Simple, Efficient and Effective Encodings of Local Deep Features for Video Action Recognition

Ionut C. Duta
University of Trento, Italy
ionutcosmin.duta@unitn.it

Bogdan Ionescu
University Politehnica of Bucharest, Romania
bionescu@imag.pub.ro

Kiyoharu Aizawa
University of Tokyo, Japan
aizawa@hal.t.u-tokyo.ac.jp

Nicu Sebe
University of Trento, Italy
niculae.sebe@unitn.it

ABSTRACT

For an action recognition system a decisive component is represented by the feature encoding part which builds the final representation that serves as input to a classifier. One of the shortcomings of the existing encoding approaches is the fact that they are built around hand-crafted features and they are not also highly competitive on encoding the current deep features, necessary in many practical scenarios. In this work we propose two solutions specifically designed for encoding local deep features, taking advantage of the nature of deep networks, focusing on capturing the highest feature response of the convolutional maps. The proposed approaches for deep feature encoding provide a solution to encapsulate the features extracted with a convolutional neural network over the entire video. In terms of accuracy our encodings outperform by a large margin the current most widely used and powerful encoding approaches, while being extremely efficient for the computational cost. Evaluated in the context of action recognition tasks, our pipeline obtains state-of-the-art results on three challenging datasets: HMDB51, UCF50 and UCF101.

KEYWORDS

Deep Feature Encoding; Video Classification; Action Recognition

1 INTRODUCTION

Human action recognition is one of the most challenging tasks in computer vision and multimedia, which receives a high attention from the research community due to a large number of potential applications. From the previous research works we can identify two main directions for this task. The first is based on hand-crafted features, represented by well-known descriptors, such as Histogram of Oriented Gradients (HOG) [6, 22], Histogram of Optical Flow (HOF) [22] and Motion Boundary Histograms (MBH) [7]. These descriptors are extracted from a video using different approaches to establish the region of extraction, such as at interest points [21], using a dense sampling [44, 49], along motion trajectories [40, 45, 47].

ICMR '17, Definitive version available at http://dx.doi.org/10.1145/3078971.3078988

DOI: http://dx.doi.org/10.1145/3078971.3078988

The second direction for this task relies on learning the features throughout a neural network, obtaining very competitive results [4, 19, 27, 35, 36, 41, 42, 51, 54]. Along the action recognition pipeline we can distinguish three main components: feature extraction, encoding and classification.

Feature encoding is used for building a final video representation which serves as input for a classifier and is one of the key components for the performance of the overall task pipeline. In this work we address the feature encoding step and propose solutions to improve it in the case of deep features. It is well-known that deep features are different from the hand-crafted ones. For instance, deep features contain high discriminative power while hand crafted features contain low level information. Therefore, a research question arises: *"Can we exploit the effectiveness of encoding with deep features?"* In general, for hand-crafted features adding more statistical information into the encoding step improves the results, this is one of the reasons for which improved Fisher Vectors (iFV) method [31] outperforms Vector of Locally Aggregated Descriptors (VLAD) [17]. However, for deep features this fact is not anymore checked. As a matter of fact, in our preliminary experiments and also in recent works such as [52], we can notice a completely different behavior of deep features compared to hand-crafted features, where a simpler encoding approach such as VLAD outperforms a method which relies on higher order information such as iFV. Thus, another question arises: *"Is it beneficial for the encoding step to rely on higher order information in case of deep features?"* Another important aspect is that with the current availability of a large number of already trained networks, many works are using these networks as a black box feature extraction tool, due to several reasons, such as the availability of not enough training data, of not enough resources, etc. Then, some techniques for hand crafted features are used to encode the extracted deep features. However, we argue that existing techniques (which are built around deep features) are not optimal also for deep features encoding. Therefore, investigating the encoding approaches for deep features is still an open area of research.

