100: using either pixel-level or image-level label. The results are shown in Table IV. The student network performs poorly when only receiving intermediate knowledge via image-level labels, even though it was later optimized with the standard KD phase. Similar to the observations made from Table II, this again suggests that the position in the parameter space, which is obtained after the intermediate knowledge transfer phase, and prior to the standard KD phase, heavily affects the final performance. Once the student network is badly initialized, even receiving soft probabilities from the teacher as additional signals does not help. Although using pixel-level label alone is better than image-level label, the best performance is obtained by combining both objectives.

B. Transfer Learning Experiments

Since transfer learning is key in the success of many applications that utilize deep neural networks, we conducted experiments in 5 transfer learning problems (Flowers [54], CUB [55], Cars [56], Indoor-Scenes [57] and PubFig83 [58]) to assess how well the proposed method works under transfer learning setting compared to others.

In our experiments, we used a pretrained ResNext50 [12] on ILSVRC2012 database as the teacher network, which is finetuned using the training set of each transfer learning task. We then benchmarked how well each knowledge distillation method transfers both pretrained knowledge and domain specific knowledge to a *randomly initialized* student network using domain specific data. Both residual (ResNet18 [59], 11.3M parameters, 1.82G FLOPs) and non-residual (a variant of AllCNN [53], 5.1M parameters, 1.35G FLOPs) architectures were used as the student.

For each transfer learning dataset, we randomly sampled a few samples from the training set to establish the validation set. All images were resized to resolution 224×224 and we used standard ImageNet data augmentation (random crop and horizontal flipping) during stochastic optimization. Based on the analysis of hyperparameters in CIFAR100 experiments, we validated KD, FitNet and SRM using $\alpha \in \{0.5, 0.75\}$ and $\tau \in \{4.0, 8.0\}$. Sparsity degree $\lambda = 0.02$ and dictionary size $\mu = 2.0$ were used for SRM. For other methods, we used the hyperparameter settings provided in CRD paper [44]. All experiments were conducted using ADAM optimizer for 200 epochs with the initial learning rate of 0.001, which is reduced by 0.1 at epochs 41 and 161. The weight decay coefficient was set to 0.0001.

Full transfer setting: in this setting, we used all samples available in the training set to perform knowledge transfer from the teacher to the student. The test accuracy achieved by different methods is shown in the upper part of Table V. It is clear that the proposed SRM outperforms other methods on many datasets, except on Cars and Indoor-Scenes datasets for AllCNN student. While KD, AT, PKT, CRD and SRM successfully led the students to better minima with both residual or non-residual students, FitNet was only effective with the residual one. The results suggest that the proposed intermediate knowledge transfer mechanism in SRM is robust to architectural differences between the teacher and the student networks.

TABLE V
FULL TRANSFER LEARNING USING ALLCNN (ACNN) AND RESNET18 (RN18) (TEST ACCURACY %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
Full Shot					
ACNN	40.80 ± 02.33	47.26 ± 00.18	61.93 ± 01.38	35.82 ± 00.43	78.47 ± 00.17
ACNN-KD	46.14 ± 00.39	51.80 ± 00.41	66.12 ± 00.17	38.44 ± 00.99	81.54 ± 00.09
ACNN-FitNet	30.10 ± 02.41	44.30 ± 01.25	60.20 ± 03.83	30.87 ± 00.35	77.61 ± 00.87
ACNN-AT	51.62 ± 00.69	51.74 ± 00.39	$\mathbf{73.89} \pm 00.06$	$\mathbf{43.56} \pm 00.52$	81.11 ± 01.51
ACNN-PKT	47.12 ± 00.52	47.60 ± 00.85	70.16 ± 00.51	37.71 ± 00.64	82.03 ± 00.26
ACNN-RKD	42.00 ± 01.10	39.99 ± 00.61	56.99 ± 02.44	30.94 ± 00.45	75.44 ± 00.50
ACNN-CRD	46.99 ± 01.13	51.12 ± 00.34	68.89 ± 00.47	42.82 ± 00.21	$\mathbf{83.02} \pm 00.11$
ACNN-SRM	$\mathbf{51.72} \pm 00.58$	$\mathbf{54.51} \pm 01.72$	71.44 ± 04.76	43.09 ± 00.60	82.89 ± 01.78
RN18	44.25 ± 00.42	44.79 ± 00.68	57.17 ± 01.95	36.72 ± 00.29	79.08 ± 00.36
RN18-KD	48.26 ± 00.33	54.91 ± 00.33	75.29 ± 00.46	43.84 ± 00.67	84.49 ± 00.28
RN18-FitNet	48.29 ± 01.24	61.28 ± 00.48	$\mathbf{85.01} \pm 00.10$	45.93 ± 01.00	89.78 ± 00.20
RN18-AT	51.49 ± 00.42	53.13 ± 00.40	77.14 ± 00.15	44.13 ± 00.55	83.60 ± 00.19
RN18-PKT	45.32 ± 00.51	45.24 ± 00.39	71.24 ± 02.23	37.27 ± 00.79	82.00 ± 00.10
RN18-RKD	42.32 ± 00.28	36.29 ± 00.58	56.87 ± 02.64	29.57 ± 00.90	70.90 ± 01.79
RN18-CRD	47.67 ± 00.05	53.25 ± 00.83	76.51 ± 00.74	43.76 ± 00.40	84.43 ± 00.40
RN18-SRM	$\mathbf{67.46} \pm 01.06$	$\mathbf{63.11} \pm 00.45$	84.40 ± 00.67	$\mathbf{54.59} \pm 00.38$	$\mathbf{89.79} \pm 00.53$

TABLE VI
5-SHOT TRANSFER LEARNING USING ALLCNN (ACNN) AND RESNET18 (RN18) (TEST ACCURACY %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
5-Shot					
ACNN	32.91 ± 00.94	13.19 ± 00.46	05.03 ± 00.11	09.21 ± 00.81	05.15 ± 00.45
ACNN-KD	35.98 ± 00.76	21.60 ± 00.83	10.61 ± 00.52	14.81 ± 00.67	11.97 ± 01.40
ACNN-FitNet	28.73 ± 01.18	14.78 ± 00.60	06.11 ± 00.37	08.36 ± 00.52	08.25 ± 01.36
ACNN-AT	38.21 ± 00.88	17.69 ± 00.94	08.52 ± 01.50	08.76 ± 00.39	08.02 ± 00.70
ACNN-PKT	33.25 ± 00.31	11.30 ± 00.20	06.16 ± 00.29	10.75 ± 01.04	06.13 ± 00.18
ACNN-RKD	30.27 ± 00.27	09.55 ± 00.34	04.82 ± 00.31	09.68 ± 00.39	04.82 ± 00.16
ACNN-CRD	35.01 ± 00.57	18.09 ± 00.91	06.77 ± 00.55	09.73 ± 00.49	06.33 ± 00.28
ACNN-SRM	$\mathbf{41.14 \pm 01.09}$	$\mathbf{22.89 \pm 01.18}$	$\mathbf{11.71 \pm 01.21}$	$\mathbf{16.63 \pm 00.34}$	$\mathbf{13.96 \pm 00.52}$
RN18	32.95 ± 00.37	11.55 ± 00.10	05.00 ± 00.27	09.04 ± 00.44	04.98 ± 00.22
RN18-KD	38.07 ± 01.47	25.53 ± 00.35	11.37 ± 00.58	14.61 ± 00.88	10.69 ± 00.99
RN18-FitNet	39.17 ± 00.62	26.50 ± 00.46	12.36 ± 01.00	13.72 ± 01.07	11.49 ± 00.66
RN18-AT	37.36 ± 01.23	18.22 ± 00.47	08.96 ± 00.92	09.93 ± 00.73	08.76 ± 00.34
RN18-PKT	33.24 ± 00.47	10.63 ± 00.18	05.88 ± 00.18	11.03 ± 00.82	05.34 ± 00.11
RN18-RKD	29.97 ± 00.55	08.83 ± 00.40	04.98 ± 00.13	08.41 ± 00.49	04.49 ± 00.24
RN18-CRD	34.24 ± 00.32	18.36 ± 02.09	06.95 ± 00.26	09.01 ± 00.23	06.36 ± 00.21
RN18-SRM	$\mathbf{51.24 \pm 00.48}$	$\mathbf{34.67 \pm 00.63}$	$\mathbf{26.63 \pm 01.82}$	$\mathbf{21.18 \pm 01.29}$	$\mathbf{29.31 \pm 00.51}$

Few-shot transfer setting: we further assessed how well the methods perform when there is a limited amount of data for knowledge transfer. For each dataset, we randomly selected 5 samples (5-shot) and 10 samples (10-shot) from the training set for training purpose, and kept the validation and test set similar to the full transfer setting. Since the Flowers dataset has only 10 training samples in total (the original split provided by the database has 20 samples per class, however, we used 10 samples for validation purpose), the results for 10-shot are similar to full transfer learning setting. The test performance (% in accuracy) under 5-shot and 10-shot transfer learning setting are reported in Table VI and Table VII, respectively. Under this restrictive regime, the proposed SRM method performs far better than other tested methods for both types of students, largely improving the baseline results.

TABLE VII
10-SHOT TRANSFER LEARNING USING ALLCNN (ACNN) AND RESNET18 (RN18) (TEST ACCURACY %)

Model	Flowers	CUB	Cars	Indoor-Scenes	PubFig83
ResNext50	89.35 ± 00.62	69.53 ± 00.45	87.45 ± 00.27	63.51 ± 00.43	91.41 ± 00.14
10-Shot					
ACNN	40.80 ± 02.33	25.53 ± 01.07	09.50 ± 00.82	16.23 ± 00.46	10.09 ± 00.19
ACNN-KD	46.14 ± 00.39	34.63 ± 00.35	23.43 ± 01.47	21.76 ± 00.53	28.11 ± 00.50
ACNN-FitNet	30.10 ± 02.41	29.40 ± 00.63	15.81 ± 02.16	15.71 ± 01.32	17.89 ± 00.67
ACNN-AT	51.62 ± 00.69	30.26 ± 00.35	25.22 ± 02.90	17.90 ± 00.40	26.27 ± 01.76
ACNN-PKT	47.12 ± 00.53	24.93 ± 01.92	13.82 ± 01.18	16.78 ± 00.59	10.67 ± 00.26
ACNN-RKD	42.00 ± 01.10	19.36 ± 00.50	09.60 ± 00.25	12.40 ± 00.76	06.99 ± 00.57
ACNN-CRD	46.99 ± 01.13	29.72 ± 00.25	16.96 ± 00.77	17.60 ± 00.85	17.24 ± 02.98
ACNN-SRM	$\mathbf{51.72 \pm 00.58}$	$\mathbf{35.86 \pm 01.39}$	$\mathbf{36.28 \pm 04.33}$	$\mathbf{24.82 \pm 00.47}$	$\mathbf{31.47 \pm 01.84}$
RN18	44.25 ± 00.42	22.72 ± 00.91	11.76 ± 01.35	15.16 ± 00.54	08.92 ± 00.73
RN18-KD	48.26 ± 00.33	40.57 ± 00.53	35.44 ± 01.13	23.85 ± 00.78	29.67 ± 00.91
RN18-FitNet	48.29 ± 01.24	43.83 ± 00.47	51.88 ± 01.72	24.62 ± 00.47	36.79 ± 01.27
RN18-AT	51.49 ± 00.42	30.47 ± 00.55	27.70 ± 02.72	17.48 ± 00.38	29.80 ± 01.41
RN18-PKT	45.32 ± 00.52	20.62 ± 02.56	11.00 ± 00.63	15.76 ± 00.62	08.80 ± 00.54
RN18-RKD	42.32 ± 00.28	17.73 ± 00.14	08.73 ± 00.71	11.82 ± 00.34	06.49 ± 00.27
RN18-CRD	47.67 ± 00.05	30.04 ± 00.13	17.11 ± 00.66	17.82 ± 00.49	16.08 ± 01.67
RN18-SRM	$\mathbf{67.46 \pm 01.06}$	$\mathbf{48.42 \pm 01.86}$	$\mathbf{61.22 \pm 06.84}$	$\mathbf{32.21 \pm 01.21}$	$\mathbf{51.99 \pm 02.95}$

TABLE VIII
TOP-1 ERROR OF RESNET18 ON IMAGENET. (*) INDICATES RESULTS OBTAINED BY 110 EPOCHS

KD	AT	Seq. KD	KD+ONE	ESKD	ESKD+AT	CRD*	SRM (our)
30.79	29.30	29.60	29.45	29.16	28.84	28.62*	28.79

C. ImageNet Experiments

For ImageNet [60] experiments, we followed similar experimental setup as in [61], [31]: ResNet34 and ResNet18 were used as the teacher and student networks, respectively. For hyperparameters of SRM, we set $\mu = 4.0, \lambda = 0.02, \tau = 4.0, \alpha = 0.3$. For other training protocols, we followed the standard setup for ImageNet, which trains the student network for 100 epochs using SGD optimizer with an initial learning rate of 0.1, dropping by 0.1 at epochs 51, 81, 91. Weight decay coefficient was set to 0.0001. In addition, the pre-training phase took 80 epochs with an initial learning rate of 0.1, dropping by 0.1 for every 20 epochs.

Table VIII shows top-1 classification errors of SRM and related methods, including KD [29], AT [31], Seq. KD [62], KD+ONE [63], ESKD [61], ESKD+AT [61], CRD [44], having the same experimental setup. The teacher network (Resnet34) achieves top-1 error of 26.70%. Using SRM, we can successfully train ResNet18 to achieve 28.79% classification error, which is lower than existing methods (except CRD, which was trained for 110 epochs), including the recently proposed ESKD+AT [61] that combines early-stopping trick and Attention Transfer [31].

V. CONCLUSION

In this work, we proposed Sparse Representation Matching (SRM), a method to transfer intermediate knowledge from one network to another using sparse representation learning. Experimental results on several datasets indicated that SRM outperforms related methods, successfully performing intermediate knowledge transfer even if there is a significant architectural mismatch between networks and/or a limited amount of data. SRM serves as a starting point for developing specific knowledge transfer use cases, e.g., data-free knowledge transfer, which is an interesting future research direction.

VI. ACKNOWLEDGEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[5] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2015.

[6] D. T. Tran, N. Passalis, A. Tefas, M. Gabbouj, and A. Iosifidis, "Attention-based neural bag-of-features learning for sequence data," *arXiv preprint arXiv:2005.12250*, 2020.

[7] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, pp. 1407–1418, 2018.

[8] Z. Zhang, S. Zohren, and S. Roberts, "Deeplob: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019.

[9] D. T. Tran, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Data normalization for bilinear structures in high-frequency financial time-series," *arXiv preprint arXiv:2003.00598*, 2020.

[10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[12] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[14] D. T. Tran, A. Iosifidis, and M. Gabbouj, "Improving efficiency in convolutional neural networks with multilinear filters," *Neural Networks*, vol. 105, pp. 328–339, 2018.

[15] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis, "Heterogeneous multilayer generalized operational perceptron," *IEEE transactions on neural networks and learning systems*, 2019.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[17] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis, "Pygop: A python library for generalized operational perceptron algorithms," *Knowledge-Based Systems*, vol. 182, p. 104801, 2019.

[18] D. T. Tran, S. Kiranyaz, M. Gabbouj, and A. Iosifidis, "Knowledge transfer for face verification using heterogeneous generalized operational perceptrons," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 1168–1172, IEEE, 2019.

[19] D. T. Tran and A. Iosifidis, "Learning to rank: A progressive neural network learning approach," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8355–8359, IEEE, 2019.

[20] F. Manessi, A. Rozza, S. Bianco, P. Napoletano, and R. Schettini, "Automated pruning for deep neural network compression," in *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 657–664, IEEE, 2018.

[21] F. Tung and G. Mori, "Clip-q: Deep network compression learning by in-parallel pruning-quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7873–7882, 2018.

[22] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[23] S.-C. Zhou, Y.-Z. Wang, H. Wen, Q.-Y. He, and Y.-H. Zou, "Balanced quantization: An effective and efficient approach to quantized neural networks," *Journal of Computer Science and Technology*, vol. 32, no. 4, pp. 667–682, 2017.

[24] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, pp. 1269–1277, 2014.

[25] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[26] D. T. Tran, M. Yamaç, A. Degerli, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning," *IEEE transactions on neural networks and learning systems*, 2020.

[27] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Multilinear compressive learning with prior knowledge," *arXiv preprint arXiv:2002.07203*, 2020.

[28] D. T. Tran, M. Gabbouj, and A. Iosifidis, "Performance indicator in multilinear compressive learning," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1822–1828, IEEE, 2020.

[29] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[30] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[31] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928*, 2016.

[32] M. Gao, Y. Shen, Q. Li, J. Yan, L. Wan, D. Lin, C. C. Loy, and X. Tang, "An embarrassingly simple approach for knowledge distillation," *arXiv preprint arXiv:1812.01819*, 2018.

[33] B. Heo, M. Lee, S. Yun, and J. Y. Choi, "Knowledge transfer via distillation of activation boundaries formed by hidden neurons," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 3779–3787, 2019.

[34] N. Passalis and A. Tefas, "Learning deep representations with probabilistic knowledge transfer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 268–284, 2018.

[35] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: algorithms and applications," *IEEE access*, vol. 3, pp. 490–530, 2015.

[36] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 791–804, 2011.

[37] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1821–1833, 2015.

[38] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *arXiv preprint arXiv:1912.10557*, 2019.

[39] L. Breiman and N. Shang, "Born again trees," *University of California, Berkeley, Berkeley, CA, Technical Report*, vol. 1, p. 2, 1996.

[40] J. Ba and R. Caruana, "Do deep nets really need to be deep?," in *Advances in neural information processing systems*, pp. 2654–2662, 2014.

[41] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

[42] C. Cao, F. Liu, H. Tan, D. Song, W. Shu, W. Li, Y. Zhou, X. Bo, and Z. Xie, "Deep learning and its applications in biomedicine," *Genomics, proteomics & bioinformatics*, vol. 16, no. 1, pp. 17–32, 2018.

[43] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3967–3976, 2019.

[44] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," in *International Conference on Learning Representations*, 2020.

[45] R. G. Lopes, S. Fenu, and T. Starner, "Data-free knowledge distillation for deep neural networks," *arXiv preprint arXiv:1710.07535*, 2017.

[46] J. Yoo, M. Cho, T. Kim, and U. Kang, "Knowledge extraction with no observable data," in *Advances in Neural Information Processing Systems*, pp. 2701–2710, 2019.

[47] N. Papernot, M. Abadi, U. Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised knowledge transfer for deep learning from private training data," *arXiv preprint arXiv:1610.05755*, 2016.

[48] A. Kimura, Z. Ghahramani, K. Takeuchi, T. Iwata, and N. Ueda, "Few-shot learning of neural networks from scratch by pseudo example optimization," *arXiv preprint arXiv:1802.03039*, 2018.

[49] J. Liu, D. Wen, H. Gao, W. Tao, T.-W. Chen, K. Osa, and M. Kato, "Knowledge representing: Efficient, sparse representation of prior knowledge for knowledge distillation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[50] H. Jain, S. Gidaris, N. Komodakis, P. Pérez, and M. Cord, "Quest: Quantized embedding space for transferring knowledge," *arXiv preprint arXiv:1912.01540*, 2019.

[51] N. Passalis and A. Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277–294, 2017.

[52] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[53] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.
[54] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729, IEEE, 2008.
[55] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.
[56] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.
[57] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420, IEEE, 2009.
[58] N. Pinto, Z. Stone, T. Zickler, and D. Cox, "Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook," in *CVPR 2011 WORKSHOPS*, pp. 35–42, IEEE, 2011.
[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[60] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.
[61] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4794–4802, 2019.
[62] C. Yang, L. Xie, S. Qiao, and A. Yuille, "Knowledge distillation in generations: More tolerant teachers educate better students," *arXiv preprint arXiv:1805.05551*, 2018.
[63] X. Zhu, S. Gong, *et al.*, "Knowledge distillation by on-the-fly native ensemble," in *Advances in neural information processing systems*, pp. 7517–7527, 2018.

7.8 Rethinking Compactness in Deep Neural Networks

The appended paper [40] follows.

Rethinking compactness in deep neural networks

Kateryna Chumachenko [*1] **Firas Laakom** [*1] **Jenni Raitoharju** [2] **Alexandros Iosifidis** [3] **Moncef Gabbouj** [1]

Abstract

Deep neural networks are a type of over-parameterized models which are able to achieve high performance despite having typically more parameters than training samples. Recently, there has been an increasing interest in uncovering and understanding the different phenomena that occur in the over-parameterized regime induced by such neural networks. In this paper, we aim to shed light on the relationship between class compactness of the learned feature representations and the model performance. Surprisingly, we find that models that learn more class-invariant features do not necessarily perform better. Moreover, we show that during training, class-wise variance increases and the models learn a less compact and more outspread representation of the classes.

1. Introduction

Deep learning is a type of non-linear models that have led to state-of-the-art results in a wide range of tasks, such as image classification (Krizhevsky et al., 2012), object detection (Zhao et al., 2019), video analysis (Oprea et al., 2020), and speech recognition (Hinton et al., 2012). However, deep neural networks are often over-parameterized, i.e., they have many more parameters than training samples. While in the past over-parameterization was known to cause overfitting and hurt generalization, recent research has shown that this is not necessarily the case (Belkin et al., 2019; Advani et al., 2020; Nakkiran et al., 2020; Allen-Zhu et al., 2019). In fact, over-parameterization is critical to find the global minimum using stochastic gradient descent (SGD) (Allen-Zhu et al., 2019). Moreover, over-parameterized models have been found to exhibit benign overfitting (Bartlett et al., 2020; Sanyal et al., 2020), as well as double descent behavior (d'Ascoli et al., 2020; Nakkiran et al., 2020). Thus, understanding the effect of over-parameterization is essential to better understand the generalization of deep learning (Zhang et al., 2016).

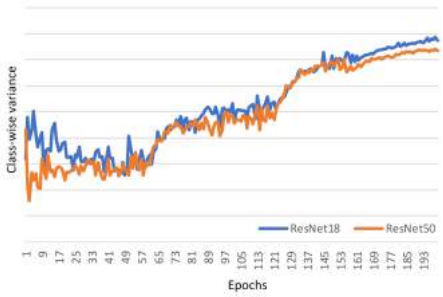


Figure 1. Class-wise variance evolution during training of neural networks.

Training of neural networks can be seen as a two-stage process, with the first stage being feature learning, followed by a task-specific output prediction. In classification tasks, the prediction is generally performed by applying a linear dense layer on top of the learned feature representation and followed by a softmax activation to transform the predictions into a class probability distribution. Hence, a prediction in such a model can be seen as a linear classification in the learned feature space. Classical statistical intuition states that a good set of features should be class-wise invariant (Abe, 2003; Ye, 2007). In other words, it is expected that the model learns to minimize the distances between the sample representations of each class in the feature space in order to learn compact class manifolds. However, in this paper, we show that surprisingly this is not necessarily the case for deep neural networks. As illustrated in Figure 1, the class-wise variance in the feature space is in fact increasing during the training rather than decreasing.

In this paper, we investigate this phenomenon for deep neural networks. We first analyze the relationship between the features' class-wise variance and the performance of the model. We show that in the context of deep neural networks, a more class-wise invariant feature space, i.e., lower relative variance, does not necessarily imply better perfor-

[*]Equal contribution [1]Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland [2]Programme for Environmental Information, Finnish Environment Institute, Jyväskylä, Finland [3]Department of Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark. Correspondence to: Kateryna Chumachenko <kateryna.chumachenko@tuni.fi>, Firas Laakom <firas.laakom@tuni.fi>.

Presented at the ICML 2021 Workshop on Overparameterization: Pitfalls & Opportunities.

mance. Then, we analyze the class-wise variance of the learned features throughout the training and discover that over-parameterized neural networks, surprisingly, do not implicitly minimize it. On the contrary, models typically converge to regimes with higher variances. Our contributions can be summarized as follows:

- We investigate the relationship between class-wise variance and model performance on a variety of architectures and show that in contrary to the classical intuition, in the deep neural network context, more invariant representations of the classes do not necessarily imply better performance.
- We analyze the relative variance of the learned features throughout the training and discover that models do not minimize compactness of class representations during training and that some over-parameterized models converge to regimes with higher variances.

2. Class-wise variance in neural networks

In this section, we analyze the class compactness of the features learned by neural networks. We quantify the class compactness using the class-wise variance in the feature space. Formally, the class-wise variance can be defined as follows:

Definition 1. *(class-wise variance) Let A be a set of data samples defined as* $A = \{(X_1, y_1), \cdots, (X_N, y_N)\} \in \mathcal{X} \times \mathbb{R}$*, where* $y_i \in \{1, \cdots, c\}$ *is the corresponding class label of* X_i*. Let* $\{\phi(X_1), \cdots, \phi(X_N)\}$ *be the feature representation of the samples in A. Then, the class-wise variance of the feature space is defined as follows:*

$$\mathbf{v}_c = \frac{1}{N} \sum_i ||\phi(X_i) - \mu_i||_2^2 \tag{1}$$

where μ_i *is the mean of the feature representations of the data belonging to class* y_i*, i.e.,* $\mu_i = \frac{1}{\sum_j \delta(y_j == y_i)} \sum_j \phi(X_j)\delta(y_j == y_i)$.

As can be seen, the class-wise variance measures the average distance of the feature representations of the samples to the corresponding class mean. Intuitively, this quantifies the compactness of the features learned by a model. Lower class-wise variance implies that the model has learned an invariant representation of each class, where samples with the same labels are mapped close to each other, whereas higher class-wise variance implies that the learned manifolds of the classes are non-compact and outspread in the feature space. Additionally, we define a normalized variant of the class-wise variance, where it is scaled using the global variance in order to quantify the relative volume of the classes compared to the total volume in the feature space. Formally, the normalized class-wise variance can be defined as follows:

Definition 2. *(normalized class-wise variance) Let A be a set of data samples defined as* $A = \{(X_1, y_1), \cdots, (X_N, y_N)\} \in \mathcal{X} \times \mathbb{R}$*, where* $y_i \in \{1, \cdots, c\}$ *is the corresponding class label of* X_i*. Let* $\{\phi(X_1), \cdots, \phi(X_N)\}$ *be the feature representation of the samples in A. Then, the normalized class-wise variance of the feature space is defined as follows*

$$\bar{\mathbf{v}}_c = \frac{\frac{1}{N} \sum_i ||\phi(X_i) - \mu_i||_2^2}{\frac{1}{N} \sum_i ||\phi(X_i) - \mu||_2^2}, \tag{2}$$

where $\mu = \frac{1}{N} \sum_j \phi(X_j)$ *is the global mean of the data in the feature space.*

The intuition that compactness is a desired property has been utilized in various classical machine learning methods in the past using the Fisher ratio (Ye, 2007; Pearson, 1901; Iosifidis et al., 2013), which is closely related to the normalized class-wise variance. In this work, we investigate the link between the (normalized) class-wise variance and the performance of over-parameterized neural network models. Surprisingly, we find that the compactness intuition does not hold in this case and that lower class-wise variance values do not necessarily result in better accuracy. Our finding can be expressed formally as follows:

Hypothesis 1. *Lower class-wise variance does not imply higher classification accuracy.*

To verify the hypothesis, we start by an experimental evaluation of several deep neural network models using MNIST dataset (Deng, 2012). Specifically, we evaluate different model architectures: Dense-1, Dense-2, Dense-3, and Dense-4 have respectively 1, 2, 3, and 4 intermediate fully-connected layers with 100 units and ReLU activation. CNN-1, CNN-2, CNN-3 have respectively 1, 2, and 3 consecutive convolutional blocks (32-channel 3×3-convolutional layer followed by a 2×2 max-pooling). The final output is flattened and connected to the output layer. We also evaluate a large CNN, which has a VGG-like structure (Simonyan & Zisserman, 2014): two 32-channel 3×3-convolutional layers followed by a 2×2 max-pooling, then two 64-channel 3×3-convolutional layers followed by a 2×2 max-pooling, and finally two 128-channel 3×3-convolutional layers followed by a 2×2 max-pooling. The output is flattened in order to obtain the feature representation, which is connected to the output layer. All the models are trained for 100 epochs using SGD with a learning rate of 0.001 and a batch size of 128. The model with the highest validation accuracy is used in the test phase to obtain the final scores.

Additionally, we perform similar experiments on CIFAR10 dataset (Krizhevsky et al., 2009). Similarly to MNIST models, we report results on CNN architectures of 1, 2, and 3

Table 1. Average test accuracy, class-wise variance, and normalized class-wise variance of validation set on different models trained on MNIST.

	Accuracy	$\mathbf{v}_c$	$\bar{\mathbf{v}}_c$
Large CNN	98.59	47659	0.58
Dense-1	94.88	15.01	0.44
Dense-2	96.14	13.43	0.32
Dense-3	96.63	15.94	0.26
Dense-4	96.73	20.43	0.23
CNN-1	96.30	336.91	0.65
CNN-2	98.16	1117.68	0.59
CNN-3	98.13	347.39	0.50

Table 2. Average test accuracy, class-wise variance, and normalized class-wise variance of validation set on different models trained on CIFAR10.

