Spatio-temporal VLAD Encoding for Human Action Recognition in Videos

Ionut C. Duta[1], Bogdan Ionescu[2], Kiyoharu Aizawa[3], and Nicu Sebe[1]

[1] University of Trento, Italy
{ionutcosmin.duta, niculae.sebe}@unitn.it
[2] University Politehnica of Bucharest, Romania
bionescu@imag.pub.ro
[3] University of Tokyo, Japan
aizawa@hal.t.u-tokyo.ac.jp

Abstract. Encoding is one of the key factors for building an effective video representation. In the recent works, super vector-based encoding approaches are highlighted as one of the most powerful representation generators. Vector of Locally Aggregated Descriptors (VLAD) is one of the most widely used super vector methods. However, one of the limitations of VLAD encoding is the lack of spatial information captured from the data. This is critical, especially when dealing with video information. In this work, we propose Spatio-temporal VLAD (ST-VLAD), an extended encoding method which incorporates spatio-temporal information within the encoding process. This is carried out by proposing a video division and extracting specific information over the feature group of each video split. Experimental validation is performed using both hand-crafted and deep features. Our pipeline for action recognition with the proposed encoding method obtains state-of-the-art performance over three challenging datasets: HMDB51 (67.6%), UCF50 (97.8%) and UCF101 (91.5%).

Keywords: Action Recognition · Video Classification · Feature Encoding · Spatio-temporal VLAD (ST-VLAD)

1 Introduction

Action recognition has become an important research area in computer vision and multimedia due to its huge pool of potential applications, such as automatic video analysis, video indexing and retrieval, video surveillance, and virtual reality. The most popular framework for action recognition is the Bag of Visual Words (BoVW) with its variations [16, 32, 34]. The BoVW pipeline contains three main steps: feature extraction and feature encoding followed by classification. In addition to these, there are several pre- and post-processing steps, such as Principal Component Analysis (PCA) with feature decorrelation or different normalizations, which can improve the performance of the system.

In what concerns the feature extraction techniques, the recent approaches based on convolutional neural networks (ConvNets) [12, 26, 25, 38, 37, 29, 20, 4]

have proven to obtain very competitive results compared to traditional hand-crafted features such as Histogram of Oriented Gradients (HOG) [5, 16], Histogram of Optical Flow (HOF) [16] and Motion Boundary Histograms (MBH) [6]. Regarding the encoding step, we can notice that in the recent approaches, super-vector based encoding methods, such as Fisher Vector [23] and VLAD [11] are presented as state-of-the-art approaches for the encoding step of action recognition tasks [34, 22, 30, 31, 18].

In this work we extend VLAD encoding method by considering feature localization within the video. There are many precursors who focus on improving VLAD representation, as this is an efficient super vector-based encoding method with very competitive results in many tasks. The work in [18] proposes to use Random Forest in a pruned version for the trees to build the vocabulary and then additionally concatenate second-order information, similar as in Fisher Vectors (FV) [23]. The work [2] uses intra-normalization to improve VLAD performance. The authors propose to L2 normalize the aggregated residuals within each VLAD block to suppress the negative effect of too large values within the vector. In our approach we use average pooling which deals with this issue. The work [14] proposes an extension to Spatial Fisher Vector (SFV) which computes per visual word the mean and variance of the 3D spatio-temporal location of the assigned features. VLAD encoding method can be viewed as a simplification of FV, which keeps only first-order statistics and performs hard assignment.

In [1], the RootSIFT normalization is proposed to improve the performance of the framework for object retrieval, by computing square root of the values to reduce the influence of large bin values. As recommended by [34] we also use this normalization for hand-crafted features. The works [17, 16] consider Spatial Pyramid approach to capture the information about features location, however, the scalability is an issue for this method, as it increases considerably the size of the final representation and it is not feasible for more than four video divisions. In [8] the performance of VLAD is boosted by using a double assignment approach. The authors of [21] suggest improving VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. Different from the above works, our method focuses on efficient capture the spatio-temporal information directly within the encoding process.