The main contributions of this work are the following: (1) we propose two encoding approaches specifically built for deep features. The first solution is extremely efficient for the computational cost, however less performant in terms of accuracy for more challenging datasets, while the other provided solution gives the best accuracy including for challenging datasets but being more demanding for the computational cost than the first solution. Our solutions take advantage of the convolutional networks, capturing the highest

feature response of the convolutional maps which actually results from the highest neuron activation from the network. Specifically, after we extract some local deep features from a video, we have two approaches to group the features: one is based on the channel information from the feature maps of the network and the other on the similarity information. In the end we perform a max-pooling over the resulted group of features. Our proposed solutions respect three essential characteristics for a highly competitive video representation: (i) simple and generic, our solutions are straightforward to implement and are highly discriminative, working even with a simple model such as a linear classifier; (ii) efficient, our proposed approaches are able to run faster than real-time; (iii) effective, our proposed methods outperform the other encoding approaches in terms of accuracy (such as, iFV and VLAD) by a large margin. (2) we propose a very competitive pipeline for action recognition to work with deep features. Our pipeline can be easily adopted to work with networks that are not specifically trained or fine-tuned for a particular dataset or even are trained on a different task. We cover all those cases in our work and we present a robust framework with state-of-the-art results on three challenging datasets for action recognition.

The rest of the paper is organized as following: Section 2 summarizes the related works. Section 3 introduces our encoding solutions. Section 4 presents the local deep feature extraction pipeline. The experimental evaluation is presented in Section 5. The conclusions are drawn in Section 6.

2 RELATED WORK

Due to the fact that feature encoding step is a key factor for the system performance, there are many works which are focused to build a powerful representation that serves as input for a classifier. Super vector-based encoding methods are among the most powerful representation generators. Improved Fisher Vectors (iFV) [31] is one of the state-of-the-art super vector-based encoding methods which performs a soft assignment of the features and incorporates first- and second-order information. Vector of Locally Aggregated Descriptors (VLAD) [17] is a simplification of iFV capturing only first-order information and performing a hard assignment of the features. Super Vector Coding (SVC) [57] method keeps the zero-order and first-order statistics, thus SVC can be seen as a combination between Vector Quantization (VQ) [37] and VLAD.

There are many works with the main goal to improve the aforementioned widely used approaches. The work in [28] proposes to improve VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. The work in [24] uses Random Forests in a pruned version for the trees to build the vocabulary and then additionally concatenate second-order information similar as iFV. The works in [22, 23] consider a 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 dividing the video in more than 4 segments. The work in [2] proposes to use intra-normalization to improve VLAD performance. The RootSIFT normalization is proposed in [1] to improve the performance of the framework for object retrieval

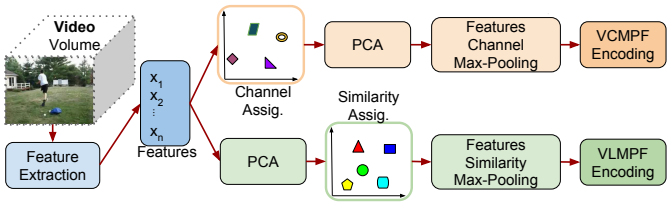

Figure 1: The pipelines for deep feature encoding.

by computing square root of the values to reduce the influence of large bin values. In [10] the VLAD approach is empowered with spatio-temporal information by considering the feature position within the video, while in [11] the performance of VLAD is boosted by using a double assignment approach. The work in [30] uses a multi-layer nested iFV encoding to boost the performance. Different from aforementioned methods which are initially built to encode hand-crafted features, our work proposes solutions specifically designed for local deep features encoding.

Recently, encouraged by deep learning breakthroughs, many works [4, 19, 27, 35, 36, 41, 42, 51, 54] encapsulate all three main steps: feature extraction, encoding and classification, in an end-to-end framework. The work in [35] uses two streams, to capture both appearance and motion information. The works in [14, 15] are based on rank pooling for encoding; the authors in [4] extend this idea to dynamic images to create a video representation. In [42] a 3D convolutional neural network is proposed which is able to learn both appearance and motion information from a video. Over the previous approaches, our proposed methods have the advantage of being able to use any available trained network without the need to train, re-train or fine tune it, obtaining impressive performance, even improving the original network results. Furthermore, our methods can easily combine different networks, with different source of information, to create an effective video representation.

3 DEEP FEATURE ENCODING

In this section we present our proposed encoding approaches for local deep features: VCMPF (Vector of Channel Max Pooled Features) and VLMPF (Vector of Locally Max Pooled Features). The overview of the main steps for each encoding method is illustrated in Figure 1.