	Accuracy	$\mathbf{v}_c$	$\bar{\mathbf{v}}_c$
ResNet18	91.29	7.76	0.46
ResNet50	92.12	7.37	0.40
Dense-1	52.80	86.88	0.84
Dense-2	53.04	48.37	0.76
Dense-3	52.90	36.45	0.70
Dense-4	51.74	29.66	0.68
CNN-1	66.36	292.41	0.89
CNN-2	74.53	222.07	0.89
CNN-3	75.68	290.47	0.76

hidden layers with the same structure. We report the results on Dense models with 1-4 hidden layers with 32 units each trained on flattened images. We additionally report the results on two deep CNN models, namely ResNet18 and ResNet50 architectures (He et al., 2016). Training is done for 200 epochs using SGD with a batch size of 128, a learning rate of 0.1, and a momentum of 0.9, which is reduced by a factor of 5 after 60, 120, and 160 epochs. Results are reported on the test set using the model selected on the validation set, with 10k images used for validation, 10k for testing, and 40k for training. Images are normalized and standard data augmentation is used. On both datasets, for each model, we report the test accuracy along with the class-wise variance and the normalized class-wise variance of the feature space prior to the output layer over the validation set. We report the average scores across three training iterations.

As can be seen in Table 1, better model performance is not necessarily associated neither with lower class-wise variance nor with normalized class-wise variance. For example, Dense-4 outperforms other Dense models, while having the highest class variance. For the normalized class-wise variance, despite common intuition, CNN-2 outperforms CNN-3, while having lower compactness (higher $\mathbf{v}_c$ and $\bar{\mathbf{v}}_c$). All the Dense models have lower normalized class-wise variances, hence, higher class compactness compared to any CNN model, while achieving lower accuracy. These findings comply with our Hypothesis 1.

Similar results can be observed in models trained on CIFAR10, as shown in Table 2, where Dense-4 and Dense-3 models have lower class-wise variance compared to Dense-2 model despite having worse performance. Similarly, CNN-2 outperforms CNN-1, although its class-wise variance is lower. At the same time, their normalized class-wise variances are equal despite the large gap in the test accuracy.

To better understand this phenomenon in the over-parameterized regime, we further investigate the evolution of the class-wise variance $\mathbf{v}_c$ and its normalized counterpart $\bar{\mathbf{v}}_c$ throughout the training in several different neural network architectures using CIFAR10 and MNIST. Figure 2 shows the evolution of the different models over the training phase on CIFAR10. Surprisingly, as we can see, all the models do not minimize the class-wise variance. On the contrary, $\mathbf{v}_c$ is increasing during the training. This leads us to our second hypothesis expressed as follows:

Hypothesis 2. *Deep neural network training implicitly maximizes the class-wise variance.*

Hypothesis 2 states that deep neural networks, on the contrary of the classical intuition, do not learn compact regimes. During the optimization phase, the models converge towards regions where the samples from the same class are outspread in the feature space. For $\bar{\mathbf{v}}_c$, we note that it is constant or slightly decreasing for the first two models and increasing for the two other models, i.e., dense models. For the two ResNet models, the average relative volume of the classes compared to the total volume in the feature space is relatively constant along the training as opposite to the other two cases where this relative volume is increasing and the models are exploring more regions in the feature space. The results on MNIST, illustrated in Figure 3 are consistent with our findings on CIFAR10. In fact, for the different models on MNIST, we note that the class-wise variance increases all along the training and that the relative volume $\bar{\mathbf{v}}_c$ undergoes two stages, in the first epochs of the training it decreases and in the second stage it consistently increases.

Additionally, it should be noted that by observing Figure 2, we can see that, interestingly, on CIFAR-10 dataset the regions of accuracy increase correspond to the regions of class-wise variance growth (increase of $\mathbf{v}_c$). In fact, the peaks in $\mathbf{v}_c$ coincide with the peaks in accuracy during training. This implies a positive correlation between the increase of variance and the improvement of performance.

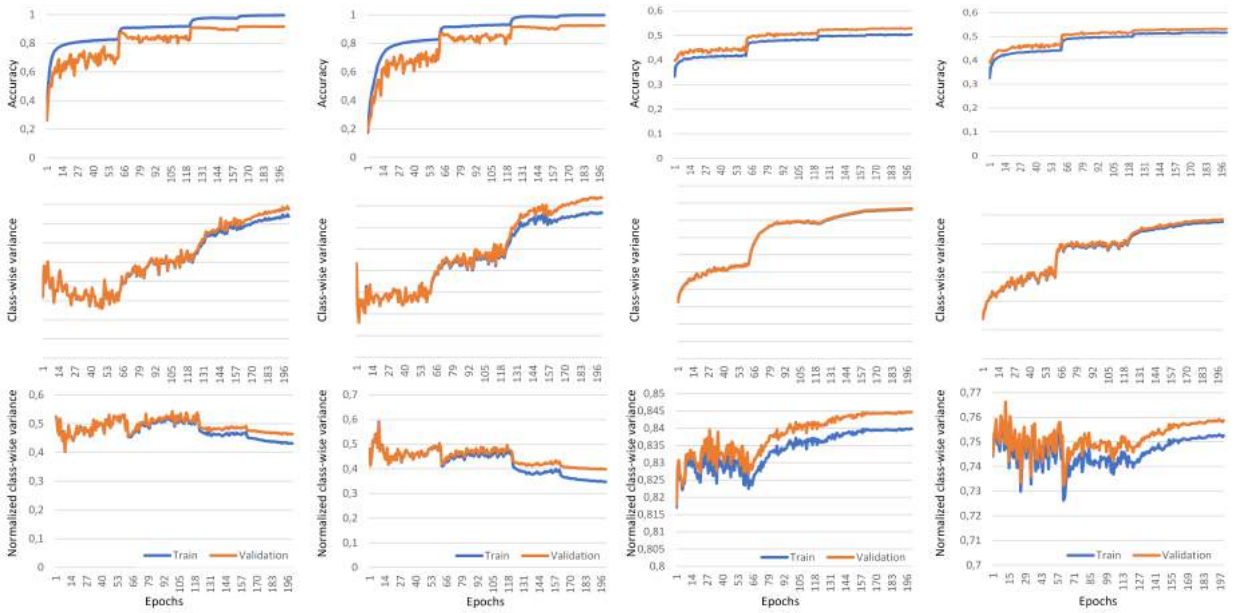


Figure 2. The average accuracy (top), class-wise variance (middle), and normalized class-wise variance (bottom) of the different models over the training epochs. Models trained on CIFAR10 from left to right: (a) ResNet18, (b) ResNet50, (c) Dense-1, (d) Dense-2

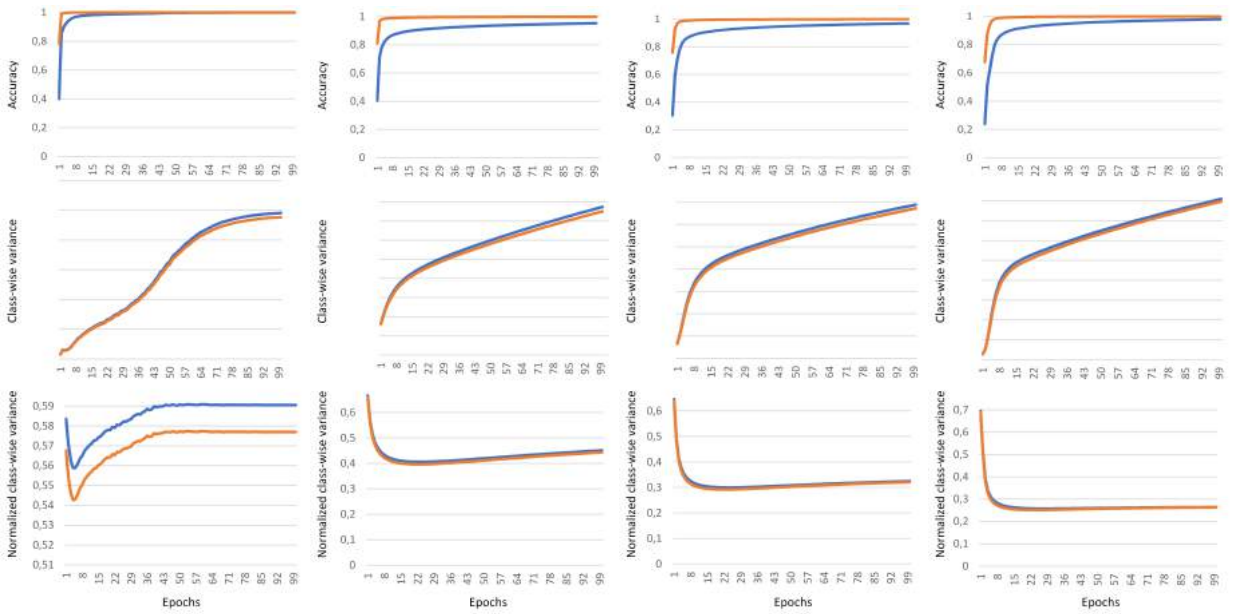


Figure 3. The average accuracy (top), class-wise variance (middle), and normalized class-wise variance (bottom) of the different models over the training epochs. Models trained on MNIST from left to right: (a) Large CNN, (b) Dense-1, (c) Dense-2, (d) Dense-3

3. Conclusion

In this paper, we studied the link between class-wise feature invariance and classification performance. We studied the relationship between class-wise compactness of the learned features and the corresponding model performance in the deep neural network context. We report two findings: First, we show that contrary to the common intuition, in deep learning *higher compactness does not imply better performance*. We validate this hypothesis over two datasets, MNIST and CIFAR10 using different architectures. Second, we show that neural network models do not implicitly learn more compact feature spaces. In fact, on the contrary *the com-*

pactness decreases during the training phase. The potential reasons and implications of such phenomena remain to be further studied.

Acknowledgements

This work was funded by the European Union's Horizon 2020 Research and Innovation Action Program under Grant 871449 (OpenDR). The work of Firas Laakom was funded by the NSF-Business Finland Center for Visual and Decision Informatics (CVDI) project AMALIA.

References

Abe, S. On invariance of support vector machines. In *Proceedings of the 4th international conference on intelligent data engineering and automated learning*, 2003.

Advani, M. S., Saxe, A. M., and Sompolinsky, H. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252. PMLR, 2019.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

d'Ascoli, S., Refinetti, M., Biroli, G., and Krzakala, F. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pp. 2280–2290. PMLR, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal processing magazine*, 29(6):82–97, 2012.

Iosifidis, A., Tefas, A., and Pitas, I. On the optiaml class representation in Linear Discriminant Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 24(9):1491–1497, 2013.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J. A., Orts-Escolano, S., Garcia-Rodriguez, J., and Argyros, A. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Pearson, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901.

Sanyal, A., Dokania, P. K., Kanade, V., and Torr, P. H. How benign is benign overfitting? *arXiv preprint arXiv:2007.04028*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ye, J. Least squares linear discriminant analysis. *International Conference on Machine Learning*, 1:1087–1093, 2007.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

7.9 Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning

The appended paper follows.

Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning

Pavlos Tiritiris
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
pavlosti@csd.auth.gr

Nikolaos Passalis
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
passalis@csd.auth.gr

Anastasios Tefas
Dept. of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
tefas@csd.auth.gr

***Abstract*—Object tracking allows for localizing moving objects in sequences of frames providing detailed information regarding the trajectory of objects that appear in a scene. In this paper, we study active object tracking, where a tracker receives an input visual observation and directly outputs the most appropriate control actions in order to follow and keep the target in its field of view, unifying in this way the task of visual tracking and control. This is in contrast with conventional tracking approaches, as typically developed by the computer vision community, where the problem of detecting the tracked object in a frame is decoupled from the problem of controlling the camera and/or the robot to follow the object. Deep Reinforcement Learning (DLR) methods hold the credentials for overcoming these issues, since they allow for tackling both problems, i.e., detecting the tracked object and providing control commands, at the same time. However, DRL algorithms require a significantly different methodology for training compared to traditional computer vision models, e.g., they rely on dynamic simulations for training instead of static datasets, while they are often notoriously difficult to converge, often requiring reward shaping approaches for increasing convergence speed and stability. The main contribution of this paper is a DRL, vision-based active tracking method, along with an appropriately designed reward shaping approach for active tracking problems. The developed methods are evaluated using a state-of-the-art robotics simulator, demonstrating good generalization on various dynamic trajectories of moving objects under a wide range of different setups.**

***Index Terms*—Active Tracking, Deep Reinforcement Learning, Reward Shaping, Webots**

I. Introduction

In recent years, object tracking has shown a significant growth and is used in a wide range of applications [1], [2], including but not limited to robotics applications [3] and human computer interaction [4]. Object tracking aims to localize a moving object in a stream of frames, enabling us to keep detailed information regarding the *trajectory* of the corresponding object through time. This information can be used either for extracting higher level information regarding the qualities and behavior the tracked object, or for controlling various parameters of the camera used for acquiring the video stream. The latter is especially important in various applications, such as autonomous cinematography [5], or even just ensuring that an appropriate view of the object of interest has been obtained, e.g., in surveillance applications [6].

It is worth noting that the goal in many tracking applications is *control* [7], instead of tracking *per se*, leading to *passive tracking* approaches, where tracking is disconnected from the actual task at hand. On the other hand, the emergence of powerful Deep Reinforcement Learning (DRL) approaches [8], [9], enabled the development of *active tracking* approaches [10], [11]. Such approaches are capable of learning end-to-end control policies, directly from raw RGB input, without any intermediate pre-processing step. In this way, the developed algorithms can be directly optimized for the task at hand, without involving separate intermediate tasks. At the same time, this unified approach also holds the credentials for providing less complex and more robust systems [11].

Despite their apparent advantages, little work has been done so far for active tracking approaches [10], [11], since active tracking requires using advanced and realistic simulation environment for simulating the effects of various control commands instead of relying on static datasets. At the same time, active tracking typically relies on reward signals for the training process instead of ground truth bounding boxes that are usually used in passive tracking approaches. However, defining the appropriate reward functions for such tasks is not trivial, since there are many alternative ways to formulate the goal of the system [12]. This is in contrast with other DRL applications, such as games [13], where the reward function is intrinsic to the problem. Indeed, the way that the reward function is defined can have a signfincant effect on the behavior of the resulting DRL model, especially for complex control tasks [14]. At the same time, providing additional rewards, in the form of *reward shaping* [15]–[17], can often allow for further increasing the stability of the training process, along with its convergence speed.

The contribution of this work is two-fold. First, we develop and evaluate an active tracking simulation environment, demonstrating that active tracking methods can be indeed trained in an end-to-end fashion operating *directly* on raw RGB inputs. To this end, a realistic robot simulation environment

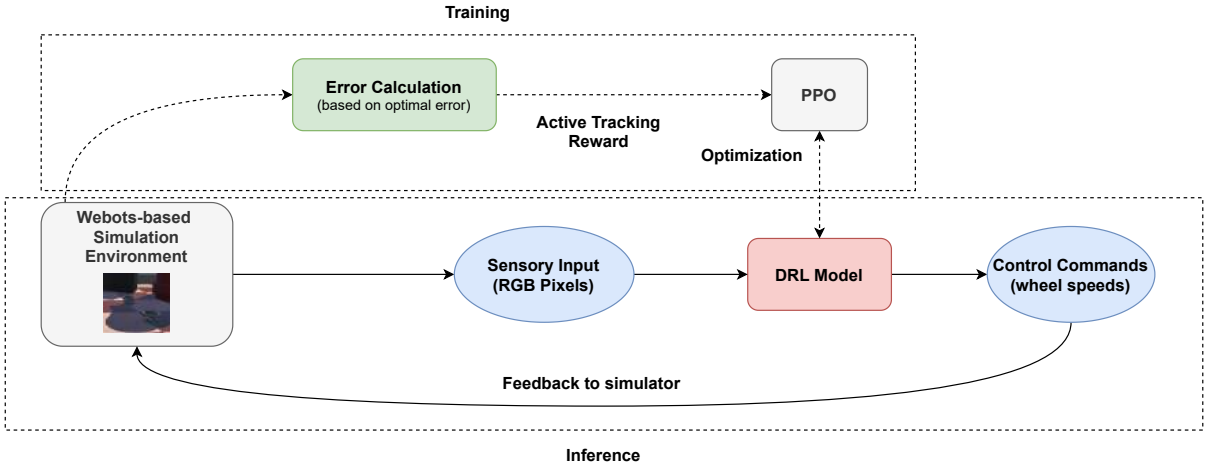


Fig. 1. Overview of the proposed DRL-based active tracking setup. First, the agent (DRL Model) is trained using the developed Webots-based simulation environment. Then, the trained agent is evaluated on a number of different scenarios (described in Section III).

was constructed using the Webots simulator [18], while the proposed method was evaluated using a real robot, the e-puck robot [19], and two different setups, involving different movement patterns of the robots. Furthermore, two different control scenarios were evaluated, corresponding to the two main approaches that are typically used for control: a) control using discrete actions/steps and b) control using continuous action spaces. Second, a simple, yet effective temporal difference-based reward function is introduced and evaluated for active tracking, improving the tracking error and allowing for directly performing control based on a tracking objective. Indeed, it is demonstrated, through extensive experiments, that the employed reward function can indeed lead to improvements in tracking accuracy, under a wide range of different setups.

The rest of the paper is structured as follows. First, background information and the proposed method are presented in Section II. Then, the experimental evaluation is proposed in Section III. Finally, conclusions are drawn in Section IV.

II. Proposed Method

The proposed approach is provided in this Section. First, we briefly introduce the developed simulation environment. Then, the employed DRL method is provided, along with the network architectures used for training the corresponding agents. Finally, three different reward functions for active tracking are introduced and discussed. An overview of the proposed approach for developing DRL agents for active tracking is provided in Fig. 1.

A. Simulation Environment and DRL agent

The simulation environment was built using the Webots simulator [18], which is an open source highly realistic robot simulator. The developed simulation environment contains two GCtronic e-puck robots [19], with the first one being the tracked target and the second one carrying an RGB camera (64×64 pixels) and aiming to track the first robot. E-puck robot moves by appropriately setting the velocity of its two wheels. Two different kinds of agents were employed in this paper: a) discrete action space agents and b) continuous action space agents. For agents that work with discrete actions, there are $2k + 1$ possible velocity values within the range $[-maxSpeed, +maxSpeed]$, splited by $2k$ equally spaced partitions. Therefore, the total number of actions for these agent is $(2k + 1)^2$, given that both wheels of the robot must be controlled. On the other hand, for continuous agents, two real numbers, one for each wheel, are used. The output of the agent is constrained between the minimum and the maximum speed supported by the robot, ensuring that all control actions will be within the hardware capabilities of the robots.

For training the agent, a state-of-the-art DRL optimization method, the Proximal Policy Optimization (PPO) [8], was employed. An actor-critic architecture was used to this end, where the actor is responsible for selecting the next action, while the critic was used for estimating the value of each state. PPO relies on the *advantage* for each performed action a when the agent is at state s:

$$A_\theta(s, a) = Q_\theta(s_t) - \sum_{t=0}^{T-1} \gamma^t r_t, \tag{1}$$

where $Q_\theta(s_t)$ denotes the critic's network estimated value, γ is the discount factor and r_t is the obtained reward at time t for a total number of T time-steps. To ensure the stability of the optimization process, PPO uses a probability ratio between new and old policy. Clipping this ratio, as proposed in [8], improves the behavior of the training process. Finally, note that in training phase PPO collects samples from the older policy using importance sampling.

The architecture used for both the actor and critic is depicted in Fig. 2. Three 2D convolutional layers were used, followed by three fully connected layer. The ReLU activation function was used after each layer [20], while average pooling

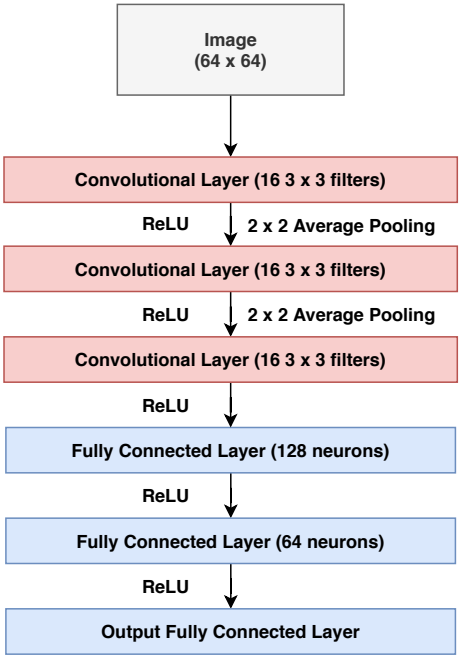


Fig. 2. Network architecture used for the actor and critic models

was employed instead of max pooling to avoid discarding potentially useful information. For discrete agents, the final layer of the actor has the same number of neurons as the number of possible actions, i.e., $(2k+1)^2$, while for continuous agents two output neurons are used (one for controlling the speed of each wheel). In the latter case, the output of the network is passed through the *tanh* activation, to ensure that it will range between -1 (maximum speed backward) and 1 (maximum speed forward). When the network outputs a value of 0, then the corresponding wheel stays stationary.

B. Reward

The RL agents must learn to follow the moving target as close as possible over a safe distance, while the camera looking at the center of the target. To this end, we define the distance error as:

$$e_d(t) = |d(t) - s|, \tag{2}$$

where $d(t)$ is the distance between the robots at time t and s is a predefined distance that must be kept from the robot (safe distance). Also, we define the angular error $e_a(t)$ at time t as the absolute value of the angle between the optimal look-at vector of the camera and the speed vector of the target robot, as shown in Fig. 3. A separate reward is calculated for each of these two errors, i.e., $r_d(t)$ and $r_a(t)$ respectively, while the total reward is calculated as the sum between the corresponding rewards:

$$r(t) = r_d(t) + r_a(t) \tag{3}$$

For defining these two individual rewards, i.e., $r_d(t)$ and $r_a(t)$, that contribute to the final reward provided in (3) several

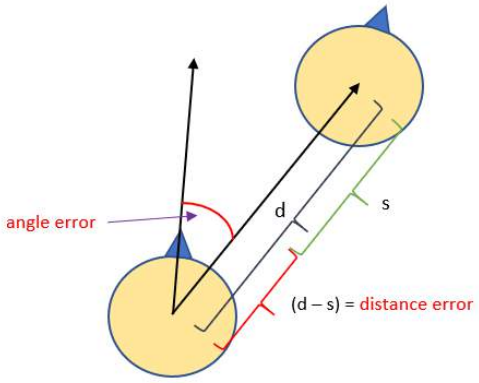


Fig. 3. Distance and angular errors

methods can be considered. Perhaps the simplest one is to directly use the negative of error, i.e.,

$$r(t) = -error(t), \tag{4}$$

where $r(t)$ refers to either $r_d(t)$ and $r_a(t)$ (according to the value used in place of $error(t)$), aiming to directly minimize the control error. However, as we experimentally demonstrate in Section III, this naive way of defining the reward leads to suboptimal results. Recent works in vision-based DRL control proposed to use hint-based rewards inspired by reward shaping methods [14], i.e.,

$$r(t) = \begin{cases} c_g, & error(t) < margin \\ c_m, & error(t) < error(t-1) \\ c_p, & \text{otherwise} \end{cases}, \tag{5}$$

where c_g corresponds to the reward for achieving the desired target, c_m to the reward for moving towards a direction that reduces the error , while c_p corresponds to the penalty the agent receives in any other case (its absolute value must be larger than c_g to avoid oscillations). We set these parameters to $c_g = 1$, $c_m = 0.5$ and $c_g = -1.2$ following the protocols proposed in the literature and experiments conducted to select the optimal parameters. However, setting these parameters can require a significant amount of effort, involving repeated experiments to determine the optimal values. Therefore, to overcome this limitation, in this work we propose using a simpler, yet more effective approach for calculating the reward based on the *temporal difference* between two subsequent error values, i.e.,

$$r(t) = error(t-1) - error(t). \tag{6}$$

This approach does not require any kind of additional fine-tuning, while it leads to more accurate control, as we demonstrate in Section III. Note that the first and third reward functions are directly linked to the (unnormalized) values of

TABLE I
HYPERPARAMETER USED FOR ALL THE CONDUCTED EXPERIMENTS

Clip parameter (PPO)	0.2
Number of update iterations (PPO)	10
Gradient norm clipping (max)	0.5
Batch size	64
Actor / Critic learning rate	0.0001 / 0.0003
Number of episodes	100

the error, therefore min-max normalization is required after the end of each episode, to ensure the stability of the optimization process. This is not necessary for the hint-based reward, since the values of this reward are already bounded.

III. EXPERIMENTAL EVALUATION

The different agents and reward functions are evaluated under different setups in this Section. The hyperparameters used for all the conducted experiments are summarized in Table I. The number of update iterations refers to the number of gradient descent steps performed after each simulation episode, while the gradient norm clipping refers to the clipping of the training gradients that are larger than the specific threshold, which improves training stability. The former was set to 10, while the later to 0.5 The safe distance was set to $s = 0.2$m (meters), while the margin for the hint-based reward function was set to 0.05.

The developed agents were evaluated in two different setups:

1) *random movement*, where the velocity of both wheels of the front robot are selected randomly (therefore the first robot moves on a non-straight trajectory). The velocities remain the same during the entire episode. This setup was used both for training and evaluation of the trained agents.
2) *random movement with velocity changing*, where the same setup as before is used, but the speed of the wheels changes every 200 steps. As a result, the trajectory of the first robot changes several times within the same episode, requiring the tracking robot to promptly adjust in order to avoid loosing the target robot. This setup was used for evaluating the agents trained with the first setup.

Each episode consists of 1000 steps, while an episode ends early when the tracking robot loses its target. For evaluation we report two different metrics:

1) *average distance error* ("Distance error"), which measures whether the agents keeps the desired distance from the tracked robot, and
2) *average control steps per episode* ("Steps per episode"), which measures the ability of the agent to track the front robot (note that an episode ends when the tracking robot loses its target).

First, the proposed method was evaluated using the first setup (random movement evaluation). The evaluation results for the discrete agent are reported in Table II, while for the continuous agent in Table III. The best discount factor was selected through validation experiments for all the evaluated reward functions ($\gamma = \{0.01, 0.25, 0.5, 0.75, 0.99\}$). First, note that for all the evaluated reward functions, the agents are indeed successfully trained, since they solve almost all the test episodes perfectly (the average number of steps is larger than 950 out of a maximum of 1000). At the same time, the temporal difference reward seems to lead to the overall best distance error (both for the discrete and continuous agent). However, it lead to slightly worse behavior for the discrete agent, failing to correctly track the target robot in a few cases, since the average number of steps per episode is reduced from 1000 to 969.

TABLE II
EVALUATION SETUP 1: DISCRETE AGENT

Reward	Distance error	Steps per episode
Negative of error	0.014	1000
Hint-based	0.054	969
Proposed	**0.009**	969

TABLE III
EVALUATION SETUP 1: CONTINUOUS AGENT

Reward	Distance error	Steps per episode
Negative of error	0.02	954
Hint-based	0.03	966
Proposed	**0.01**	1000

However, the opposite behavior is observed in the second evaluation setup reported in Table IV, where the proposed reward leads to both the best distance error, as well as leads to never loosing the target robot for all the evaluated episodes. The same results are achieved for the hint-based reward, even though hint-based rewards seem to always lead to higher distance error. This can be explained, since hint-based rewards

TABLE IV
EVALUATION SETUP 2: DISCRETE AGENT

Reward	Distance error	Steps per episode
Negative of error	0.016	985
Hint-based	0.035	1000
Proposed	**0.007**	1000

TABLE V
EVALUATION SETUP 2: CONTINUOUS AGENT

Reward	Distance error	Steps per episode
Negative of error	0.014	1000
Hint-based	0.030	974
Proposed	**0.006**	1000

TABLE VI
EVALUATION SETUP 2 (WITH OBSTACLES): CONTINUOUS AGENT

Reward	Distance error	Steps per episode
Negative of error	0.037	961
Hint-based	0.072	853
Proposed	**0.007**	965

are disconnected from the actual value of the error. Even though in previous works this has been shown to improve the training stability when combined with Q-learning based algorithms [14], this reward function does not seem to lead to similar improvements for the problem at hand. The ability of the proposed reward function to improve the distance error is again validated using the second evaluation setup using continuous action spaces (Table V).

Finally, we also evaluated the ability of the trained agents to work on an even more challenging environment by including additional obstacles, in the second setup. The evaluation results for the continuous agent are reported in Table VI. Again, it is confirmed that the proposed reward leads to the best results for the continuous agents, both regarding the distance error and the ability of the agent to keep the target in view, despite the presence of obstacles.

IV. CONCLUSIONS

In this paper a deep reinforcement learning-based approach for solving active tracking problems was presented. The proposed method can be trained in end-to-end fashion, while operating directly on raw RGB inputs without requiring solving any intermediate tracking problem, like most of the existing passive tracking methods. At the same time, a simple, yet effective reward function for active tracking problems was introduced and demonstrated that it can lead to improved tracking performance, under two different evaluation setups and DRL agents using a realistic simulation environment developed using the Webots simulator and a model of a real robot, the e-puck robot. The results provided in this paper pave the way for developing more sophisticated active tracking methods, ranging from methods that can track a wide variety of objects, to methods that can control simultaneously both the robot and the camera, choosing the most appropriate action to be performed, increasing the perception accuracy. Furthermore, the proposed method also consists a first step towards developing advanced active perception algorithms for other tasks [21], such as object detection, human recognition, etc.

