Recurrent Bag-of-Features for Visual Information Analysis

Marios Krestenitis[a,*], Nikolaos Passalis[a], Alexandros Iosifidis[c], Moncef Gabbouj[b], Anastasios Tefas[a]

[a]*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*
[b]*Faculty of Information Technology and Communication, Tampere University, Tampere, Finland*
[c]*Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark*

Abstract

Deep Learning (DL) has provided powerful tools for visual information analysis. For example, Convolutional Neural Networks (CNNs) are excelling in complex and challenging image analysis tasks by extracting meaningful feature vectors with high discriminative power. However, these powerful feature vectors are crushed through the pooling layers of the network, that usually implement the pooling operation in a less sophisticated manner. This can lead to significant information loss, especially in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., videos, which often require extracting fine-grained temporal information. A novel stateful recurrent pooling approach, that can overcome the aforementioned limitations, is proposed in this paper. The proposed method is inspired by the well-known Bag-of-Features (BoF) model, but employs a stateful trainable recurrent quantizer, instead of plain static quantization, allowing for efficiently processing sequential data and encoding both their temporal, as well as their spatial aspects. The effectiveness of the proposed Recurrent BoF model to enclose spatio-temporal information compared to other competitive methods is demonstrated using six different datasets and two different tasks.

Keywords: Bag-of-Features, Recurrent Neural Networks, Pooling Operators, Activity Recognition

*Corresponding Author
Email addresses: `mikreste@csd.auth.gr` (Marios Krestenitis), `passalis@csd.auth.gr` (Nikolaos Passalis), `alexandros.iosifidis@eng.au.dk` (Alexandros Iosifidis), `moncef.gabbouj@tuni.fi` (Moncef Gabbouj), `tefas@csd.auth.gr` (Anastasios Tefas)

1. Introduction

A vast variety of visual information analysis techniques has been proposed in the field of computer and robotics vision, as one of the most active and continuously expanding research fields. The pipeline of a typical visual information analysis method consists of two fundamental information processing steps: a) feature extraction, which typically extracts lower level information from small spatial or temporal segments of the data, and b) feature aggregation, which fuses the information extracted during the previous step into a compact representation that can be used for various tasks, e.g., classification, retrieval, etc. For example, handcrafted feature vectors can be used to describe local image regions [1], while the set of the extracted features can be then aggregated into a more compact representation using feature aggregation methods, such as the Bag-of-Feature model [2] and Vectors of Locally Aggregated Descriptors (VLAD) [3].

With the advent of deep learning (DL) these two steps were, to some extent, unified and replaced with deep trainable feature extraction layers, e.g., convolutional layers, that are combined with naive pooling operators, e.g., max or average pooling, to lower the complexity of the model and provide translation invariance. Indeed, the outstanding performance of Convolutional Neural Networks (CNNs) in complex and challenging image analysis tasks, has confirmed their ability to extract meaningful feature vectors with high discriminative power for a wide variety of different computer vision tasks [1]. However, these powerful feature vectors are crushed through the pooling layers of the network, that usually implement the pooling operation in a less sophisticated manner, often leading to significant information loss, as further discussed in [4, 5]. This is even more evident in cases where the informative content of the data is sequentially distributed over the spatial or temporal dimension, e.g., videos, which often require extracting fine-grained temporal information, which is discarded by these pooling approaches.

The aforementioned limitations can be better understood through the following example. Consider the task of activity recognition in videos, where the action of *standing up* must be distinguished from the action of *sitting down*. By employing a deep CNN, robust feature vectors can be extracted from every video frame or a sequence of them. However, the pooling layers, as the weak point of the network, dull the expressiveness of the extracted feature vectors and produce less discriminative representations by pooling over the time dimension. For example, assume that a sequence of feature vectors are extracted from a given video instance of action *sitting down*. Let that sequence, notated as $\mathbf{S}_1 = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3]$, be composed of three feature vectors $\mathbf{a}_1 = [0, 0, 1]^T$, $\mathbf{a}_2 = [0, 1, 0]^T$, and $\mathbf{a}_3 = [1, 0, 0]^T$, which are the feature vectors that correspond to the sub-actions *standing above a chair*, *bending knees* and *sitting on a chair*, respectively. Similarly, consider the same feature vector sequence, but in reverse order, i.e., $\mathbf{S}_2 = [\mathbf{a}_3, \mathbf{a}_2, \mathbf{a}_1]$, that represents a video instance of the activity *standing up*. Also, let $\mathbf{s}_i$ denote the aggregated representation extracted for the i-th video. Note that when average or max pooling is applied over both sequences, then the same representation is extracted for both videos i.e., $\mathbf{s}_1 = \mathbf{s}_2 = [\max_i[\mathbf{a}_i]_1, \max_i[\mathbf{a}_i]_2, \max_i[\mathbf{a}_i]_3]^T = [1, 1, 1]^T$ for max pooling (where the notation $[\mathbf{x}]_i$ is used to refer to the i-th element of vector $\mathbf{x}$) or $\mathbf{s}_1 = \mathbf{s}_2 = \frac{1}{3}\sum_{i=1}^{3} \mathbf{a}_i = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]^T$ for average pooling. Therefore, even though the employed CNN was capable of perfectly recognizing the fundamental sub-actions from still frames, the resulting deep model cannot discriminate between the two actions due to the employed pooling layer. Based on this observation, in this work we characterize the average and max pooling layers as *weak pooling* layers, since they are not capable of capturing the fine-grained spatial or temporal interactions between the feature vectors extracted from a given sequence. Note that in other cases, instead of employing a pooling layer, the extracted feature map can be flattened to a vector and fed to the fully connected layer. However, this approach makes it impossible for the network to handle inputs of arbitrary size, while it significantly reduces the invariance of the network to temporal or spatial translations (the features must always arrive at the exact same moment or position).

Before delving into the details of the proposed pooling method that is capable of overcoming the aforementioned limitations, it is worth revisiting a well-known feature aggregation model, the Bag-of-Features (BoF) model [2], also known as Bag-of-Visual-Words (BoVW). BoF has been extensively used in a wide range of machine learning and computer vision problems [6, 7, 8]. Initially inspired by the Bag-of-Words (BoW) model, which was designed for text analysis, BoF creates a constant-length representation of a multimedia object, e.g., video, audio, time-series, etc., by compiling a histogram over its quantized feature vectors. Assuming a set of objects is given, the usual framework of BoF is as follows:

1. At first, a feature extraction procedure takes place, where every input object is translated to a set of feature vectors. This set defines the *features space*, where every object is projected into.

2. A set of representative feature vectors is learned, which is known as *dictionary* or *codebook*. Every feature vector of the codebook is called codeword, while the learning process is called

dictionary learning.

3. Every feature vector of an input object is quantized based on the learned codebook. This step is referred as *feature encoding* or *feature quantization*.
4. A histogram is created by counting the number of feature vectors that were quantized to each codeword.

Following this pipeline, every object can be represented as a fixed-length histogram over the learned codewords. The codewords/dictionary can be either learned using generative/reconstruction approaches [2, 9], or by employing discriminative dictionary learning methods [10, 11], which usually better fit classification tasks. Note that BoF can be also combined with deep neural networks to provide more powerful trainable pooling layers that can better withstand distribution shifts, while handling inputs of various sizes [12].

Despite the remarkable results in various tasks and the ability to handle variable size inputs, the main drawback of BoF-based methods is the loss of spatial and temporal information, i.e. their inability to capture the geometry of input data [13]. These drawbacks severely limit the ability of BoF to process temporal or sequential data, such as video data, since it is not capable of capturing the temporal succession of events. Therefore, BoF-based methods are still considered as weak pooling layers, since the quantization of feature vectors to codewords does not take into account the features' sequential order and thus the temporal information is entirely discarded. To overcome this limitation, the quantization process should take into account the order in which the features arrive, allowing for forming temporal codewords that also capture the interrelation between the feature vectors. As a result, a BoF method employing a stateful recurrent quantizer would be able to quantize the vector $\mathbf{a}_2$ (bending knees) into a different codeword depending on whether it was following the vector $\mathbf{a}_1$ (standing above a chair) or the vector $\mathbf{a}_3$ (sitting on a chair), allowing for extracting a representation that can discriminate between the standing up action from the sitting down action.

