Human Action Recognition using Recurrent Bag-of-Features Pooling

Marios Krestenitis[1], Nikolaos Passalis[1] Alexandros Iosifidis[2], Moncef Gabbouj[3], and Anastasios Tefas[1]

[1] Dept. of Informatics, Aristotle University of Thessaloniki, Greece
{mikreste, passalis, tefas}@csd.auth.gr
[2] Dept. of Engineering, Electrical and Computer Engineering, Aarhus University, Denmark, ai@eng.au.dk
[3] Faculty of Information Technology and Communication, Tampere University, Finland, moncef.gabbouj@tuni.fi

Abstract. Bag-of-Features (BoF)-based models have been traditionally used for various computer vision tasks, due to their ability to provide compact semantic representations of complex objects, e.g., images, videos, etc. Indeed, BoF has been successfully combined with various feature extractions methods, ranging from handcrafted feature extractors to powerful deep learning models. However, BoF, along with most of the pooling approaches employed in deep learning, fails to capture the temporal dynamics of the input sequences. This leads to significant information loss, especially when the informative content of the data is sequentially distributed over the temporal dimension, e.g., videos. In this paper we propose a novel stateful recurrent quantization and aggregation approach in order to overcome the aforementioned limitation. 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 effectively encoding the temporal dimension of the data. The effectiveness of the proposed approach is demonstrated using three video action recognition datasets.

1 Introduction

Computer vision is one of the most active and continuously expanding research fields, while with the advent of deep learning (DL) many powerful visual information analysis methods for high dimensional data have been recently proposed [8]. The typical pipeline of a visual information analysis approach involves at least the following two steps: a) feature extraction, in which lower level information is extracted from small spatial or temporal segments of the data, and b) feature aggregation, in which the extracted information is fused into a compact representation that can be used for the subsequent tasks, e.g., classification, retrieval, etc. DL unified, to some extent, these two steps by employing 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 [8]. 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. As we will demonstrate through this paper, 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 requires 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 *sitting down* must be distinguished from the action of *standing up*. Feature vectors can be extracted from every video frame or a sequence of them by using a deep CNN. 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 is 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. Therefore average and max pooling layers 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 may be flatten to a vector and fed to the subsequent 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 shifts (the features must always arrive at the exact same moment).

In this paper, we propose a novel stateful recurrent pooling approach that is capable of overcoming these limitations. The proposed method builds upon the well-known Bag-of-Feature (BoF) model [11], which is capable of creating a constant-length representation of a multimedia object, e.g., video, audio, etc., by compiling a histogram over its quantized feature vectors. Therefore, every object is represented using a fixed-length histogram over the learned codewords.

The codewords/dictionary can be either learned using generative/reconstruction approaches [11], or by employing discriminative dictionary learning methods [4], which usually better fit classification tasks. This scheme found application in numerous computer vision tasks, including scene recognition [7], texture classification [16], etc. 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 [9]. Despite its remarkable success in various tasks and its ability to handle variable size inputs, the main drawback of BoF-based methods is the loss of spatial and temporal information, as well as their inability to capture the geometry of input data [7].

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. To overcome this limitation, we suggest that 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" (given in the previous examples) 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". In this way, it would be possible to extract a representation that can discriminate the standing up action from the sitting down action.

The proposed method is inspired by the BoF model, but employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach to overcome the limitation of existing BoF formulations. 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. In this work, 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 other naive pooling layers, that can lead to significant loss of temporal information, the extracted feature vectors are quantized to a number of codewords in a recurrent manner, enabling us 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, while at the same time keeping the overall space and time complexity low compared to competitive approaches. In this way, the proposed method holds the credentials for providing fast and efficient human-centric perception methods for various embedded and robotics applications [6]. To the best of our knowledge, in this paper we propose the first stateful recurrent Bag-of-Features model that is capable of effectively modeling the temporal dynamics of video sequences. It is also worth

noting that existing BoF formulations, e.g., [9], merely provide models that fully ignore the temporal information.

The rest of the paper is structured as follows. In Section 2 the proposed Recurrent BoF (ReBoF) method is analytically derived. Then, the experimental evaluation of the ReBoF method is provided in Section 3. A thoroughly discussion on how ReBoF could be used for practical applications, along with its limitations, are provided in Section 4, while conclusions are drawn and future work is discussed in Section 5.

2 Proposed Method

In this Section, we first briefly introduce the regular (non-recurrent) Bag-of-Features model. Then, we derived the proposed Recurrent Bag-of-Features formulation, draw connections with the regular Bag-of-Features model and discuss how it can be used for video classification tasks.