REFERENCES

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13–58, 2006.

[2] P. Li, D. Wang, L. Wang, and H. Lu, "Deep visual tracking: Review and experimental comparison," *Pattern Recognition*, vol. 76, pp. 323–338, 2018.

[3] C. T. Chou, J.-Y. Li, M.-F. Chang, and L. C. Fu, "Multi-robot cooperation based human tracking system using laser range finder," in *Proceesings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 532–537.

[4] M. Abbaszadegan, S. Yaghoubi, and I. S. MacKenzie, "Trackmaze: A comparison of head-tracking, eye-tracking, and tilt as input methods for mobile games," in *Proceedings of the International Conference on Human-Computer Interaction*, 2018, pp. 393–405.

[5] J. Capitan, A. Torres-Gonzalez, and A. Ollero, "Autonomous cinematography with teams of drones [j]," in *Proceedings of the Workshop on Aerial Swarms. IEEE International Conference on Intelligent Robots and Systems*, vol. 1, 2019, pp. 1–3.

[6] Z. Pan, S. Liu, A. K. Sangaiah, and K. Muhammad, "Visual attention feature (vaf): a novel strategy for visual tracking based on cloud platform in intelligent surveillance systems," *Journal of Parallel and Distributed Computing*, vol. 120, pp. 182–194, 2018.

[7] W. Ye, Z. Li, C. Yang, J. Sun, C.-Y. Su, and R. Lu, "Vision-based human tracking control of a wheeled inverted pendulum robot," *IEEE Transactions on Cybernetics*, vol. 46, no. 11, pp. 2423–2434, 2015.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Proceedings oft he AAAI Conference on Artificial Intelligence*, 2018.

[10] W. Luo, P. Sun, F. Zhong, W. Liu, T. Zhang, and Y. Wang, "End-to-end active object tracking via reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2018.

[11] ——, "End-to-end active object tracking and its real-world deployment via reinforcement learning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[12] R. Pocius, D. Isele, M. Roberts, and D. W. Aha, "Comparing reward shaping, visual hints, and curriculum learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 1928–1937.

[14] N. Passalis and A. Tefas, "Deep reinforcement learning for controlling frontal person close-up shooting," *Neurocomputing*, vol. 335, pp. 37–47, 2019.

[15] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the International Conference on Machine Learning*, vol. 99, 1999, pp. 278–287.

[16] A. C. Tenorio-Gonzalez, E. F. Morales, and L. Villasenor-Pineda, "Dynamic reward shaping: training a robot by voice," in *Ibero-American conference on artificial intelligence*. Springer, 2010, pp. 483–492.

[17] S. M. Devlin and D. Kudenko, "Dynamic potential-based reward shaping," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, 2012, pp. 433–440.

[18] O. Michel, "Cyberbotics ltd. webots™: professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[19] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the Conference on Autonomous Robot Systems and Competitions*, vol. 1, 2009, pp. 59–65.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[21] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, 2018.

7.10 EfficientLPS: Efficient LiDAR Panoptic Segmentation

The appended paper [195] follows.

EfficientLPS: Efficient LiDAR Panoptic Segmentation

Kshitij Sirohi*,[1], Rohit Mohan*,[1], Daniel Büscher[1], Wolfram Burgard[1,2], and Abhinav Valada[1]

***Abstract*—Panoptic segmentation of point clouds is a crucial task that enables autonomous vehicles to comprehend their vicinity using their highly accurate and reliable LiDAR sensors. Existing top-down approaches tackle this problem by either combining independent task-specific networks or translating methods from the image domain ignoring the intricacies of LiDAR data and thus often resulting in sub-optimal performance. In this paper, we present the novel top-down Efficient LiDAR Panoptic Segmentation (EfficientLPS) architecture that addresses multiple challenges in segmenting LiDAR point clouds including distance-dependent sparsity, severe occlusions, large scale-variations, and re-projection errors. EfficientLPS comprises of a novel shared backbone that encodes with strengthened geometric transformation modeling capacity and aggregates semantically rich range-aware multi-scale features. It incorporates new scale-invariant semantic and instance segmentation heads along with the panoptic fusion module which is supervised by our proposed panoptic periphery loss function. Additionally, we formulate a regularized pseudo labeling framework to further improve the performance of EfficientLPS by training on unlabelled data. We benchmark our proposed model on two large-scale LiDAR datasets: nuScenes, for which we also provide ground truth annotations, and SemanticKITTI. Notably, EfficientLPS sets the new state-of-the-art on both these datasets.**

***Index Terms*—Scene Understanding, Semantic Segmentation, Instance Segmentation, Panoptic Segmentation.**

I. Introduction

AUTONOMOUS vehicles are required to operate in challenging urban environments that consist of a wide variety of agents and objects, making comprehensive perception a critical task for robust and safe navigation. Typically, perception tasks are focused on independently reasoning about the semantics of the environment and recognition of object instances. Recently, panoptic segmentation [1] which unifies semantic and instance segmentation has emerged as a popular scene understanding problem that aims to provide a holistic solution. Panoptic segmentation simultaneously segments the scene into 'stuff' classes that comprise of background objects or amorphous regions such as road, vegetation, and buildings, as well as 'thing' classes that represent distinct foreground objects such as cars, cyclists, and pedestrians. Panoptic segmentation has been extensively studied in the image domain [1]–[4], facilitated by the ordered structure of images being supported by well-researched convolutional networks. However, only a handful of methods have been proposed for panoptic segmentation of LiDAR point clouds [5], [6]. LiDARs have become an indispensable sensor for autonomous vehicles due to their illumination independence and geometric description of the scene, making scene understanding using LiDAR point clouds an essential capability. However, the typical unordered, sparse, and irregular structure of point clouds pose several unique challenges.

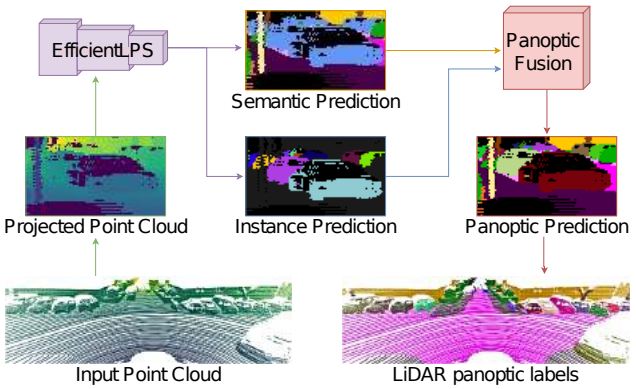


Fig. 1: Overview of the top-down EfficientLPS architecture that consists of a shared backbone to learn spatially-aware features from the projected point cloud and individual heads to learn semantic and instance specific features which are fused in the panoptic fusion module. The network explicitly utilizes the range information in the backbone, semantic head and fusion module to mitigate the problems due to the projection and distance-dependent sparsity of LiDAR point clouds.

To this end, deep learning methods that rely on grid based convolutions to address these challenges typically follow two different directions. They either project the point cloud into the 3D voxel space and employ 3D convolutions on them [7], [8], or they project the point cloud into the 2D space [6], [9], [10] and employ the well-researched 2D Convolutional Neural Networks (CNNs). While voxel-based method achieve high accuracy, they are computationally more expensive and require substantial memory to store the voxelized point clouds. Methods such as [8], [11] leverage the sparse nature of occupied voxel grids to improve the runtime and memory consumption. The 2D projection based methods on the other hand, yield a more denser representation and require comparatively lesser computational resources, but they suffer from information loss during projection, blurry CNN outputs, and incorrect label assignment to the occluded points during re-projection. Therefore, there is a need to bridge this gap with a method that has the advantages of fast and memory-efficient 2D convolutions while mitigating the problems due to the projection.

In this work, we present the novel Efficient LiDAR Panoptic Segmentation (EfficientLPS) architecture that effectively addresses the aforementioned challenges by employing a 2D

*These authors contributed equally.
[1]Department of Computer Science, University of Freiburg, Germany
[2]Toyota Research Institute, Los Altos, USA.

CNN for the task while explicitly utilizing the unique 3D information provided by point clouds. EfficientLPS consists of a shared backbone comprising our novel Proximity Convolution Module (PCM), an encoder, the proposed Range-aware FPN (RFPN) and the Range Encoder Network (REN). We build the encoder and REN based on the EfficientNet [12] family, therefore we follow the convention of naming our model with the Efficient prefix. EfficientLPS also consists of a novel distance-dependent semantic segmentation head and an instance segmentation head, followed by a fusion module that provides the panoptic segmentation output. Our network makes several new contributions to address the problems that persist in LiDAR cylindrical projections. We propose the Proximity Convolution Module (PCM) that alleviates the problems caused by the fixed geometric grid structure of a standard convolution which is incapable of modeling object transformations such as scaling, rotation and deformation [13]. The problem of distance-dependent sparsity further exacerbates the limited transformation modeling capability of standard convolutions. The PCM models the transformations of objects in the scene by leveraging the contributions of nearby points, effectively reshaping the convolution kernel depending on range values.

When LiDAR points are projected into the 2D domain, objects tend to be closer to each other. Hence, the network in these cases often ignores smaller objects in favor of larger overlapping objects. Although this overlap is more distinguishable in the range channel of the projections, the features computed over all the projection channels begin to lose track of this distinction as they try to capture more and more complex representations in the deeper layers of the encoder. To alleviate this problem and enable the network to better distinguish adjacent objects, we propose the Range-aware Feature Pyramid Network (RFPN). We employ the REN parallel to the encoder to solely encode the range channel of the projection and selectively fuse it with the FPN outputs to compute range-aware multi-scale features.

Moreover, there is a large variation in the scale of objects due to the projection of the point cloud into the 2D domain. The objects that are closer tend to be larger in scale, and objects at a farther distance tend to be smaller. Hence, the 2D projection consists of objects that have distance-dependent scale variations. Typically, the instance head of top-down methods cope with it to a certain extent using many predefined anchors at different scales. However, the semantic head that predominantly aggregates multi-scale features tends to suffer [2], [14], [15]. In order to mitigate this problem, we propose the distance-dependent semantic head that consists of modules that incorporate our range-guided depth-wise atrous separable convolutions in addition to fixed multi-dilation rate convolutions to generate features that cover a relatively large scale range in terms of the receptive field in a dense manner.

Furthermore, segmented objects in the projection domain often tend to have inaccurate boundaries. In the image domain, these inaccuracies only span a few pixels so they have little or negligible effect on the overall performance. However, when the segmented output in the projection domain is re-projected back into point clouds, it causes leakage of object boundaries into the background and significantly affects the performance of the model. To address this problem, we introduce the novel panoptic periphery loss function that operates on the logits of the panoptic fusion module to effectively combine the outputs of both heads. Our proposed loss function refines 'thing' instance boundaries by maximizing the range separation between the foreground boundary pixels, i.e., the 'thing' instance boundary and the neighboring background pixels.

Most supervised learning methods require large amounts of annotated training data and manually labeling point clouds is an extremely arduous task. As an alternative solution to this problem, we explore the viability of generating pseudo labels from the abundantly available unlabeled point cloud datasets. We formulate a new framework for computing regularized pseudo labels from unlabeled data, given some labeled data with similar properties. The regularized pseudo labels aim to reduce the incorrect predictions on the unlabeled dataset to prevent confirmation bias. To the best of our knowledge, this is the first work to propose a pseudo labeling technique for any point cloud scene understanding task.

We perform extensive evaluations on the SemanticKITTI [16] and nuScenes [17] datasets which have point clouds with different densities to demonstrate the generalization ability of our model. As the nuScenes dataset itself does not provide panoptic segmentation labels, we compute the annotations from the publicly available semantic segmentation and 3D bounding box annotations. We provide several baselines and make the nuScenes panoptic segmentation dataset publicly available to encourage future research using sparse point clouds. Our proposed EfficientLPS consistently outperforms existing methods, thereby setting the new state of the art on both datasets and is ranked #1 on the SemanticKITTI leaderboard. Finally, we present detailed ablation studies that demonstrate the novelty of the various architectural contributions that we make in this work.

To summarize, the main contributions of this work are as follows:

1) A novel top-down architecture that consists of a shared backbone with task-specific heads that incorporate our proposed range enforced components and a fusion module supervised by our panoptic periphery loss function.
2) The proximity convolution module which boosts the transformation modeling capacity of the shared backbone by leveraging range proximity between neighboring points.
3) The novel range-aware feature pyramid network that reinforces bidirectionally aggregated semantically rich multi-scale features with spatial awareness.
4) The new semantic head that captures scale-invariant rich characteristic and contextual features using our range-guided depth-wise atrous separable convolutions.
5) The novel panoptic periphery loss function that refines the segmentation of 'thing' instances by maximizing the range separation between foreground boundary pixels and neighboring background pixels.
6) A new framework for improving panoptic segmentation of LiDAR point clouds by exploiting large unlabelled datasets via regularized pseudo label generation.
7) Exhaustive quantitative and qualitative evaluations of our model along with comprehensive ablation studies of our

proposed architectural components.
8) We made the code and models publicly available at http://rl.uni-freiburg.de/research/lidar-panoptic

II. Related Works

Panoptic segmentation has emerged as a popular scene understanding task since its introduction by Kirillov *et al.* [1]. By unifying semantic segmentation and instance segmentation, it aims at holistic scene understanding and reasoning. While panoptic segmentation has been extensively studied in the 2D domain using RGB images, only a handful of methods address this task in the 3D domain of LiDAR point clouds. In the following, we first discuss different convolution kernels and different techniques of representing point cloud data. We then discuss recent works that address the various scene segmentation tasks using point clouds in autonomous driving scenarios, namely: semantic, instance, and panoptic segmentation.

A. Convolution kernels

Standard 2D convolutions lack the ability to model geometric relations in the 3D domain due to the nature of the convolution operation that samples from fixed locations. Some works model these relations with the help of non-regular grids [13] where the learned offsets change the shape of the sampling grid in convolutions. Using RGB-D images, Wang *et al.* [18] directly use the depth value to weight the contribution of neighboring pixels for the convolution output. While the convolution shaping methods using RGB images learn the lacking spatial information, we propose a shaping mechanism using the already available spatial information in point clouds. We devise the Proximity-aware Convolution Module (PCM) that reshapes the convolution grid to capture local contextual information from neighboring pixels. This is especially helpful for capturing contextual information of very distant points in LiDAR point cloud that suffer from distance-induced sparsity.

B. Data Representation

The methods that are typically employed on point clouds can be broadly classified into three categories, namely: point-based, volumetric, and projection-based techniques. PointNet [19] is one of the first pioneering point-based methods which learns features using MLPs followed by max-pooling to extract global context. More recent methods [20]–[22] develop convolution operations and kernels specially designed to work with point clouds. Kernel Point Convolution (KPConv) [22] is one such method with flexible kernel points in the 3D space, which are learned in a similar manner on point clouds as in 2D convolutions. On the other hand, volumetric methods transform the point clouds into regular voxel grids and apply 3D convolutions [7]. Most methods that rely on 3D convolution are both memory and computationally intensive which limits the resolution of the voxels and hence the overall performance. However, to account for the sparse nature of the voxel grid, sparse 3D convolutions [8], [11] have been proposed to decrease the runtime and memory footprint.

In projection-based methods, the point cloud is projected onto an intermediate regular 2D grid representation to facilitate the use of well researched 2D convolution architectures. To obtain pseudo sensor data such as the grid representation, existing methods project points either using the spherical projection [23], [24] or using scan unfolding [25]. Conversely, point clouds are also projected into a bird's eye view (BEV) [26] to exploit the radial nature and obtain better spatial segregation. In this work, we employ scan unfolding due to its ability to recover dense representations, similar to the original format that a LiDAR sensor provides.

C. Scene Understanding using LiDAR Point Clouds

1) Semantic Segmentation: The challenges posed by the unordered and sparse nature of point clouds has hindered the progress in LiDAR semantic segmentation for autonomous driving. Considerably lesser number of techniques have been proposed to address this task using point clouds in comparison to methods in the visual domain. Dewan *et al.* [27] propose an approach to classify points into movable, non-movable, and dynamic classes, by combining deep learning-based semantic cues and rigid motion based motion cues in a Bayesian framework. Wu *et al.* [24] propose a projection-based approach that builds upon SqueezeNet and introduces the fire module which is incorporated into the encoder and decoder. DeepTemporalSeg [28] employs Bayesian filtering to obtain temporally consistent semantic segmentation.

The release of the SemanticKITTI [16] dataset motivated many works in semantic segmentation of LiDAR point clouds. Milioto *et al.* [23] propose a 2D CNN architecture that operates on spherically projected point clouds and employs a kNN based post-processing step to account for the occlusions due to the projection. SalsaNext [9] uses spherical projection for semantic segmentation and also performs uncertainty estimation. PolarNet [26] projects the point cloud in the birds-eye view and employs a ring convolutions on the radially defined grids. Some methods follow a hybrid approach by combining 2D and 3D convolutions. KPRNet [29] uses 2D convolutions for semantic segmentation followed by KPConv-based [22] post-processing using point-wise convolutions. SPVNAS [30] automates architecture design by employing neural architectural search to search for efficient 3D convolution-based models.

2) Instance Segmentation: Similar to the image domain, instance segmentation of point clouds can be classified into two categories: proposal based and proposal free methods. Proposal based methods perform 3D bounding box detection followed by point-wise mask generation for the points in each bounding box. 3D Bonet [31] follows this approach using two separate 3D bounding box proposal generation and mask generation branches. GSPN [32] generates proposals using a shape aware proposal generation for different instances. On the other hand, proposal free methods directly predict the instances by detecting keypoints such as the centriod of the instance, or the similarity between points, which is followed by clustering [33]. SGPN [34] learns a similarity matrix between the points which is used to cluster points with higher similarity scores between them. PointGroup [35] extracts semantic information using 3D sparse convolutions to cluster points towards the instance centroid, followed by

using the original points and the clustered points to obtain the final prediction. VoteNet [36] predicts an offset vector to the centroid of every point and then employs clustering.

3) Panoptic Segmentation: Panoptic segmentation methods can also be classified into proposal-free (bottom-up) or proposal-based (top-down) techniques. Bottom-up methods group points belonging to the same instances either by a voting scheme or based on pixel-pair affinity pyramid while simultaneously learning the semantic labels [37]. On the other hand, top-down approaches [10] tackle the problem in two a stage manner with a dedicated instance segmentation branch for detecting and segmenting 'thing' classes, and a semantic segmentation branch for segmenting the 'stuff' classes.

Miliotto *et al.* [6] and Gasperini *et al.* [5] adopt the bottom-up approach where instances are detected without region proposals. Miliotto *et al.* use spherical projection of point clouds and predict offsets to the centroids for aiding clustering. They also use 3D information available in the range images for trilinear upsampling in the decoder. Panoster [5] uses an instance head which directly provides the instance ids of the points from learnable clustering without any explicit grouping requirement. In addition to the spherical projection-based method Rangenet++ [23] and Panoster [5] also show the implementation of their clustering mechanism using the point-based method KPConv [22] for semantic segmentation. PanopticTrackNet [10] further unifies panoptic segmentation with multi-object tracking and provides temporally consistent instance labels.

4) Semi-Supervised Learning: Semi-supervised learning (SSL) for LiDAR panoptic segmentation has not been explored. Thus, we discuss the two prominent SSL approaches for 3D object detection. SESS [38] trains a EMA-based teacher model and a student model simultaneously with consistency loss between them whereas 3DIoUMatch [39] employs mean-teacher [40] based framework using an IoU prediction filtering mechanism. Both of the approaches use augmentation of point clouds for which the point cloud is available. In contrast, our proposed pseudo labeling framework exploits external unlabeled point cloud datasets with a separate teacher and student model.

In this work, we present a novel LiDAR panoptic segmentation network that effectively exploits the advantages of projection-based top-down methods. Our proposed architecture comprises of a shared backbone that incorporates the proposed proximity convolution module in the beginning to boost its geometric transformation modeling capacity and the novel range-aware FPN at the end to capture spatially aware and semantically rich multi-scale features. It further consists of a modified Mask R-CNN [41] instance head and a new semantic head that fuses distance-dependent fine and long-range contextual features with distance-independent features for enhanced scale-invariance. We also propose the novel panoptic periphery loss function that refines the 'thing' object class boundaries by maximizing the range difference between the foreground and background pixels of the instance boundaries. All of the aforementioned modules effectively leverage the intricacies of LiDAR data to address the issues prevalent in projection-based LiDAR segmentation. Additionally, we explore the viability of pseudo labeling for LiDAR panoptic segmentation and thus propose a novel regularized pseudo labeling framework for the same.

III. Technical Approach

In this section, we present a brief overview of our proposed EfficientLPS architecture and then detail each of its constituting components. Fig. 2 illustrates the topology of EfficientLPS that follows the top-down layout. First, we project the point cloud from the LiDAR scanner into the two-dimensional space using scan unfolding [25]. The projected representation comprises of five channels: range, intensity and the (x, y, z) coordinates. We then employ our novel shared backbone which consists of our proposed Proximity Convolution Module (PCM) to aid in modeling geometric transformations, followed by a modified EfficientNet-B5 encoder with the 2-way FPN [2]. We employ the proposed Range Encoder Network (REN) in parallel which takes the range channel of the projected point cloud as input. We then fuse the multi-scale outputs of the encoder and REN to obtain the range-aware feature pyramid that enhances the ability to distinguish adjacent objects at different distances. The entire shared backbone is enclosed with green dashed lines in Fig. 2.

Following the backbone, we employ the parallel semantic segmentation (depicted in orange) and instance segmentation (depicted in purple) heads. The semantic head consists of Dense Prediction Cells (DPC) [15] and Large Scale Feature Extractor (LSFE) [2] units, which we extend with our proposed range-guided depth-wise atrous separable convolutions to enable capturing scale invariant long-range contextual and fine features. We use a variant of Mask R-CNN [41] for the instance head. We then fuse the logits from both the heads in the panoptic fusion module [2] which is supervised by our panoptic periphery loss to facilitate object boundary refinement and yield the panoptic segmentation in the projection domain. Finally, we re-project the predictions into the 3D space to obtain the final panoptic segmentation output of the input point cloud. During training, we employ our proposed pseudo labeling technique to train our model with both labeled and unlabeled data. In the rest of this section, we describe each of the aforementioned components in detail.

A. Projection using Scan Unfolding

We employ scan unfolding [25] to project the point cloud into the 2D range image format. Scan unfolding aims to mitigate the problems due to the alternate spherical projection method which suffers from point occlusions due to ego-motion correction. Scan unfolding yields a much denser representation than spherical projection. LiDAR sensors typically provide raw data in a range image-like format, with each pixel describing the range value at a particular row and column. Each column of this range image consists of measurements taken by individual modules stacked vertically within the sensor at a particular time and each row represents the consecutive measurements of one module taken during spinning of the sensor. However, most of the publicly available datasets provide LiDAR measurements as a list of 3D Cartesian coordinates, without any information about the column or row indices. Hence, we assign these

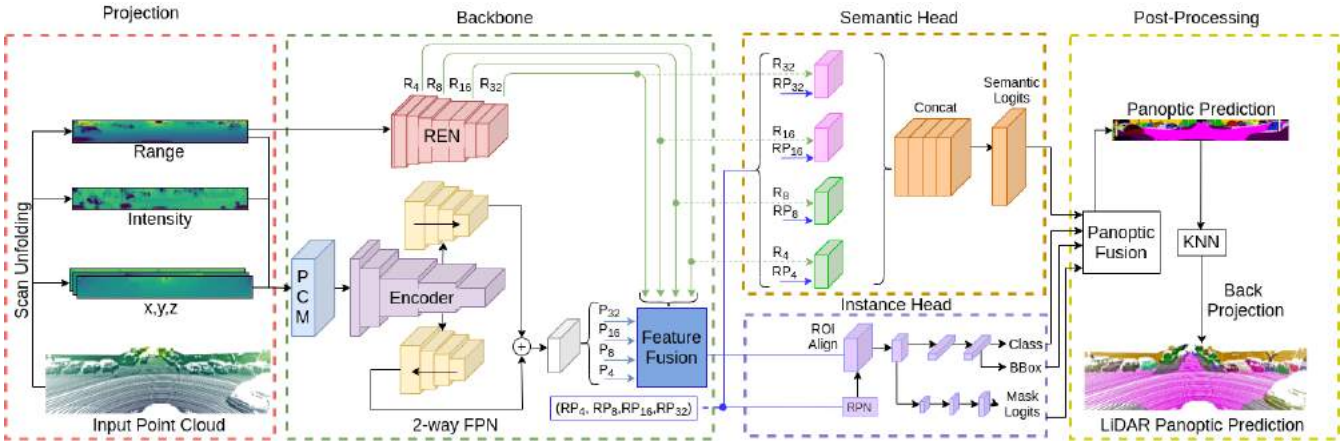


Fig. 2: Illustration of our proposed EfficientLPS architecture for LiDAR panoptic segmentation. The point cloud is first projected into the 2D domain using scan unfolding and fed as an input to our Proximity Convolution Module (PCM). Subsequently, we employ the shared backbone consisting of the EfficientNet encoder with the 2-way FPN and the Range Encoder Network (REN) in parallel. The output of these two modules are fused and fed as input to the semantic and instance heads. The logits from both heads are then combined in the panoptic fusion module which is supervised by the panoptic periphery loss function. Finally, the output of the panoptic fusion module is projected back to the 3D domain using a kNN algorithm.

indices to every point in the laser scan for recovering the range image-like representation.

In order to project the LiDAR scan represented as a point cloud to the range image, we assign row and column indices to every point in the scan corresponding to an image of size $W \times H$. The list of points provided by datasets such as KITTI [42] is typically constructed by just concatenating the rows of each scan. Hence, it is still ordered by horizontal and vertical indices. The scan unfolding algorithm takes advantage of this information and sequentially computes the yaw difference between the consecutive points to recover the vertical index. A jump is detected when the yaw difference is above a predefined threshold, which increments the vertical index of the following points in the list. We chose a threshold value of 310 deg, since the yaw angle typically drops from a value near 360 deg to near 0 between the rows. The horizontal index is computed as $\lfloor (0.5(1-\phi/\pi)W) \rfloor$, where ϕ is the yaw angle of each point in the range $[-\pi, \pi]$. The points are projected to their corresponding rows and columns with range, intensity and (x, y, z) coordinates represented as separate channels. This results in a tensor of shape $(5 \times H \times W)$ which is fed as input to the network. Note that the value of H is the number of vertically placed sensor modules in a sensor, which is 64 for the KITTI dataset. Other datasets that already contain the vertical index information, such as nuScenes, only require the computation of the horizontal index.

B. EfficientLPS Architecture

1) Backbone: The backbone consists of our proposed Proximity Convolution Module (PCM), followed by an encoder and the novel Range-aware Feature Pyramid Network (RFPN). We detail each of these components in the following section.

Proximity Convolution Module: The core of the PCM is the proximity convolution operation. The standard convolution operation performs sampling over a feature map followed by a weighted sum of the sampled values to yield an output feature map y. The convolution at pixel p is computed as

$$y(p) = \sum_{p_o \in R} w(p_o) \cdot x(p + p_o), \tag{1}$$

where R is a regular sampling grid containing the sampling offset locations around p in the input feature map x weighted by the kernel w.

The standard convolution is limited in its geometric transformation modeling capacity due to its fixed grid structure. The distance-dependent sparsity present in the LiDAR data further exacerbates the effects of this limitation. To tackle this constraint, we propose the proximity convolution which exploits range information to augment the spatial sampling locations for effectively improving the transformation modeling ability. Formally, in the proximity convolution, for each pixel p in the projected range image $R \in \mathbb{R}^{h \times w}$, we compute its nearest neighbors using the k-Nearest-Neighbors (kNN) algorithm. Here, we use range difference of the corresponding points from range image as the distance in the 3D space to find the nearest neighbors. Subsequently, we sort the nearest neighbors in the ascending order of their range difference to the query pixel. We now adapt Eq. (1) as

$$y(p) = \sum_{p_n \in N} w(p_n) \cdot x(p + p_n), \tag{2}$$

where N is no longer a regular grid but consists of offsets for the n nearest neighbors of pixel p. Please note that the grid N includes the offset to the query pixel and its $n-1$ nearest neighbors. The weights w are learned in the same manner as in standard convolutions. The search grid for the kNN algorithm is always larger than the learnable weight matrix and value of k is the product of kernel size dimensions.