3.1 VCMPF

The upper part of Figure 1 represents the path to obtain the VCMPF representation for a given video. We exploit the nature of deep features and obtain an automatic codebook represented by the channels of the network layer from where we extract the features (see Section 4 for details about feature extraction), thus we have 512 channels. Therefore, each local deep feature extracted from a video is a 512 dimensional vector, and each dimension represents the response for its corresponding channel. Formally, we can represent the codebook for the channels as $Ch=\{ch_1, ch_2, ..., ch_{512}\}$ and the extracted features for a given video are represented by $X=\{x_1, x_2, ..., x_n\} \in R^{n \times d}$, where d is the feature dimensionality and n is the total number of the local features of the video. In the assignment, for each local feature x_j (j=1, ..., n) we obtain the index

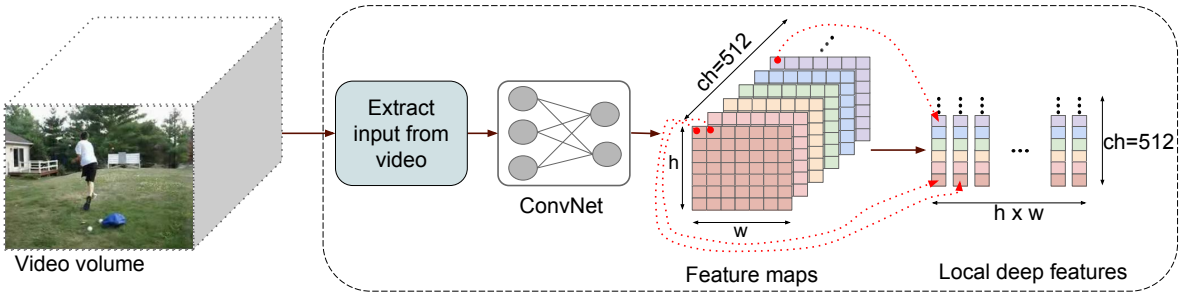

Figure 2: The pipeline for local deep feature extraction.

(within its vector) where the highest response (the maximum absolute value of the feature vector) is located and then we assign the feature to the corresponding channel with the obtained index. In this case the assignment step is extremely fast as it is not necessary to compute any distances, which in general is demanding for the computational cost.

The next step is to apply PCA over all features. This is an optional step, however, it can be useful in practice due to the high dimensionality of the local deep features. We apply PCA after the assignment step with the goal to conserve all 512 dimensions of the initial feature vector, and thus, keep all the initial channels as a codebook. The assignment step provides a division of the features in 512 groups. For each group i (i =1, ..., 512) we compute a vector representation v^{ch_i}=$[v_1^{ch_i}, v_2^{ch_i}, ..., v_{d^*}^{ch_i}]$, where d^* is the feature dimensionality after PCA. Each $v_s^{ch_i}$ (s iterates over the dimensions of the feature, s=1, ..., d^*) is formally computed as:

$$v_s^{ch_i} = \text{sign}(x_{j,s}) \max_{x_j \in ch_i} |x_{j,s}| \quad (1)$$

where sign() returns the sign of a number and |.| represents the absolute value. The equation above performs a max pooling over each group of features based on the channel assignment. With this approach we capture the highest feature responses over each channel. The concatenation of all $[v^{ch_1}, v^{ch_2}, ..., v^{ch_{512}}]$ gives the VCMPF encoding, which is a row vector with the size 512×d^*.

3.2 VLMPF

For VLMPF, we initially learn a codebook with k-means, using a large set of randomly selected features extracted from a subset of videos. The resulted codebook C=$\{c_1, c_2, ..., c_k\}$ contains k visual words. In the bottom part of Figure 1 are presented the main steps to obtain the VLMPF representation. After we obtain the features for a video, we can optionally apply PCA to reduce their dimensionality. Then we perform a hard assignment of the features, where each local feature is assigned (based on the Euclidian distance) to its nearest visual word from the learned codebook C. Similar as above, for each resulted group of features we compute a vector representation v^{c_i}=$[v_1^{c_i}, v_2^{c_i}, ..., v_{d^*}^{c_i}]$, where $v_s^{c_i}$ is formally computed as:

$$v_s^{c_i} = \text{sign}(x_{j,s}) \max_{x_j : NN(x_j)=c_i} |x_{j,s}| \quad (2)$$

where $NN(x_j)$ denotes the nearest neighborhood visual word of the codebook C for the feature x_j.