The encoding method which provides the final video representation is crucially important for the performance of an action recognition system as influences directly the classifier predicted class. The information regarding the location within the video and the feature grouping based on this can provide an additional useful source of information for the performance of the system. Besides the effectiveness of the super vector-based encoding method VLAD, the information regarding the spatio-temporal locations of the features is not considered. In this paper we tackle this limitation by proposing an encoding approach which incorporates the spatio-temporal information into the encoding step. We present Spatio-temporal VLAD (ST-VLAD), an encoding approach which aims to divide the video in different parts and performs one pooling based on the specific group of features regarding their location within the video, the other pooling is executed

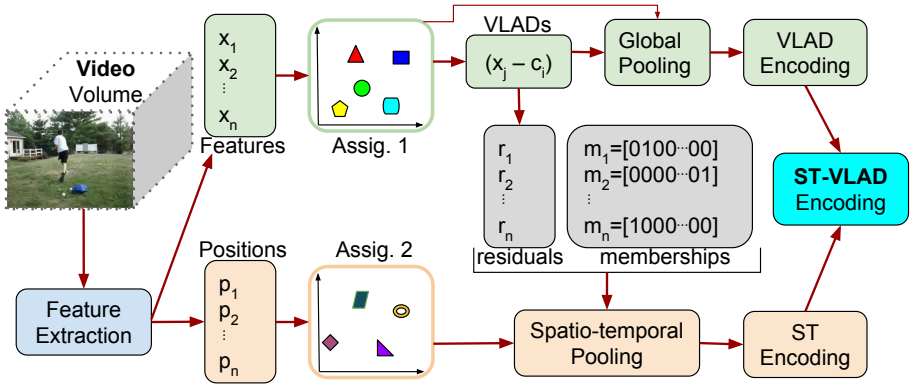


Fig. 1: The ST-VLAD framework for features encoding.

based on the similarity of the features. This provides an important additional information which contributes on improving the overall system performance.

The reminder of the paper is as following. Section 2 introduces the preposed encoding method. In Section 3 is presented local deep feature extraction pipeline. The experimental evaluation and the final results are outlined in Section 4. Finally, Section 5 concludes this work.

2 Proposed ST-VLAD encoding method

In this section we present our approach for video features encoding, Spatio-temporal VLAD (ST-VLAD). The goal of our method is to provide a better video representation by taking into consideration the spatio-temporal information of the features. VLAD is a super vector-based encoding method proposed initially in [11]. We embed in VLAD encoding method the information regarding the position of the features within the video. To include spatio-temporal information into the encoding step, apart from the features of a videos, we also retain their positioning within the video. To each feature we associate a position p:

$$\mathrm{p} = (\bar{x}, \bar{y}, \bar{t});\ \bar{x} = \frac{x}{h},\ \bar{y} = \frac{y}{w},\ \bar{t} = \frac{t}{\#fr} \tag{1}$$

where h, w and $\#fr$ represent the height, width and the number of frames of the video respectively. Therefore, $\bar{x}$, $\bar{y}$, $\bar{t}$ correspond to the normalized x, y, t position with respect to the video. This normalization guarantees that the position values range between the same interval [0 1] for any video input.

We initially learn a codebook with the size of $k1$ visual words with k-means C=$\{c_1, c_2, ..., c_{k1}\}$, which are the means for each cluster. The codebook C is learnt from a subset of features extracted from a subset of videos. In parallel we learn another codebook of $k2$ visual words with k-means PC=$\{pc_1, pc_2, ..., pc_{k2}\}$, which represents the points codebook. The codebook PC is computed from the location information of the features used for the first codebook C. This is an automatic way to propose a spatio-temporal video $k2$ divisions, which is independent from the feature extraction algorithm. Therefore, can be applied

without any modification to any type of method for establishing the region of the feature extraction, such as dense [36, 31] or at specific locations [34].

Fig. 1 presents an illustration of our pipeline for feature encoding with the path that a given video traverses to obtain its final representation, which serves as the input for a classifier. First, the video information is represented by computing certain content features. Then we aim at performing two hard assignments using the codebooks obtained before. The first assignment is performed as in standard VLAD, where each local video feature $\mathrm{x_j}$ from the set $X = \{\mathrm{x_1}, \mathrm{x_2}, ..., \mathrm{x_n}\} \in R^{\mathrm{n}\times\mathrm{d}}$ is assigned to its nearest visual word (d is the feature dimensionality). After computing the residuals (the difference between the local feature and its assigned centroid), which we can call also the result VLADs vectors, we perform two actions. The first action is to perform a global pooling over the residuals for each centroid. Instead of doing sum pooling as in the standard VLAD, we perform average pooling over the residuals of each visual word:

$$v_i = \frac{1}{N_i}\sum_{j=1}^{N_i}(\mathrm{x_j} - c_i) \tag{2}$$

where N_i is the number of features assigned to the cluster c_i. This division by the number of descriptors, that switches sum pooling to average pooling, is a very simple technique to deal with the problem of burstiness when some parts of VLAD vector can dominate the entire representation and may influence negatively the performance of the classifier. In the rest of the paper we refer to VLAD under this version. The concatenation of all v_i gives the video representation of VLAD encoding, which is a vector with size $k1\times\mathrm{d}$.

The second action is to save the residuals, therefore, to the local set of features $X \in R^{\mathrm{n}\times\mathrm{d}}$ is associated with the residuals $R = \{\mathrm{r_1}, \mathrm{r_2}, ..., \mathrm{r_n}\} \in R^{\mathrm{n}\times\mathrm{d}}$. With the objective of preserving the associated similarity-based cluster information, we retain, together with residuals, the information regarding their centroid membership. We represent the membership information by a vector m with the size equal to the number of visual words $k1$, where all the elements are zero, except one value (which is equal to 1) that is located on the position corresponding to the associated centroid. For instance m=[0100...00] maps the membership feature information to the second centroid of the codebook C.

At the bottom part of Fig. 1 is represented the encoding path where we consider the position of the extracted features P=$\{\mathrm{p_1}, \mathrm{p_2}, ..., \mathrm{p_n}\} \in R^{\mathrm{n}\times 3}$. We perform a second assignment based on the feature location within the video. This step proposes $k2$ splits of the obtained residuals. We perform the spatio-temporal pooling over each proposed division of the video by doing average pooling of the residuals part and sum pooling over the memberships. We obtain a new video representation, spatio-temporal encoding (ST encoding) by concatenating all the pooling results over the $k2$ divisions. The size of the resulted vector of ST encoding is $k2\times(d+k1)$.

After we obtain the representations of VLAD and ST encoding, we concatenate them in a vector which represents the final representation for a video. The size of the resulted final representation is $k1\times\mathrm{d} + k2\times(d+k1)$. Besides the ad-

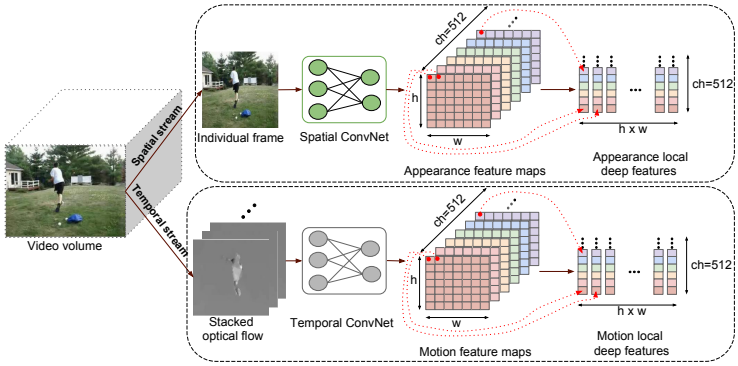


Fig. 2: Two-stream deep feature extraction pipeline for action recognition.

ditional information provided, this proposed representation has the advantage of the scalability over the number of video divisions. For instance, if we consider other methods such as Spatial Pyramid (SP) [17, 16], for a division into 3 parts, the final video description is the concatenation of the resulted encoding for (3+1) times (+1 is for the whole video). If we apply VLAD with the classic 256 clusters and the feature dimensionality is 54 then the size of final representation resulted with SP is 55,296 ($256 \times 54 \times 4$), while our ST-VLAD generates a final representation with the size 14,754 ($(256 \times 54) + (3 \times (256 + 54))$). If we increase the number of video divisions, for instance to 32, then the final vector size generated by SP explodes to more than 456K, which makes it very demanding for the computational resources, while the size of ST-VLAD is still reasonable (23,744). We make the code for ST-VLAD publicly available[4].

3 Local deep feature extraction

This section presents the framework to extract deep local features for a given video. The approaches based on convolutional networks (ConvNets) [12, 26, 25, 38, 37, 29, 20, 4] obtained very competitive results over traditional hand-crafted features. In this work we also consider deep features resulted from ConvNets. Based on the fact that videos contain two main source of information, appearance and motion information, we consider also two ConvNets for capturing separately each information type. The pipeline for deep feature extraction is similar to [8].