In this paper, a novel stateful recurrent pooling approach that can overcome the aforementioned limitations is proposed. The proposed method is inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach. In this way, the proposed method harness the power of a novel powerful recurrent quantization formulation in order to capture the temporal information of input data, which is crucial in classification tasks, such as activity recognition, while still maintaining all the advantages of the BoF model. The proposed Recurrent BoF (abbreviated as "ReBoF") layer is used between the last feature extraction layer and the fully connected layer of a network. Therefore, instead of using weak pooling layers, the extracted feature vectors are quantized to a number of codewords in a recurrent manner, enabling to encode the fine-grained temporal information contained in the original feature vectors. The resulting network can be trained in an end-to-end fashion using plain gradient descent and back-propagation, since the proposed ReBoF formulation is fully differentiable. This allows for building powerful deep learning models for various visual information analysis tasks, as thoroughly demonstrated in this paper. Furthermore, the proposed layer enables the network to operate with arbitrary sized inputs, while it can also reduce the size of the fully connected layer by extracting a compact, yet powerful constant length representation. Finally, it is worth noting that ReBoF can be also used for encoding the spatial information, instead of merely the temporal one. For example, the spatial information encoded in the feature vectors extracted from static images can be encoded by manipulating the extracted feature map as a sequence of feature vectors. This allows for overcoming one long-standing limitation of regular BoF formulation that led to the need of using complicated spatial segmentation schemes [13].

The rest of the paper is structured as follows. In Section 2 the related work is presented and compared to the proposed approach. Next, the ReBoF model is elaborately described in Section 3. The experimental evaluation of ReBoF over six different datasets, along with comparisons to other related techniques, are presented in Section 4. Finally, conclusions are drawn in Section 5.

2. Related Work

Over the years, Bag-of-Features has been a widely used method for image analysis tasks. Its ability to provide better scale and position invariance of local keypoint features attracted significant research attention [2]. Moreover, its capability to handle variable sized images made BoF a common choice in image classification and retrieval tasks. The remarkable results in these research areas among with its simplicity motivated researchers to employ BoF in more complex tasks, such as video activity recognition and video semantic retrieval [14, 15]. Video-based human activity recognition consists one of the most challenging problems in computer vision community, that gains increasing interest over the years. Early approaches were focused on crafting discriminative features to describe each activity, such as space-time trajectories of human joints [16]. This is not an easy task, since the extracted low- or mid-level features must be robust to noise and intra-class variations, such as viewpoint and illumination changes, different motion speeds, etc. Moreover, there are human actions that are associated with specific objects or tools, e.g., cooking, hammering, etc., distinct human poses, e.g., sitting, playing guitar, or executed in a distinguishing environment, e.g., surfing, playing tennis. These cases highlight the need for high-level descriptors that are also capable of capturing the high-level semantic information of the video. The proposed ReBoF model goes beyond these methods, since it not only provides an end-to-end trainable model that allows for extracting representations tailored exactly to the task at hand, but it is also capable of modeling the temporal information of the input feature streams.

Previous efforts tried to tackle some of these issues by exploiting the ability of BoF to process objects of various lengths and providing an orderless representation of the input local keypoint features, extracted from RBG streams, optical flow and joint annotations. Feature descriptors, such as HoG/HoF [17], and MBH [14], were used to perform feature extraction. Furthermore, Dollar et al. proposed a feature extractor ensemble [18], which among with other local features, such as Dense Trajectories (DTs), improved Dense Trajectories (iDTs), Space Time Interest Points (STIPs), and skeleton joint and body shape features [19], were combined with the BoF model to better capture the spatio-temporal information of a video. However, these methods employed BoF just as a post-processing step for aggregating the pre-computed features, while the selection of codewords is usually performed in a fully unsupervised way, leading to less discriminative representations and restricting the accuracy of these approaches.

On the other hand, the abrupt expansion of DL in a variety of computer vision problems has also been imported in activity recognition tasks [20, 21]. One of the initial approaches was to use a neural network, which was pretrained on an image classification task, to extract features from each frame and then merge them to classify the whole video [20]. Donahue et al. proposed to feed the features extracted from each frame to a recurrent layer composed of Long Short-Term Memory (LSTM) units [22], allowing for encoding the temporal dimension of the sequence. Later, more powerful methods employed convolutional networks with 3D kernels (C3D) [23, 24] or fused the RGB frame-based features with features from a stack of optical flow frames, through a two-stream network [25], allowing for effectively capturing the motion features of the video. Li et al. [26] proposed a deformable C3D network enhanced with an attention submodule in order to capture the temporal and spatial irregularities of video streams. Current state-of-the-art methods use Inflated

3D Convolutional (I3D) networks [21], that are capable of effectively combining both the RGB and optical flow streams. In spite of the impressive results, all these methods use powerful feature extractors combined with weak pooling layers.

The limitations of these approaches led to the development of more advanced pooling layers. Earlier works extended traditional feature aggregation methods into differentiable models that can be combined with DL models and trained in an end-to-end fashion, e.g., Bag-of-Features was extended to Neural BoF pooling [12], while Vector of Locally Aggregated Descriptors (VLAD) was extended into NetVLAD pooling [27]. However, these approaches still suffer from the same limitations, since they fail to capture the temporal information contained in the input feature stream. It is worth noting that even more advanced pooling approaches, such as attention-based pooling [28], which dynamically adjusts the feature aggregation on each temporal segment, and learnable pooling with context gating [29], cannot extract compact representations that can capture the temporal information. More recently proposed methods attempt to overcome this limitation by further encoding the spatial information either by employing pyramid-based pooling schemes [30], or by further extending the VLAD model to capture spatio-temporal information [31]. However, these approaches either do not employ memory or use VLAD-based features of significantly higher dimensionality compared to those typically used in BoF-based models. On the other hand, the proposed method is capable of equipping BoF with memory (by employing a context-aware recurrent quantizer), overcoming all these limitations, while keeping the ability of the BoF model to extract compact, yet discriminative, histogram-based representations.

This work aims to bridge the gap between earlier BoF approaches and the recently proposed DL models by employing a powerful recurrent BoF formulation, which can be readily combined with DL architectures, as well as effectively process sequential data. The proposed method manages to successfully adapt the BoF model to temporal tasks by overcoming the limitations of other recurrent models, e.g., LSTMs and GRUs, that often fail to correctly model the temporal information extracted from videos, as also noted in [29]. It is worth noting that previous methods successfully managed to combine neural networks with BoF, where the codewords are learned via end-to-end training of the model [12]. However, they were mainly limited to image analysis problems, while suffering from spatial information loss, due to BoF layer architecture. Furthermore, it has been shown that BoF can be modeled as a (stateless) recurrent neural network that simply sums over the quantized features [32]. In contrast to these approaches, the proposed method employs a stateful recurrent quantizer that exploits the sequential nature of the extracted features. Therefore proposed ReBoF model retains the valuable assets of BoF models, while, at the same time, it can effectively encode the spatio-temporal characteristics of the data. To the best of our knowledge, ReBoF is the first method that provides a stateful recurrent BoF formulation, which can capture the spatio-temporal dynamics of the data and be enclosed in a convolutional network, providing an end-to-end trainable DL model for several visual information analysis tasks.

3. Proposed Method

The proposed recurrent BoF model is analytically derived in Section 3.2, after briefly introducing the standard Neural BoF model in Section 3.1. Then, the suggested methodology to apply ReBoF in two use cases, i.e., video and image classification, is described in Section 3.3

3.1. Standard BoF

Assuming that a set $\mathcal{X} = \{x_i\}_{i=1}^{N}$ of N objects is given, N_i feature vectors are extracted from each object and notated as $\mathbf{x}_{ij} \in R^D (j = 1, ..., N_i)$, where D is the dimensionality of the feature space. For example, for image classification tasks each of the N images is represented by feature vectors extracted using a deep convolutional network or hand-crafted feature descriptors [1].