The sampling operation of the proximity convolution in comparison to the standard convolution is illustrated in Fig. 3. In the example, the convolutions are performed at the border of the objects. As shown in Fig. 3, the proximity convolution forms the kernel according to the shape of the object, while the

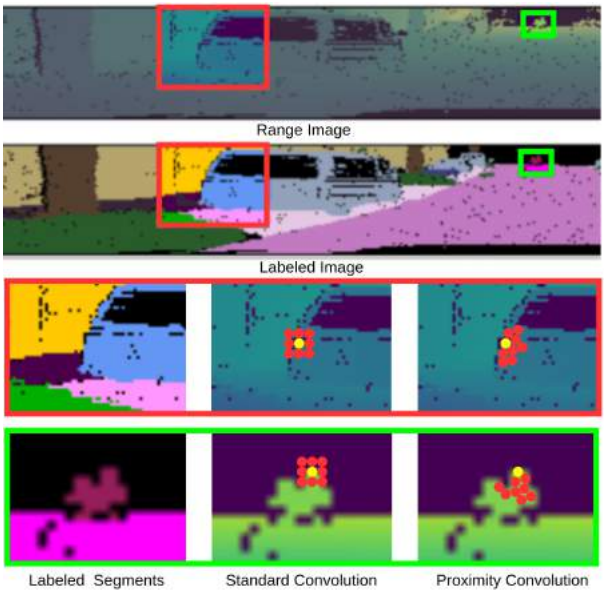


Fig. 3: Comparison of pixel sampling while applying standard convolution and our proposed proximity convolution. The highlighted red box contains a car and green box contains a bike. The convolution is applied by placing the kernel at the yellow dot, and the sampled neighboring pixels are represented by red dots. Observe that the neighboring pixels are sampled adaptively based on the range difference of the corresponding points in the range image.

standard convolution obtains information outside the objects. Particularly, the standard convolution is not able to obtain information for the bike rider (green rectangle), since the large distance of the object from the sensor causes increased sparsity. Since the proximity convolution is less dependent on this distance-induced sparsity, it successfully represents the shape of the bike rider. Therefore, the proximity convolution models the geometric transformations more effectively, especially for farther away objects that suffer from distance-induced sparsity.

The proximity convolution module comprises of the proposed proximity convolution, synchronized Inplace Activated Batch Normalization (iABNsync) [3] and Leaky ReLU activation. We use iABNsync in this layer and all the subsequent parts of the network in contrast to the vanilla batch normalization layer, as it provides a better estimate of the gradients and reduces the GPU memory footprint. We study the performance of the proximity convolution module in the ablation study presented in Sec. IV-D2.

Encoder: We adopt the EfficientNet [12] topology for the main encoder as well as the Range Encoder Network (REN). We remove the Squeeze and Excitation (SE) connections to enable better localization of features and contextual elements. Similar to the proposed proximity convolution module, we replace the batch normalization layers with iABNsync and Leaky ReLU activation. The EfficientNet architecture comprises of nine blocks where blocks 2, 3, 5, and 9 yield multi-scale features that correspond to the down-sampling factors of $\times 4$, $\times 8$, $\times 16$ and $\times 32$ respectively.

EfficientNet employs compound scaling to scale the base network efficiently. Here, width, depth, and the resolution of the network are the coefficients available for scaling. We choose the scaling coefficients for the main encoder as 1.6, 2.2, and 456 respectively and the coefficients for the REN as 0.1, 0.1, and 224 respectively, which we obtain via grid search optimization. The output of the PCM is fed as input to the main encoder and the REN takes the projected range image as input. Fig. 2 depicts the main encoder in light purple and the REN in red.

Range-Aware FPN: Our proposed range-aware FPN (RFPN) reinforces the coherently aggregated fine and contextual features of Feature Pyramid Networks (FPNs) with distance awareness. This enables the network to better segregate adjacent objects with different range variations. We build upon the 2-way FPN [2] that enables bidirectional flow of information using two parallel branches that aggregate multi-scale from the main encoder in a top-down and bottom-up manner respectively. The 2-way FPN is depicted with yellow blocks in Fig. 2. The outputs from both the parallel branches at each resolution are summed together and passed through a 3×3 convolution with 256 output channels to yield the outputs: P_4, P_8, P_{16}, and P_{32}. Note that we use standard convolutions in the 2-way FPN instead of separable convolutions used in [2] to learn richer representations at the expense of additional parameters.

Our proposed RFPN consists of the aforementioned 2-way FPN, the REN module, and the feature fusion module as shown in Fig. 4 (a). The REN module is employed in parallel to the 2-way FPN. We find that enabling the REN to learn to encode range information explicitly in different scales rather than direct downsampling of range data, yields better performance as shown in the ablation studies presented in Sec. IV-D3. The outputs of REN which are at four different resolutions (R_4, R_8, R_{16} and R_{32}) and the outputs of the 2-way FPN (P_4, P_8, P_{16} and P_{32}) are fed as input to the Feature Fusion module which computes range-aware pyramid features for each of the corresponding resolution (RP_4, RP_8, RP_{16} and RP_{32}). Here, 4, 8, 16 and 32 denote different downsampling factors with respect to the input.

The purpose of the fusion module is two-fold. First, to fuse the inputs R_s and P_s, where s denotes the downsampling factor. Second, to enable the network to emphasize on the more informative features between the 2-way FPN features and the corresponding fused features, hence incorporating distance awareness selectively. As shown in Fig. 4 (b), the feature fusion module consists of two branches. The first branch takes the concatenated tensors R_s and P_s as input, and feeds them through two 3×3 convolution layers sequentially to yield the fused features G_s. Additionally, we use iABNsync and leaky ReLU layers after each 3×3 convolution. We compute the weight factors for this branch (w_{fs}) by employing a 1×1 convolution followed by a sigmoid activation. The second branch propagates P_s in parallel and the weight factor of this branch is computed as $1 - w_{fs}$. Then the output (RP_s) of this module is given by

$$RP_s = w_{fs} * G_s + (1 - w_{fs}) * P_s, \tag{3}$$

For the sake of simplicity, the fusion module is depicted as one blue box in Fig. 2, whereas in practice each resolution has its own exclusive feature fusion module depicted in Fig. 4 (b). We present detailed analysis of different components of the range-aware FPN in the ablation study in Sec. IV-D3.

2) Distance-Dependent Semantic Head: The main component of our proposed distance-dependent semantic head is the

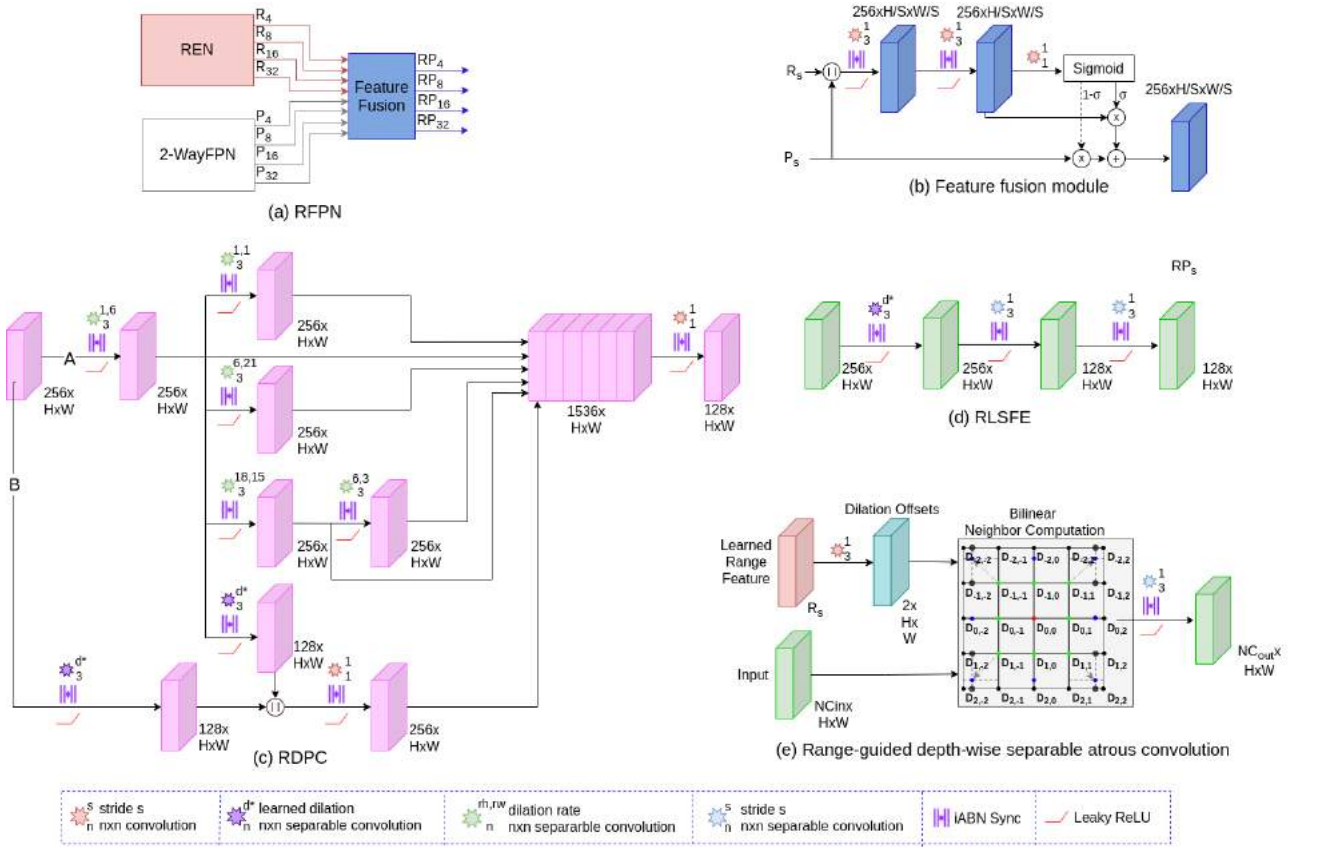


Fig. 4: Topology of the different proposed architectural components in EfficientLPS. (a) Range-Aware FPN (RFPN) and (b) Feature fusion module is used for the fusion of range encoded features with FPN features in RFPN. (c) Range-guided Dense Prediction Cells (RDPC) and (d) Range-guided Large Scale Feature Extractor (RLSFE) modules are part of the proposed semantic head. Lastly, (e) Range-guided depth-wise atrous separable convolution is the mechanism for controlling dilation offsets in the RDPC and RLSFE modules.

novel range-guided depth-wise atrous separable convolution operation. We essentially encode the range using the REN module and compute the dilation factor to apply at each central pixel from the encoded features. We then employ the depth-wise atrous separable convolution operation with the computed dilation factor, thereby enabling the receptive field to be adaptable to the range data. As shown in Fig. 4 (e), we employ a 3×3 convolution on the encoded range features from the REN module to obtain the dilation offsets for each pixel. Subsequently, we compute the bilinearly interpolated neighbors from the input at the corresponding offsets for each pixel. We then employ a 3×3 depth-wise separable convolution on the computed neighbors to generate the final output. Note that the input, the REN encoded features, and the dilation offsets have the same spatial resolution. We also use a parameter D_{max} in this convolution to set the maximum value for dilation offsets.

The range-guided depth-wise atrous separable convolution learns scale-invariant features as the dilation rate of the convolution kernel changes based on the distance, and so does the scale of objects. We take advantage of this scale invariance in the proposed semantic head of our EfficientLPS architecture. We extend the semantic head proposed in [2] consisting of Dense Prediction Cells (DPC), Large Scale Feature Extractor (LSFE), and Mismatch Correction Module (MC) with bottom-up path augmentation connections. We effectively retain the MC module and the bottom-up path augmentation connections but redesign the DPC and LSFE modules by incorporating our range-guided depth-wise atrous separable convolutions. We refer to this new LSFE variant as Range-guided Large Scale Feature Extractor (RLSFE) that comprises a 3×3 range-guided depth-wise atrous separable convolution with 256 output filters and $D_{max} = 3$ followed by an iABNsync and a Leaky ReLU activation function. The D_{max} parameter is set to a lower value in this module as it captures fine features that can get distorted with higher dilation rates. We employ two 3×3 separable convolutions with 128 output filters, each followed by iABNsync and a Leaky ReLU activation to further perform channel reduction and learn deeper features. Fig. 4 (d) shows the topology of our proposed RLSFE module.

The topology of our Range-guided Dense Prediction Cells (RDPC) module is depicted in Fig. 4 (c). We refer to the 3×3 depth-wise separable convolution with 256 output channels and a dilation rate of (1,6) as branch A. Similarly, the parallel 3×3 range-guided depth-wise atrous separable convolution with 128 output filters with $D_{max} = 24$ is referred to as branch B. Branch A further splits into four parallel branches. Three of the branches consist of a 3×3 depth-wise separable convolution with 256 output channels with dilation rates (1,1), (6,21), and (18,15) respectively. The fourth branch is an identity connection that goes to the end in parallel to all the branches.

The fifth branch concatenates with Branch B and consists of a 3×3 range-guided depth-wise atrous separable convolution with 128 output filters with $D_{max} = 24$, and runs parallel to other branches. The branch with (18,15) dilation rates further branches out into an identity connection and a branch with a 3×3 depth-wise separable convolution with 256 output channels and dilation rate (6,3). In the end, there are a total of six parallel branches that are concatenated together to yield a tensor with 1536 channels. Finally, a 1×1 convolution with 256 output channels generates the output of the RDPC module. Please note that each of the aforementioned convolutions in the RDPC module is followed by an iABNsync and Leaky ReLU activation.

The RDPC module essentially integrates range-guided depth-wise atrous separable convolutions with fixed multi-dilation rates ones to generate features that cover a relatively large scale range in terms of receptive field in a dense manner. In summary, our proposed distance-dependent semantic head consists of two RDPC modules employed at $\times 32$ and $\times 16$ downsampling factor whose inputs are RP_{32}, R_{32} and RP_{16}, R_{16} respectively. It also utilizes two RLSFE modules for $\times 8$ and $\times 4$ downsampling factor with RP_8, R_8 and RP_4, R_4 as inputs. These modules are subsequently followed by the MC module [2] with bottom-up path augmentations. In the last step, we apply a 1×1 convolution with $N_{stuff+thing}$ output filters and upsample to the resolution of the input image. We train our semantic head with equally weighted per-pixel log-loss (L_{pp}) [43] and *Lovász-Softmax* loss (L_{LS}) [44] as

$$L_{semantic}(\Theta) = L_{pp} + L_{LS}, \tag{4}$$

We evaluate the performance of our proposed semantic head in the ablation study presented in Sec. IV-D5.

3) Instance Head: We adopt the Mask R-CNN [41] topology for the instance head and make certain modifications. We replace the batch normalization and ReLU activations with iABNsync and Leaky ReLU layers respectively. Fig. 2 shows the instance head depicted in purple blocks which consists of two stages. In the first stage, the Region Proposal Network (RPN) employs a fully convolutional network to generate object proposals and objectness scores for each output resolution of the RFPN module. The RPN is trained with the objectness score loss L_{os} [2] and object proposal loss L_{op} [2]

In the subsequent stage, ROI align extracts features by directly pooling from the n^{th} channel of the FPN encodings with a 14×14 spatial resolution bounded within the object proposals obtained in the previous stage. These extracted features are then fed to specialized bounding box regression, object classification and mask segmentation networks. The second stage is trained with the classification loss L_{cls} [2], bounding box loss L_{bbx} [2] and mask segmentation loss L_{mask} [2].

The overall loss of the instance segmentation head is the equally weighted summation of the aforementioned losses as

$$L_{instance} = L_{os} + L_{op} + L_{cls} + L_{bbx} + L_{mask}. \tag{5}$$

Note that the gradient from the losses L_{cls}, L_{bbx} and L_{mask} are allowed to flow only through the network backbone and not through the RPN.

C. Panoptic Fusion

We fuse the outputs of the semantic and instance heads using the heuristic proposed in [2] to yield the panoptic predictions in the projection domain. This heuristic enables us to adaptively fuse the predictions of both the heads which alleviates the inherent overlap problem. The instance head outputs a set of object instances comprising of class prediction, confidence score, bounding box, and mask logits for each instance. While, the semantic head outputs semantic logits of $N_{stuff} + N_{thing}$ channels. We first compute the mask logit ML_A for the object instances by applying a series of operations on the outputs of the instance head, consisting of thresholding, sorting, scaling, resizing, padding, and overlap filtering. Subsequently, we compute the mask logit ML_B for the corresponding object instances from the outputs of the semantic head by channel selection based on the class of the object instances and suppress the logits for that channel outside the instance bounding box. Finally, we adaptively fuse the two logits ML_A and ML_B to yield the fused mask logits FL of the instances as

$$FL = (\sigma(ML_A) + \sigma(ML_B)) \odot (ML_A + ML_B), \tag{6}$$

where $\odot$ is the Hadamard product, and $\sigma(\cdot)$ is the sigmoid function. In the next step, we concatenate the 'stuff' logits from the output of the semantic head and the fused logits, followed by applying the softmax function. Subsequently, we apply the argmax function along the channel dimension to obtain the intermediate panoptic prediction. To compute the final output, we replace the non-'thing' class predictions in the intermediate prediction with the 'stuff' class predictions of the semantic head while ignoring the classes that have an area smaller than a pre-defined area threshold min_{sa}.

D. Panoptic Periphery Loss

We propose the panoptic periphery loss function which exploits range information to refine the boundaries of the 'thing' class objects. By minimizing this loss, the boundary pixels of instances are adapted to maximize the range difference to the adjacent background pixels. This is motivated by the fact that there is typically a range gap at the borders of object instances. Consider the network provides a set of instances I of 'thing' class objects, and for each instance, we have its foreground and background pixels. Then for a given range image R, the panoptic periphery loss function is defined as

$$L_{refine} = -\frac{1}{|B|} \sum_{i \in I} \sum_{b \in B_i} [\max_{n \in N} (k_n * (r_b - r_n)^2)], \tag{7}$$

where $|B|$ is the total number of boundary points over all instances, B_i is the set of boundary pixels for instance i, N is the set of the four immediate neighbors of pixel location b, r_b and r_n are the range value at pixel location b and n, respectively. $k_n = 1$ for n being a background pixel and 0 otherwise. The negative sign ensures that the loss decreases when the range difference between boundary and background increases.

The overall loss L for training EfficientLPS is given by

$$L = L_{sem} + L_{instance} + L_{refine}, \tag{8}$$

where L_{sem} is the semantic head loss, $L_{instance}$ is the instance head loss and L_{refine} is the panoptic periphery loss for boundary refinement.

E. Back-projection

During the projection to point clouds, different points may get assigned to the same pixel in the projected image, which leads to the assignment of the same label to all overlapping points. Moreover, due to the downsampling operations in the network, the convolutions produce blurry outputs in the decoder which leads to leaking of the labels at the boundaries of the instances during back-projection to the 3D domain.

We use a k-nearest neighbor (kNN) based back-projection scheme [23] to mitigate these issues. For every point in the point cloud, the nearest k neighbors to the point vote for its semantic and instance labels. We obtain the labels of the selected neighbors from the corresponding pixels in the projected output prediction. To compute the nearest neighbors, we search for nearest neighbors within a pre-defined window around the pixel in the projected range image, out of which we select k nearest points based on the differences in their absolute range value. The entire post-processing is GPU optimized and is only employed during inference.

F. Psuedo Labeling

Due to the arduous task of annotating point-wise panoptic segmentation labels in point clouds and the effectiveness of pseudo labeling in the image domain, we explore its utility for LiDAR panoptic segmentation. We formulate a novel heuristic to improve the performance of EfficientLPS without requiring any additional manual human-annotations or model augmentations. We make the following assumptions while formulating the proposed heuristic. First, the unlabeled dataset is drawn from the same data distribution as the labeled dataset. Second, the model which is used to generate the labels for the unlabeled dataset and the model learning from these generated pseudo labels are the same, i.e., both the models have the same representation capacity. Finally, the precision and recall of the model generating the pseudo labels are tunable during inference time via adjustment of one or more hyperparameters.

The first assumption is dataset-specific to ensure high-quality pseudo labels, since the model trained on the same distribution as the unseen dataset tends to generalize better than the dataset from a different distribution. In our case, we choose the KITTI RAW dataset [42] as the unlabeled dataset since SemanticKITTI [16] which is the labeled dataset, is a subset of the former. We use the same EfficientLPS model to generate the pseudo labels with the goal of improving its performance, hence satisfying the second assumption. This assumption ensures that the label generating model can provide meaningful pseudo labels and the learning model has the representational capacity to capture it. To satisfy the third condition, the performance of EfficientLPS can be tuned using softmax confidence thresholding or by tuning the thresholds of the panoptic fusion module in EfficientLPS or a combination of both. This condition is required to design an effective regularization strategy for pseudo label generation in order to minimize the confirmation bias while learning.

We first train EfficientLPS on the labeled dataset and use this model to generate pseudo labels for the unlabelled dataset. We refer to this model as the Pseudo Label Generator (PLG) and the parameters that can be used to control the performance of the model as control parameters as a whole. In the next step, we use grid search to find the most optimal control parameter combination that maximizes the given ratio $(TP-FP)/TP$ until the PQ score is higher than the PQ_{cutoff} parameter on the validation dataset of the labeled dataset. Here, TP is the true positives, and FP is false positives that are computed over the validation set. PQ_{cutoff} is the minimum value of PQ to which the performance of PLG is allowed to drop. By maximizing the aforementioned ratio, we make the generated pseudo label to be more accurate by having relatively fewer false positives and higher true positives. Subsequently, we use the optimal control parameter setting to generate pseudo labels with PLG. As a post-processing step, we then discard all the instances with the number of points less than a pre-defined limit P_{limit} in the pseudo labels. This improves the quality of the generated pseudo labels by discarding incorrect predictions that were made due to the lack of sufficient points. We then train EfficientLPS from scratch on this pseudo labeled dataset, followed by fine-tuning the model on the labeled dataset to improve the overall performance. We comprehensively evaluate the performance of our proposed heuristic in the ablation study presented in Sec. IV-D7.

IV. Experimental Evaluation

In this section, we first briefly describe the datasets that we report the results on in Sec. IV-A, followed by the training protocol that we employ in Sec. IV-B and detailed comparisons as well as benchmarking results in Sec. IV-C. Subsequently, we present comprehensive ablation studies on the various proposed architectural components in EfficientLPS in Sec. IV-D and detailed qualitative analysis in Sec. IV-E. In all the experiments presented in this section, we use the PQ metric [1] as the main evaluation criteria as defined by the benchmarks. For completeness, we also report the mean Intersection-over-Union (mIoU), Segmentation Quality (SQ), and Recognition Quality (RQ), as well as the aforementioned metrics for the 'stuff' and 'thing' classes separately.

A. Dataset

We evaluate the performance of our approach on two datasets that were collected with LiDARs of different resolutions to test the generalization of our network. The first dataset is SemanticKITTI [16] which contains sequences (00-21) consisting of point-wise semantic and temporally consistent instance labels for the 43,552 LiDAR scans. The dataset is split into 20,351 scans (sequences 00-10) that are available for training, while the rest of the sequences (11-21) are withheld and used by the benchmarking server for evaluations. The dataset provides labels of 28 classes out of which 19 classes are considered for evaluation.

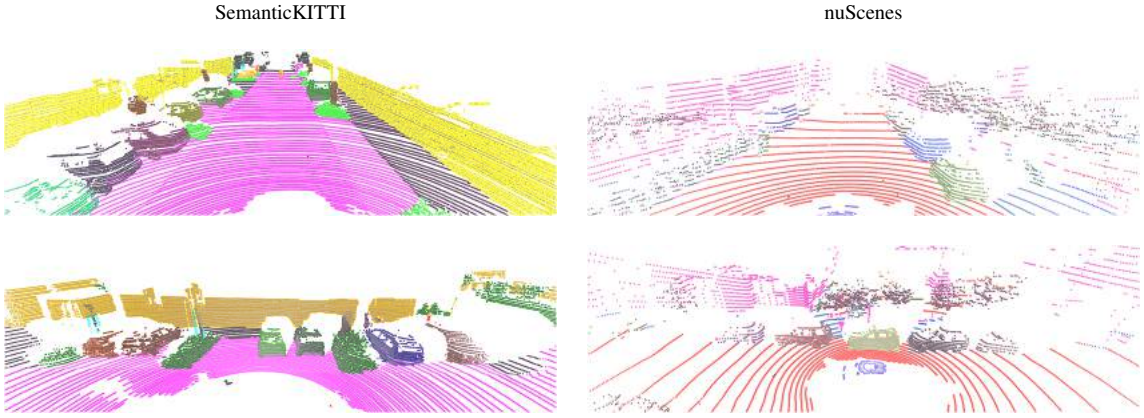


Fig. 5: Example groundtruth visualization from SemanticKITTI and nuScenes datasets. The SemanticKITTI dataset was collected using a 64-beam LiDAR, hence it provides a fairly dense representation of the environment in comparison to the nuScenes dataset which was collected using a 32-beam LiDAR.

For the second dataset, we use nuScenes [17] which is a large-scale dataset for autonomous driving that consists of point-wise semantic labels for 32 classes out of which 16 are considered for evaluation. The dataset contains 1000 scenes, out of which 700 scenes are used for the training set, 150 scenes for the validation set, and the rest 150 scenes for the test set. The dataset itself does not provide any panoptic segmentation labels but it contains 3D bounding box annotations for the 'thing' object classes. To obtain the panoptic segmentation labels, we extract the points that lie inside the 'thing' class bounding boxes and assign unique instance-ids. The SemanticKITTI dataset ignores object instances that have less than 50 points. We follow a similar scheme for the nuScenes dataset and ignore object instances that have less than 15 points, since nuScenes is roughly $3\times$ sparser than SemanticKITTI. While the vertical field of view of the LiDAR that was used in nuScenes is slightly higher than SemanticKITTI, the number of vertical lines is only 32 compared to 64 in SemanticKITTI. Fig. 5 shows example groundtruth panoptic segmentation of point clouds from both these datasets.

B. Training Protocol

We train our network on projected point clouds of 4096×256 resolution. We use bilinear interpolation on the projections obtained from scan unfolding and nearest neighbor interpolation on the ground truth point clouds. We initialize the main encoder of our EfficientLPS architecture with weights from the EfficientNet-B5 model pre-trained on the ImageNet [46] dataset. Furthermore, we initialize the weights of the iABNsync layers to 1 and use Xavier initialization for the other layers. We also employ zero constant initialization for the biases and set the slope of Leaky ReLU to 0.01. We use the same hyperparameters for the instance head as Mask R-CNN [41]. For the panoptic fusion module, we set $c_t = 0.5$, $o_t = 0.5$ and $min_{sa} = 128$.

We use Stochastic Gradient Descent (SGD) with a momentum of 0.9 for training our models. We employ a multi-step learning rate schedule, where we start with an initial base learning rate of 0.07 and reduce it by a factor of 10 after 16,000 and 22,000 iterations. We train our models for a total of 25,000 iterations with a batch size of 16 on 8 NVIDIA TITAN RTX GPUs. We first train the model on the pseudo labeled dataset until the first reduction in learning rate by a factor of 10, followed by continuing the training on the labeled dataset.

C. Comparison with the State-of-the-Art

In this section, we evaluate the performance of EfficientLPS and compare with state-of-the-art methods for panoptic segmentation of LiDAR point clouds.

SemanticKITTI: We compare with three state-of-the-art methods, LPSAD [6], PanopticTrackNet [10], and Panoster [5], as well as the two baselines, (RangeNet++ [23] + PointPillars [45]), and (KPConv [22] + PointPillars [45]). Tab. I presents the results on the SemanticKITTI test set which was evaluated by the benchmark server. Our proposed EfficientLPS achieves a PQ score of 57.4%, which is an improvement of 4.7% over the previous state of the art Panoster. EfficientLPS also outperforms all the existing methods in all the metrics and sets the new state-of-the-art on this benchmark. The higher overall SQ score of EfficientLPS can be primarily attributed to the proposed panoptic periphery loss function, which improves the segmentation quality of 'thing' class objects by refining their boundaries. This is evident from the increase of 4.5% in SQ^{th} score compared to Panoster. Moreover, the proposed distance-dependent semantic head enables the recognition of objects in a scale-invariant manner by incorporating range encoded information to achieve a higher recognition quality, especially for the 'stuff' classes. This yields an improvement of 6.4% in the RQ^{st} score, which enables it to achieve the best overall RQ score of 68.7%. Additionally, the backbone of EfficientLPS equipped with the proposed PCM module which models the geometric transformations of different objects and the RFPN which learns spatially consistent features, contributes to the improvement of 3.7% in PQ^{Th}, 5.4% in PQ^{St} and 1.5% in mIoU scores.

In Tab. II, we present a comparison of the class-wise PQ scores on the SemanticKITTI test set. EfficientLPS achieves the highest PQ score for most of the 'stuff' class objects, with the exception of *trunk* and *pole* classes which are outperformed by Panoster. This shows that incorporating range encoded features into the semantic head helps achieve better overall semantic

TABLE I: Comparison of LiDAR panoptic segmentation performance on SemanticKITTI test set. All scores are in [%].

Method	PQ	PQ†	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}	mIoU
RangeNet++ [23] + PointPillars [45]	37.1	45.9	75.9	47.0	20.2	75.2	25.2	49.3	76.5	62.8	52.4
KPConv [22] + PointPillars [45]	44.5	52.5	80.0	54.4	32.7	81.5	38.7	53.1	79.0	65.9	58.8
LPSAD [6]	38.0	47.0	76.5	48.2	25.6	76.8	31.8	47.1	76.2	60.1	50.9
PanopticTrackNet [10]	43.1	50.7	78.8	53.9	28.6	80.4	35.5	53.6	77.7	67.3	52.6
Panoster [5]	52.7	59.9	80.7	64.1	49.4	83.3	58.5	55.1	78.8	68.2	59.9
EfficientLPS (ours)	**57.4**	**63.2**	**83.0**	**68.7**	**53.1**	**87.8**	**60.5**	**60.5**	**79.5**	**74.6**	**61.4**

TABLE II: Class-wise PQ scores on SemanticKITTI test set. R.Net, P.P, KPC refer to RangeNet++, Point Pillars, KPConv respectively. All scores are in [%].