This equation performs a max pooling over each group of features resulted from the similarity-based information. Basically, VLMPF retains the maximum response for each dimension and for each visual word separately. All concatenated vectors $[v^{c_1}, v^{c_2}, ..., v^{c_k}]$ provide the VLMPF encoding, which is represented by a row vector of k×d^* dimensions.

These two proposed approaches for deep feature encoding, represent a competitive and practical solution to encode the features extracted with convolutional neural network from the entire video.

4 DEEP FEATURE EXTRACTION

This section presents the framework to extract the local deep features for a video. The approaches based on convolutional networks (ConvNets) [4, 19, 27, 35, 36, 41, 42, 51, 54] have recently obtained very competitive results over traditional hand-crafted features.

Our framework for local feature extraction uses three streams: a spatial stream for capturing the appearance, a temporal stream for capturing the motion and a spatio-temporal stream for capturing at the same time both appearance and motion information. All these three networks are individually applied on a video to extract the local deep features. Figure 2 illustrates the framework for local deep feature extraction, where we initially obtain from a given video the requested input necessary for each network and then we extract the feature maps with spatial information.

In our framework the appearance information is obtained by using the VGG ConvNet in [36], which is a network with 19 layers. VGG19 is also characterized by a smaller size of the convolutional filters of 3×3 and the stride is only 1 pixel. These characteristics enable the network to explore finer-grained details from the feature maps. This network is trained on the ImageNet dataset [8], with state-of-the-art results for image classification. The VGG19 ConvNet requests as input an image with 224×224 pixels and three channels for the color information. We initially extract the individual frames from a video and then we accordingly resize them to the required input size of the network. For each individual frame we take the output of the last convolutional layer with spatial information, which for this network is represented by pool5. The motivation of choosing this layer is related to the fact that deeper layers provide high discriminative information. By taking a layer with spatial information we can extract local deep features for each frame of the video, instead of a global deep feature, increasing the robustness of the approach. The pool5 layer provides as output a

feature map with a spatial size of 7×7 and 512 channels. From the obtained feature map we extract the local deep features by taking individually each spatial location and concatenate the values along all 512 channels, obtaining local deep features with 512 dimensions. Therefore, from each frame we extract 7×7=49 local deep features and each feature is a 512 dimensional vector. For each video we obtain in total $\#frames$×49 local deep features. In the experimental section we refer to the local deep features extracted from the Spatial Convolutional Network as SCN.

The motion information extraction is based on the re-trained network in [51]. This deep network, also VGG, is initially proposed in [36] and contains 16 layers. The authors in [51] re-trained the VGG ConvNet for action recognition 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 to prevent overfitting. The VGG ConvNet is re-trained for action recognition task using the UCF101 dataset [39]. For the temporal ConvNet the input consists in 10-stacked optical flow fields, each field provides an image for the vertical motion and another image for the horizontal motion information. Thus, the input for this network is represented by 20-stacked optical flow images. From a given video we extract the optical flow fields using the OpenCV implementation of the TVL1 algorithm [55]. Similar as in the previous ConvNet, for the temporal ConvNet we also take the output of the last convolutional layer with structure information represented by pool5 layer. The spatial size of this pool5 layer is also 7×7 and 512 channels. As illustrated in Figure 2, 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 extracted for a video when using temporal ConvNet is ($\#frames$−9)×49. We further refer to the local deep features extracted with Temporal Convolutional Network as TCN.

For the spatio-temporal stream we use the 3D ConvNet [42]. This network is trained on Sports-1M dataset [19] and contains 16 layers: 8 convolutional, 5 max-pooling, 2 fully connected layers, and the softmax output layer. The network is designed to capture both appearance and motion information by using 3D convolutional kernels. The input of the network is a 16 frame-long clip extracted from the video. Similar to the previous two networks used in our pipeline, we use a sampling step size of one frame to iterate over the frames of the video for creating the input clips. The last layer with spatial information has spatial size of the feature maps of 4×4. Due to this relatively small spatial size of the feature maps for this layer and also for having an equal size of the feature maps for all three networks, we extract for this spatio-temporal network the conv5b layer. The conv5b layer contains two feature maps, each of them 7×7×512. For obtaining a single feature map out of this two, we take maximum value for each position of the both feature maps from conv5b. After this step, we obtain the same size as previous two networks, therefore, we can apply the same framework to extract the local deep features. The total number of local deep features obtained with this network for a video is ($\#frames$−15)×7×7. We further refer to the features extracted with this Convolutional 3D network as C3D.