BoF aims to aggregate these extracted feature vectors into a fixed-length histogram. This is achieved using a two-stage procedure, where during the first stage each feature vector is quantized using a predefined number of codewords and then, during the second stage, the N_i quantized feature vectors of each object are accumulated to form the final histogram. Note that for the first stage, a codebook $\mathbf{V} \in R^{N_K \times D}$ must be employed, where N_K is the number of codewords. This codebook is usually learned by clustering all feature vectors into N_K clusters [2]. Common clustering algorithms, such as k-means, can be used in this step, with each centroid, $\mathbf{v}_k \in R^D (k = 1, ..., N_K)$, corresponding to a codeword.

Several feature quantization approaches have been proposed [2, 12]. In this work, we focus on a soft quantization approach that allows for learning the codebook using regular back-propagation, along with the rest of the parameters of the model [12]. This can significantly improve the discriminative power of the extracted representation, since the codebook is fined-tuned for the task at hand. The feature vector $\mathbf{x}_{ij}$, extracted from the i-th object, is quantized by measuring its similarity with each of the N_k codewords as following:

$$[\mathbf{d}_{ij}]_k = \exp\left(\frac{-\left\|\mathbf{v}_k - \mathbf{x}_{ij}\right\|_2}{\sigma}\right) \in [0,1], \tag{1}$$

where σ controls the fuzziness of the quantization assignment (also known as the *width* of the Gaussian kernel). Then, the fuzzy membership vector of each feature vector is obtained after normalizing the observed similarities:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\left\|\mathbf{d}_{ij}\right\|_1} \in R^{N_K}. \tag{2}$$

Finally, the normalized membership vectors are accumulated in a histogram for every object:

$$\mathbf{s}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{u}_{ij} \in R^{N_K}. \tag{3}$$

Note that the histogram $\mathbf{s}_i$ has unit l_1 norm, regardless the number of the extracted feature vectors, and provides an aggregated representation of the feature vectors extracted from the corresponding object. Then, the compiled histogram can be fed to a multilayer perceptron (MLP) to classify the object or used in other tasks, such as regression or retrieval.

3.2. Recurrent BoF

Note that the representation extracted using the standard BoF formulation described by (1), (2) and (3) completely discards any spatial or temporal information encoded by the input feature vectors. To overcome this limitation, BoF is extended by employing a recurrent stateful quantizer that can capture and effectively encode the temporal information. Note that in case of sequential data, the feature vector $\mathbf{x}_{ij}$ corresponds to the j-th timestep of the i-th sequence. As we will demonstrate in Subsection 3.3, this is without loss of generality, since the same approach can be also used to capture part of the spatial information of the input.

Before deriving a recurrent quantization approach, it is worth revisiting the quantization process involved in the BoF model from a probabilistic perspective. The probability of observing each feature vector $\mathbf{x}_{ij}$, given an input object x_i, can be trivially estimated using Kernel Density Estimation [9] as:

$$p(\mathbf{x}_{ij}|x_i) = \sum_{k=1}^{N_K}[\mathbf{s}_i]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \in [0,1], \tag{4}$$

where $K(\cdot)$ is a kernel function and the histogram (image-specific parameters) $\mathbf{s}_i \in \mathbb{R}^{N_K}$ separately adjust the density estimation. Then, the histogram can be calculated using a maximum likelihood estimator:

$$\mathbf{s}_i = \arg\max_{\mathbf{s}} \sum_{j=1}^{N_i} \log \left(\sum_{k=1}^{N_K} [\mathbf{s}]_k K(\mathbf{x}_{ij}, \mathbf{v}_k) \right). \tag{5}$$

It can be easily shown [9], that these parameters can be estimated as $\mathbf{s}_i = \dfrac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij}$, where

$$[\mathbf{u}_{ij}]_k = \frac{K(\mathbf{x}_{ij}, \mathbf{v}_k)}{\sum_{l=1}^{N_K} K(\mathbf{x}_{ij}^{(t)}, \mathbf{v}_l)} \in [0, 1], \tag{6}$$

giving rise to the regular BoF with soft-assignments, as described in the previous subsection. This formulation allows for replacing the Gaussian kernel used in (1), which requires tuning the width σ of the kernel, with an easier to use hyperbolic kernel that involves no hyper-parameters. The main reason for this choice is that the Gaussian kernel proved to be quite unstable, when employed in a recurrent quantizer. On the other hand, the proposed hyperbolic-based formulation was significantly stabler and easier to use. Therefore, a hyperbolic (sigmoid) kernel is used to compute the similarity between each feature vector and the codewords:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{x}_{ij}^T \mathbf{v}_k) \in \mathbb{R}, \tag{7}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid function. Note that this formulation still ignores the temporal information, since it provides no way to encode the current *state* of the quantizer. Therefore, in order to take into account the temporal information the current state, as expressed by the histogram compiled using the feature vectors fed to the network until the current time-step, (7) is extended to:

$$\mathbf{d}_{ij} = \sigma\big(\mathbf{V}\mathbf{x}_{ij} + \mathbf{V}_h(\mathbf{r}_{ij} \odot \mathbf{s}_{i,j-1})\big) \in \mathbb{R}^{N_K}, \tag{8}$$

where $\mathbf{V}_h \in \mathbb{R}^{N_K \times N_K}$ is weight matrix that is used to transfer the gated histogram vector into the quantization space and is learned during the training process, $\mathbf{s}_{i,j-1}$ is the histogram extracted from previous quantizations (states) and $\mathbf{r}_{ij} \in \mathbb{R}^{N_K}$ is the output of a reset gate, introduced to ensure the long-term stability of the model. The reset gate is inspired by the GRU model [33] and is defined as:

$$\mathbf{r}_{ij} = \sigma(\mathbf{W}_r \mathbf{x}_{ij} + \mathbf{U}_r \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \tag{9}$$

where $\mathbf{W}_r \in \mathbb{R}^{N_K \times D}$ and $\mathbf{U}_r \in \mathbb{R}^{N_K \times N_K}$ are the weight matrices used to implement the reset gate.

Then, similarly to standard BoF approach, the l_1 normalized membership vector is computed as in (2):

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1}. \tag{10}$$

Note that in order to ensure that quantizer's output is always a properly normalized membership vector, the initial state $\mathbf{s}_{i,0}$ is set equal to $\mathbf{s}_{i,0} = \frac{1}{N_K}\mathbf{1}$, where N_K is the number of recurrent codewords and $\mathbf{1} \in R^{N_K}$ is a vector of all ones. Hence, the fixed-length histogram is recurrently updated as follows:

$$\mathbf{s}_{i,j} = (\mathbf{1} - \mathbf{z}_{ij}) \odot \mathbf{s}_{i,j-1} + \mathbf{z}_{ij} \odot \mathbf{u}_{ij} \in \mathbb{R}^{N_K}, \tag{11}$$

where the update gate $\mathbf{z}_{ij}$ controls how much the current histogram, which also encodes the state of the quantizer, will be updated and it is defined as:

$$\mathbf{z}_{ij} = \sigma(\mathbf{W}_z\mathbf{x}_{ij} + \mathbf{U}_z\mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \tag{12}$$

where $\mathbf{W}_z \in \mathbb{R}^{N_K \times D}$ and $\mathbf{U}_z \in \mathbb{R}^{N_K \times N_K}$ are parameters of the update gate. Finally, the total histogram is compiled by averaging the intermediate state histograms as:

$$\mathbf{s}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} \mathbf{s}_{i,j} \in \mathbb{R}^{N_K}. \tag{13}$$

All the parameters of the ReBoF model can be learned using regular back-propagation. Note that ReBoF is capable of recursively processing the input feature vectors, while capturing their temporal information. First, the feature vectors are quantized using the proposed recurrent stateful quantizer, as described by (8) and (10). Then, the quantized vectors are used to appropriately update the state of the quantizer, as given by (11), and finally compile the resulting histogram. It is worth noting that ReBoF, similarly to all BoF-based models, is capable of processing varying-length input sequences.