For capturing the appearance information in our spatial stream we use the VGG ConvNet in [26]. This is a very deep network with 19 layers, including 16 convolutional layers and 3 fully connected layers. Together with a very high depth, VGG19 is characterized by a smaller size of the convolutional filters i.e., 3×3 and the stride is only 1 pixel. These enable the network to explore finer-grained details from the feature maps. This network is trained on the ImageNet dataset [7], with state-of-the-art results for image classification. The used pipeline for the deep feature extraction is depicted in Fig. 2.

The input of VGG19 ConvNet is an image of spatial size of 224×224 with three channels for the color information. After we extract the individual frames

[4] `http://disi.unitn.it/~duta/software.html`

from a video, we accordingly resize them for the request input of the network. For each individual frame we take the output of the last convolutional layer with spatial information, pool5, of the spatial ConvNet. Our choice for the convolutional layer is motivated by the fact that the deeper layers provide high discriminative information and by taking a layer with spatial information we can extract local deep features for each frame of the video. The output of pool5 is the feature maps with a spatial size of 7×7 and 512 channels. For extracting local deep feature from feature maps we take each spatial location from all channels and concatenate them to create the local deep features with 512 dimensions. Hence, from each frame we obtain $7 \times 7 = 49$ local deep features and each feature is a 512 dimensional vector. Therefore, for each video we obtain in total $\#frames \times (7 \times 7)$ local deep features.

To capture the motion information we use the re-trained network in [37]. This deep network, also VGG, is initially proposed in [26] and contains 16 layers. The authors of [37] re-trained the VGG ConvNet for a new task with new input data using several good practices for the network re-training, such as pre-training to initialize the network, smaller learning rate, more data augmentation techniques and high dropout ratio. The VGG ConvNet is re-trained for action recognition task using the UCF101 dataset [28]. The input for the temporal ConvNet is 10-stacked optical flow fields, therefore, in total there are 20-staked optical flow images as one input for the network. To extract optical flow fields we use the OpenCV implementation of TVL1 algorithm [39]. For the temporal ConvNet we also take the output of the last convolutional layer with structure information: pool5. The pool5 layer has the spatial size of feature maps of 7×7 and 512 channels. The final local deep features for an input are obtained by concatenating the values from each spatial location along all the channels, resulting in 49 local features for an input. The total number of local deep features for a video for the temporal ConvNet are $(\#frames - 9) \times 7 \times 7$. In the case of deep local features, the normalized positions, needed for ST-VLAD, are extracted based on the localization on the feature maps.

4 Experimental evaluation

This part shows the experimental evaluation of our method, ST-VLAD, to create a final representation for a video. We validate our approach using both hand-crafted and deep features. The pipeline for extraction of deep local features is presented in the previous section, which provides us two features for the spatial and temporal stream: Spatial ConvNet (SCN) and Temporal ConvNet (TCN). Authors of [37] provide three networks re-trained for each split of UCF101 dataset. For the local deep feature extraction with temporal network another detail is that for the datasets UCF50 and HMDB51 we use the re-trained network from [37] for split1 and for UCF101 we accordingly use the re-trained networks for each split.

As hand-crafted features, we use Improved Dense Trajectories (iDT) approach with the code provided by the authors [34], keeping the default parameter settings recommended to extract four different descriptors: HOG, HOF, MBHx

and MBHy. This is one of the state-of-the-art hand-crafted approaches for feature extraction. iDT is an improved version of [32] which removes the trajectories that are generated by camera motion. Each of these four descriptors is extracted along all valid trajectories and the resulted dimensionality is 96 for HOG, MBHx and MBHy, and 108 for HOF.

As default setting we use k-means for generating the codebook from 500K random selected features. Before encoding step, we apply only for hand-crafted features the RootSIFT normalization [1] similar to [34], then for both hand-drafted and deep features we perform PCA to reduce the dimensionality by a factor of two and decorrelate the features. Therefore, the feature size of HOG, MBHx and MBHy is 48, for HOF is 54, and for the resulted deep feature size for SCN and TCN is 256.

