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 paper 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 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 [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 $\boldsymbol{x}_i \in \Re^D$ 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 $\mathcal{X} \subseteq \Re^D$ and an output space $\mathcal{F} \subseteq \Re^C$, 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:

$$\begin{aligned} QMI(Y,C) = \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c)^2 d\boldsymbol{y} + \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y})^2 P(c)^2 d\boldsymbol{y} \\ -2 \sum_{c=1}^{N} \int_{\boldsymbol{y}} p(\boldsymbol{y},c) p(\boldsymbol{y}) P(c) d\boldsymbol{y}. \end{aligned} \tag{6}$$

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:

$$K_P = \|\mathbf{y}_i - \mathbf{y}_j\|^d \tag{11}$$

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.

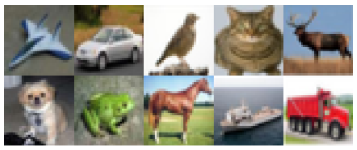


Figure 1: Sample images of the Cifar-10 dataset.

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.

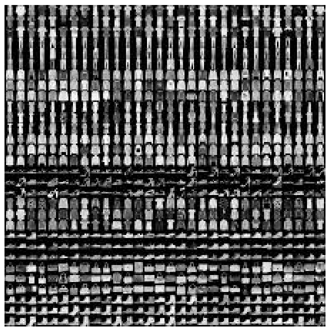


Figure 3: Sample images of the Fashion MNIST dataset.

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

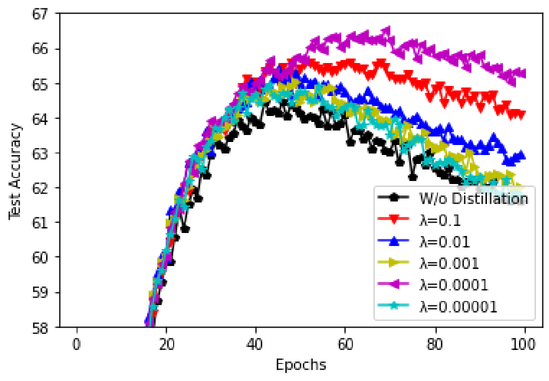


Figure 6: On the parameter λ in eq. (12).

.

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

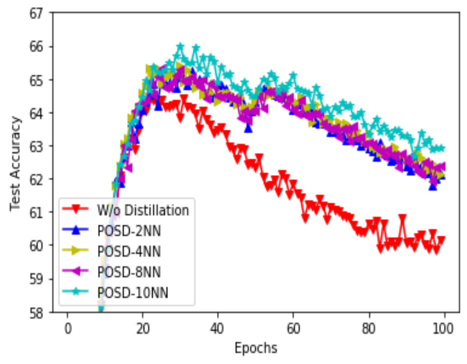
Method	Cifar-10	SVHN-10	Fashion MNIST
W/o Distillation	64.734% ± 0.654%	88.706% ± 0.306%	91.214% ± 0.141%
POSD-2NN	65.472% ± 0.914%	89.625% ± 0.307%	91.624% ± 0.111%
POSD-4NN	**66.782% ± 0.691%**	89.534% ± 0.500%	91.650% ± 0.138%
POSD-8NN	66.140% ± 1.013%	**89.912% ± 0.250%**	**91.730% ± 0.157%**
POSD-10NN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

Table 3: Test Accuracy - Batch Size: 64

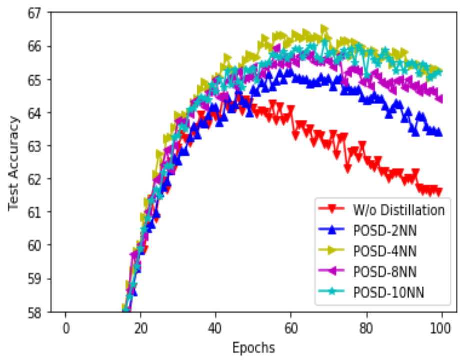
Method	Cifar-10	SVHN-10	Fashion MNIST
W/o Distillation	65.048% ± 0.620%	88.013% ± 0.083%	91.058% ± 0.130%
POSD-2NN	66.248% ± 0.592%	89.387% ± 0.433%	**91.728% ± 0.175%**
POSD-4NN	66.362% ± 0.726%	**89.946% ± 0.433%**	91.636% ± 0.154%
POSD-8NN	**66.760% ± 0.680%**	89.491% ± 0.363%	91.724% ± 0.121%
POSD-10NN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

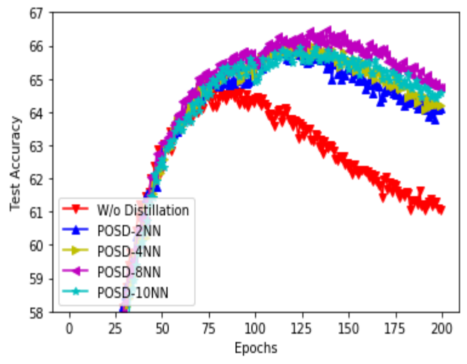


(a) Cifar-10: Batch Size of 32 samples.

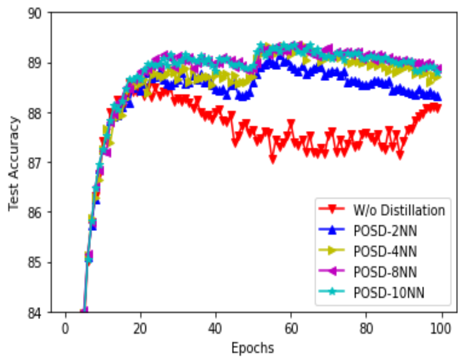


(b) Cifar-10: Batch Size of 64 samples.