The pipeline of the proposed ReBoF pooling method is summarized in Fig. 1. The input feature vectors $\mathbf{x}_{ij}$ are first fed to the reset gate which controls how much the previous histogram $\mathbf{s}_{i,j-1}$ will contribute to the quantization process. Note that recurrent connections are plotted in red in Fig. 1. Then, the proposed recurrent quantizer is employed to extract the membership vector $\mathbf{u}_{ij}$, which is then fed to the update gate. The update gate controls how much the previous histogram will be updated based on the output of the recurrent quantizer $\mathbf{u}_{ij}$ and the previous histogram $\mathbf{s}_{i,j-1}$, leading to the updated histogram $\mathbf{s}_{ij}$. The final histogram representation of an

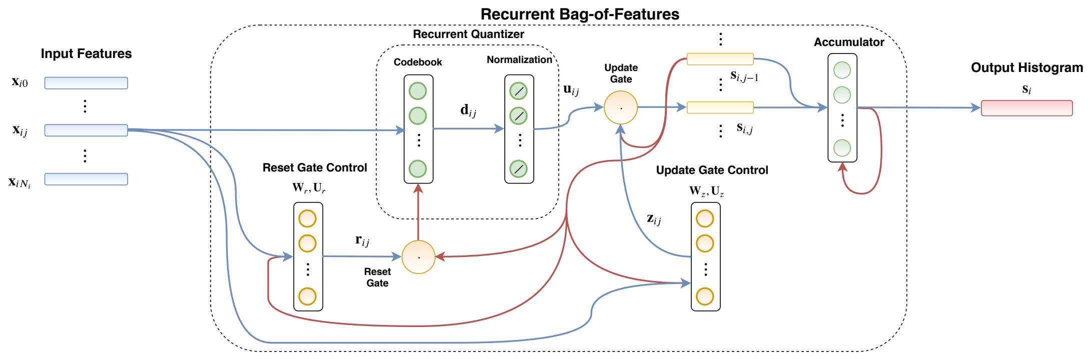


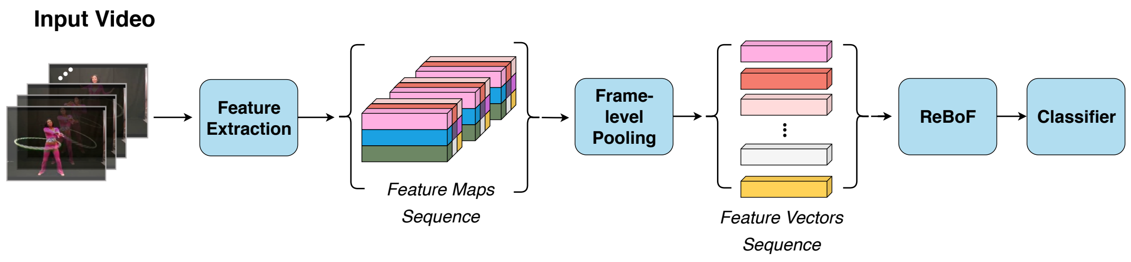
Figure 1: Pipeline of the proposed ReBoF pooling approach. Figure depicts the main information processing steps involved in ReBoF. Red lines denote recurrent connections/information flow, while blue lines denote feed-forward connections/information flow. Figure best viewed in color.

input object is obtained by incrementally accumulating the extracted histogram vectors into the final one $\mathbf{s}_i \in \mathbb{R}^{N_K}$. It is worth noting that compared to other Neural BoF approaches [12], the proposed one is the first one that employs a recurrent quantizer by incorporating an additional reset gate and update gate. The reset gate is used to equip the quantizer with (resetable) memory, allowing for performing context-aware quantization, while the update gate allows for dynamically updating the histogram only when appropriate (e.g., ignoring features that are not relevant to the current context).

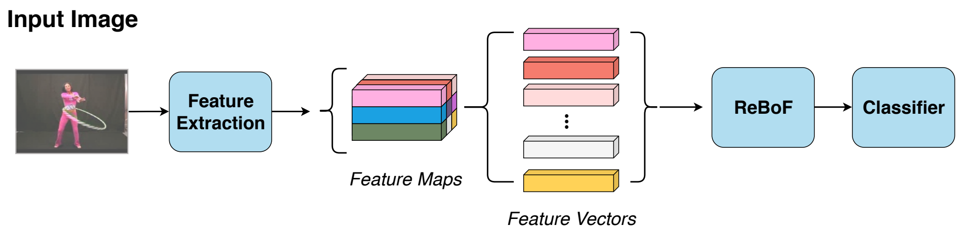
3.3. Using Recurrent BoF for Visual Information Analysis Tasks

ReBoF can be directly used in any DL architecture without any modification between the last feature extraction layer and the first fully connected one. The extracted representation $\mathbf{s}_i$ depends only on the number of used codewords N_K and, as a result, ReBoF is capable of processing inputs of variable size without any modification. Three different ways to employ ReBoF in DL models are presented: a) video analysis using CNN frame-based feature extractors, b) image analysis using CNNs and c) spatio-temporal video analysis.

ReBoF for video analysis: As presented in 3.2, the advantage of ReBoF is its ability to capture the temporal information of sequential data. This can be readily exploited for tackling challenging video analysis problems, e.g., activity recognition, video retrieval, etc. Assuming that every frame of the video is described by an extracted feature vector $\mathbf{x}_{ij}$ and that the i-th video is composed of N_i frames, then ReBoF can be directly applied by feeding to it the sequence of



(a)



(b)

Figure 2: Using Recurrent BoF for two different tasks: (a) video analysis and (b) image analysis

the frame-based feature vectors. This allows ReBoF to process videos of arbitrary duration, while creating fixed-length compact representations of them. This process is illustrated in Fig. 2a, where a feature vector is extracted from every frame of a video. Note that usually a high-dimensional tensor is first extracted from every video frame when convolutional feature extractors are used. This tensor is first aggregated into a feature vector, using a frame-level pooling approach (ReBoF can be also used for this task, as discussed below), and then the feature vectors sequence is fed into the ReBoF model to extract a video-level compact representation. This representation can be then used for the task at hand, e.g., classification.

ReBoF for image analysis: Moreover, ReBoF can be utilized also for processing non-sequential input data, while preserving crucial spatial information. For example, in case of image classification, let $\mathbf{X} \in R^{W \times H \times C}$ be the $3D$ feature map extracted from a CNN layer, with width, height and number of channels equal to W, H and C respectively. Several ways to "scan" the image can be employed to flatten the feature map into a set of vectors. In the simplest case considered in this paper, the image is scanned from up to down and left to right. Therefore, a total of $N_i = W \cdot H$ feature vectors are extracted from each image. Then, this flattened sequence of feature vectors can be fed into the ReBoF model. This process is illustrated in Fig. 2b. Note that since the feature vectors are always produced following the same scanning pattern, the model can learn the spatial interactions between the extracted feature vectors, effectively capturing the spatial information.

ReBoF for spatio-temporal analysis: Finally, it is possible to combine two ReBoFs in order to capture the spatio-temporal content of the input video. Similarly to the image analysis approach, the first ReBoF model can be used to create a histogram for every video frame, i.e., perform the frame-level pooling, as shown in Fig. 2b. Then, a time series of histograms is compiled and fed to the second ReBoF that captures the temporal information of the video (as shown in Fig. 2a) and thus leads to a compact spatio-temporal representation that is subsequently fed to the classifier.

4. Experimental Evaluation

In this section the proposed ReBoF model is evaluated on six different datasets. The experimental procedure along with the results are presented and discussed in the following two main subsections. In the first one, the proposed method is validated over three video datasets for activity recognition, while in the second one, evaluation is conducted over three datasets for image classification.

4.1. Video Activity Recognition

Datasets: The proposed ReBoF model is initially validated in 3 different video datasets for activity recognition. The specific task was selected to demonstrate the ability of ReBoF to effectively capture the temporal dimension of video sequences, which is expected to contain essential information for classification tasks.