2.1 Bag-of-Features

Let $\mathcal{X} = \{x_i\}_{i=1}^{N}$ be a set of N videos to be represented using the standard BoF model. From each video N_i feature vectors are extracted: $\mathbf{x}_{ij} \in R^D (j = 1, ..., N_i)$, where D is the dimensionality of each feature vector. BoF provides a way to efficiently aggregate these features into a fixed-length histogram. To this end, each feature vector is first quantized into a predefined number of codewords, by employing a codebook $\mathbf{V} \in R^{N_K \times D}$, where N_K is the number of codewords. This codebook is usually learned by clustering all feature vectors into N_K clusters [11]. Clustering algorithms, such as k-means, can be used to this end, with each centroid, $\mathbf{v}_k \in R^D (k = 1, ..., N_K)$, corresponding to a codeword. Then, the quantized feature vectors of each object are accumulated to form the final histogram. Even though several feature quantization approaches have been proposed [11], this work focuses on using a soft quantization approach that allows for learning the codebook using regular back-propagation, along with the rest of the parameters of the model [10]. This can significantly improve the discriminative power of the model, since the codebook is adapted for the task at hand.

More specifically, each 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:

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

where σ controls the fuzziness of the quantization process. Then, for each feature vector we obtain a fuzzy membership vector, after normalizing the observed similarities as:

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

Finally, the final histogram is extracted by accumulating all the normalized membership vectors as:

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

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

2.2 Proposed Recurrent BoF

The histogram extracted using the regular BoF formulation, as described previously, discards any spatial or temporal information encoded by the feature vectors. To overcome this limitation, in this work we propose using a recurrent stateful quantizer, which allows for capturing and effectively encoding the temporal information expressed by the order in which the feature vectors arrive to the model. Note that in the case of video, we assume that the feature vector $\mathbf{x}_{ij}$ corresponds to the j-th timestep of the i-th video sequence. Before deriving the proposed recurrent quantization approach, it is worth examining, from a probabilistic perspective, the quantization process involved in the BoF model. Using Kernel Density Estimation [1], we can estimate the probability of observing the feature vector $\mathbf{x}_{ij}$, given an input object x_i, 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 the histogram $\mathbf{s}_i \in \mathbb{R}^{N_K}$ separately adjust the density estimation, while $K(\cdot)$ is a kernel function. Then, a maximum likelihood estimator can be used to actually calculate the histogram:

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

The involved parameters (histogram) can be estimated as $\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{u}_{ij}$, as shown in [1], where

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

Using this formulation, we can re-derive the regular BoF with soft-assignments. Also, note that we can also replace the Gaussian kernel used in (1), which typically requires tuning the width σ, with an easier to use hyperbolic kernel, which

does not require tuning any hyper-parameter. The hyperbolic kernel also proved to be stabler and easier to use when the proposed method was combined with deep neural networks. Therefore, we can now measure the similarity between each feature vector and the codewords in order to quantize the feature vectors as:

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

where

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{8}$$

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, we extend (7) in order to take into account the temporal information, as expressed by the histogram extracted until the current step, as:

$$[\mathbf{d}_{ij}]_k = \sigma(\mathbf{V}\mathbf{x}_{ij} + \mathbf{V}_h(\mathbf{r}_t \odot \mathbf{s}_{i,j-1})) \in \mathbb{R}^{N_K}, \tag{9}$$

where $\mathbf{V}_h \in \mathbb{R}^{N_K \times N_k}$ is a weight matrix that is used to transfer the gated histogram vector into the quantization space, $\mathbf{s}_{i,j-1}$ is the histogram extracted from previous quantizations (state) and $\mathbf{r}_j \in \mathbb{R}^{N_K}$ is the output of a reset gate, introduced to ensure the long-term stability of the model. The additional parameters introduced in this formulation are learned during the training process using back-propagation. The proposed method also employs a reset gate, inspired by the GRU model [2], to further increase the stability of the learning process. Therefore, the reset gate is defined as:

$$\mathbf{r}_{ij} = \sigma(\mathbf{W}_r \mathbf{x}_{ij} + \mathbf{U}_r \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \tag{10}$$