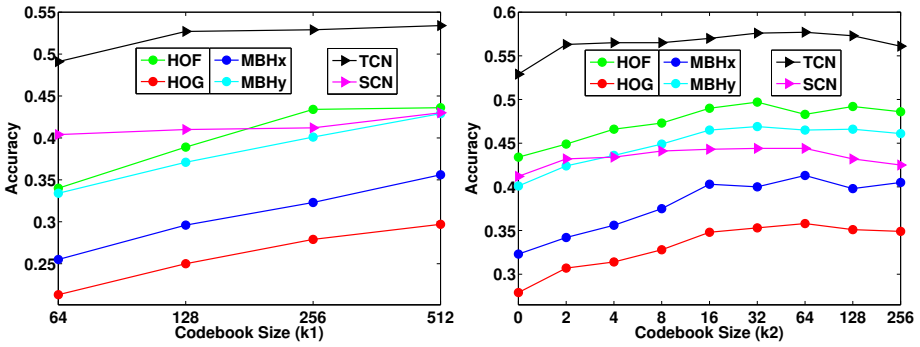
After feature encoding we apply Power Normalization (PN) followed by L2-normalization ($||sign(\mathrm{x})|\mathrm{x}|^{\alpha}||$, where $0 \leq \alpha \leq 1$ is the normalization parameter). For all experiments, we fix $\alpha = 0.1$ for hand-crafted features and $\alpha = 0.5$ for deep features. For the classification step we use a linear one-vs-all SVM with C=100. It is very important when different sources of information are combined to have values between the same range. Otherwise, the feature components with a bigger absolute value will dominate the vector representation and will weight more for the classifier, thus, may influence negatively the performance. The PN has a positive effect in reducing the picks within the vector, and α controls the level of penalization, by giving a smaller α, the large values are shrinked more. We combine different source of information, for instance the sum pooling over the membership information may give big values, therefore it is necessarily to balance the values within the final video representation vector. The reason of applying a different normalization level α depending on the feature type is related with the initial VLAD values before the concatenation with ST representation. For instance, the resulted values for VLAD representation of any iDT feature can range between [-0.5 0.5] but for the deep features can range between [-200 200], and ST representation can come with values until around 1500 (from the sum pooling of membership information). Therefore the absolute values resulted for deep features are more balanced after concatenation and we apply only PN with α=0.5, but for iDT as the values are less balanced, it is necessarily a bigger penalization of the vector picks, thus, we perform PN with smaller α of 0.1.

4.1 Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: HMDB51 [15], UCF50 [24] and UCF101 [28].

The HMDB51 dataset [15] contains 51 action categories, with a total of 6,766 video clips. It is one of the most challenging dataset with realistic settings. We use the original non-stabilized videos, and we follow the original protocol using three train-test splits [15]. We report average accuracy over the three splits as performance measure.

In total the UCF50 dataset [24] contains 6,618 realistic videos taken from YouTube. There are 50 human action categories mutually exclusive. The videos are split into 25 predefined groups. We follow the recommended standard proce-



(a) (b)

Fig. 3: Evaluation of the vocabulary size: (a) for VLAD, (b) for ST-VLAD.

dure and perform leave-one-group-out cross validation and report average classification accuracy over all 25 folds.

The UCF101 dataset [28] is a widely adopted benchmark for action recognition, consisting in 13,320 realistic videos, which are divided in 25 groups for each action category. This dataset contains 101 action classes and there are at least 100 video clips for each class. We follow for evaluation the recommended default three training/testing splits. We report the average recognition accuracy over these three splits.

4.2 Parameter tuning

In this set of experiments we evaluate different sizes of the vocabularies (C and PC) on the HMBD51 dataset. First, we tested different numbers of visual words ($k1$) for VLAD codebook. Fig. 3a represents the graph evolution for VLAD performance when the codebook size varies. By increasing the codebook size the performance of VLAD is improved, especially for iDT features, where we can notice a consistent improvement. The impact of increasing the codebook size for the deep features is less steep, this is probably due to the fact that in general for a video, the number of generated iDT features is higher than in the case of deep features. While the codebook size of 512 gives the best results, we choose VLAD with the codebook size of 256 as a standard size of the vocabulary, since this is the best trade-off between the performance and the size of the resulted final representation.