The first dataset is the UTKinect-Action3D [34] that consists of 10 types of human activities in indoor settings. Samples for each action are collected from 10 different people that perform every activity two times. The demonstrated actions are: *walk*, *sit down*, *stand up*, *pick up*, *carry*, *throw*, *push*, *pull*, *wave hands*, and *clap hands*. For each video instance, three streams were recorded: RGB, depth and skeleton joint locations. The RGB images were used for all of the conducted experiments. Since there is no official training/testing split provided, a 50%-50% subject-based split strategy was employed, i.e., the videos of the first five subjects were included in the training set and the rest of them were used to form the testing set. Hence, a quite challenging setup was created, as the activities belonging in the testing set were performed from unseen subjects.

The second database is the well-known UCF101 dataset[35], that is widely used for benchmarking action recognition models. The dataset contains $13,320$ action instances belonging in 101 classes, that can be grouped in five generic categories, namely: 1. Human-Object Interaction, 2. Body-Motion Only, 3. Human-Human Interaction, 4. Playing Musical Instruments, and 5. Sports. UCF101 also contains three different evaluation splits. For all the experiments conducted in this paper, the first split was used.

We also created a challenging and more complex dataset based on the UCF101 to better demonstrate the ability of the proposed method to capture the temporal dynamics of video data. To this end, we compiled a dataset by mixing instances from different activities of the UCF101 dataset together. More specifically, 10 activities of UCF101 (split 1) were selected to be joined together. Every action of this subset was joined with each one of the remaining, leading to 90 complex actions. One can further understand the significance of encoding the temporal information of these instances by considering that a sample of action "A" combined with one of action "B" (let name this complex activity class "AB") must be separated from samples of complex activities from class "BA". Note that "AB" and "BA" videos contain the same set of frames (for a specific instance of "A" and "B"), but in a different order. A typical example of such two *complex* actions would be the action sequences of "sitting to a chair" (AB) and "rising from a chair" (BA), where A is the action of getting closer to the chair and B the action of actually being on the chair. A method that does not capture the temporal information cannot discriminate between these two actions, since the videos would contain the same frames, but in the reverse order. Hence, 114 samples were selected for each class, as this was the minimum number of instances contained in the selected initial classes. The selected 10 action classes were the following: *ApplyEyeMakeup*, *ApplyLipstick*, *Billiards*, *BoxingPunchingBag*, *BoxingSpeedBag*, *Haircut*, *Hammering*, *TableTennisShot*, *TennisSwing*, and *Typing*. These classes were selected to offer a challenging dataset of complex activities, where every primary action has familiar content with at least one of the rest. The i-th sample of initial class "A" is combined with the i-th sample of initial class "B" and so on, leading to $7,380$ training and $2,880$ testing data equally balanced among the 90 classes. The compiled dataset is called "Complex UCF" in the rest of this paper.

Evaluation Setup: The video analysis approach described in Subsection 3.3 was employed in order to evaluate the proposed method over the first two datasets. Every video instance was uniformly sampled in time in order to extract a predefined number of frames, denoted by N_f. The number of extracted frames was set to $N_f = 30$ for the UTKinect-Action3D dataset and to $N_f = 40$ for the UCF101 dataset. Shorter videos were looped over as many times as necessary to ensure that each video contains at least N_f frames, following the methodology described in the relevant literature [24].

Every frame was fed to an Inception V3 model [36], pretrained in ImageNet, and a feature representation was extracted from each frame from the last average pooling layer of the network. Therefore, every frame is represented by a 2048-dimensional feature vector, building a sequence of N_f features for every video sample. This sequence is then fed to the ReBoF layer, which is followed by a fully connected layer block composed of two fully connected layers. The first one is composed of 512 neurons using the ReLU activation function and dropout with rate of 0.5, while the output (classification) layer has as many neurons as the classes of each dataset and employs the softmax activation function. The cross-entropy loss is used during the training. The resulting network was trained from scratch using the Adam optimizer and a learning rate of 10^{-5}, apart from the pretrained feature extractors which were kept frozen. We also experimentally found out that scaling the extracted histogram by N_K improved the convergence of the proposed method, especially when training the whole architecture from scratch. This scaling ensures that the gradients from the fully connected layers will not diminish as they are back-propagated to the previous layers. For the UTKinect-Action3D dataset the models were trained for 800 epochs, while for the UCF101, all models were trained for 50 epochs and the training/evaluation procedure was repeated 3 times.

The performance of the proposed method is compared with three other pooling methods, that also use the exact same feature extraction and classification layer blocks (in order to ensure a fair comparison). First, global average pooling over the temporal dimensions of the input sequence is used in place of the proposed ReBoF layer. This method is called "Average Pooling" in the rest of this paper. For the second method, a recently proposed state-of-the-art variant of the BoF method that is adapted for use with DL architectures, the Linear BoF [37], is employed. Furthermore, a powerful recurrent aggregation model, the GRU [33], which is used to aggregate the features, while also capturing the temporal dimension of the data, is also compared to the proposed method. To ensure a fair comparison, we use the same number of codewords or hidden units for the Linear BoF or GRU model, respectively, as the number of codewords used in our method, except otherwise stated.

A slightly different setup was used for the third dataset (Complex UCF101). A 16-frame sliding window, with overlap of 4 frames was applied on every activity instance of UCF101 dataset, creating 16-frame clips. A 3D ResneXt-101, pretrained on UCF101, as deployed in [24], was used for feature extraction. Every clip was then fed to the network and features were extracted from the last average pooling layer of the network. After this step, every video clip is represented by a 2048-dimensional vector. Note that contrary to the previous setup, feature vectors already enclose some spatio-temporal information, since 3D convolutions were used during the feature extraction. Then, for every activity instance of class “AB”, 16 sequential feature vectors were selected from each one of the classes “A” and “B”, starting from the beginning of the video. If a subsequence of features is shorter, then it was looped over. The two subsequences were then stacked, forming a 32-length sequence ($N_f = 32$) of feature vectors for the action “AB” of the ComplexUCF dataset.

Similar to the previous setup, the performance of ReBoF is compared to the “Average Pooling”, ”Linear BoF” and “GRU” methods. All methods share the same classification block as before and the networks were trained from scratch, excluding the weights of feature extractors which were pretrained. In case of Average Pooling, the network was trained for 50 epochs, while in the rest, the training process stopped when 99.9% accuracy was achieved in training set. The evaluation of ReBoF and GRU methods is repeated 3 times and the mean accuracy and standard deviation on the test set are reported.

Experimental Results: The experimental evaluation for the UTKinect-Action3D dataset is provided in Table 1. The proposed method outperforms the other three evaluated methods, providing the highest accuracy (54.64%) using 512 codewords. GRU also achieves its best accuracy (47.71%) for the same number of codewords, while for the Linear BoF method the highest performance (44.47%) is achieved for 256 codewords. Both ReBoF and GRU significantly outperform the Average Pooling and Linear BoF methods, since they are capable of effectively modeling the temporal dynamics of the input video sequences and distinguishing between similar activities, such as *stand up* and *sit down*.

Moreover, in Fig. 3 the effect of using a wider range of codewords and number of GRU units in the classification accuracy on the UTKinect-Action3D dataset is evaluated using the two methods that achieve the highest performance (GRU and ReBoF). In all cases, the proposed method leads to higher accuracy compared to the GRU method. Furthermore, the proposed method allows for reducing the size of the extracted representation, since it outperforms the best GRU model (512 units) using just 128 codewords. This allows for reducing the size of the extracted representation

Table 1: UTKinect-Action3D Evaluation

Method	**# Codewords - GRU Units**	**Test Accuracy** (%)
Average Pooling	-	40.83
Linear BoF	256	44.47
GRU	512	47.71
ReBoF	512	**54.64**

and, as a result, the number of parameters in the subsequent fully connected layer. Both methods achieve their best performance for 512-dimensional representations. After this point, the accuracy for both models drops, mainly due to overfitting phenomena.