Method	car	truck	bicycle	motorcycle	other vehicle	person	bicyclist	motorcyclist	road	sidewalk	parking	other ground	building	vegetation	trunk	terrain	fence	pole	traffic sign	PQ
R.Net [23]+ P.P. [45]	66.9	6.7	3.1	16.2	8.8	14.6	31.8	13.5	90.6	63.2	41.3	6.7	79.2	71.2	34.6	37.4	38.2	32.8	47.4	37.1
KPC [22] + P.P. [45]	72.5	17.2	9.2	30.8	19.6	29.9	59.4	22.8	84.6	60.1	34.1	8.8	80.7	77.6	53.9	42.2	49.0	46.2	46.8	44.5
LPSAD [6]	76.5	7.1	6.1	23.9	14.8	29.4	29.7	17.2	90.4	60.1	34.6	5.8	76.0	69.5	30.3	36.8	37.3	31.3	45.8	38.0
PanopticTrackNet [10]	70.8	14.4	17.8	20.9	27.4	34.2	35.4	7.9	91.2	66.1	50.3	10.5	81.8	75.9	42.0	44.3	42.9	33.4	51.1	43.1
Panoster [5]	84.0	18.5	36.4	44.7	30.1	61.1	**69.2**	**51.1**	90.2	62.5	34.5	6.1	82.0	77.7	**55.7**	41.2	48.0	**48.9**	59.8	52.7
EfficientLPS (ours)	**85.7**	**30.3**	**37.2**	**47.7**	**43.2**	**70.1**	66.0	44.7	**91.1**	**71.1**	**55.3**	**16.3**	**87.9**	**80.6**	52.4	**47.1**	**53.0**	48.8	**61.6**	**57.4**

TABLE III: Comparison of LiDAR panoptic segmentation performance on SemanticKITTI validation set. All scores are in [%].

Method	PQ	PQ†	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}	mIoU
RangeNet++ [23] + PointPillars [45]	36.5	-	73.0	44.9	19.6	69.2	24.9	47.1	75.8	59.4	52.8
KPConv [22] + PointPillars [45]	41.1	-	74.3	50.3	28.9	69.8	33.1	50.1	77.6	62.8	56.6
LPSAD [6]	36.5	46.1	-	-	-	-	28.2	-	-	-	50.7
PanopticTrackNet [10]	40.0	-	73.0	48.3	29.9	76.8	33.6	47.4	70.3	59.1	53.8
Panoster [5]	55.6	-	79.9	66.8	56.6	-	65.8	-	-	-	61.1
EfficientLPS (ours)	**59.2**	**65.1**	**75.0**	**69.8**	**58.0**	**78.0**	**68.2**	**60.9**	**72.8**	**71.0**	**64.9**

segmentation performance. The resulting spatial awareness is significantly beneficial for the *other-ground* class, which every method struggles to classify due to the presence of the more dominating *road* and *sidewalk* classes. We also observe a similar effect with the *parking* class. EfficientLPS achieves an improvement in the PQ score by 20.8% for *parking* and 10.2% for *other-ground* in comparison to Panoster. EfficientLPS also achieves the best performance for all 'thing' class objects, with the exception of *motocyclist* and *bicyclist* classes. This is due to the fact that both these classes share almost the same properties and they are relatively small objects which are represented by only a few points. These objects also always coexists with other classes such as *bicycle* and *motorcycle*, and are adversely affected during the projection of point cloud, rendering them very close to other objects. Hence, the point-based backbone KPConv and the clustering used by Panoster provides an advantage in this case. Nevertheless, EfficientLPS outperforms the other methods in the PQ score for all the other classes with large margins of 12.1% for *other vehicles*, 11.8% for *truck* and 9% for *person* classes. Therefore, the overall state-of-the-art performance obtained from our network is not just due to the improvement in scores for a particular object class, rather is a result of collective improvement across different semantic object classes with a variety of structural properties.

Tab. III presents the results on the SemanticKITTI validation set. '-' indicates that the corresponding methods do not report the specific metric. We observe a similar trend here as the test set where EfficientLPS outperforms all the other methods in all of the metrics. It achieves a PQ score of 59.2% and 75.0%, 69.8% and 64.9% for SQ, RQ and mIoU scores, respectively.

nuScenes: As there are no existing panoptic segmentation methods that have been benchmarked on the nuScenes dataset, we trained two baseline methods by combining individual semantic and instance segmentation models, namely (KPConv [22] + Mask R-CNN [41]) and (RangeNet++ [23] + Mask R-CNN [41]), as well as the established PanopticTrackNet [10] which is a projection-based panoptic segmentation model for LiDAR point clouds. For all these models, we use the original code provided by the authors and optimized the hyperparameters to the best of our ability. We trained Mask R-CNN on the projected point cloud images using the approach described in Sec. III-A and project the predictions back to the 3D domain using the post-processing described in Sec. III-E. We also make these trained baselines publicly available.

Tab. IV presents the results on the nuScenes validation set. Among the baselines, (KPConv + Mask R-CNN) achieves the highest PQ score of 51.5%, closely followed by PanopticTrackNet which achieves a PQ score of 51.3%. (KPConv + Mask R-CNN) achieves a higher PQ^{St} score than PanopticTrackNet which demonstrates that ability of point-based methods to perform better at semantic segmentation. During the projection

TABLE IV: Comparison of LiDAR panoptic segmentation performance results on nuScenes validation set. All scores are in [%].

Method	PQ	PQ†	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}	mIoU
RangeNet++ [23] + Mask R-CNN [41]	46.6	52.6	79.5	58.4	39.9	80.5	52.1	57.8	77.9	68.8	56.6
PanopticTrackNet [10]	51.4	56.2	80.2	63.3	45.8	81.4	55.9	60.4	78.3	75.5	58.0
KPConv [22] + Mask R-CNN [41]	51.5	56.8	80.3	63.5	44.6	81.3	53.9	62.9	78.8	79.4	58.9
EfficientLPS (ours)	**62.0**	**65.6**	**83.4**	**73.9**	**56.8**	**83.2**	**68.0**	**70.6**	**83.8**	**83.6**	**65.6**

TABLE V: Class-wise results on nuScenes validation set. All scores are in [%].

Method	barrier	bicycle	bus	car	cvehicle	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other flat	sidewalk	terrain	man-made	vegetation	PQ
RangeNet++ [23] + Mask R-CNN [41]	40.3	25.7	51.7	62.5	14.6	48.3	38.8	41.8	32.7	43.0	77.1	41.5	59.2	42.1	58.9	67.9	46.6
PanopticTrackNet [10]	47.1	32.9	57.9	66.3	22.8	51.1	42.8	46.8	38.9	51.0	81.5	42.3	61.8	45.1	60.9	70.9	51.4
KPConv [22] + Mask R-CNN [41]	46.7	31.5	56.8	65.7	21.9	50.4	41.6	44.9	37.6	49.1	83.5	43.1	63.5	48.6	73.9	71.5	51.5
EfficientLPS (ours)	**56.8**	**37.8**	**52.4**	**75.6**	**32.1**	**65.1**	**74.9**	**73.5**	**49.9**	**49.7**	**95.2**	**43.9**	**67.5**	**52.8**	**81.8**	**82.4**	**62.0**

of points, distant objects in the 3D domain end up close to each other in the projected 2D domain. Hence, projection based methods that solely operate in the 2D domain find it hard to distinguish between them. The proposed distance-dependent semantic head in EfficientLPS exploits range encoded features and achieves an improvement of 7.7% in the PQ^{St} score over (KPConv + Mask R-CNN). On the other hand, the top-down architecture of PanopticTrackNet which has a dedicated instance segmentation head, achieves a better performance in segmenting instances of 'thing' class objects, thereby achieving a higher PQ^{Th} score than (KPConv + Mask R-CNN). The RFPN moodule along with proposed panoptic periphery loss that we use for training EfficientLPS enables it to achieve an improvement of 12.2% in the PQ^{Th} score over PanopticTrackNet. Overall, EfficientLPS achieves a PQ score of 62.0%, SQ score of 83.4%, RQ score of 73.9%, and mIoU of 65.6%, outperforming all the methods in each of the metrics and sets the new state of the art on the nuScenes dataset. The consistent state-of-the-art performance, even on the sparse nuScenes dataset demonstrates the effectiveness and the generalization ability of our proposed modules in tackling different challenges such as distance-dependent sparsity, severe occlusions, large scale-variations, and re-projection errors.

In Tab. V, we present the class-wise PQ scores on the nuScenes validation set. The sparsity of the point clouds especially affects segmentation of smaller objects such as *pedestrian*, *bicycle*, *motorcycle*, *barrier* and *traffic cone*. The operation of the proposed proximity convolution module is independent of the sparsity of the point cloud, and models the geometric transformations of objects, only based on the proximity between the points. This is one of the key factors that enables EfficientLPS to achieve a higher performance for all 'thing' object classes, with an improvement of 12.9% for *pedestrian*, 8.4% for *bicycle*, and 13.2% for *motorcycle* compared to PanopticTrackNet. The sparse nature of point clouds also makes it harder to recognize and distinguish between different semantic object classes, especially for the object classes that often appear close to each other such as *driveable surface*, *sidewalk* and *terrain*. The explicit incorporation of range encoded information in RFPN and the distance-dependent semantic head introduces spatial consistency in the learned features. This especially helps 'stuff' object classes, leading to an PQ improvement by 14.0% for *driveable surface*, 7.2% for *terrain*, and 7.0% for *sidewalk* compared to (KPConv + Mask R-CNN). Overall, the superior performance obtained for all classes demonstrates the efficacy of EfficientLPS for accurately segmenting different semantic object classes, even for very sparse point clouds.

D. Ablation Studies

In this section, we present extensive ablation studies on the various proposed architectural components in EfficientLPS to warrant our design choices and study the impact of each module on the overall performance of our architecture. We begin with a comprehensive analysis on the EfficientLPS architecture that shows the effect of our proposed proximity input layer, range-aware FPN, distance-dependent semantic head, panoptic periphery loss, and pseudo labeling framework, on the overall performance of our network. We then study the influence of parameters for the search grid and kernel size on the performance of the proximity convolution module. Subsequently, we study the effect of the REN and the different approaches to incorporate it with the 2-way FPN in our proposed range-aware FPN. We then present detailed analysis of the distance-dependent semantic head to show the impact of different architectural design choices. Furthermore, we quantitatively show the boundary refinement achieved with the panoptic periphery loss function using the border IoU metric and compare the performance of the proposed pseudo labeling heuristic to a naive heuristic. Finally, to demonstrate the generalization ability of our proposed modules, we present results by incorporating our architectural components into other well-known top-down panoptic segmentation networks in a straight forward manner.

1) Comprehensive Analysis of EfficientLPS: In this section, we study the improvement due to the incorporation of various architectural components proposed in EfficientLPS. The results of this experiment are shown in Tab. VI. Fig. 6 also shows

TABLE VI: Ablative analysis on the various proposed architectural components in EfficientLPS. The model variants consists of the marked (✓) modules in their respective columns with Per. Loss denoting the Panoptic and P. Labels denoting Labels. The results are reported on the SemanticKITTI validation set.

Model Variant	PCM	RFPN	RDPC	Panoptic Periphery Loss	Pseudo Labels	PQ (%)	SQ (%)	RQ (%)	PQ^{St} (%)	PQ^{Th} (%)	mIoU (%)	Runtime (ms)
M1	✗	✗	✗	✗	✗	53.0	73.1	63.9	53.3	52.5	58.6	153.84
M2	✓	✗	✗	✗	✗	53.9	73.9	64.4	54.3	53.3	59.8	175.43
M3	✓	✓	✗	✗	✗	55.7	74.4	65.8	55.7	55.8	60.9	192.31
M4	✓	✓	✓	✗	✗	56.6	75.0	66.8	56.4	56.9	62.4	212.76
M5	✓	✓	✓	✓	✗	57.4	75.0	67.6	56.5	58.7	62.5	212.76
M6	✓	✓	✓	✓	✓	**59.2**	**75.0**	**69.8**	**58.0**	**60.9**	**64.9**	212.76

the improvement in performance for each of the models described in this section. We begin with the base model M1 that consists of EfficientNet-B5 followed by the 2-way FPN as the shared backbone with the semantic head from [2] and a Mask R-CNN based instance head. Subsequently, the logits from both heads are fused in the panoptic fusion module at inference time. We train this model with an input resolution of 4096×256 as it allows the anchor scales defined in [41] to be used directly. We use scan unfolding for point cloud projection and the kNN-based post-processing for re-projecting the predicted labels back to the 3D domain. Further, we employ the *Lovász-Softmax* loss in addition to weight per-pixel log loss while training. This model M1 achieves a *PQ* score of 53.0% with PQ^{st} and PQ^{th} scores of 53.3% and 52.5% respectively. This model has a runtime of 153.84 ms. To compute the runtime, we use a single NVIDIA Titan RTX GPU and an Intel Xenon@2.20GHz CPU. We average over 1000 runs on the same LiDAR point cloud. In the case of parallel components in the architecture, maximum runtime among all the components contribute to the total runtime. In the subsequent model M2, we incorporate our proposed proximity convolution module which achieves a *PQ* score of 53.9% and an *mIoU* of 59.8%, which constitutes to an improvement 0.9% and 1.2% respectively over the model M1. This improvement can be attributed to the enhanced transformation modeling capability imparted due to the incorporation of the proximity convolution module, as shown qualitatively in Fig. 7 (a). The model M1 fails to recognize and segment far-away objects (motorcyclist in the figure) as the points become more sparse with increasing distance. In contrast, the model M2 is able to accurately capture the distant objects. Additionally, M2 has a runtime of 175.43 ms.

The model M3 builds upon the model M2 by incorporating the range-aware FPN. This model achieves an improvement of 1.8% in *PQ* score that constitutes to an improvement of 1.4% in the PQ^{St} and 2.5% in the PQ^{Th} scores while having a runtime of 192.31 ms. This performance improvement can be attributed to the distance-aware reinforcement of coherently aggregated fine and contextual features that enables accurate segmentation of small occluded objects. In Fig. 7 (b), the car is occluded by the vegetation. The model M3 is able to successfully segment the occluded classes due to the incorporation of range-aware features, whereas model M2 falsely predicts both the occluded and the occluder objects as either the foreground or the background class. The next model M4 which incorporates our distance-dependent semantic head into model M3 achieves an improvement of 1.5% in *mIoU* and 0.9% in the *PQ* score

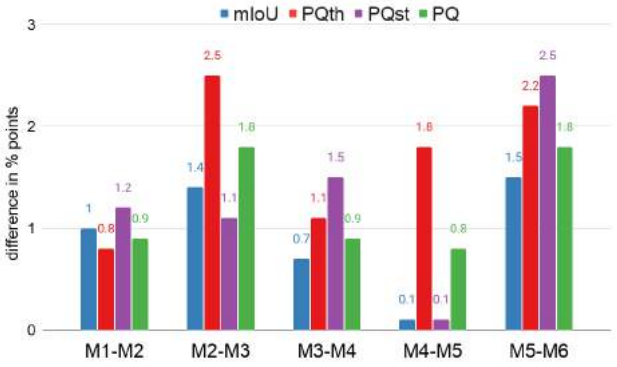


Fig. 6: Ablation analysis on the percent change in the metrics for the incorporation of various architectural components shown in Tab. VI. Where M(n)-M(n+1) denotes the % improvement in the metrics that model M(n+1) achieves over model M(n).

with runtime of 212.76 ms. In order to visualize the qualitative improvement of model M4 over model M3, we present an example in Fig. 7 (c) where the model M4 segments the *wall* class much more accurately than the model M3. Our proposed semantic head effectively handles the scale variation depicted in the examples. It benefits from the adaptable receptive field which results from combining fixed multi-dilation rate convolutions with range-guided depth-wise atrous separable convolutions.

Subsequently, the model M5 extends model M4 by using our proposed panoptic periphery loss during training. This model achieves PQ^{th} score of 58.7% which is a substantial gain of 1.8% over model M4. The model *M*5 achieves a overall *PQ* and *mIoU* scores of 57.4% and 62.5% respectively. Fig. 7 (d) shows a qualitative comparison of the boundary refinement improvement. We further improve the performance of our network in model M6 by employing our pseudo labeling framework as described in Sec. IV-D7. Training with an unlabeled dataset in combination with the labeled dataset leads to a large improvement of 1.5% in *mIoU* and 1.8% in *PQ* scores. The final model M6 achieves state of the art performance on SemanticKITTI, with a *PQ* score of 59.2%. Moreover, we do not observe any changes in runtime of models M5 and M6 since there is no additional architectural overhead in these models. Model M6 is essentially EfficientLPS with a runtime of 212.76 ms. In the following sections, we further analyze the individual architectural components of the *M*5 model in more detail.

2) Influence of Proximity Convolution Parameters: The proposed Proximity Convolution (PC) is the core of the

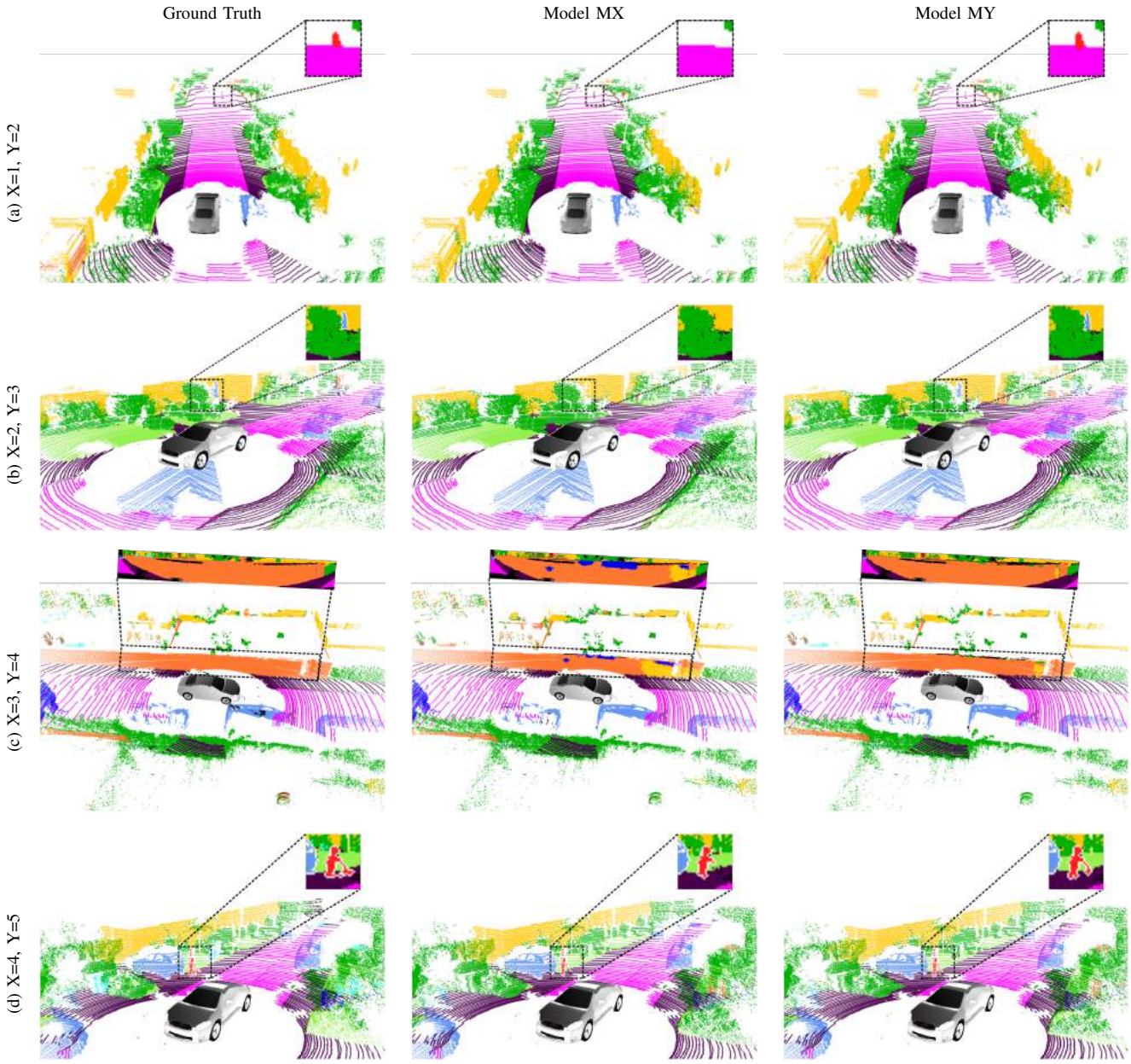


Fig. 7: Qualitative comparison of different models described in Tab. VI. The model numbers denoted by X and Y are shown in the first column. In Fig (a), the incorporation of the proximity convolution module in model $M2$ enables accurate segmentation of small and distant objects such as the motorcyclist in the image. Fig. (b) compares model $M2$ with $M3$ which incorporates the Range-aware FPN, enabling it to segment occluded objects such as the van in blue which is occluded by the surrounding vegetation. Fig. (c) shows the performance improvement due to the new distance guided semantic head incorporated into model $M4$ which enables consistent segmentation of the sidewall shown in orange. Lastly, Fig. (d) shows the improvement due to the panoptic periphery loss used in training model $M5$ which enables segmenting the entire instance of the human shown in red. The enlarged images are taken from the corresponding projected range image for better visualization (PDF best viewed at $\times 4$ zoom scale).

TABLE VII: Effect of the search area and kernel size on the performance of the proximity convolution module. All scores are in [%].

Model	Search grid	Kernel size	PQ	PQ^{St}	PQ^{Th}	mIoU
$M5_1$	-	3x3	56.6	55.7	57.9	61.8
$M5_2$	5x5	3x3	**57.4**	56.5	**58.7**	**62.5**
$M5_3$	7x7	3x3	57.1	**56.8**	57.4	**62.5**
$M5_4$	7x7	5x5	**57.4**	56.5	58.5	62.2

proximity convolution module. It employs the kNN algorithm to find the k closest neighbors of each pixel in the projected image where value of k is the product of kernel size dimensions of the proximity convolution. This algorithm is also parameterized by the search grid size that defines a grid around a pixel within which the algorithm performs the search. Tab. VII presents the results of the experiment where we vary the search grid and kernel sizes in the M5 model from Sec. IV-D1. In the first model $M5_1$, we employ the standard convolution with a kernel size of 3×3 in the proximity convolution module of model M5. This model achieves a PQ score of 56.6% and a $mIoU$ of 61.8. The second model $M5_2$ uses the PC with a search grid

TABLE VIII: Influence of the range encoder on the overall panoptic segmentation performance. All scores are in [%].

	PQ	PQ^{St}	PQ^{Th}	SQ	RQ	mIoU
Downsampled	54.8	54.6	55.1	74.1	65.2	59.5
Range-Encoded	**57.4**	**56.5**	**58.7**	**75.0**	**67.6**	**62.5**

of 5×5 and a kernel size of 3×3. The model $M5_2$ with the proximity convolution achieves an improvement of 0.8% in the *PQ* score. Here, more than one-third of the search space can be captured by the convolution weights.

In model $M5_3$, we increase the search grid of the PC from 5×5 to 7×7 while keeping the kernel size the same as model $M5_2$. Here, almost one-fifth of the search space can be captured by the convolution weights. Hence, although the search area increases, there are not enough convolution weights to efficiently capture all the close neighbors. This leads to a decrease in performance of 0.3% in the *PQ* score. The model $M5_4$ increases the kernel size of the PC in model $M5_3$ to 5×5. This model performs similar to the model $M5_2$, although the convolution weights can capture half of the search space. This result indicates that predominantly the top nine neighbors computed in model $M5_2$ were adequate to capture the underlying transformations. Therefore, in the proposed EfficientLPS architecture, we employ the search grid size of 5×5 and a proximity convolution kernel size of 3×3.

3) Evaluation of Range Encoder Network: The Range Encoder Network (REN) is a small CNN based on the EfficientNet [12] topology which encodes range information. We use compound scaling coefficients of 0.1 for the width, 0.1 for the depth, and 224 for the resolution. Our proposed range-aware FPN and the distance-dependent semantic head both employ the multi-scale features encoded by the REN. The multi-scale features of the REN reinforce coherently aggregated fine and contextual output features of the 2-way FPN with spatial awareness. The dilation offsets for the range-guided depth-wise atrous separable convolutions that are employed at different scales are computed in the distance-dependent semantic head. In this experiment, we show the importance of the REN features compared to the direct incorporation of the range channel in the EfficientLPS architecture. Tab. VIII presents the results from this experiment.

We compare the performance of two models in this experiment. The first model referred to as downsampled, removes the REN from the model M5 and directly downsamples the range channel with downsampling factors $\times 4$, $\times 8$, $\times 16$, and $\times 32$. These downsampled versions of the range channels are then fed to the relevant modules. This model achieves a *PQ* score of 54.8% and a *mIoU* of 59.5%. The second model is essentially the model M5 that already has the REN as part of the network, and we refer to it as the range-encoded model. This model achieves a *PQ* score of 57.4% and a *mIoU* of 62.5%. The range-encoded model achieves an improvement in *PQ* and *mIoU* scores of 2.6% and 3.0% respectively, demonstrating that the direct downsampling is not sufficient to propagate the spatial information to the main network and a dedicated encoder such as the REN is required.

TABLE IX: Evaluation of different methods to incorporate learned range features in the RFPN. All scores are in [%].

	PQ	PQ^{St}	PQ^{Th}	SQ	RQ	mIoU
Additive	53.9	53.7	54.1	74.0	64.6	60.8
Concatenative	56.5	55.6	57.8	74.8	66.4	61.6
Fusion	**57.4**	**56.5**	**58.7**	**75.0**	**67.6**	**62.5**

TABLE X: Evaluation of different variations of the proposed semantic head. All scores are in [%].

RLSFE	RDPC	PQ	PQ^{St}	PQ^{Th}	SQ	RQ	mIoU
✗	✗	56.2	55.5	57.1	74.7	66.8	61.1
✓	✗	56.6	56.0	57.4	74.7	67.0	61.4
✗	✓	57.1	56.2	58.4	74.9	67.5	62.1
✓	✓	**57.4**	**56.5**	**58.7**	**75.0**	**67.6**	**62.5**

4) Evaluation of Range-Aware FPN: There are two main components of our proposed range-aware FPN: the REN and the fusion to incorporate REN features into the FPN. In the previous section, we discuss the importance of the REN for the functioning of range-aware FPN. In this section, we evaluate different fusion methods to incorporate REN features into the FPN. Tab. IX shows the results from this experiment on the M5 model from Sec. IV-D1. We identify three major techniques to fuse features from multiple network streams: Addition, Concatenation, and Feature fusion. For the additive model, the REN features at each scale are expanded to 256 channels and are summed with the corresponding output of the 2-way FPN to yield the output of the range-aware FPN. In the case of the concatenative model, the outputs of REN and 2-way FPN are concatenated at each scale, followed by two sequential 3×3 convolution layers to yield the final output of the range-aware FPN. Each of the convolution layers is followed by an iABNsync and leaky ReLU activation. For the feature fusion model, we employ the mechanism from Eq. (3).

The additive model yields the lowest performance with a *PQ* score of 53.9% as it treats the features from the REN and the main network equally, even though the representational capacity of the two networks are significantly different. Instead of supporting the main network to capture richer representation, the REN reduces the quality of the overall features. The concatenative model yields a better performance with a *PQ* score of 56.5%. This model provides the network with the flexibility of determining which features are more significant to impart the required spatial awareness and hence achieves a substantial improvement over the additive model. Nevertheless, the fusion model with a *PQ* score of 57.4% outperforms the other fusion methods. This model further makes the fusion more flexible by extending the concatenative model with additional selectivity control which improves the performance. We expect that more adaptive fusion techniques [47]–[49] can further improve the performance of our range-aware FPN.

5) Evaluation of Different Semantic Head Topologies: The semantic head incorporates our proposed range-guided variants of the dense prediction cells (DPC) [15] and the large scale feature extractor (LSFE) [2] modules with the bottom-up path connections. In this section, we compare the performance of the original versions of each of these two modules with their

TABLE XI: Evaluation of the panoptic periphery loss using the border IoU metric for the person (Pe), car (Ca), bicyclist (Bi) and other vehicle (Ov) classes. All scores are in [%].

Model	Per. loss	$bIoU^{Pe}$	$bIoU^{Ca}$	$bIoU^{Bi}$	$bIoU^{Ov}$	PQ^{Th}
M4	✗	66.8	89.6	74.3	40.9	56.9
M5	✓	**70.0**	**90.3**	**75.1**	**47.5**	**58.7**

proposed range-guided counterparts. Tab. X presents the results of this experiments. We train model M5 from Sec. IV-D1 with the semantic head consisting of the original DPC and LSFE modules. This model attain a PQ score of 56.2%, PQ_{st} score of 55.5%, PQ_{th} score of 57.1%, and an $mIoU$ of 61.1%. For the second model, we replace the original LSFE module in the first model with our range-guided LSFE (RLSFE) module. This model achieves an improvement of 0.4%, 0.5%, 0.3%, and 0.3% in the PQ, PQ_{st}, PQ_{th} and $mIoU$ scores respectively, compared to the first model.