Fig. 3b illustrates the graph with the results for our proposed encoding method, ST-VLAD (Spatio-temporal VLAD). For this set of experiments we fix the size of $k1$ for VLAD to standard value of 256 visual words and we perform the tuning of $k2$, for the spatio-temporal divisions. The "0" value on the graph for the codebook size is for showing the results only for VLAD (with 256 visual words) without the contribution of the spatio-temporal encoding. By increasing the codebook size, the final results for all considered features continue to increase the performance up to the value of 32 video divisions, except for HOG and MBHx for which the accuracy continues to grow. Considering this and also taking into account the trade-off between accuracy and the final video

Table 1: The performance (mean accuracy %) comparison of our encoding approach, ST-VLAD, to the baseline.

	HMDB51 (%)						UCF50 (%)					
	HOF	HOG	MBHx	MBHy	SCN	TCN	HOF	HOG	MBHx	MBHy	SCN	TCN
VLAD256	43.4	27.9	32.3	40.1	41.2	52.9	82.9	70.3	78.7	82.7	84.5	96.0
VLAD512	43.6	29.7	35.6	42.9	43.0	53.4	84.0	72.3	80.4	84.3	84.2	95.8
ST-VLAD	**49.7**	**35.3**	**40.0**	**46.9**	**44.4**	**57.6**	**85.6**	**77.5**	**82.3**	**85.1**	**85.5**	**96.8**

	UCF101 (%)					
	HOF	HOG	MBHx	MBHy	SCN	TCN
VLAD256	70.8	58.2	66.3	69.5	75.9	85.1
VLAD512	72.9	59.6	69.2	72.0	75.4	85.5
ST-VLAD	**75.0**	**66.0**	**71.6**	**73.7**	**78.0**	**86.3**

representation size, we fix the codebook size PC to 32 for all features in the next experiments. Therefore, we have the default settings for ST-VLAD of 256 for $k1$ and 32 for $k2$.

The performance of all iDT features has a continuous significant gain until $k2$=32, the algorithm behind iDT approach removes a consistent number of trajectories. Therefore, the features are not extracted anymore densely, instead they are extracted only at some positions, and by using an encoding method that provides information about the grouping of trajectories, brings an important source of information for the final representation.

4.3 Comparison to baseline

The exact numbers of the comparison of our approach, ST-VLAD, with the baseline over all three considered datasets, can be visualized in Table 1. The parameters of ST-VLAD are fixed in the subsection above, and for the direct comparison with the baseline we take VLAD with the standard codebook sizes of 256 and 512. The ST-VLAD encoding approach outperforms VLAD by a large margin over all 6 features for each dataset.

The performance growth is prominent, especially on the HMDB51 dataset, which is one of the most challenging video collections for action recognition, where there is a considerable room for improvement. For instance, on HOF and HOG features, by adding the spatio-temporal information to VLAD256 boosts the performance with 6.3 and 7.4 percentage points respectively. Increasing the vocabulary size for VLAD to 512 may improve the performance, however, our approach still outperforms VLAD512 by a large margin, besides the fact that VLAD512 generates a higher size for the final video representation than our proposed approach. For instance, the size of the video representation using VLAD512 in the case of HOF descriptor is 27,648, though, our representation is 23,744; and in the case of deep local features (SCN or TCN) the video representation size for VLAD512 is 131,072, nevertheless, ST-VLAD representation size is 81,920.

Regarding the computational cost, we randomly sampled 500 videos form HMDB51 dataset, and report the number of frames per second for each encoding method, choosing as feature type TCN for the measurements which obtains

Table 2: The final feature fusion results for ST-VLAD.

	HMDB51 (%)			UCF50 (%)			UCF101 (%)		
	iDT	2St	iDT+2St	iDT	2St	iDT+2St	iDT	2St	iDT+2St
Early	58.4	61.2	66.5	90.6	97.6	97.1	82.9	90.2	91.3
sLate	56.3	59.2	64.6	89.8	97.1	95.6	81.0	88.7	88.0
wLate	56.7	60.1	65.5	90.0	97.6	97.6	81.2	88.9	89.9
sDouble	57.4	60.8	66.0	90.1	97.4	96.1	81.9	89.2	88.7
wDouble	**59.0**	**62.1**	**67.6**	**90.7**	**97.8**	**97.9**	**83.0**	**90.2**	**91.5**

the best performance over all the other features. Despite that our method outperforms by a large margin VLAD512 in terms of performance, ST-VALD is not more demanding for the computational cost. In fact, ST-VLAD is slightly faster, being able to process 1,233 frames/sec, while VLAD512 runs at a frame rate of 1,059. Obviously, the fastest method is VLAD256, with 1,582 frames/sec.