Similar conclusions can be drawn from the evaluation results on the UCF101 dataset, which are reported in Table 2. The experiments were repeated three times and the mean accuracy and standard deviation on the test set are reported. Even though UCF101 is a less challenging dataset, in terms of temporal dependence, the proposed method still outperforms the rest of the evaluated methods, achieving the highest accuracy of 72.02% for 1024 codewords. The effect of using different number of codewords and recurrent units is also evaluated in Fig. 4. Again, the proposed method outperforms the GRU method regardless the number of used codewords, while it achieves comparable accuracy using 2 to 4 times smaller representations.

Table 2: UCF101 Evaluation

Method	**# Codewords - GRU Units**	**Test Accuracy** (%)
Average Pooling	-	70.32 ± 0.43
Linear BoF	1024	71.11 ± 0.35
GRU	2048	71.04 ± 0.20
ReBoF	1024	$\mathbf{72.02} \pm 0.68$

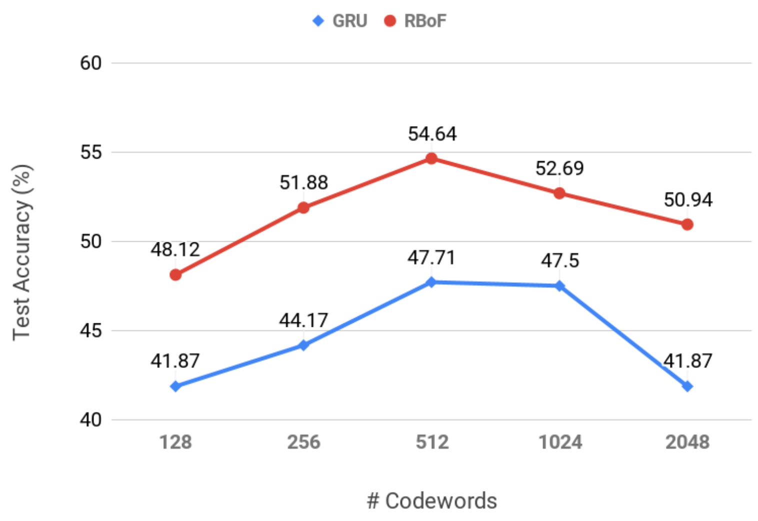


Figure 3: UTKinect-Action3D Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

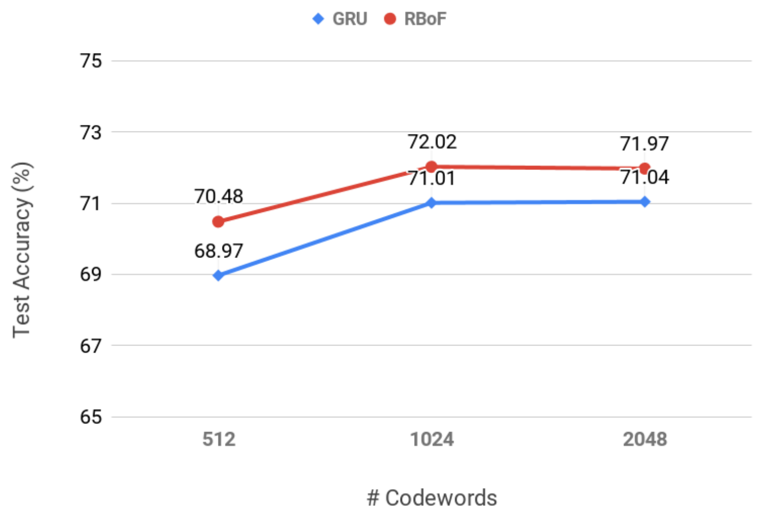


Figure 4: UCF101 Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

Next, the evaluation on the Complex UCF dataset is provided in Table 3. Note that the Average Pooling and Linear BoF methods fail to overpass the 50% test accuracy barrier, since "AB" activities cannot be distinguished from those of "BA", due to discarding crucial temporal information. Note that the Average Pooling and Linear BoF methods obtain significantly lower accuracy on this dataset, since capturing the temporal information plays a less crucial role on the other two datasets, in which many videos that belong to certain actions can be categorized just by one frame, leading to a smaller gap between the Average Pooling method and the ones that capture the temporal information, i.e., GRU and ReBoF. Again, ReBoF achieves the highest accuracy equal to 89.29% for 1024 codewords, outperforming the GRU method.

Table 3: Complex UCF101 Evaluation

Method	**# Codewords - GRU Units**	**Test Accuracy** (%)
Average Pooling	-	48.95
Linear BoF	1024	43.88
GRU	512	88.86 ± 2.04
ReBoF	512	89.25 ± 1.08
GRU	1024	88.62 ± 1.02
ReBoF	1024	$\mathbf{89.29 \pm 0.89}$

Moreover, qualitative results on Complex UCF101 dataset are presented in Fig. 5 in order to further illustrate the ability of ReBoF to encode efficiently the temporal information of video data. ReBoF is compared with the rest of the methods using the confidence scores for the top-k ($k = 1, 2, 3$) predictions. The corresponding labels are reported for each prediction. Fig. 5 clearly demonstrates the inefficiency of Average Pooling and Linear BoF methods to distinguish the combination of activities "TennisSwing + ApplyLipstick" ("AB") from the inverted combination of activities "ApplyLipstick + TennisSwing" ("BA"), due to the complete loss of the temporal information. GRU method presents to some extent better performance since the correct predicted label is found on top-2 ranking. Yet, the model is overconfident over the wrong class, probably due to the difficulty of distinguishing the second part ("ApplyLipstick") of the complex action from a similar one ("ApplyEyeMakeup"). On the other hand, the proposed ReBoF method is more robust

Target: TennisSwing+ApplyLipstick

Method	top-1	top-2	top-3
Average Pooling	ApplyLipstick+TennisSwing 0.3215	ApplyEyeMakeup+TennisSwing 0.2699	TennisSwing+ApplyEyeMakeup 0.2521
Linear BoF	ApplyLipstick+TennisSwing' 0.41984	ApplyEyeMakeup+TennisSwing 0.2382	TennisSwing+ApplyEyeMakeup 0.1537
GRU	TennisSwing+ApplyEyeMakeup 0.5018	**TennisSwing+ApplyLipstick** 0.4034	TableTennisShot+ApplyLipstick 0.0344
ReBoF	**TennisSwing+ApplyLipstick** **0.4776**	TennisSwing+ApplyEyeMakeup 0.4582	ApplyEyeMakeup+TennisSwing 0.0103

Figure 5: Qualitative results on the Complex UCF101 dataset: For a given video sequence the first three top-k ($k = 1, 2, 3$) prediction scores among with the corresponding predicted label are reported for each of the evaluated methods. In the first column (top-1 results) red and green color is used to denote false and correct predictions, respectively.

to such phenomena, more efficiently capturing the temporal details and managing to accurately classify the input sample, contrary to rest of the methods. It should be also noted that the inverted complex action "ApplyLipstick+TennisSwing" is not present in the first top-3 ranking and thus, the confidence score for the specific class is at least lower than 0.01, signifying the effectiveness of ReBoF's recurrent quantizer to enclose the temporal characteristics of the video sequence.

Finally, in Table 4 ReBoF is compared to other similar approaches reported in the literature on UCF101 dataset. Activity recognition based on a single RGB frame of the video achieved 67.4% accuracy across all three splits of UCF101. LRCN model presented in [22] improved the results by employing an LSTM to capture the temporal information of the data. The "slow fusion" scheme presented in [20] aimed to conceive the temporal information by applying a CNN network over a time window of the RGB stream, yet the reported results are lower compared to the case of single frame network, failing to exploit the temporal information contained in the video input. On the other hand, the proposed ReBoF formulation manages to effectively capture the spatial information of video streams, outperforming the rest of the evaluated methods. It is worth noting that other approaches that use C3D networks, such as the I3D network [21], can further boost the action recognition performance. However, these more powerful architectures usually also require using significantly larger datasets in order to be trained effectively and avoid over-fitting, as also demonstrated in Table 4 (e.g., training 3D-ConvNets from scratch leads to lower recognition accuracy), while at the same time these architectures also significantly increase the computational complexity of the model compared to the proposed method.