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, the l_1 normalized membership vector is computed similarly to the regular BoF model as:

$$\mathbf{u}_{ij} = \frac{\mathbf{d}_{ij}}{\|\mathbf{d}_{ij}\|_1}. \tag{11}$$

Note that the initial state $\mathbf{s}_{i,0}$ is initialized to:

$$\mathbf{s}_{i,0} = \frac{1}{N_K}\mathbf{1}, \tag{12}$$

where N_K is the number of codewords and $\mathbf{1} \in R^{N_K}$ is a vector of all ones. This ensures that quantizer's output will be always a properly normalized membership vector. Therefore, the histogram is recurrently updated as:

$$\mathbf{s}_{i,j} = (\mathbf{1} - \mathbf{z}_{ij}) \odot \mathbf{s}_{i,j-1} + \mathbf{z}_j \odot \mathbf{u}_{ij} \in \mathbb{R}^{N_K}. \tag{13}$$

The update gate $\mathbf{z}_{i,j}$, which controls how much the current histogram will be updated, is defined as:

$$\mathbf{z}_{ij} = \sigma(\mathbf{W}_z \mathbf{x}_{ij} + \mathbf{U}_z \mathbf{s}_{i,j-1}) \in \mathbb{R}^{N_K}, \tag{14}$$

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, to compile the final histogram we average all the intermediate histograms as:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbf{s}_{i,j} \in \mathbb{R}^{N_K}. \tag{15}$$

Back-propagation can be directly used to learn all the parameters of the proposed ReBoF model. Note that ReBoF allows for capturing their temporal information, since it is capable of recursively processing the input feature vectors. First, the proposed recurrent stateful quantizer is employed to quantize the input feature vectors, as described by (9) and (11). Then, these vectors are employed to update the state of the quantizer, as descibed by (13), and allowing for compiling the resulting histogram. It is worth noting that ReBoF, similarly to all BoF-based models, is capable of processing varying-length input sequences.

ReBoF provides a significant advantage over existing BoF formulations, since it allows to capture the temporal information of sequential input data. This alows the ReBoF model to effective tackle challenging video analysis problems, e.g., video retrieval, activity recognition, etc. Therefore, to apply ReBoF we: a) use a convolutional neural network to extract a feature vector $\mathbf{x}_{ij}$ from each frame of a video, and b) feed the extracted feature vectors to the ReBoF model in order to extract a compact representation for each video. This allows ReBoF to process videos of arbitrary duration, while creating fixed-length compact representations of them. Note that the whole architecture can be trained in an end-to-end fashion, since the proposed ReBoF formulation is fully differentiable.

3 Experimental Evaluation

The proposed method was evaluated using three video activity recognition datasets, the UTKinect-Action3D [15] dataset, the UCF101 dataset [12] and a more challenging dataset, the Complex UCF101, which was designed to evaluate the ability of the methods to capture the temporal dimension of video sequences, as it will be described below. The UTKinect-Action3D [15] 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 provided RGB frames 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 UCF101 dataset [12] 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. For all the experiments conducted in this paper, the first evaluation split of the dataset was used.

We also created a challenging and more complex dataset based on the UCF101 dataset 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. 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*. Then, 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.

Table 1: UTKinect-Action3D Evaluation

Method	**# Codewords / GRU Units**	**Test Accuracy** (%)
Average Pooling	-	40.83
GRU	512	47.71
ReBoF	512	**54.64**

Table 2: UCF101 Evaluation

Method	**# Codewords / GRU Units**	**Test Accuracy** (%)
Avg. Pooling	-	70.32 ± 0.43
GRU	2048	71.04 ± 0.20
ReBoF	1024	$\mathbf{72.02} \pm 0.68$

For the UTKinect-Action3 and UCF101 datasets, every video instance was uniformly sampled in time in order to extract a predefined number of frames,

Table 3: Complex UCF101 Evaluation

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

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 [3]. Then, an Inception V3 model [13], pretrained in ImageNet, was used to extract a feature representation from each frame from the last average pooling layer of the network. Therefore, from each video, a sequence of N_f feature vectors was extracted. This sequence was then fed to the proposed ReBoF layer, followed by a hidden fully connected layer with 512 neurons and dropout with rate 0.5, as well as by the final classification layer. The ReLU activation function was used for the hidden layer, while the cross-entropy loss was used for training the model. The network was trained using the Adam optimizer and a learning rate of 10^{-5}, apart from the pretrained feature extractors which were kept frozen. Furthermore, note that we also experimentally found out that scaling the extracted histogram by N_K significantly improved the convergence of the proposed method, especially when training 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. The network was trained for 800 epochs for the UTKinect-Action3D dataset and for 500 epochs for the UCF101 (the training/evaluation procedure was also repeated 3 times and the mean and standard deviation is reported). For the Complex UCF101 a slightly different setup was used. First, a 16-frame sliding window, with overlap of 4 frames, was applied on every activity instance of UCF101 dataset, while a pretrained 3D ResneXt-101 [3] was used to extract a feature vector from each window. The features were extracted from the last average pooling layer of the network. Therefore, a 32-length sequence ($N_f = 32$) of 2048-dimensional feature vectors were extracted for each video action. The training process stopped when 99.9% accuracy was achieved in training set (for the average pooling baseline, the network was trained for 50 epochs). The evaluation of ReBoF and GRU methods was repeated 3 times and the mean accuracy and standard deviation on the test set are reported.