Figure 7: Evaluating the test accuracy during the training process for different methods.



(a) Cifar-10: Batch Size of 128 samples.

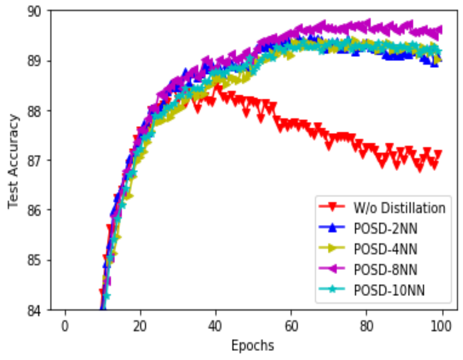


(b) SVHN-10: Batch Size of 32 samples.

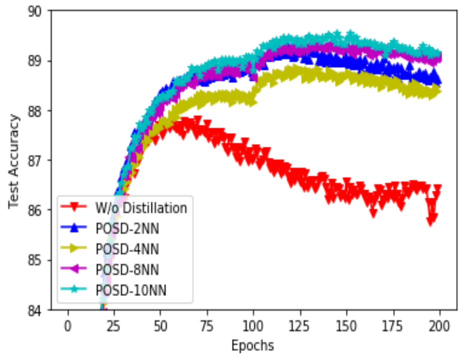
Figure 8: Evaluating the test accuracy during the training process for different methods

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-

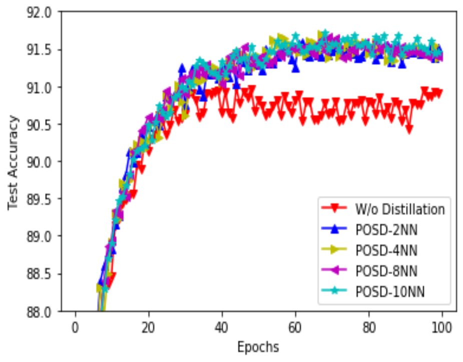


(a) SVHN-10: Batch Size of 64 samples.

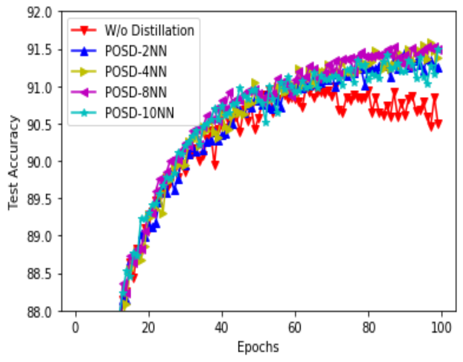


(b) SVHN-10: Batch Size of 128 samples.

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

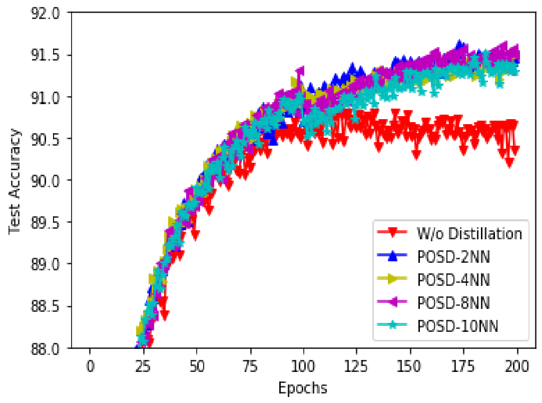


Figure 11: Fashion MNIST: Evaluating the test accuracy during the training process for different methods, batch Size of 128 samples.

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

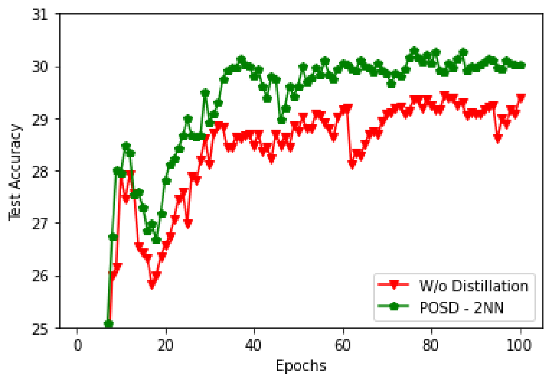


Figure 12: Tiny-ImageNet: Evaluating the test accuracy during the training process for the baseline and training with the POSD methods with the optimal number of NNs.

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.

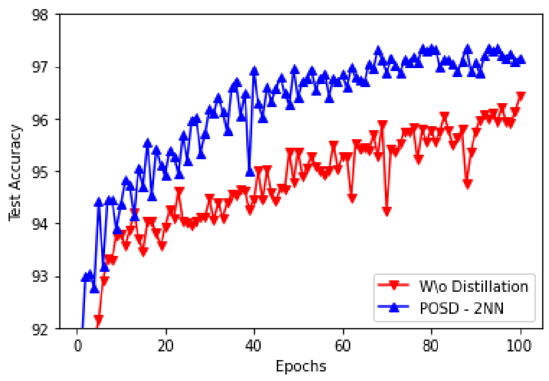


Figure 13: Crowd-drone dataset: Evaluating the test accuracy during the training process for the baseline and training with the POSD methods with the optimal number of NNs.

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. Buciluă, 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.