We observe a similar improvement in performance of the third model that employs our proposed range-guided DPC module (RDPC). This model achieves an improvement of 0.9% in the PQ score and 1.3% in the $mIoU$ compared to the first model. Interestingly, the improvement in the PQ^{th} score is higher than the improvement in the PQ^{st} score which is 0.7% and 1.3% respectively. This indicates a larger improvement in semantic segmentation of 'thing' classes with the denser and relatively larger receptive field of RDPC compared to 'stuff' classes. Finally, we train the first model with both RLSFE and RDPC modules which achieves a further improvement with a PQ score of 57.4% and an $mIoU$ of 62.5%. This experiment demonstrates that our semantic head effectively learns scale-invariant features due to its distance-dependent receptive fields.

6) Influence of Panoptic Periphery Loss: We evaluate the boundary refinement performance due to the panoptic periphery loss using the *border-IoU* metric. The *border-IoU* metric enables us to analyze the bleeding or shadowing effects that are observed while projecting the predicted labels back to point clouds. Our proposed loss exploits spatial information to refine the boundaries of panoptic 'thing' class objects. Tab. XI presents the results using the *border-IoU* metric for the top four 'thing' classes that achieve the highest improvement, namely person, car, bicyclist, and other-vehicle.

We compute the *border-IoU* for a border width of 2 pixels for the model M4 and the model M5 from Sec. IV-D1 which are trained with and without the panoptic periphery loss. We observe the highest improvement of 6.6% in *bIoU* for the other-vehicle class, followed by 3.2% improvement in *bIoU* for the person class. We also observe an improvement for the car and bicyclist class. The other-vehicle class consists of different types of vehicles such as a trailer, bus, and train, which makes it challenging to accurately segment the boundaries. Similarly, the person class is often only represented with few pixels for extended body parts which again makes it challenging to accurately segment these boundaries. The inaccuracies in the border segmentation are further exacerbated while projecting back to the 3D domain. Hence, explicitly focusing on refining the boundaries using our proposed periphery loss yields substantial improvement.

TABLE XII: Evaluation of the proposed heuristic for pseudo labeling. All scores are in [%].

Heuristic	PQ	PQ^{St}	PQ^{Th}	SQ	RQ	mIoU
✗	57.4	56.5	58.7	**75.0**	67.6	62.5
M_{naive}	58.0	57.2	59.1	**75.0**	69.3	64.0
M_{ours}	**59.2**	**58.0**	**60.9**	**75.0**	**69.8**	**64.9**

TABLE XIII: Evaluation of the generalization ability of our proposed architectural components. All scores are in [%].

Model	PQ	PQ^{St}	PQ^{Th}	SQ	RQ	mIoU
Panoptic $FPN_{vanilla}$	48.7	49.6	47.5	71.7	63.0	55.2
$Seamless_{vanilla}$	50.6	50.7	50.5	72.8	63.6	56.9
Panoptic FPN_{ours}	50.8	50.9	50.6	72.5	63.4	56.3
$Seamless_{ours}$	53.4	52.9	54.1	73.1	64.2	58.4

7) Evaluation of Pseudo Labeling: In this section, we evaluate the performance of our proposed heuristic for generating pseudo labels from unlabeled datasets. We first define two pseudo label generators, one for generating naive pseudo labels (PLG_N) and another using our proposed heuristic (PLG_O). PLG_N is the M5 model from Sec. IV-D1 which achieves the highest PQ score on the SemanticKITTI validation dataset. Whereas, PLG_O is the M5 models with its hyperparameters set such that it maximizes the ratio $(TP-FP)/TP$ on the validation dataset. The hyperparameters that we tune via grid search are the overlap threshold, minimum stuff area, confidence threshold, and softmax threshold, for the panoptic fusion module; NMS IoU, and score threshold, for the RCNN; the number of proposals for the RPN. For naive labeling of the unlabeled dataset, we use the output of PLG_N as the final pseudo labels. For our heuristic-based labeling, we employ the post-processing technique described in Sec. IV-D7 on the predictions of PLG_O to obtain the final pseudo labels. Fig. 8 shows example pseudo labels generated from both the methods. In example 1, the train is misclassified as a truck in the output from the naive approach, whereas it is classified as unlabeled in the output from heuristic-based method. In example 2, the naive approach misclassifies one of the two persons as a bicyclist, and our heuristic-based approach classifies both the people as unlabeled. These examples demonstrate that our heuristic-based approach assigns objects as unlabeled than risking misclassification wherever possible.

Tab. XII presents the quantitative results from this experiment. Both M_{naive} and M_{ours} models have the same architecture as the M5 model from Sec. IV-D1 and are trained using the training scheme described in Sec. IV-D7. The difference between the two models is that M_{naive} is trained using the naive approach, whereas M_{ours} is trained using our heuristic-based technique. We observe that both the models achieve a higher PQ score than the model trained without the pseudo labeled dataset. Moreover, our M_{ours} achieves the highest PQ score of 59.2% which is an improvement of 1.8% over M_{naive}. These results demonstrate that training our model on a pseudo labeled dataset improves the performance and we obtain a larger improvement if we further optimize by employing a form of regularization on the pseudo labels such as using our proposed heuristic.

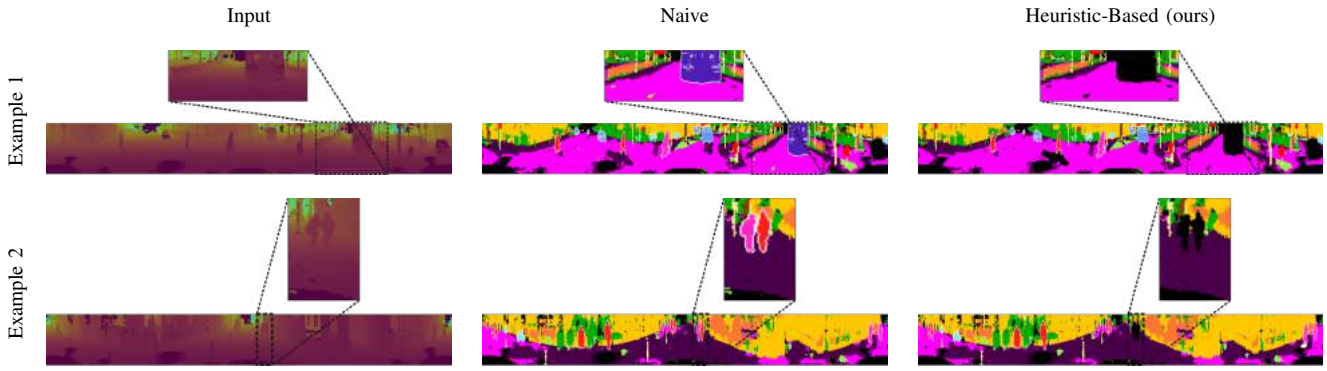


Fig. 8: Comparison of pseudo labels generated naively and using our proposed heuristic. On the left, the full range image is shown with zoomed in regions. Each example shows the predictions of the full image with zoomed in regions and the input range image. The labels generated using the naive heuristic misclassifies the truck (purple) in example 1 and a person as bicyclist (light purple) in example 2. Our proposed heuristic assigns the misclassified pixels as unlabeled in both examples.

8) Generalization Ability of Proposed Modules: In this experiment, we study the effectiveness and generalization ability of our proposed modules by directly incorporating them into other well-known top-down image panoptic segmentation networks. We choose two state-of-the-art panoptic image segmentation networks: Panoptic FPN [50] and Seamless [3]. In the vanilla version of both the networks, we use the panoptic fusion module to compute the final panoptic segmentation predictions. We train the vanilla versions with an input resolution of 4096×256 using scan unfolding based projection and kNN-based post-processing. We employ Lovász Softmax in addition to weighted per-pixel log loss during training. Tab. XIII shows the results from this experiment. The Panoptic $\text{FPN}_{\text{vanilla}}$ model achieves a *PQ* score of 48.7% and an *mIoU* of 55.2%. While the $\text{Seamless}_{\text{vanilla}}$ model achieves a *PQ* score of 50.6% and an *mIoU* of 56.9%.

In our version of both the networks, we introduce the proximity convolution module before the encoder, followed by employing the REN in parallel to the main encoder and switching the standard FPN to our range-aware FPN. Additionally, we use the panoptic periphery loss function during training. Since Panoptic FPN only employs a 3×3 convolution in the semantic head for each of its FPN scales, we replace this convolution with our proposed 3×3 range-guided depth-wise atrous separable convolution with $D_{max} = 3$. We keep D_{max} value low to compute denser features with a small receptive field in contrast to a high value that will output sparse features with a large receptive field. In the case of the Seamless network, we extend the miniDL module in the semantic head with an additional parallel branch that consists of our proposed 3×3 range-guided depth-wise atrous separable convolution with $D_{max} = 12$. As a result, Panoptic FPN_{ours} achieves an improvement of 2.1% in the *PQ* score and 1.1% in *mIoU* over the vanilla version. We also observe a similar improvement of 2.8% in the *PQ* score and 1.5% in *mIoU* for the $\text{Seamless}_{\text{ours}}$ model. This demonstrates the generalization ability of our proposed architectural modules as the direct incorporation of these modules without any tuning of parameters still achieves substantial improvement over the vanilla version. These results also show that any future top-down panoptic segmentation network can easily be transformed into a LiDAR panoptic segmentation network by incorporating our proposed architectural components.

E. Qualitative Evaluations

In this section, we qualitatively evaluate the panoptic segmentation performance of our proposed EfficientLPS model on the validation set of SemanticKITTI and nuScenes datasets. We compare the results of EfficientLPS with the best performing baseline from Sec. IV-C for the respective datasets. To this end, we compare with PanopticTrackNet and (KPConv + Mask R-CNN) for SemanticKITTI and nuScenes datasets respectively. Fig. 9 shows two comparisons for each of these datasets. We also present the improvement/error map for each of the comparisons and enlarge the segments of the outputs that show significant misclassification. The improvement/error map depicts the points that are misclassified by EfficientLPS with respect to the groundtruth in red and the points that are correctly predicted by EfficientPS but are misclassified by the baseline model in blue.

Fig. 9 (a) and Fig. 9 (b) show examples from the SemanticKITTI dataset in which the improvement over the baseline output (PanopticTrackNet) can be seen in the more accurate segmentation of the other-vehicle class as well as the improvement in distinguishing inconspicuous 'thing' classes such as person and bicyclist. EfficientLPS also demonstrates a more clear separation of object instances while segmenting cluttered classes. This can be primarily attributed to the proximity convolution module and the range-aware FPN that enables the network to have enhanced transform modeling capacity along with distance-aware semantically rich multi-scale features. This enables our model to learn highly discriminative features to accurately classify semantically related classes but at the same time the spatial awareness allows it to accurately segment object instances that are very close to each other. In Fig. 9 (a) the baseline fails to classify the other-vehicle class, whereas our model accurately classifies this object. In Fig. 9 (b), the baseline model segments the bicyclist but classifies is it as a person depicted in red (label color for person class). In contrast, our model correctly classifies it as a bicyclist depicted in magenta (label color for bicyclist class). Our model also

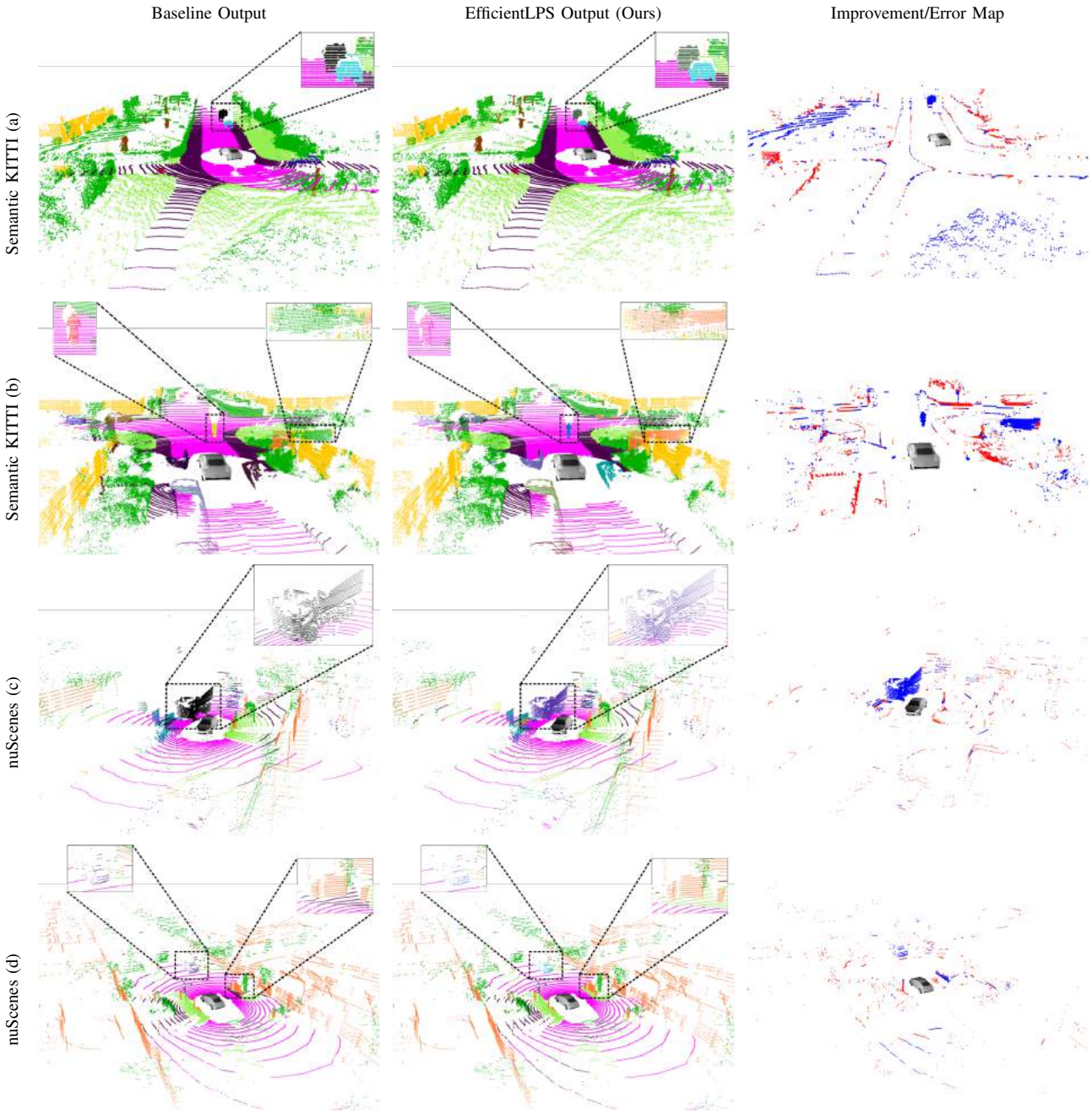


Fig. 9: Qualitative comparison of LiDAR panoptic segmentation results. We compare against PanopticTrackNet on SemanticKITTI and (KPConv + Mask R-CNN) on nuScenes validation sets. We also show the improvement/error map which shows the points that are misclassified by EfficientLPS in red and the points that are misclassified by the baseline but correctly predicted by EfficientLPS in blue.

successfully segments both vegetation and fence classes, as opposed to the baseline which misclassifies most of the fence as vegetation. Additionally, the effects of the boundary refinement is prominent in the segmentation of the car in Fig. 9 (a).

In Fig. 9 (c) and Fig. 9 (d), we qualitatively compare the performance on the sparse nuScenes dataset. We observe that in Fig. 9 (c) the truck is not segmented in the output of the baseline model (KPConv + Mask R-CNN), while our EfficientLPS model accurately segments the instance of the truck. In Fig. 9 (d) the baseline model segments the oncoming car instance but misclassifies it as a truck. It is significantly hard to accurately classify this instance as only a few points lie on the object. Nevertheless, our model still classifies these points as a car, which can be attributed to the shared backbone and the adaptable dense receptive field of the semantic head as well as the instance head. In Fig. 9 (d) the terrain class is accurately segmented by the EfficientLPS model, whereas the baseline misclassifies it as sidewalk.

V. Conclusion

In this work, we presented a novel top-down approach for LiDAR panoptic segmentation using a 2D CNN that effectively

leverages the unique spatial information provided by LiDAR point clouds. Our EfficientLPS architecture achieves state-of-the-art performance by incorporating novel architectural components that mitigate the problems caused by projecting the point cloud into the 2D domain. We proposed the proximity convolution module that effectively models geometric transformations of points in the projected image by exploiting the proximity between points. Our novel range-aware FPN demonstrates effective fusion of range encoded features with that of the main encoder to aggregate semantically rich multi-scale range-aware features. We proposed a new distance-dependent semantic head that incorporates our range-guided depth-wise atrous separable convolutions with adaptive dilation rates that cover large receptive fields densely to better capture the contextual semantic information. We further introduced the panoptic periphery loss function that refines object boundaries by utilizing the range information for maximizing the gap between the object boundary and the background. Moreover, we presented a new heuristic for generating pseudo labels from an unlabeled datasets to assist the network with additional training data.

We introduced panoptic ground truth annotations for the sparse LiDAR point clouds in the nuScenes dataset which we made publicly available. Additionally, we provided several baselines for LiDAR panoptic segmentation on this new dataset. We presented comprehensive benchmarking results on SemanticKITTI and nuScenes datasets that show that EfficientLPS sets the new state-of-the-art. EffcientLPS is ranked #1 on the SemanticKITTI panoptic segmentation leaderboard. We presented exhaustive ablation studies with quantitative and qualitative results that demonstrate the novelty of the proposed architectural components. Furthermore, we made the code and models publicly available.

ACKNOWLEDGEMENTS

This work was partly funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR, a research grant from Eva Mayr-Stihl Stiftung, and partly financed by the Baden-Württemberg Stiftung gGmbH.

REFERENCES

[1] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 9404–9413.

[2] R. Mohan and A. Valada, "Efficientps: Efficient panoptic segmentation," *arXiv preprint arXiv:2004.02307*, 2020.

[3] L. Porzi, S. R. Bulo, A. Colovic, and P. Kontschieder, "Seamless scene segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 8277–8286.

[4] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-deeplab," *arXiv preprint arXiv:1910.04751*, 2019.

[5] S. Gasperini, M.-A. N. Mahani, A. Marcos-Ramiro, N. Navab, and F. Tombari, "Panoster: End-to-end panoptic segmentation of lidar point clouds," *arXiv preprint arXiv:2010.15157*, 2020.

[6] A. Milioto, J. Behley, C. McCool, and C. Stachniss, "Lidar panoptic segmentation for autonomous driving," in *Int. Conf. on Intelligent Robots and Systems*, 2020.

[7] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Int. Conf. on Intelligent Robots and Systems*, 2015, pp. 922–928.

[8] B. Graham, M. Engelcke, and L. Van Der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 9224–9232.

[9] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving," *arXiv preprint arXiv:2003.03653*, 2020.

[10] J. V. Hurtado, R. Mohan, and A. Valada, "Mopt: Multi-object panoptic tracking," *arXiv preprint arXiv:2004.08189*, 2020.

[11] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[12] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[13] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Int. Conf. on Computer Vision*, 2017.

[14] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. of the Europ. Conf. on Computer Vision*, 2018.

[15] L.-C. Chen, M. Collins, Y. Zhu, G. Papandreou, B. Zoph, F. Schroff, H. Adam, and J. Shlens, "Searching for efficient multi-scale architectures for dense image prediction," in *Advances in Neural Information Processing Systems*, 2018, pp. 8713–8724.

[16] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Int. Conf. on Computer Vision*, 2019.

[17] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 11 621–11 631.

[18] W. Wang and U. Neumann, "Depth-aware cnn for rgb-d segmentation," in *Proc. of the Europ. Conf. on Computer Vision*, 2018, pp. 135–150.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[20] A. Boulch, "Convpoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, 2020.

[21] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proc. of the Europ. Conf. on Computer Vision*, 2018, pp. 87–102.

[22] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Int. Conf. on Computer Vision*, 2019.

[23] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4213–4220.

[24] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *Int. Conf. on Robotics & Automation*, 2018.

[25] L. T. Triess, D. Peter, C. B. Rist, and J. M. Zöllner, "Scan-based semantic segmentation of lidar point clouds: An experimental study," *arXiv preprint arXiv:2004.11803*, 2020.

[26] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, "Polarnet: An improved grid representation for online lidar point clouds semantic segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 9601–9610.

[27] A. Dewan, G. L. Oliveira, and W. Burgard, "Deep semantic classification for 3d lidar data," in *Int. Conf. on Intelligent Robots and Systems*, 2017.

[28] A. Dewan and W. Burgard, "Deeptemporalseg: Temporally consistent semantic segmentation of 3d lidar scans," in *Int. Conf. on Robotics & Automation*, 2020, pp. 2624–2630.

[29] D. Kochanov, F. K. Nejadasl, and O. Booij, "Kprnet: Improving projection-based lidar semantic segmentation," *arXiv preprint arXiv:2007.12668*, 2020.

[30] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *Proc. of the Europ. Conf. on Computer Vision*, 2020, pp. 685–702.

[31] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni, "Learning object bounding boxes for 3d instance segmentation on point clouds," *Advances in neural information processing systems*, vol. 32, pp. 6740–6749, 2019.

[32] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "Gspn: Generative shape proposal network for 3d instance segmentation in point cloud," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 3947–3956.

[33] F. Zhang, C. Guan, J. Fang, S. Bai, R. Yang, P. H. Torr, and V. Prisacariu, "Instance segmentation of lidar point clouds," in *Int. Conf. on Robotics & Automation*, 2020, pp. 9448–9455.
[34] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.
[35] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, "Pointgroup: Dual-set point grouping for 3d instance segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.
[36] Z. Ding, X. Han, and M. Niethammer, "Votenet: a deep learning label fusion method for multi-atlas segmentation," in *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 2019.
[37] N. Gao, Y. Shan, Y. Wang, X. Zhao, Y. Yu, M. Yang, and K. Huang, "Ssap: Single-shot instance segmentation with affinity pyramid," in *Int. Conf. on Computer Vision*, 2019, pp. 642–651.
[38] N. Zhao, T.-S. Chua, and G. H. Lee, "Sess: Self-ensembling semi-supervised 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 079–11 087.
[39] H. Wang, Y. Cong, O. Litany, Y. Gao, and L. J. Guibas, "3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 615–14 624.
[40] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.
[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Int. Conf. on Computer Vision*, 2017, pp. 2961–2969.
[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *Int. Journal of Robotics Research*, 2013.
[43] S. R. Bulo, G. Neuhold, and P. Kontschieder, "Loss max-pooling for semantic image segmentation," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 7082–7091.
[44] M. Berman, A. Rannen Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 4413–4421.
[45] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.
[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
[47] A. Valada, R. Mohan, and W. Burgard, "Self-supervised model adaptation for multimodal semantic segmentation," *Int. Journal of Computer Vision*, pp. 1–47, 2019.
[48] A. Valada, A. Dhall, and W. Burgard, "Convoluted mixture of deep experts for robust semantic segmentation," in *IEEE/RSJ International conference on intelligent robots and systems (IROS) workshop, state estimation and terrain perception for all terrain mobile robots*, 2016.
[49] A. Valada, G. Oliveira, T. Brox, and W. Burgard, "Towards robust semantic segmentation using deep fusion," in *Robotics: Science and Systems (RSS 2016) Workshop, Are the Sceptics Right? Limits and Potentials of Deep Learning in Robotics*, 2016.
[50] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019, pp. 6399–6408.

Kshitij Sirohi is a Ph.D. student in the Autonomous Intelligent Systems group headed by Wolfram Burgard. He received his M.S. degree in Embedded Systems Engineering from the University of Freiburg in 2020. His research focuses on Deep learning and probabilistic approaches for robot perception, localization and mapping.

Rohit Mohan is a Ph.D. student in the Robot Learning Lab headed by Abhinav Valada. He received his M.S. degree in Computer Science from the University of Freiburg in 2021. His research focuses on robot perception and navigation using deep learning.

Daniel Büscher is a postdoctoral researcher in the Autonomous Intelligent Systems group headed by Wolfram Burgard. He received his Ph.D. degree from the University of Freiburg in 2012. His research focuses on autonomous robot navigation and perception using deep learning.

Wolfram Burgard is Vice President for Automated Driving Technology at the Toyota Research Institute in Los Altos, USA. He is on leave from a Professorship for Computer Science at the University of Freiburg, Germany where he heads the Laboratory for Autonomous Intelligent Systems. He received his Ph.D. degree in computer science from the University of Bonn in 1991. His areas of interest lie in robotics and artificial intelligence. Wolfram Burgard and his group developed several innovative techniques for robot navigation, perception and control. Wolfram Burgard is head of the BrainLinks-BrainTools Center at the University of Freiburg. He is fellow of the AAAI, the EurAi, the IEEE and the ELLIS Society. Wolfram Burgard is also member of the German Academy of Sciences Leopoldina and the Heidelberg Academy of Sciences and Humanities.

Abhinav Valada is an Assistant Professor and head of the Robot Learning Lab at the University of Freiburg. He is a member of the Department of Computer Science, a principal investigator at the BrainLinks-BrainTools Center, and a core faculty in the European Laboratory for Learning and Intelligent Systems (ELLIS) unit in Freiburg. He received his Ph.D. in Computer Science from the University of Freiburg in 2019 and his M.S. degree in Robotics from Carnegie Mellon University in 2013. His research lies at the intersection of robotics, machine learning and computer vision with a focus on tackling fundamental robot perception, state estimation and control problems using learning approaches in order to enable robots to reliably operate in complex and diverse domains. Abhinav Valada is a scholar of the ELLIS Society.

7.11 GEM: Glare or Gloom, I Can Still See You - End-to-End Multi-Modal Object Detection

The appended paper [131] follows.

GEM: Glare or Gloom, I Can Still See You – End-to-end Multi-modal Object Detection

Osama Mazhar[1], Robert Babuška[1,2] and Jens Kober[1]

***Abstract*—Deep neural networks designed for vision tasks are often prone to failure when they encounter environmental conditions not covered by the training data. Single-modal strategies are insufficient when the sensor fails to acquire information due to malfunction or its design limitations. Multi-sensor configurations are known to provide redundancy, increase reliability, and are crucial in achieving robustness against asymmetric sensor failures. To address the issue of changing lighting conditions and asymmetric sensor degradation in object detection, we develop a multi-modal 2D object detector, and propose deterministic and stochastic sensor-aware feature fusion strategies. The proposed fusion mechanisms are driven by the estimated sensor measurement reliability values/weights. Reliable object detection in harsh lighting conditions is essential for applications such as self-driving vehicles and human-robot interaction. We also propose a new "r-blended" hybrid depth modality for RGB-D sensors. Through extensive experimentation, we show that the proposed strategies outperform the existing state-of-the-art methods on the FLIR-Thermal dataset, and obtain promising results on the SUNRGB-D dataset. We additionally record a new RGB-Infra indoor dataset, namely L515-Indoors, and demonstrate that the proposed object detection methodologies are highly effective for a variety of lighting conditions.**

***Index Terms*—Object Detection, Segmentation and Categorization, Sensor Fusion, Deep Learning for Visual Perception, Computer Vision for Automation, RGB-D Perception**

I. INTRODUCTION

MODERN intelligent systems such as autonomous vehicles or assistive robots should have the ability to reliably detect objects in challenging real-world scenarios. Object detection is one of the widely studied problems in computer vision. It has been addressed lately by employing deep convolutional neural networks where the state-of-the-art methods have achieved fairly accurate detection performances on the existing datasets [1–4]. However, these vision models are fragile and do not generalize across realistic unconstrained scenarios, such as changing lighting conditions or other environmental circumstances which were not covered by the training data [5]. The failure of the detection algorithms in such conditions could lead to potentially catastrophic results, as in the case of self-driving vehicles.

Fig. 1. Output samples of the proposed multi-modal object detector. The blue/green bar at the top illustrates the contribution/reliability of each sensor modality in obtaining the final output. Images from two modalities are merged diagonally only for illustration purposes. (a) Shows the results on the FLIR-Thermal dataset with RGB and thermal sensor modalities, (b) Shows the output on the SUNRGB-D dataset with RGB and our proposed "r-blended" hybrid depth modality.

One way of addressing this problem is to employ a data-augmentation strategy [6]. It refers to the technique of perturbing data without altering class labels, and it has been proven to greatly improve robustness and generalization performance [7]. Nevertheless, this is insufficient for the cases where the sensor fails to acquire information due to malfunction or its technical limitations. For example, the output of standard passive cameras degenerates with reduced ambient light, while thermal cameras or LiDARs are less affected by illumination changes.

Multi-sensor configurations are known to provide redundancy and often enhance the performance of the detection algorithms. Moreover, efficient sensor fusion strategies minimize uncertainties, increase reliability, and are crucial in achieving robustness against asymmetric sensor failures [8]. Although, increasing the number of sensors might enhance the performance of detection algorithms, this comes with a considerable computational and energy cost. This is not desirable in mobile robotic systems, which typically have constraints in terms of computational power and battery consumption. In such cases, intelligent choice and combination of sensors are crucial.