4.4 Comparison to state-of-the-art

For the comparison with state-of-the-art we evaluate the performance of our approach, ST-VLAD, by using three feature combinations. The first one is the combination of HOF, HOG, MBHx and MBHy to obtain the improved Dense Trajectories representation (iDT), the second one is to combine both deep features (SCN and TCN) to get the two-stream representation (2St). The last one is to combine all the 6 features for obtaining the final representation (iDT+2St).

For those three feature combination we tested both early and late fusion approaches. Table 2 presents the feature fusion results for all three datasets. For early fusion we concatenate for each specific combination the resulted individual video representation for each feature. Then we apply again the normalization, but only L2 norm, and this time we apply it over the resulted vector of the concatenation of the features. In the end we give the final vector as input to the linear classifier to get the predicted classes.

Late fusion is performed in two manners. For the first one we just make the sum (sLate) of the classifier output over each feature. For the second one we perform a wighted sum (wLate), where we give different weights for each feature representation classifier output, and then we perform the sum. The weights combination are tuned by taking values between 0 and 1 with the step 0.1. In addition to early and late fusion, we perform a double fusion where to the sum of the classifier output of the individual feature representation we add also the classifier output of their early fusion. Similar as for late fusion, we have sum double fusion (sDouble) and weighted sum double fusion (wDouble), the difference from late fusion is that in this case there is one additional classifier output (from the early fusion) which contributes to the sum or the wighted sum tuning.

From the Table 2 we can see that in general early fusion performs better than late fusion. Double fusion combines the benefits of early and late fusion, boosting the performance and obtains the best results for all three datasets. In general the combination of hand-crafted features (iDT) with deep features (2St) raises the performance, in special on more challenging datasets such as HMDB51, while

Table 3: Comparison to the state-of-the-art.

HMDB51 (%)		UCF50 (%)		UCF101(%)	
Jain et al. [10]	52.1	Kliper-Gross et al. [13]	72.7	Karpathy et al. [12]	65.4
Zhu et al. [40]	54.0	Solmaz et al. [27]	73.7	Wang et al. [35]	85.9
Oneata et al. [19]	54.8	Reddy et al. [24]	76.9	Wang et al. [33]	86.0
Park et al. [20]	56.2	Uijlings et al. [31]	81.8	Peng et al. [22]	87.9
Wang et al. [34]	57.2	Wang et al. [32]	85.6	Simonyan et al. [25]	88.0
Sun et al. [29]	59.1	Wang et al. [34]	91.2	Sun et al. [29]	88.1
Simonyan et al. [25]	59.4	Wang et al. [33]	91.7	Ng et al. [38]	88.6
Wang et al. [33]	60.1	Peng et al. [22]	92.3	Park et al [20]	89.1
Peng et al. [22]	61.1	Ballas et al. [3]	92.8	Bilen et al. [4]	89.1
Bilen et al. [4]	65.2	Duta et al. [9]	93.0	Wang at al. [37]	**91.4**
ST-VLAD	**67.6**	ST-VLAD	**97.8**	ST-VLAD	**91.5**

on the less challenging ones such as UCF50, the hand-crafted features does not bring significant contribution.

The comparison of our method, ST-VLAD, with the state-of-the-art is presented in Table 3. We obtain state-of-the-art results on UCF50 with a performance of 97.8% and with 91.5% for UCF101. The outstanding performance of our method on the challenging dataset HMDB51 of 67.6% outperforms the state-of-the-art, including on the very recent works based on deep learning, such as [4, 29, 20], showing the effectiveness of our representation.

5 Conclusion

This paper introduces Spatio-temporal VLAD (ST-VLAD), an encoding method which captures the spatio-temporal information within the encoding process. In the experimental part we show that our approach of grouping the features and the specific information extraction brings significant additional information to the standard representation. Our pipeline obtains state-of-the-art results on three of the most challenging datasets for action recognition: HMDB51, UCF50 and UCF101. For the future work we will focus more also on capturing the relationships between the features from the same group.

Acknowledgement: This work has been supported by the EC FP7 project xLiMe.