The performance of the proposed method was also compared to two other established pooling methods. The same feature extractors and classification block was used to ensure a fair comparison between the methods. First, global average pooling (denoted by "Average Pooling" in the rest of the paper), over the temporal dimensions of the input sequence, was used instead of the proposed ReBoF method. Furthermore, a more powerful recurrent aggregation model, a GRU [2], was also employed to aggregate the extracted features.

First, the proposed method was evaluated on the UTKinect-Action3D, while the results are reported in Table 1. The proposed method greatly outperforms the other two evaluated methods, leading to the highest accuracy (54.64%) using 512 codewords. For the GRU also 512 units were employed, since at this point GRU achieved its best accuracy, which is however significantly lower (47.71%) compared to the proposed ReBoF method. Both ReBoF and GRU outperform the plain Average Pooling, since they are capable of effectively modeling the temporal dynamics of the input video sequences allowing for better discriminating between similar activities, such as *stand up* and *sit down*.

Moreover, in Fig. 1 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 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.

Again, similar conclusions can be drawn from the evaluation results on the UCF101 dataset, which are reported in Table 2. Note that 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 (72.02%) for 1024 codewords. Again, all the methods were tuned to use the best number of codewords/representation length to ensure a fair comparison. Note that the proposed method outperforms the GRU while using representations with half the size of the ones used by the GRU. The effect of using different number of codewords and recurrent units is also evaluated in Fig. 2. 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.

Finally, the proposed method was also evaluated on the Complex UCF dataset. The results are provided in Table 3. As expected, Average Pooling fails to overpass the 50% test accuracy, since "AB" activities cannot be distinguished from those of "BA", due to employing global averaging, which completely discards the temporal information. On the other hand, ReBoF again achieves the highest accuracy (89.29% when 1024 codewords are used). It is worth noting that even

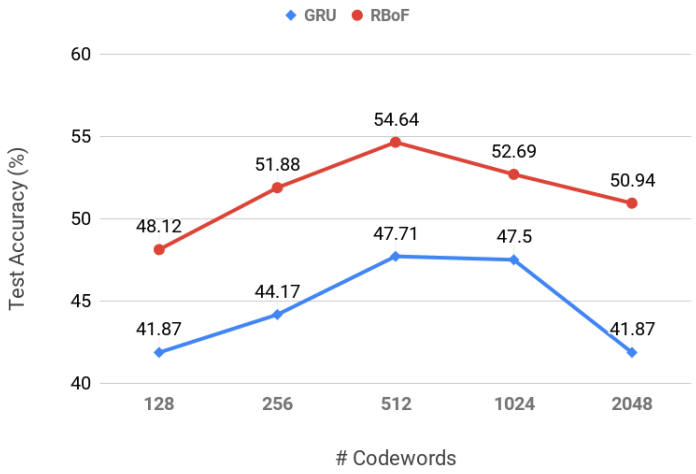

Fig. 1: UTKinect-Action3D Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

though spatio-temporal information is already encapsulated in the extracted feature vectors, since 3D kernels were used in feature extraction block, GRU again achieves lower accuracy compared to the ReBoF for both representation sizes.

4 Discussion

In this section, some further practical details regarding the employment of ReBoF method are presented, providing more insight into the proposed method compared to our previous works, e.g., [5]. A more thorough inspection of equations (9), (10), (13) and (14) implies that ReBoF shares some similarities with GRU units [2], since both ReBoF and GRUs use a similar update-reset gate structure. Therefore, the RBoF method can be implemented by modifing an existing (and optimized) GRU implementation by a) replacing the output activation function (in order to ensure the quantization of the feature vectors), b) setting the initial state $\mathbf{s}_{i,0}$ (to ensure that the histogram vector will maintain a unit l^1 norm) and c) appropriately initializing the codebook. This allows us to simply modify existing and highly optimized GRU implementations to provide highly efficient ReBoF implementations. As far as the number of employed codeworks, a rule of thumb, based on the current work's analysis, could be to initialize ReBoF with 512 codewords, since it can provide a balanced trade-off among feature space dimensionality and model accuracy.