Table 4: Comparison with other methods on UCF101 (split 1)

Method	**Test Accuracy** (%)
Single RGB frame (all splits) [22]	67.4
"Slow fusion" spatio-temporal ConvNet (all splits) [20]	65.4
Spatial LRCN (all splits) [22]	68.2
3D-ConvNet (trained from scratch) [33]	51.6
ReBoF	72.02

4.2. Image Classification

Datasets: ReBoF is also able to capture the spatial information of the extracted feature vectors, as discussed in Section 3.3. To this end, ReBoF is also evaluated on three image datasets: the MNIST database of handwritten digits (MNIST) [38], the Fashion MNIST database of fashion images [39], and the CIFAR-10 database of color images with varying content [40].

The MNIST dataset is widely used in the computer vision field and contains $60,000$ training and $10,000$ testing grayscale images of handwritten digits. The size of each image is 28×28 pixels and the data are distributed over 10 different classes, one for each digit (0 to 9). The Fashion MNIST is an MNIST-like dataset that contains labeled fashion grayscale images. Similarly to MNIST, it consists of $60,000$ training and $10,000$ testing images of size 28×28 pixels. The CIFAR-10 dataset contains totally $60,000$ colour images of size 32×32 pixels. The dataset is separated in $50,000$ images for training and $10,000$ images for testing. Data are equally divided in 10 classes of various generic categories, e.g., *airplane*, *cat*, *ship*, etc.

Evaluation Setup: The proposed method is evaluated in the aforementioned image datasets, following the approach presented in Section 3.3. The ReBoF model is combined with a convolutional neural network, creating a model that consists of three main layer blocks: 1. a feature extraction block, 2. a ReBoF layer, and 3. a classification layer block. The feature extraction block is composed of two convolutional sub-blocks. The first sub-block is composed of two convolutional layers that use 32 3×3 filters, followed by a 2×2 max pooling layer. The second sub-block consists of two additional convolutional layers that use 64 3×3 filters, followed by a 2×2 max pooling layer. In addition, dropout with rate 0.25 is applied to the input of second block. The output of the last convolutional block is used to extract feature vectors for each image. The final feature map is scanned from left to right and from top to bottom to extract feature vectors that are sequentially fed to the ReBoF block, after applying dropout with rate of 0.25. The extracted features are recurrently quantized in N_K-dimensional vectors and accumulated in a N_K-length histogram. The output of the ReBoF layer is then fed to the classification layer block that is composed of a hidden fully connected layer with 512 neurons and an output layer with 10 neurons, while dropout with rate of 0.2 and 0.5 is applied to the input of the first and second fully connected layers, respectively. The ReLU activation function is used for all the hidden layers, while the network is trained to minimize the cross-entropy loss. The network is trained from scratch in an end-to-end fashion for 100 epochs using the Adam optimizer and learning rate equal to 0.001.

The proposed method is also compared to using an average pooling baseline, as well as to the Linear BoF method, similarly to the activity recognition task. In the average pooling method, the output of the employed feature extraction block is fed to a global average pooling layer and then forwarded to the employed classification layer block. In this case, no dropout is applied to the feature extractor's output. In the Linear BoF method, the ReBoF block is substituted by a Linear BoF, while the rest of the network is kept the same to ensure a fair comparison between the methods. Both networks were trained from scratch, using the same training parameters as in the case of ReBoF.

Evaluation Results: The proposed method is compared with two other evaluated methods in Table 5. ReBoF outperforms both the Average Pooling and Linear BoF methods in all the evaluated cases, regardless the used dataset. For the MNIST and Fashion MNIST datasets the highest performance is achieved when 512 codewords are used, while for the CIFAR-10 dataset for 128 codewords. Similarly, for the Linear BoF method the best results for the MNIST dataset are achieved when 64 codewords are employed, whereas in the case of Fashion MNIST and CIFAR-10 when 128 codewords are used. The performance improvements are marginal for the simpler MNIST and Fashion MNIST datasets. However, significant accuracy improvements are observed for the more complex CIFAR-10 dataset, confirming the ability of the proposed method to capture

Table 5: Image classification evaluation

Method	MNIST	Fashion MNIST	CIFAR-10
Average Pooling	99.44	92.80	79.71
Linear BoF	99.46	92.97	80.25
ReBoF	**99.50**	**93.28**	**82.25**

(Classification accuracy is reported (%))

the spatial information encoded in the extracted feature vectors. This is also confirmed in the results reported in Fig. 6, where the effect of varying the number of codewords for the ReBoF method is examined. ReBoF achieves remarkable results in the MNIST and Fashion MNIST datasets, even for a small number of codewords. It is worth noting that when a very large number of codewords is used, i.e., 1024 or 2048, the accuracy begins to decrease due to overfitting. The effect of the number of codewords on the model's accuracy can be also clearly demonstrated for CIFAR-10, which contains larger and more complex images. In this case, the best accuracy is achieved when 128 codewords are used.

Also, the ability of ReBoF method to capture spatial patterns from images is further demonstrated in Fig 7, where qualitative results are presented for the MNIST dataset. In case of Average Pooling method, the prediction model is overconfident over the wrong digit in all of the three evaluated cases. Linear BoF presents a more balanced distribution over the prediction scores, yet it also fails in some cases to classify the input image correctly. On the contrary, ReBoF, which is capable of capturing the spatial information contained in the feature vector sequence extracted from each MNIST digits, leads to the best results, accurately identifying each digit, while also providing the highest confidence on the correct label.

5. Conclusions

In this paper a novel Recurrent Bag-of-Features formulation has been proposed, designed to effectively process sequential input data. ReBoF was inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach. This enables ReBoF to harness the power of a powerful recurrent quantizer that is able to capture the temporal

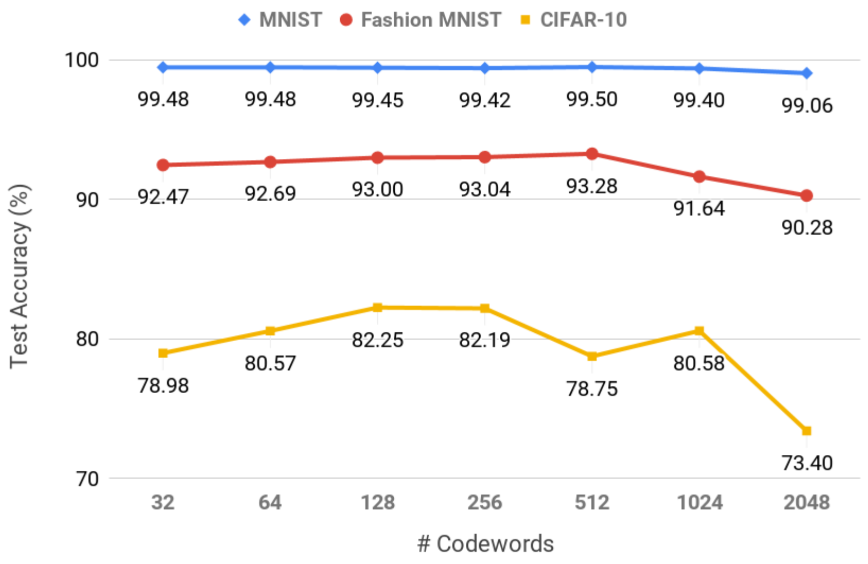


Figure 6: Test accuracy (%) of ReBoF in three datasets for different number of codewords