Furthermore, multi-modal data fusion often requires an estimate of the sensor signal uncertainty to guarantee efficient fusion and reliable prediction without a priory knowledge of the sensor characteristics [9]. The existing multi-modal object detection methods fuse the sensor data streams without explicitly modeling the measurement reliability. This may have severe consequences when the data from an individual sensor degrades or is missing due to sheer sensor failure.

To address the above problems, we propose sensor-aware multi-modal fusion strategies for object detection in harsh

Manuscript received: Feb, 24, 2021; Revised May, 11, 2021; Accepted June, 8, 2021.

This paper was recommended for publication by Eric Marchand upon evaluation of the Associate Editor and Reviewers' comments. The research presented in this article was carried out as part of the OpenDR project, which has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 871449.

[1]Cognitive Robotics Department, Delft University of Technology, Delft, The Netherlands {O.Mazhar, R.Babuska, J.Kober}@tudelft.nl

[2]Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Czech Republic

Digital Object Identifier (DOI): see top of this page.

lighting conditions, thus the title " GEM: Glare or gloom, I can still see you - End-to-end Multimodal object detection". The output samples of GEM are shown in Figure 1. Two fusion methods are proposed: deterministic weighted fusion and stochastic feature fusion. In the deterministic weighted fusion, the measurement certainty of each sensor is estimated either by learning scalar weights or masks through separate neural networks. The learned weights are then assigned to the feature maps extracted from the feature extractor backbones for each sensor modality. The weighted feature maps can be fused either by averaging or concatenation. Moreover, we can visualize and interpret the measurement certainty of each sensor in the execution phase, which provides deeper insights into the relative strengths of each data stream. The stochastic feature fusion creates a one-hot encoding of the feature maps of each sensor, which can be assumed as a discrete switch that allows only the dominant/relevant features to pass. The obtained selected features are then concatenated before they are passed to the object detection and classification head. The proposed sensor-aware multi-modal object detector, referred to simply as GEM in the rest of the paper, is trained in an end-to-end fashion along with the fusion mechanism.

Most modern object detectors, including YOLO, Faster-RCNN and SSD employ many hand-crafted features such as anchor generation, rule-based assignment of classification and regression targets as well as weights to each anchor, and non-maximum suppression postprocessing. The overall performance of these methods often relies on careful tuning of the above-mentioned hyper-parameters. Following their success in sequence/language modeling, transformers have lately emerged in vision applications, outperforming competitive baselines and demonstrating a strong potential in this field. Therefore, we employ transformers in our work as in [10], which thanks to their powerful relational modeling capability eliminates the need of hand-crafted components in object detection. Our main contributions in this paper are:

- Evaluation of feature fusion in two configurations, i.e., deterministic weighted fusion and stochastic feature fusion for multi-modal object detection.
- Estimation of measurement reliability of each sensor as scalar or mask multipliers through separate neural networks for each modality to efficiently drive the deterministic weighted fusion.
- Use of transformers for multi-modal object detection to harness the efficacy of self-attention in sensor fusion.

II. Related Work

In this section, we first review deep learning-based object detection strategies, followed by a discussion on existing methods for multi-modal fusion methods in relevant tasks.

A. Deep learning-based Object Detection

Detailed literature surveys for deep learning-based object detectors have been published in [11, 12]. Here we briefly discuss some of the well-known object detection strategies. Typically, object detectors can be classified into two types, namely two-stage and singe-stage object detectors.

1) Two-stage object detection: Two-stage object detectors exploit a region proposal network (RPN) in their first stage. RPN ranks region boxes, alias *anchors*, and proposes the ones that most likely contain objects as candidate boxes. The features are extracted by region-of-interest pooling (RoIPool) operation from each candidate box in the second stage. These features are then utilized for bounding-box regression and classification tasks.

2) Single-stage object detection: Single-stage detectors propose predicted boxes from input images in one forward pass directly, without the region proposal step. Thus, this type of object detectors are time efficient and can be utilized for real-time operations. Lately, an end-to-end object detection strategy has been proposed in [10] that eliminates the need for hand-crafted components like anchor boxes and non-maximum suppression. The authors employ transformers in an encoder-decoder fashion, which have been extremely successful and become a de facto standard for natural language processing tasks. The transformer implicitly performed region proposals instead of using an R-CNN. The multi-head attention module in transformers jointly attended to different semantic regions of an image/feature maps and linearly aggregates the outputs through learnable weights. The learned attention maps can be visualized without requiring dedicated methods, as in the case of convolutional neural networks. The inherent non-sequential architecture of transformers allows parallelization of models. Thus, we opted to build upon the methodology of [10] for our multi-modal object detector for harsh lighting conditions.

B. Sensor Fusion

Sensor fusion strategies can be roughly divided into three types according to the level of abstraction where fusion is performed or in which order transformations are applied compared to feature combinations, namely low-level, mid-level, and high-level fusion [13]. In low-level or early fusion, raw information from each sensor is fused at pixel level, e.g., disparity maps in stereovision cameras [14]. In mid-level fusion, a set of features is extracted for each modality in a pre-processing stage, while multiple approaches [15] are exploited to fuse the extracted features. Late-fusion often employs a combination of two fusing methods, e.g., convolution of stacked feature maps followed by several fully connected layers with dropout regularization [16]. In high-level fusion or ensemble learning methods, predictions are obtained individually for each modality and the learnt scores or hypotheses are subsequently combined via strategies such as weighted majority votes [17]. Deep fusion or cross fusion [18] is another type of fusion strategy which repeatedly combines inputs, then transforms them individually. In each repetition, the transformation learns different features. For example in [8], features from the layers of VGG network are exchanged among all modalities driven by sensor entropy after each pooling operation.

C. Multi-sensor Object Detection

Most of the efforts on multi-modal object detection in the literature are focused on pedestrian or vehicle detection in

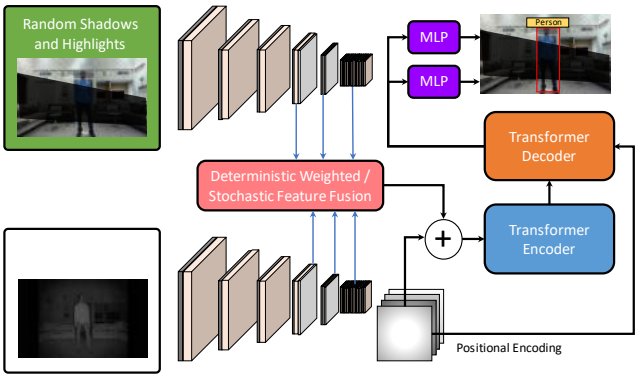


Fig. 2. Our proposed pipeline for a multi-modal object detector with transformers. The features from each backbone are fused and passed to the transformer encoder-decoder network. The decoder output is subsequently exploited by Multilayer Perceptrons (MLPs) for bounding box regression and object classification.

the automotive context. Sensor fusion strategies are typically proposed for camera-LiDAR, camera-radar, and camera-radar-LiDAR setups. Here, we briefly go through the relevant state-of-the-art methods.

The authors in [8] proposed an entropy-steered multi-modal deep fusion architecture for adverse weather conditions. The sensor modalities exploited in their method include RGB camera, gated camera (NIR band), LiDAR, and radar. Instead of employing BeV projection or point cloud representation for LiDAR, the authors encoded depth, height, and pulse intensity on an image plane. Moreover, the radar output was also projected onto an image plane parallel to the image horizontal dimension. Considering the radar output invariant along the vertical image axis, the scan was replicated across the horizontal image axis. They utilized a modified VGG architecture for feature extraction, while features were exchanged among all modalities driven by sensor entropy after each pooling operation. Fused feature maps from the last 6 layers of the feature extractors were passed to the SSD object detection head. In [19], the authors proposed a pseudo multi-modal object detector from thermal IR images in a Faster-RCNN setting. The features from ResNet-50 backbones for the two modalities are concatenated and a 1×1 convolution is applied to the concatenated features before they are passed to the rest of Faster-RCNN network. They exploited I2I translation networks, namely CycleGAN [20] and UNIT [21] to transform thermal images from the FLIR Thermal [22] and KAIST [23] datasets to the RGB domain, thus the names MM-CG and MM-UNIT.

Here, we also discuss some fusion strategies which were originally proposed for applications other than object detection but are relevant to our work. In [24], the authors proposed a sensor fusion methodology for RGB and depth images to steer a self-driving vehicle. The latent semantic vector from an encoder-decoder segmentation network trained on RGB images was fused with the depth features. The fusion architecture proposed by [24] is similar to the gating mechanism driven by the learned scalar weights presented in [25]. The method proposed in [26] is closest to our work. The authors proposed two sensor fusion strategies for Visual-Inertial Odometry (VIO), namely soft fusion and hard fusion. In soft fusion, they learned soft masks which were subsequently assigned to each element in the feature vector. Hard fusion employed a variant of the Gumbel-max trick, which is often used to sample discrete data from categorical distributions. Learning masks equal to the size of feature vectors might introduce computational overhead. Therefore, we learn dynamic scalar weights for each sensor modality, which adapt to the environmental/lighting conditions. These scalar weights represent the reliability or relevance of the sensor signals. Moreover, we also learn single-channel masks with a spatial size equal to that of the feature maps obtained from the feature extractor backbone. Nevertheless, we also implement the Gumbel-Softmax trick for comparison, as a stochastic feature fusion for multi-modal object detectors.

III. Sensor-aware Multi-modal Fusion

In this work, we propose a new method for sensor-aware feature selection and multi-modal fusion for object detection. We actually evaluate feature fusion in two configurations, i.e., deterministic weighted fusion with scalar and mask multipliers, and stochastic feature fusion driven by the Gumbel-Softmax trick that enables sampling from a discrete distribution. The overall pipeline of the proposed multi-modal object detector is illustrated in Figure 2. The proposed methodologies are trained and evaluated on datasets with RGB and thermal or depth images. However, it can be extended to include data from other sensors like LiDAR or radar, either by projecting the sensor output onto an image plane as proposed in [8] or by employing sensor-specific feature extractors such as [27].

A. Deterministic Weighted Fusion

The proposed deterministic weighted fusion scheme is conditioned on the measurement certainty of each sensor. These values are obtained either by learning scalar weights or masks through separate neural networks. Subsequently, the weights are assigned to the feature maps extracted from the backbones as (scalar or mask) multipliers for each sensor modality. Given the output of the backbone feature extractors $\mathbf{s}$ for a single modality, the neural network f optimizes parameters $\boldsymbol{\theta}$ to obtain measurement certainty w of the corresponding sensor as described as follows:

$$w = f(\mathbf{s}, \boldsymbol{\theta}) \times \frac{1}{\text{rows} \times \text{cols} \times k} \sum_{l=1}^{\text{rows}} \sum_{m=1}^{\text{cols}} \sum_{n=1}^{k} \mathbf{s}(l, m, n) \quad (1)$$

where k is the selected number of channels. The network f learns the parameters $\boldsymbol{\theta}$ in an end-to-end fashion. In the case of sensor degradation, the output of the neurons in the early layers of the corresponding backbone will remain close to zero. Thus, we multiply the output of the network f by the mean of first k feature maps, 16 in our case, from $\mathbf{s}$ in a feed-forward setting to obtain w. This allows f to dynamically condition its output to changing lighting/sensor degradation scenarios, which subsequently guides the transformers to focus on the dominant sensor modality for object detection. Furthermore, the multiplication of the raw output of f with the mean

of the selected feature maps is performed without gradient calculation to prevent the distortion of the feature maps in the back-propagation phase.

The weighted feature maps are fused by either taking an average of the two feature sets, or by concatenating them. The fused features are then passed to the transformer for object detection and localization. Our scalar fusion functions g_{sa} (averaging) and g_{sc} (concatenating) are represented as:

$$\begin{aligned} g_{\text{sa}}(\mathbf{s}_{\text{RGB}}, \mathbf{s}_{\text{IR}}) &= \phi(w_{\text{RGB}} \odot \mathbf{s}_{\text{RGB}}, w_{\text{IR}} \odot \mathbf{s}_{\text{IR}}) \\ g_{\text{sc}}(\mathbf{s}_{\text{RGB}}, \mathbf{s}_{\text{IR}}) &= [w_{\text{RGB}} \odot \mathbf{s}_{\text{RGB}}; w_{\text{IR}} \odot \mathbf{s}_{\text{IR}}] \end{aligned} \tag{2}$$

where ϕ denotes the mean operation, $\mathbf{s}_{\text{RGB}}$ and $\mathbf{s}_{\text{IR}}$ are feature maps obtained from the backbone feature extractor for RGB and thermal/IR imagers respectively, while w_{RGB} and w_{IR} are the sensor measurement certainty weights obtained through Equation (1). Similar to the scalar fusion method, feature selection is also modelled by learning masks for each modality, in this case $\mathbf{m}_{\text{RGB}}$ and $\mathbf{m}_{\text{IR}}/\mathbf{m}_{\text{depth}}$, with a spatial size equal to that of the features maps. The fusion scheme with mask multipliers is represented as:

$$\begin{aligned} g_{\text{ma}}(\mathbf{s}_{\text{RGB}}, \mathbf{s}_{\text{IR}}) &= \phi(\mathbf{m}_{\text{RGB}} \odot \mathbf{s}_{\text{RGB}}, \mathbf{m}_{\text{IR}} \odot \mathbf{s}_{\text{IR}}) \\ g_{\text{mc}}(\mathbf{s}_{\text{RGB}}, \mathbf{s}_{\text{IR}}) &= [\mathbf{m}_{\text{RGB}} \odot \mathbf{s}_{\text{RGB}}; \mathbf{m}_{\text{IR}} \odot \mathbf{s}_{\text{IR}}] \end{aligned} \tag{3}$$

B. Stochastic Feature Fusion

In addition to the weighted fusion schemes, we exploit a variant of the Gumbel-max trick to learn a one-hot encoding that either propagates or blocks each component of the feature maps for intelligent fusion. The Gumbel-max resampling strategy allows to draw discrete samples from a categorical distribution during the forward pass through a neural network. It exploits the reparametrization trick to separate out the deterministic and stochastic parts of the sampling process. However, it adds Gumbel noise instead of that from a normal distribution, which is actually used to model the distribution of the maximums for samples taken from other distributions. Gumbel-max then employs the $\arg\max$ function to find the class that has the maximum value for each sample.

Considering α be the n-dimensional probability variable conditioned for every row on each channel of the feature volume such that $\alpha = [\pi_1, \ldots, \pi_n]$, representing the probability of each feature at location n, the Gumbel-max trick can be represented by the following equation:

$$Q = \arg\max_i (\log \pi_i + G_i) \tag{4}$$

where, Q is a categorical variable with class probabilities $\pi_1, \pi_2, \ldots, \pi_n$ and $\{G_i\}_{i \leq n}$ is an i.i.d. sequence of standard Gumbel random variables which is given by:

$$G = -\log(-\log(U)), \quad U \sim \text{Uniform}[0, 1] \tag{5}$$

The use of $\arg\max$ makes the Gumbel-max trick *non*-differentiable. However, it can be replaced by *Softmax* with a temperature factor τ, thus making it a fully-differentiable resampling method [28]. Softmax with temperature parameter τ can be represented as:

$$f_\tau(x)_i = \frac{\exp(x_i/\tau)}{\Sigma_{j=1}^{n} \exp(x_j/\tau)} \tag{6}$$

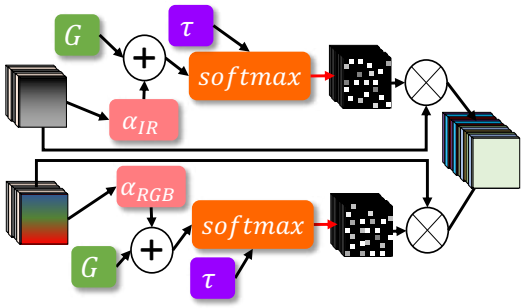


Fig. 3. Illustration of our stochastic feature fusion strategy that employs the Gumbel-Softmax sampling trick.

where τ determines how closely the Gumbel-Softmax distribution matches the categorical distribution. With low temperatures, e.g., $\tau = 0.1$ to $\tau = 0.5$, the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable [28]. The Gumbel-Softmax resampling function can therefore be written as

$$Q_i^\tau = f_\tau(\log \pi_i + G_i) = \frac{\exp((\log \pi_i + G_i)/\tau)}{\Sigma_{j=1}^{n} \exp((\log \pi_j + G_j)/\tau)} \tag{7}$$

with $i = 1, \ldots, n$.

We set $\tau = 1$ and obtain feature volume approximate one-hot categorical encodings for each modality $\mathbf{e}_{\text{RGB}}$ and $\mathbf{e}_{\text{IR}}$. Then a Hadamard product is taken between the encodings and the feature volumes and the resultants are subsequently concatenated and passed on to the bounding box regressor and classification head. We illustrate our selective fusion process developed for multi-modal object detector in Figure 3, while the selective fusion function g_{sf} is given as follows

$$g_{\text{sf}}(\mathbf{s}_{\text{RGB}}, \mathbf{s}_{\text{IR}}) = [\mathbf{e}_{\text{RGB}} \odot \mathbf{s}_{\text{RGB}}; \mathbf{e}_{\text{IR}} \odot \mathbf{s}_{\text{IR}}]. \tag{8}$$

IV. Experiments

A. Datasets

Three datasets are utilized in the training and evaluation of GEM, including the FLIR Thermal, SUNRGB-D [29] and a new L515-Indoor dataset that we recorded for this research. The FLIR Thermal dataset provides 8,862 training and 1,366 test samples of thermal and RGB images recorded in the streets and highways in Santa Barbara, California, USA. Only the thermal images in the dataset are annotated with four classes, i.e., *People*, *Bicycle*, *Car* and *Dog*. The given RGB images in the dataset are neither annotated nor aligned with their thermal counterparts, while the camera matrices are also not provided. Thus, to utilize this dataset in a multi-modal setting, the given RGB images must be annotated or aligned with their corresponding thermal images. One way to address this problem is to create artificial RGB images from input thermal images through GANs or similar neural networks as performed in [19]. However, we opted to employ the concept of homography by manually selecting matching features in multiple RGB and thermal images. The selected feature points are then employed to estimate a transformation matrix between

the two camera modalities. RGB images are subsequently transformed with the estimated homography matrix such that they approximately align with their thermal equivalents. The *Dog* class constitutes only 0.29% of all the annotations in the aligned FLIR-Thermal dataset, thus it is not included in our experiments.

The SUNRGB-D dataset contains 10,335 RGB-D images taken by Kinect v1, Kinect v2, Intel RealSense, and Asus Xtion cameras. The annotations provided consist of 146,617 2D polygons and 64,595 3D bounding boxes, while 2D bounding boxes are obtained by projecting the coordinates of 3D bounding boxes onto the image plane. Although the dataset contains labels for approximately 800 objects, we evaluate our method on the selected 19 objects similar to [29]. We first divide the dataset into three subsets such that the training set consists of 4,255 images, the validation set has 5,050 images, while the test set contains 1,059 images.

The L515-Indoor dataset provides 482 training and 207 validation RGB and IR images recorded with Intel RealSense L515 camera with various ambient light conditions in an indoor scene. It contains annotations of 1,819 2D bounding boxes of 6 object categories in total. The IR images are aligned with their RGB counterparts through a homography matrix which is computed in a similar fashion as explained for the FLIR-Thermal dataset. The population distributions of the datasets are illustrated in Figure 4.

B. Pre-processing Sensor Outputs

For the FLIR-Thermal and L515-Indoor datasets, aligned RGB and thermal/IR images are fed into our feature extractor backbones without any pre-processing. However, techniques that exploit datasets with depth images including [29] often apply HHA encoding [30] on the depth sensor modality for early feature extraction prior to being fed into the neural networks. HHA is a geocentric embedding for depth images that encodes horizontal disparity, height above ground, and angle with gravity for each pixel. In a multi-threading setup on a 12-Core Intel® Core™ i7-9750H CPU, HHA encoding of a batch of 32 images takes approximately 119 seconds, which is far from its application in real-time object detection or segmentation tasks.

To address this problem, assuming that we are working with RGB and depth modalities, we create a new hybrid image that introduces scene texture in a depth image. As the red light is scattered the least by air molecules, we blend the depth images and the red image channels through a blend weight α. Thus, we name our hybrid depth image as "r-blended" depth image.

$$\text{img}_{\text{r-blended}} = \alpha\ \text{img}_{\text{depth}} + (1-\alpha)\ \text{img}_{\text{red}} \tag{9}$$

The value of α is set to 0.9 for depth images while the weight value for the red channels becomes 0.1. This is to make sure that when the neural network is trained with "r-blended depth" image, it should focus on learning the depth features while information from the red channel only complements the raw depth map. The idea to blend the red channel is also supported by the fact that CMOS cameras are often more sensitive to green and red light. We first train our multi-modal object detector on RGB and HHA encoded depth images. Later, we fine-tune the trained model by replacing HHA encoded images with r-blended depth images and achieve comparable results in terms of detection accuracy, while the fine-tuned model can indeed be used for real-time multi-modal object detection.

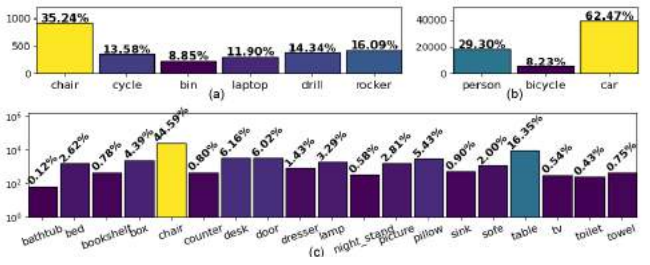


Fig. 4. Class distributions of the datasets (a) L515-Indoor (b) FLIR-Thermal (c) SUNRGB-D. The number of annotations in (c) are presented in the logarithmic scale.

C. Training

GEM is trained with scalar fusion and mask fusion methods, i.e., g_{sa}, g_{sc}, g_{ma} and g_{mc} for deterministic weighted fusion driven by Equations (2) and (3), while it is also trained with g_{sf} for stochastic feature fusion. The backbone feature extractors for both sensor streams and the transformer block are pre-trained on MSCOCO dataset on RGB images as in [10]. For the FLIR thermal dataset, each model is trained on a cluster with 2 GPUs for 100 epochs while the models for the SUNRGB-D dataset are trained with 4 GPUs for 300 epochs. Similarly, the models for L515-Indoor are trained on a cluster with 2 GPUs for 300 epochs with a batch size of 1. The batch size for the FLIR-Thermal and SUNRGBD datasets is set to 2, while the learning rate for the feature extractor backbones, fusion networks, and transformer block is set to 8×10^{-6} for all datasets. We employ ResNet-50 as the feature extractor, while we also train g_{sc} on MobileNet v2 [31] for the FLIR thermal dataset. To guide the fusion process and mimic harsh lighting conditions for the RGB sensor, we also employ Random Shadows and Highlights (RSH) data augmentation as proposed in [7]. RSH develops immunity against lighting perturbations in the convolutional neural networks, which is desirable for real world applications. We additionally implement SSD512 object detector with VGG16 backbones in a multi-modal setting in two configurations, i.e., a simple averaging fusion as the baseline method (SSD-BL) and a weighted averaging fusion scheme (SSD-WA) similar to g_{sa}. The anchor/default boxes are configured for both SSD-BL and SSD-WA in a fashion similar to that for the MS-COCO dataset. These models are trained for 800 epochs in a single GPU setup with a batch size 1 and a learning rate 1×10^{-4} which decays with a decaying factor 0.2 after the first 520,000 iterations.

D. Evaluation

FLIR-Thermal: performance evaluation of the proposed networks on the FLIR-Thermal dataset is shown in Table I. We show Average Precision (AP) values at Intersection over Union (IoU) of 0.5 for each dominant class, while the mean Average Precision (mAP) is also estimated with and without lighting

TABLE I
PERFORMANCE EVALUATION ON FLIR-THERMAL DATASET.

Model w/ RSH	w/o Random Shadows and Highlights				w/ RSH
	AP@IoU=0.5			mAP@	mAP@
	Person	Bicycle	Car	IoU=0.5	IoU=0.5
FLIR Baseline	0.794	0.580	0.856	0.743	-
rgb-only	0.383	0.168	0.638	0.395	0.376
thermal-only	0.683	0.499	0.783	0.655	0.316
MM-UNIT [19]	0.644	0.494	0.707	0.615	-
MM-CG [19]	0.633	0.502	0.706	0.614	-
SSD-BL	0.450	0.341	0.719	0.503	0.478
SSD-WA	0.526	0.314	0.718	0.519	0.516
avg-baseline	0.801	0.562	0.879	0.747	0.731
conc-baseline	0.533	0.417	0.675	0.541	0.492
g_{sa}	**0.828**	0.593	**0.891**	**0.770**	**0.769**
g_{sc}	0.809	**0.637**	0.862	0.769	0.764
g_{ma}	0.803	0.575	0.862	0.746	0.744
g_{mc}	0.800	0.611	0.857	0.755	0.755
g_{sf}	0.790	0.584	0.874	0.749	0.756
g_{sc} m-net	0.696	0.472	0.823	0.663	0.664

perturbations. These lighting corruptions are introduced by creating Random Shadows and Highlights (RSH) on the test RGB images. The evaluation with lighting perturbation is performed for 10 trials in all experiments, while the average of the obtained mAP is shown in the table. The results are compared with the single modality object detector, the multi-modal baseline fusion networks, and the existing state-of-the-art methods on this dataset. In the baselines, the features from the backbones are fused in two configurations: averaged and concatenated, without any weighing or re-sampling mechanism. Additionally, we compare the performance of SSD-BL and SSD-WA on the FLIR-Thermal dataset. It is clear from the evaluation results, that our proposed methodologies, i.e., g_{sa}, g_{sc}, g_{ma}, g_{mc}, and g_{sf}, outperform the previously reported results on this dataset. Our methods also demonstrate robustness against lighting perturbation, while a significant performance drop of single modality and baseline methods can be seen when tested with RSH. The "avg-baseline" obtained comparable results, but as it is only a blind fusion, hence no sensor contribution or reliability measure can be obtained with this methodology. Additionally, its performance can significantly drop in the case of asymmetric sensor failure. This can partially be observed in Table I where the baselines are tested with RSH perturbations. Concerning the evaluation of multi-modal SSD on the FLIR-Thermal dataset, SSD-WA certainly improves the results compared to SSD-BL, specifically in terms of robustness against lighting perturbations introduced by RSH. The overall performance of SSD-based detectors turned out to be inferior to that of our transformer-based multi-modal object detection methods.

Among our proposed fusion methods, g_{sa} obtained the best overall performance on the FLIR-Thermal dataset. Scalar multiplication amplifies the information in the feature maps by retaining the learned structure. Nevertheless, mask multiplication may amplify a certain spatial portion of the feature maps in some channels, but it could also potentially distort the learned information depth-wise. Concatenation might be useful when the feature spaces of the utilized sensor modalities differ, e.g., image versus point cloud. However, in our case of image modalities, the averaging features g_{sa} performed better than concatenation g_{sc}. Similarly, switching off the features with selective fusion g_{sf} has affected the performance of the model adversely. We plan to explore this method further in our future research, especially in the cases when information from the sensor modalities of dissimilar domains are fused, e.g., camera versus LiDAR/radar.

SUNRGB-D: The evaluation results on SUNRGB-D dataset are shown in Table II. We not only present a comparison of single vs. multi-modal settings on the selected 19 categories of the SUNRGB-D dataset, but also between raw vs. processed depth images. The table only shows the results for eight categories due to limited space. Two single modality networks are trained, one with RGB images and the other with HHA-encoded depth images. We also evaluate the performance of "conc-baseline" and "avg-baseline" with RGB and HHA-encoded depth modalities. Motivated by the performance of g_{sa} and g_{sc} on the FLIR-Thermal dataset, we chose to evaluate their performance on SUNRGB-D dataset exclusively. Since HHA-encoding introduces a significant computational burden inhibiting the possibility of real-time object detection, we first train g_{sa} and g_{sc} with on RGB and HHA-encoded depth images, later we fine-tune these models on raw-depth images as well as on our "r-blended" hybrid depth images. It is evident in Table II that both g_{sa} and g_{sc} obtain promising results on this dataset with RGB and "r-blended" depth images. Further analysing the results of Table I, we observe that the comparative performance of the models on the *Bicycle* class is not stable. Looking at the distribution of the datasets in Figure 4, we realize that the *Bicycle* class only constitutes 8.23% of the dataset. This indicates its comparative inconsistent performance on various models. However, analysing the results in Table II, we realize this performance instability might also be related to the object size. The proposed networks are able

TABLE II
PERFORMANCE EVALUATION ON SUNRGB-D DATASET.