Furthermore, the advantages of employing ReBoF architecture in the corresponding framework for video analysis might be limited under certain circumstances. In case that temporal information is not crucial to distinguish different

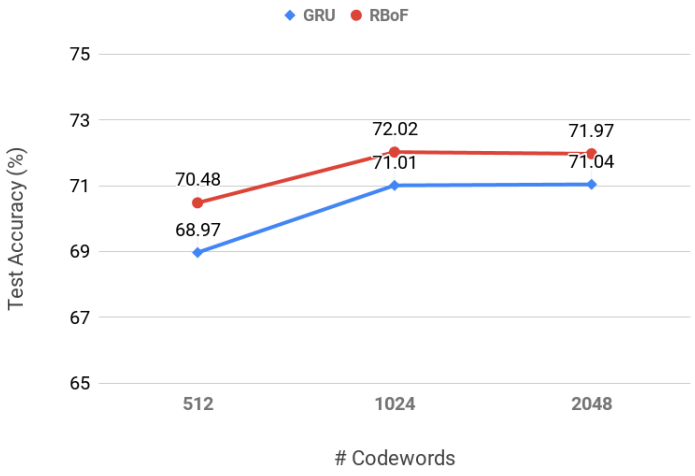

Fig. 2: UCF101 Evaluation: Effect of using different number of codewords/recurrent units for the ReBoF and GRU methods

video instances, the employment of ReBoF is not guaranteed that will lead to significantly higher performance compared to a simpler architecture designed to analyze video instances by a single or a small set of frames. Yet, any method that discards temporal information is incapable to cope with tasks where crucial information is contained over temporal dimension of data. This is clearly demonstrated via the experimental results on UCF101 and Complex UCF101 datasets. In the first case, the employment of ReBoF slightly improves the model accuracy, since performing classification of the videos of UCF101 can considered a not so challenging task in terms of temporal dependence, given that in many cases we can achieve quite high recognition accuracy just using one frame from the videos. On the contrary, in the second case of Complex UCF101, enclosing temporal information is crucial to distinguish different video activities and thus, the employment of ReBoF leads to significant performance improvements compared to methods that discard temporal information. Finally, it should be noted that feeding to the ReBoF layer feature vectors that a priori enclose temporal information might lead to improvements to some extent however, this could also disproportionately increase overall the complexity of the models.

5 Conclusions

A novel stateful Recurrent Bag-of-Features (ReBoF) model was proposed in this paper. ReBoF employs a stateful trainable recurrent quantizer, instead of a plain static quantization approach, as the one used in existing BoF formulations. This

allows ReBoF to capture the temporal information of the 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 in DL models and 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.

ReBoF opens several interesting future research directions. First, 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 [7]. Furthermore, 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 spatiotemporal histograms. Finally, using ReBoF for other tasks, such as video retrieval and hashing [14], is expected to further boost the performance of existing methods by extracting compact, yet more discriminative representations.

References

1. Bhattacharya, S., Sukthankar, R., Jin, R., Shah, M.: A probabilistic representation for efficient large scale visual recognition tasks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2593–2600 (2011)
2. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6546–6555 (2018)
4. Iosifidis, A., Tefas, A., Pitas, I.: Discriminant bag of words based representation for human action recognition. Pattern Recognition Letters **49**, 185–192 (2014)
5. Krestenitis, M., Passalis, N., Iosifidis, A., Gabbouj, M., Tefas, A.: Recurrent bag-of-features for visual information analysis. Pattern Recognition p. 107380 (2020)
6. Lane, N.D., Bhattacharya, S., Mathur, A., Georgiev, P., Forlivesi, C., Kawsar, F.: Squeezing deep learning into mobile and embedded devices. IEEE Pervasive Computing **16**(3), 82–88 (2017)
7. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. vol. 2, pp. 2169–2178 (2006)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
9. Passalis, N., Tefas, A.: Learning bag-of-features pooling for deep convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5755–5763 (2017)
10. Passalis, N., Tefas, A.: Neural bag-of-features learning. Pattern Recognition **64**, 277–294 (2017)

11. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: Proceedings of the International Conference on Computer Vision. p. 1470 (2003)
12. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
13. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)
14. Tang, J., Lin, J., Li, Z., Yang, J.: Discriminative deep quantization hashing for face image retrieval. IEEE Transactions on Neural Networks and Learning Systems (2018)
15. Xia, L., Chen, C., Aggarwal, J.: View invariant human action recognition using histograms of 3d joints. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 20–27 (2012)
16. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. International Journal of Computer Vision **73**(2), 213–238 (2007)