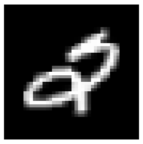
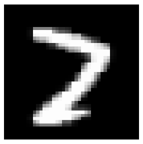
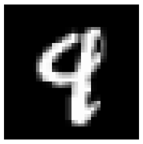
Input	Average Pooling	Linear BoF	ReBoF	Ranking
	8: 0.8842	8: 0.5824	**2**: **0.9848**	top-1
	2: 0.0654	**2**: 0.4172	8: 0.0134	top-2
	9: 0.0483	9: 9.4e-05	0: 0.0008	top-3
	2: 0.7978	2: 0.5845	**7**: **0.5897**	top-1
	7: 0.2014	**7**: 0.4151	2: 0.4097	top-2
	3: 0.0003	0: 0.0001	3: 0.0003	top-3
	4: 0.7532	**9**: **0.8120**	**9**: **0.9273**	top-1
	9: 0.2467	4: 0.1879	4: 0.07226	top-2
	8: 5.4e-05	8: 1.8e-05	7: 0.0002	top-3

Figure 7: Qualitative results on the MNIST dataset: Top-k predictions for $k = 1, 2, 3$ among with the corresponding confidence scores are reported for the three evaluated methods. Correct and false predictions are denoted with green and red color, respectively, in the first row (top-1 predictions) for each sample.

information of input data, which is crucial in classification tasks, such as activity recognition, while still maintaining all the advantages of the BoF model. ReBoF can be directly used between the last feature extraction layer and the fully connected layer of a network, while the resulting architecture can be trained in an end-to-end fashion using back-propagation, allowing for building powerful deep learning models for various visual information analysis tasks. The performance of ReBoF model was extensively evaluated in various activity recognition tasks using three video datasets. In all cases, the proposed method outperformed the rest of the evaluated methods, demonstrating the ability of ReBoF to capture the temporal information and increase the classification accuracy. Furthermore, ReBoF is also capable of encoding the spatial information, as it was also demonstrated in image classification tasks using three more datasets.

ReBoF opens several interesting future research directions. First, the proposed approach for activity recognition can be further enhanced by combining a ReBoF layer over the spatial dimension, followed by a ReBoF layer over the temporal dimension. In this way, the spatio-temporal information can be more properly encoded by creating a space-time histogram, further improving the performance over the applications presented in this paper. Second, ReBoF can be employed to replace the intermediate weak pooling layers of CNNs, aiming to increase the accuracy of the models. Finally, employing the proposed method for other tasks, such as video retrieval and hashing, is expected to boost the performance of existing methods by extracting compact, yet more discriminative representations.

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.

References

[1] L. Nanni, S. Ghidoni, S. Brahnam, Handcrafted vs. non-handcrafted features for computer vision classification, Pattern Recognition 71 (2017) 158–172.

[2] J. Sivic, A. Zisserman, Video google: A text retrieval approach to object matching in videos, in: Proc. Int. Conf. on Computer Vision, 2003, p. 1470.

[3] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, C. Schmid, Aggregating local image descriptors into compact codes, IEEE Trans. on Pattern Analysis and Machine Intelligence 34 (9) (2012) 1704–1716.

[4] N. Passalis, A. Tefas, Learning bag-of-features pooling for deep convolutional neural networks, in: Proc. IEEE Int. Conf. on Computer Vision, 2017, pp. 5755–5763.

[5] C.-Y. Ma, M.-H. Chen, Z. Kira, G. AlRegib, Ts-lstm and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition, Signal Processing: Image Communication 71 (2019) 76–87.

[6] F. B. Silva, R. d. O. Werneck, S. Goldenstein, S. Tabbone, R. d. S. Torres, Graph-based bag-of-words for classification, Pattern Recognition 74 (2018) 266–285.

[7] N. Passalis, A. Tefas, Learning bag-of-embedded-words representations for textual information retrieval, Pattern Recognition 81 (2018) 254–267.

[8] I. F. J. Ghalyan, Estimation of ergodicity limits of bag-of-words modeling for guaranteed stochastic convergence, Pattern Recognition 99 (2020) 107094.

[9] S. Bhattacharya, R. Sukthankar, R. Jin, M. Shah, A probabilistic representation for efficient large scale visual recognition tasks, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2011, pp. 2593–2600.

[10] A. Iosifidis, A. Tefas, I. Pitas, Discriminant bag of words based representation for human action recognition, Pattern Recognition Letters 49 (2014) 185–192.

[11] A. Iosifidis, A. Tefas, I. Pitas, Multidimensional sequence classification based on fuzzy distances and discriminant analysis, IEEE Trans. on Knowledge and Data Engineering 25 (11) (2012) 2564–2575.

[12] N. Passalis, A. Tefas, Neural bag-of-features learning, Pattern Recognition 64 (2017) 277–294.

[13] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Vol. 2, 2006, pp. 2169–2178.

[14] Y.-G. Jiang, C.-W. Ngo, J. Yang, Towards optimal bag-of-features for object categorization and semantic video retrieval, in: Proc. ACM Int. Conf. on Image and Video Retrieval, 2007, pp. 494–501.

[15] A. Iosifidis, A. Tefas, I. Pitas, View-invariant action recognition based on artificial neural networks, IEEE Trans. on Neural Networks and Learning Systems 23 (3) (2012) 412–424.

[16] Y. Yacoob, M. J. Black, Parameterized modeling and recognition of activities, Computer Vision and Image Understanding 73 (2) (1999) 232–247.

[17] I. Laptev, M. Marszałek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, in: Proc. IEEE Conf. on Computer Vision & Pattern Recognition, 2008, pp. 1–8.

[18] P. Dollár, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, 2005.

[19] A. Jalal, Y.-H. Kim, Y.-J. Kim, S. Kamal, D. Kim, Robust human activity recognition from depth video using spatiotemporal multi-fused features, Pattern Recognition 61 (2017) 295–308.

[20] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

[21] J. Carreira, A. Zisserman, Quo vadis, action recognition? a new model and the kinetics dataset, arXiv:1705.07750 [cs]ArXiv: 1705.07750.
URL `http://arxiv.org/abs/1705.07750`

[22] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, K. Saenko, Long-term recurrent convolutional networks for visual recognition and description, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015, p. 2625–2634. `doi:10.1109/CVPR.2015.7298878`.
URL `http://ieeexplore.ieee.org/document/7298878/`

[23] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE Trans. on Pattern Analysis and Machine Intelligence 35 (1) (2013) 221–231. `doi:10.1109/TPAMI.2012.59`.

[24] K. Hara, H. Kataoka, Y. Satoh, Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2018, pp. 6546–6555.

[25] C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, arXiv:1604.06573 [cs]ArXiv: 1604.06573.
URL `http://arxiv.org/abs/1604.06573`

[26] J. Li, X. Liu, M. Zhang, D. Wang, Spatio-temporal deformable 3d convnets with attention for action recognition, Pattern Recognition 98 (2020) 107037.

[27] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, Netvlad: Cnn architecture for weakly supervised place recognition, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2016, pp. 5297–5307.

[28] R. Girdhar, D. Ramanan, Attentional pooling for action recognition, in: Proc. Advances in Neural Information Processing Systems, 2017, pp. 34–45.

[29] A. Miech, I. Laptev, J. Sivic, Learnable pooling with context gating for video classification, arXiv preprint arXiv:1706.06905.

[30] P. Wang, Y. Cao, C. Shen, L. Liu, H. T. Shen, Temporal pyramid pooling-based convolutional neural network for action recognition, IEEE Trans. on Circuits and Systems for Video Technology 27 (12) (2016) 2613–2622.

[31] Y. Xu, Y. Han, R. Hong, Q. Tian, Sequential video vlad: training the aggregation locally and temporally, IEEE Trans. on Image Processing 27 (10) (2018) 4933–4944.

[32] A. Richard, J. Gall, A bag-of-words equivalent recurrent neural network for action recognition, Computer Vision and Image Understanding 156 (2017) 79–91. `doi:10.1016/j.cviu.2016.10.014`.

[33] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078.

[34] L. Xia, C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3d joints, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops, 2012, pp. 20–27.

[35] K. Soomro, A. R. Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, arXiv preprint arXiv:1212.0402.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

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

[38] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[39] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017). `arXiv:cs.LG/1708.07747`.

[40] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. rep., Citeseer (2009).