Bibliography

[1] Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: CVPR (2012)
[2] Arandjelovic, R., Zisserman, A.: All about VLAD. In: CVPR (2013)
[3] Ballas, N., Yang, Y., Lan, Z.Z., Delezoide, B., Prêteux, F., Hauptmann, A.: Space-time robust representation for action recognition. In: ICCV (2013)
[4] Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: CVPR (2016)
[5] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
[6] Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: ECCV (2006)
[7] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)

[8] Duta, I.C., Nguyen, T.A., Aizawa, K., Ionescu, B., Sebe, N.: Boosting VLAD with double assignment using deep features for action recognition in videos. In: ICPR (2016)
[9] Duta, I.C., Uijlings, J.R.R., Nguyen, T.A., Aizawa, K., Hauptmann, A.G., Ionescu, B., Sebe, N.: Histograms of motion gradients for real-time video classification. In: CBMI (2016)
[10] Jain, M., Jégou, H., Bouthemy, P.: Better exploiting motion for better action recognition. In: CVPR (2013)
[11] Jégou, H., Perronnin, F., Douze, M., Sanchez, J., Perez, P., Schmid, C.: Aggregating local image descriptors into compact codes. TPAMI 34(9), 1704–1716 (2012)
[12] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
[13] Kliper-Gross, O., Gurovich, Y., Hassner, T., Wolf, L.: Motion interchange patterns for action recognition in unconstrained videos. In: ECCV (2012)
[14] Krapac, J., Verbeek, J., Jurie, F.: Modeling spatial layout with fisher vectors for image categorization. In: ICCV (2011)
[15] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: Hmdb: a large video database for human motion recognition. In: ICCV (2011)
[16] Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)
[17] Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
[18] Mironică, I., Duţă, I.C., Ionescu, B., Sebe, N.: A modified vector of locally aggregated descriptors approach for fast video classification. Multimedia Tools and Applications (in press, 2016)
[19] Oneata, D., Verbeek, J., Schmid, C.: Action and event recognition with fisher vectors on a compact feature set. In: ICCV (2013)
[20] Park, E., Han, X., Berg, T.L., Berg, A.C.: Combining multiple sources of knowledge in deep cnns for action recognition. In: WACV (2016)
[21] Peng, X., Wang, L., Qiao, Y., Peng, Q.: Boosting vlad with supervised dictionary learning and high-order statistics. In: ECCV (2014)
[22] Peng, X., Wang, L., Wang, X., Qiao, Y.: Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. arXiv:1405.4506 (2014)
[23] Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV (2010)
[24] Reddy, K.K., Shah, M.: Recognizing 50 human action categories of web videos. Machine Vision and Applications 24(5), 971–981 (2013)
[25] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)
[26] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
[27] Solmaz, B., Assari, S.M., Shah, M.: Classifying web videos using a global video descriptor. Machine vision and applications (2013)
[28] 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)
[29] Sun, L., Jia, K., Yeung, D.Y., Shi, B.E.: Human action recognition using factorized spatio-temporal convolutional networks. In: ICCV (2015)
[30] Uijlings, J.R.R., Duta, I.C., Rostamzadeh, N., Sebe, N.: Realtime video classification using dense hof/hog. In: ICMR (2014)
[31] Uijlings, J.R.R., Duta, I.C., Sangineto, E., Sebe, N.: Video classification with densely extracted hog/hof/mbh features: an evaluation of the accuracy/computational efficiency trade-off. International Journal of Multimedia Information Retrieval (2015)
[32] Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. IJCV 103(1), 60–79 (2013)
[33] Wang, H., Oneata, D., Verbeek, J., Schmid, C.: A robust and efficient video representation for action recognition. IJCV (2015)
[34] Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV (2013)
[35] Wang, H., Schmid, C.: Lear-inria submission for the thumos workshop. In: ICCV Workshop (2013)
[36] Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC (2009)
[37] Wang, L., Xiong, Y., Wang, Z., Qiao, Y.: Towards good practices for very deep two-stream convnets. arXiv preprint arXiv:1507.02159 (2015)
[38] Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR (2015)
[39] Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime tv-l 1 optical flow. In: Pattern Recognition (2007)
[40] Zhu, J., Wang, B., Yang, X., Zhang, W., Tu, Z.: Action recognition with actons. In: ICCV (2013)