Models	Tested without Random Shadows and Highlights (RSH)											w/ RSH
	AP@IoU=0.5										mAP@	mAP@
	bathtub	bed	bookshelf	box	chair	...	door	dresser	lamp	night stand	IoU=0.5	IoU=0.5
RGB-only	0.116	0.461	0.038	0.084	0.457	...	0.370	0.085	0.185	0.095	0.224	0.169
HHA-only	0.355	0.409	0.002	0.020	0.413	...	0.113	0.024	0.199	0.057	0.165	0.093
conc-baseline	0.333	0.440	0.002	0.068	0.456	...	0.367	0.056	0.195	0.041	0.211	0.155
avg-baseline	0.174	0.461	0.032	0.062	0.470		0.339	0.030	0.220	0.049	0.207	0.166
g_{sc}(raw-depth)	0.404	0.411	0.008	0.073	0.487	...	0.360	0.051	0.225	0.044	0.226	0.209
g_{sc}(r-blended)	0.350	0.457	0.040	0.085	**0.490**	...	**0.381**	**0.107**	**0.226**	**0.108**	**0.242**	0.230
g_{sa}(raw-depth)	0.204	0.399	0.033	**0.087**	0.478		0.344	0.102	0.220	0.025	0.221	0.214
g_{sa}(r-blended)	0.253	0.423	**0.106**	0.080	0.474		0.379	0.035	0.219	0.079	0.239	**0.236**

Fig. 5. Qualitative analysis of our multi-modal object detector, g_{sa} in this case. Columns (a), (b) and (d) are the outputs of g_{sa} in various asymmetric sensor failure conditions imitated artificially, which are mentioned on the upper-right corner of each image in row I. The top blue/green bar represents the contribution of each sensor modality in obtaining the final results (RGB: blue and Thermal/Infra: green). (c) and (e) are the outputs from single modal baselines. (f) is the ground-truth. Rows I and II are from FLIR-Thermal dataset while III and IV are from L515-Indoor dataset. Row IV represents a true sensor failure case when IR camera gets saturated due to sun-light even in indoors.

TABLE III
PERFORMANCE EVALUATION ON L515-INDOOR DATASET.

Model w/ RSH	w/o Random Shadows and Highlights					w/ RSH
	AP@IoU=0.5				mAP@ IoU=0.5	mAP@ IoU=0.5
	Chair	Cycle	Bin	Laptop		
rgb-only	0.909	0.912	0.920	0.911	0.912	0.769
ir-only	0.141	0.557	0.012	0.690	0.386	0.311
g_{sa} m-net	0.851	0.895	0.705	0.740	0.811	0.685
g_{sa}	0.968	0.998	0.997	0.979	0.982	0.945

to distinguish large sized objects even if their contribution in the dataset is relatively small e.g., *Baththub* and *Bed* classes. This problem can be traced back to [10] which itself struggles to perform equally on detecting small sized objects.

L515-Indoor: Table III presents the evaluation results of L515-Indoor dataset. We tested the performance RGB-only and IR-only networks, as well as the g_{sa} variant of GEM on this dataset. Evidently, g_{sa} outperformed both single modality detectors providing an additional functionality of switching between the dominant sensors in changing lighting conditions. The performance of g_{sa} with MobileNet v2 backbone is also presented in the table. The qualitative results on all three datasets are shown in Figures 5 and 6.

MobileNet v2: On a mobile platform having a 12-Core Intel® Core™ i7-9750H CPU, and Nvidia GeForce RTX 2080 GPU, with ResNet-50 backbones, it takes approx 106.0 ms for a single forward pass on the proposed multi-modal object detector. However, with MobileNet v2 backbone feature extractors, the time for a single forward pass reduces to 49.7 ms obtaining approximately 20.1 fps. The drop in prediction accuracy of the deep models with the decrease in the number of network parameters for faster detection speed, is a well-known dilemma (e.g., in our case 23 million parameters for ResNet-50 to 3.4 million parameters for MobileNet v2). A compromise on prediction accuracy should only be made in non-critical cases where human safety is not at stake. Otherwise, the use of lightweight backbones should be avoided

Fig. 6. (a) Sample output of GEM (g_{sc}) on the SUNRGB-D dataset with RGB images and "r-blended" depth modality. In (b), the output of single modal object detector trained only on RGB images is shown, while (c) is the ground truth.

V. CONCLUSION

In this paper, we propose GEM, a novel sensor-aware multi-modal object detector, with immunity against adverse lighting scenarios. Among the proposed sensor fusion configurations, the scalar averaging variant of the deterministic weighted fusion outscored the state-of-the-art and other fusion methods. The mask multipliers may amplify a certain spatial portion of the feature maps, but could also potentially distort the learned features depth-wise. Concatenation might be useful in cases where the feature spaces of the utilized sensor modalities differ. Regarding RGB-D data, the proposed "r-blended" hybrid depth modality has proven to be a promising and lightweight alternative to the commonly employed HHA-encoded depth images. However, instead of employing a fixed blend weight α, dynamic adaptation driven by ambient light intensity could demonstrate a more realistic use of the proposed hybrid image. GEM brings along the shortcomings of [10] in multi-modal object detection setting as well, e.g., it struggles to detect small objects and suffers from the computational complexity of the attention layers. These issues will be addressed in the future work.

REFERENCES

[1] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Proc. of the IEEE Intl. Conf. on Computer Vision*. 2017, pp. 2980–2988.

[2] Wei Liu et al. "SSD: Single Shot Multibox Detector". In: *European Conf. on Computer Vision*. Springer. 2016, pp. 21–37.

[3] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767* (2018).

[4] Shaoqing Ren et al. "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks". In: *arXiv preprint arXiv:1506.01497* (2015).

[5] Dong Yin et al. "A Fourier Perspective on Model Robustness in Computer Vision". In: *arXiv preprint arXiv:1906.08988* (2019).

[6] Dan Hendrycks et al. "The Many Faces of Robustness: A Critical Analysis of Out-of-distribution Generalization". In: *arXiv preprint arXiv:2006.16241* (2020).

[7] Osama Mazhar and Jens Kober. "Random Shadows and Highlights: A New Data Augmentation Method for Extreme Lighting Conditions". In: *arXiv preprint arXiv:2101.05361* (2021).

[8] Mario Bijelic et al. "Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather". In: *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*. 2020, pp. 11682–11692.

[9] Clara Marina Martinez et al. "Feature Uncertainty Estimation in Sensor Fusion Applied to Autonomous Vehicle Location". In: *Intl. Conf. on Information Fusion*. 2017, pp. 1–7.

[10] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *arXiv preprint arXiv:2005.12872* (2020).

[11] Licheng Jiao et al. "A Survey of Deep Learning-based Object Detection". In: *IEEE Access* 7 (2019), pp. 128837–128868.

[12] Li Liu et al. "Deep Learning for Generic Object Detection: A Survey". In: *Intl. journal of Computer Vision* 128.2 (2020), pp. 261–318.

[13] Fernando Garcia et al. "Sensor Fusion Methodology for Vehicle Detection". In: *IEEE Intelligent Transportation Systems Magazine* 9.1 (2017), pp. 123–133.

[14] Johann Weichselbaum et al. "Accurate 3D-vision-based Obstacle Detection for An Autonomous Train". In: *Computers in Industry* 64.9 (2013), pp. 1209–1220.

[15] Eunbyung Park et al. "Combining Multiple Sources of Knowledge in Deep CNNs for Action Recognition". In: *2016 IEEE Winter Conf. on Applications of Computer Vision (WACV)*. IEEE. 2016, pp. 1–8.

[16] Guido Borghi et al. "Poseidon: Face-from-depth for Driver Pose Estimation". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2017, pp. 4661–4670.

[17] Tran Tuan Nguyen et al. "Multisource Fusion for Robust Road Detection using Online Estimated Reliabilities". In: *IEEE Transactions on Industrial Informatics* 14.11 (2018), pp. 4927–4939.

[18] Luca Caltagirone et al. "LIDAR-Camera Fusion for Road Detection using Fully Convolutional Neural Networks". In: *Robotics and Autonomous Systems* 111 (2019), pp. 125–131.

[19] Chaitanya Devaguptapu et al. "Borrow from Anywhere: Pseudo Multi-modal Object Detection in Thermal Imagery". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshops*. 2019.

[20] Phillip Isola et al. "Image-to-image Translation with Conditional Adversarial Networks". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2017, pp. 1125–1134.

[21] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised Image-to-image Translation Networks". In: *arXiv preprint arXiv:1703.00848* (2017).

[22] *FLIR Thermal Dataset*. https://www.flir.eu/oem/adas/adas-dataset-form/.

[23] "Multispectral Pedestrian Detection: Benchmark Dataset and Saseline". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2015, pp. 1037–1045.

[24] Qadeer Khan, Torsten Schön, and Patrick Wenzel. "Towards Self-supervised High Level Sensor Fusion". In: *arXiv preprint arXiv:1902.04272* (2019).

[25] Naman Patel et al. "Sensor Modality Fusion with CNNs for UGV Autonomous Driving in Indoor Environments". In: *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. 2017, pp. 1531–1536.

[26] Changhao Chen et al. "Selective Sensor Fusion for Neural Visual-Inertial Odometry". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2019, pp. 10542–10551.

[27] Charles R Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[28] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparameterization with Gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[29] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite". In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. 2015, pp. 567–576.

[30] Saurabh Gupta et al. "Learning rich features from RGB-D images for object detection and segmentation". In: *European Conf. on Computer Vision*. Springer. 2014, pp. 345–360.

[31] Andrew G Howard et al. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". In: *arXiv preprint arXiv:1704.04861* (2017).

7.12 Random Shadows and Highlights: A new data augmentation method for extreme lighting conditions

The appended paper [132] follows.

RANDOM SHADOWS AND HIGHLIGHTS: A NEW DATA AUGMENTATION METHOD FOR EXTREME LIGHTING CONDITIONS

Osama Mazhar, Jens Kober

Cognitive Robotics Department
Delft University of Technology
Delft, The Netherlands

ABSTRACT

In this paper, we propose a new data augmentation method, Random Shadows and Highlights (RSH) to acquire robustness against lighting perturbations. Our method creates random shadows and highlights on images, thus challenging the neural network during the learning process such that it acquires immunity against such input corruptions in real world applications. It is a parameter-learning free method which can be integrated into most vision related learning applications effortlessly. With extensive experimentation, we demonstrate that RSH not only increases the robustness of the models against lighting perturbations, but also reduces over-fitting significantly. Thus RSH should be considered essential for all vision related learning systems. Code is available at: https://github.com/OsamaMazhar/Random-Shadows-Highlights.

***Index Terms*—** Robotic vision, Harsh conditions, Generalization, Data augmentation, Convolutional Neural Networks

1. INTRODUCTION

It is often claimed that the performance of deep learning algorithms have attained super-human capabilities. However, real-world effectiveness of deep learning computer vision algorithms often fail to match the published performance on benchmarks [1]. The vision models are largely fragile and do not generalize across realistic unconstrained scenarios [2], e.g., when they encounter instances of classes, textures, or environmental conditions that were not covered by the training data [3]. This could have serious consequences where the failures could lead to potentially catastrophic results as in the case of self-driving vehicles.

The authors of [4] studied the existing hypotheses concerning the methods to improve the robustness of the deep networks. Of particular relevance to our work, the technique of perturbing data without altering class labels, also known as data augmentation, has been proven to greatly improve model robustness and generalization performance [5, 6]. It artificially inflates a dataset through label-preserving transforms to derive new samples from the originals. Data augmentation offers straightforward strategies to learn invariances that are challenging to encode architecturally. The most prevalent forms of data augmentation methodologies rely mostly on hand crafted features or include geometric distortions such as random cropping, zooming, rotation and flipping [7]. Other methods include color space transformation, linear intensity scaling and elastic deformation. These methods are successful at teaching mainly orientation and scale invariance, but are insufficient for other practical cases like occlusion, texture and complex illumination variations. This is however often achieved by domain randomization on rendered images in simulation for Sim2Real transfer [8].

Fig. 1. Samples of our proposed Random Shadows and Highlights (RSH) data augmentation scheme.

Extreme lighting conditions such as dark shadows, intense highlights or change in brightness substantially affect the performance of deep models in image classification or object detection tasks. To address such illumination variations and bridging the reality gap, this paper introduces a new data augmentation methodology, Random Shadows and Highlights (RSH) for real camera images. The proposed strategy mimics high-contrast extreme lighting conditions by creating random shadows and highlights in the images. RSH is

This research is funded by H2020 EU project OpenDR, Grant Agreement No. 871449.

a parameter-learning free lightweight method which can be integrated with most vision related learning models without the need to change the learning strategy. The proposed data augmentation method is utilized to imitate the limitations of conventional RGB sensors to handle extreme light conditions.

2. RELATED WORKS

Data augmentation is a strategy that aims to improve robustness of the models to unforeseen data shifts which they might encounter in real-world. It explicitly teaches invariance to whichever transformation is used on the dataset. For example, robustness against occlusions can be learned through Cutout [9] and Random Erasing [10]. Cutout randomly masks out square regions of input to generate new images to simulate occluded examples. Random Erasing additionally varies size and aspect ratio of the masked region. It also allows to replace pixel values with random noise. Instead of occluding an image, CutMix [11] substitutes a portion of image with a portion of another image while the ground truth labels are also mixed proportionally to the area of the patches. Mixup [12] is a another regularization strategy that produces elementwise convex combination of two images. AugMix [13] mixes together the results of several shorter augmentation chains in convex combinations to prevent image degradation while maintaining augmentation diversity. Alpha-blending two images generate pixel-level features that a camera might never produce and that could potentially affect model performance. To prevent this, RICAP [14] spatially blend four cropped images to create a new image. It performs occupancy estimation instead of classification, by mixing the four class labels with ratios proportional to the areas of the cropped images. The method proposed in [15] is closest to our proposed strategy. Nevertheless, they only create illumination circles on the images which does not represent the real world lighting perturbations often seen in indoor environments or even outdoors caused by shadows of the buildings. Separate from hand-crafted methodologies are learned augmentation methods such as AutoAugment [16], patch Gaussian augmentation [6], learned image transformations [5] and DeepAugment [4].

3. DATASETS

For **image classification**, we evaluate RSH on Tiny-ImageNet [17], CIFAR-10 and CIFAR-100 [18]. Tiny-ImageNet is a miniature version of the ImageNet dataset with 200 classes. Each class contains 500 training and 50 validation images yielding 100,000 training and 10,000 validation samples. All images are of size $64 \times 64 \times 3$. CIFAR-10 consists of 60,000 colour images in 10 classes with images of objects like airplanes, birds, trucks, etc. Each class consists of 6,000 images. The training set contains 50,000 images while 10,000 images are in the validation set. CIFAR-100 is similar to CIFAR-10 except that it has 100 classes with 600 images in each class. Each image comes with a fine and a coarse label. For **object detection**, we utilize the PASCAL VOC 2007 dataset [19] which provides 9,963 images of realistic scenes containing 24,640 annotated objects. The dataset is split into 50% for training/validation and 50% for testing. There are 20 classes of objects which are almost equally distributed in the images.

4. OUR APPROACH

Shadows cast by buildings in outdoors often take the form of trapezoids. Furthermore, luminous light entering through windows or doors inside the buildings also take similar shapes. We devise an algorithm to create Random Shadows and Highlights on images imitating trapezoidal shape with its base aligned with the vertical axis of the image. For an image I, Random Shadows and Highlights is applied with a probability p in training. For probability $1 - p$, the image is left unchanged. RSH is driven by twelve input parameters to create random shadows and highlights. These variables

Algorithm 1: Random Shadows and Highlights

Input: Input image I;
Image size W, H and c;
RSH probability p;
Highlights range H_l, H_h;
Shadows range S_l, S_h;
Left edge ranges l_{ul}, l_{uh} and l_{ll}, l_{lh};
Right edge ranges r_{ul}, r_{uh} and r_{ll}, r_{lh}.
Output: Transformed image I^*
Initialization: $p_1 \leftarrow \text{Rand}(0, 1)$.

if $p_1 \geq p$ **then**
  $I^* \leftarrow I$;
  return I^*.
else
  $L_{lf} \leftarrow \text{Rand}(l_{ll}, l_{lh}) \times H$;
  $L_{hf} \leftarrow \text{Rand}(l_{ul}, l_{uh}) \times H$;
  $R_{lf} \leftarrow \text{Rand}(r_{ll}, r_{lh}) \times H$;
  $R_{hf} \leftarrow \text{Rand}(r_{ul}, r_{uh}) \times H$;
  $P_{bl} \leftarrow (0, L_{lf} + L_{hf})$;
  $P_{tl} \leftarrow (0, L_{hf})$;
  $P_{br} \leftarrow (W, R_{lf} + R_{hf})$;
  $P_{tr} \leftarrow (W, R_{hf})$;
  $I_{mask} \leftarrow$
   $\text{ContourFill}(\text{Zeros}(H, W, c), [P_{tl}, P_{tr}, P_{br}, P_{bl}])$;
  $I_{inv\_mask} \leftarrow \neg I_{mask}$;
  $S_f \leftarrow \text{Rand}(S_l, S_h)$;
  $H_f \leftarrow \text{Rand}(H_l, H_h)$;
  $I_{shadow} \leftarrow \text{adjustBrightness}(I, S_f)$;
  $I_{high} \leftarrow \text{adjustBrightness}(I, H_f)$;
  $I^* \leftarrow I_{shadow}(I_{mask}) + I_{high}(I_{inv\_mask})$;
  return I^*.
end

include highlights range H_l, H_h and shadows range S_l, S_h. These also include left edge ranges l_{ul}, l_{uh} and l_{ll}, l_{lh}, and right edge ranges r_{ul}, r_{uh} and r_{ll}, r_{lh}, to randomly obtain four points in total i.e., two on each vertical axis of the image which are subsequently employed to draw a trapezoid. For an input image of size $W \times H$, our algorithm randomly creates a contour on the image driven by the provided parameters ranges. The brightness of the pixels inside the contour is reduced by a factor S_f which is randomly chosen within the predefined range S_l and S_h. Similarly, the brightness of the pixels outside the contour is increased by a factor H_f obtained from within H_l and H_h. The detailed procedure of creating the contour mask(s) and ultimately creating an image with shadows and highlights is shown in Alg 1.

5. EXPERIMENTS

5.1. Image Classification

Through extensive experimentation, we study the impact of Random Shadows and Highlights (RSH) in comparison to a set of related augmentations methods. Among the commonly employed strategies, the most relevant ones to our work are *Gamma Correction* and *Color Jitter*. Varying powers of gamma darkens or lightens the shadows in the image. While in Color Jitter, we randomly change the brightness, contrast, saturation and hue of the image. Moreover, we also compare our results with the method proposed in [15], we call it *Disk Illumination*. To study over-fitting, we quantify the Train-Test Difference (TTD), as the name suggests, difference in error on the train set and test set, with and without RSH perturbations. When lighting corruption is applied to the test set, its probability is set to 1. The RSH parameters which were chosen in our experiments are: $H_l = 1$, $H_h = 2$, $S_l = 0$, $S_h = 1$, l_{ul} and $r_{ul} = 0$, l_{uh} and $r_{uh} = 0.3$, l_{ll} and $r_{ll} = 0.4$, l_{lh} and $r_{lh} = 0.8$. Gamma is randomly changed from 0 to 1.5 while brightness, contrast and saturation ranges are 0 to 2, and hue changes from -0.5 to 0.5 randomly.

Architectures: We adopted two architectures on Tiny-ImageNet, CIFAR-10 and CIFAR 100: EfficientNet [20] and AlexNet [21]. More precisely, we employed the EfficientNet-B0 architecture. The models are pre-trained on the ImageNet dataset except the *SoftMax* layer which is adopted for each dataset and initialized with random weights. The training is performed only for 20 epochs at a learning rate of $1e^{-3}$ and a momentum of 0.9 with the *SGD* optimizer.

5.1.1. Classification Evaluation

The performances of the selected data augmentation strategies are evaluated at different activation probabilities p, ranging from 0.0 to 1.0 with a step size of 0.1. Table 1 presents the results of these experiments at $p = 0.5$ and $p = 1$ only. At a first glance, it may appear that the performance with RSH decreases slightly on test sets when no lighting perturbations are applied. To some extent, this is true and is justified by the observations made in [22] that argues that standard accuracy is often compromised to obtain robust models. However, when tested with lighting corruptions, our strategy out-scores all other methods with a considerable margin by obtaining least test errors at both probabilities. In fact, Figure 2 illustrates that RSH is extremely robust against lighting perturbations at all values of p. Table 1, also shows that for lighting perturbations at $p = 1$ on the test set, the performance of the model trained with RSH is increased by approximately 25% at both $p = 0.5$ and $p = 1$ when compared to RGC which performed best among other strategies. Furthermore, we estimate the Train-Test Difference (TTD) to measure over-fitting for cases with and without lighting corruption. In addition to the increased robustness, our RSH augmentation method proves to have reduced over-fitting significantly as well.

Method	Train Error	Test Error	Test Error RSH (1)	TTD w/o RSH	TTD RSH (1)
Baseline	0.303	0.408	0.630	0.105	0.327
RGC (0.5)	0.301	0.396	0.603	0.095	0.302
RCJ (0.5)	0.415	0.406	0.612	−0.009	0.197
RDI (0.5)	0.493	0.410	0.627	−0.083	0.134
RSH (0.5)	0.353	0.407	**0.449**	**0.054**	**0.096**
RGC (1)	0.323	0.400	0.600	0.077	0.277
RCJ (1)	0.485	0.459	0.656	−0.026	0.171
RDI (1)	0.639	0.631	0.780	−0.008	0.141
RSH (1)	0.411	0.434	**0.450**	**0.023**	**0.039**

Table 1. Performance evaluation of EfficientNet-B0 on Tiny-ImageNet. For comparison, models are trained with Random Gamma Correction (RGC), Random Color Jitter (RCJ) and Random Disk Illumination (RDI) as well. Apart from train and test error computation, performance is measured with lighting perturbations as well by applying RSH in the test set with $p = 1$. Moreover, the Train-Test Difference (TTD) is also computed to quantify over-fitting of the models. Negative TTD implies over-regularization which is not desired.

Impact of hyperparameters: We also conduct experiments to study the impact of RSH hyperparameters on the model performance. Figure 3 illustrates the results obtained when training AlexNet on CIFAR-10 with increasing RSH probability from 0.0 to 1.0. The probability of lighting corruption is set to 1 in the test set. It is evident, that the model overfits severely when no RSH augmentation is applied during training. Nevertheless, RSH p increases in the training, robustness of the network increases continuously until we obtain almost the same train-test prediction accuracy, which is desired in most cases. We also evaluate the impact of changing other hyperparameters. Figure 4 illustrates that increasing the range of highlights and shadows improves not only the accuracy of the model, but enhances the TTD by 3.125% as well. The significance of the "shadows area" is also assessed

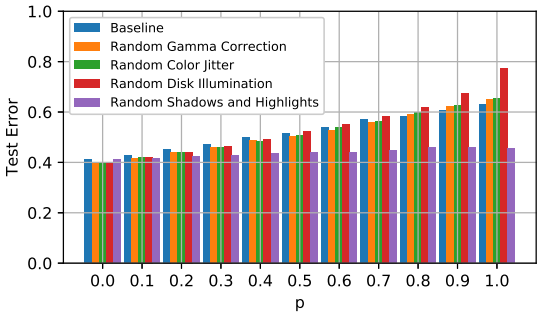


Fig. 2. Test error comparison between different data augmentation strategies for different levels of lighting perturbations. p represents data augmentation activation probabilities for each individual technique in training as well as that of lighting perturbations in test set.

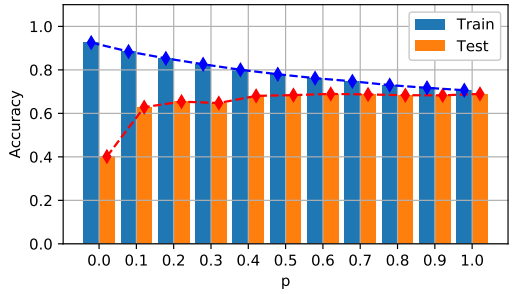


Fig. 3. Performance evaluation on CIFAR-10 when trained with different activation probabilities of RSH. The probability of lighting corruptions is set to 1 for all values of p. It is evident that with the increase in p, over-fitting decreases continuously until the model fits almost perfectly for both train and test sets.

on CIFAR-100 by changing the values of l_{lh}, r_{lh}, l_{ul} and r_{ul} while keeping l_{ll}, r_{ll}, l_{uh} and r_{uh} constant. A marginal improvement of 1.7% is observed with the values presented in Section 5.1 as compared to the the least performing ones.

5.2. Object Detection

In our experiments for object detection, we employ a single-stage end-to-end object detector DETR proposed in [23]. It is among the pioneering works that exploited transformers in the image domain. However, DETR technically is still a combination of CNN and self-attention. We utilize ResNet-50 as the backbone features extractor while the object detector head is adopted to suit the number of classes in the Pascal VOC 2007 dataset. The weights are initialized from a checkpoint pretrained on the COCO object dataset. Thus, the hyperparameters of the transformer in DETR are set to default with 6 encoder and decoder layers, 8 attention heads, and 100 number of queries. The learning rate for both backbone and transformer layers is set to 8×10^{-6} while training is terminated after 150 epochs for each experiment. We perform experiments with and without Random Shadows and Highlights augmentation on the Pascal VOC 2007 dataset.

Model	mAP@IoU=0.5			TTD w/o RSH	TTD w/ RSH (1)
	Train	Test	Test w/ RSH (1)		
Baseline	0.888	0.751	0.639	0.137	0.249
RSH (1)	0.872	**0.751**	**0.705**	**0.121**	**0.167**

Table 2. Performance evaluation of our RSH augmentation strategy for object detection on the Pascal VOC 2007 dataset.

5.2.1. *Detection Evaluation*

The performance evaluation of DETR with RSH on the Pascal VOC 2007 dataset is presented in Table 2. For each setting, mean of 20 evaluation trials is computed. It is evident from the results that the model trained with our RSH augmentation not only maintained its standard accuracy but also improved the robustness against lighting corruption by 10.328% compared to the baseline. Clear improvements in TTD with and without RSH perturbations in the test set can also be seen.

6. CONCLUSION

A new lightweight learning-free data augmentation strategy is presented in this paper to acquire robustness against lighting corruption. This is achieved by creating random shadows and highlights on the images during training. The models trained with the proposed augmentation method not only out-scored other similar methods against lighting perturbations but also reduced over-fitting on the training data significantly. The results in the performed experiments demonstrate that RSH should be considered essential in CNN model training. In the future, we plan to apply our method on other applications, e.g, facial recognition and person re-identification.

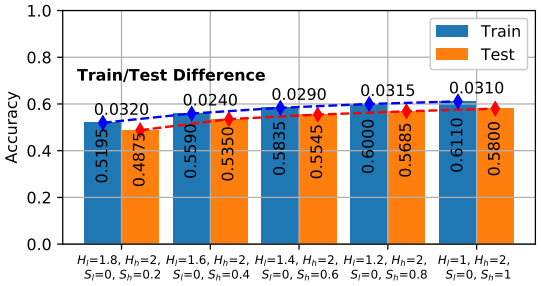


Fig. 4. Impact of changing H and S hyperparameters. For all cases we set $H_h = 2$ while $S_l = 0$. The X axis represents increasing span of choice for highlights and shadows. Clearly, with wider range, RSH performs best in terms of accuracy.

7. REFERENCES

[1] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, et al., "The limits and potentials of deep learning for robotics," *The Intl. Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[2] D. Yin, R. Gontijo Lopes, Jon S., Ekin D. C., and J. Gilmer, "A Fourier perspective on model robustness in computer vision," in *Advances in Neural Information Processing Systems*, 2019, pp. 13276–13286.

[3] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide, "Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020, pp. 11682–11692.

[4] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, et al., "The many faces of robustness: A critical analysis of out-of-distribution generalization," *arXiv preprint arXiv:2006.16241*, 2020.

[5] B. Zoph, E. D. Cubuk, G. Ghiasi, T. Lin, J. Shlens, and Q. V. Le, "Learning data augmentation strategies for object detection," in *European Conf. on Computer Vision*. Springer, 2020, pp. 566–583.

[6] R. G. Lopes, D. Yin, B. Poole, J. Gilmer, and E. D. Cubuk, "Improving robustness without sacrificing accuracy with patch Gaussian augmentation," *arXiv preprint arXiv:1906.02611*, 2019.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[9] T. DeVries and G. W Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[10] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *Proc. of the AAAI Conf. on Artificial Intelligence*, vol. 34, no. 07, pp. 13001–13008, Apr. 2020.

[11] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2019, pp. 6023–6032.

[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[13] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," *arXiv preprint arXiv:1912.02781*, 2019.

[14] R. Takahashi, T. Matsubara, and K. Uehara, "Data augmentation using random image cropping and patching for deep CNNs," *IEEE Trans. on Circuits and Systems for Video Technology*, 2019.

[15] D. Sakkos, H. P. Shum, and E. S. Ho, "Illumination-based data augmentation for robust background subtraction," in *13th Intl. Conf. on Software, Knowledge, Information Management and Applications (SKIMA)*. IEEE, 2019, pp. 1–8.

[16] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[17] Y. Le and X. Yang, "Tiny ImageNet Visual Recognition Challenge," `https://tiny-imagenet.herokuapp.com/`, 2015.

[18] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," *Tech Report*, 2009.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[20] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[22] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," *arXiv preprint arXiv:1805.12152*, 2018.

[23] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *arXiv preprint arXiv:2005.12872*, 2020.