5 EXPERIMENTS

This section presents the experimental part where we evaluate the proposed approach for action recognition.

5.1 Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: HMDB51 [20], UCF50 [32], and UCF101 [39].

The HMDB51 dataset [20] 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 [20]. We report average accuracy over the three splits as performance measure.

The UCF50 dataset [32] contains 6,618 realistic videos taken from YouTube. There are 50 human action categories mutually exclusive and the videos are split into 25 predefined groups. We follow the recommended standard procedure and perform leave-one-group-out cross validation and report average classification accuracy over all 25 folds.

The UCF101 dataset [39] is a widely adopted benchmark for action recognition, consisting in 13,320 realistic videos and 101 action classes. We follow for evaluation the recommended default three training/testing splits and report the average recognition accuracy over these three splits.

5.2 Experimental setup

For the motion stream of the local deep feature extraction pipeline, the work in [51] provides three trained models for each split of the UCF101 dataset. We accordingly use the models for each split of the UCF101 for feature extraction. For the other two datasets, HMDB51 and UCF50, we use only the model trained on the split1 of UCF101 to extract the local deep features. We compare our proposed encodings (VCMPF and VLMPF) with two state-of-the-art super vector-based encoding approaches represented by Vector of Locally Aggregated Descriptors (VLAD) [17] and Fisher Vectors (iFV) [31]. We set the size of the codebook to 256 visual words for our VLMPF, VLAD and iFV. This is a standard used size of the codebook in many works, and by keeping this setting we can easily and fairly compare different approaches. The codebook is learned from a randomly 500k sampled features from a subset of videos. Before classification, for VCMPF and VLMPF we individually normalize each representation resulted from each feature type by applying L2 normalization. Many works, such as [47, 44], show that iFV and VLAD perform better if after feature encoding the Power Normalization (PN) is applied followed by L2-normalization ($||sign(x)|x|^{\alpha}||$). Thus, we also adopted this normalization strategy for iFV and VLAD, setting α to the standard widely used value of 0.5. The reason for which iFV and VLAD work better when using PN is due to the fact that their resulted final representation contains large peaks within the vector and PN helps to reduce them and make the vector smoother. Instead, our proposed encodings do not provide a final vector containing large peaks, therefore, it is not necessary to apply also PN. For the classification part, in all the experiments we use a linear one-vs-all SVM with the parameter C=100.

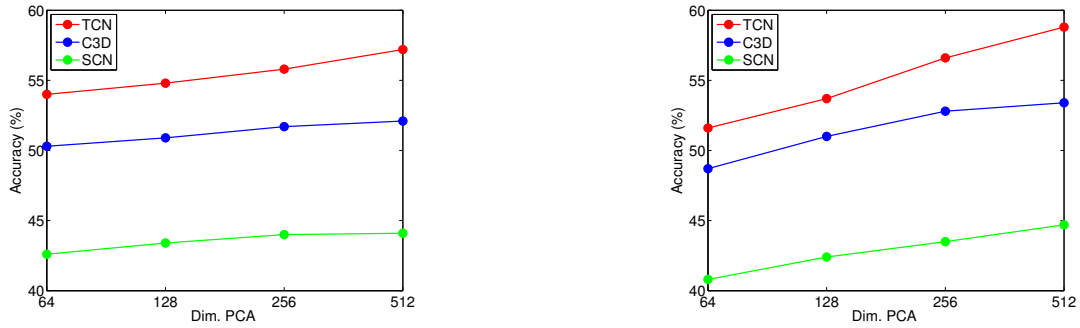

Figure 3: Evaluation of the dimensionality reduction with PCA for VCMPF (left) and VLMPF (right).

Table 1: Accuracy comparison on all three datasets. Best results are in bold.

	HMDB51 (%)						UCF50 (%)						UCF101 (%)					
	SCN		TCN		C3D		SCN		TCN		C3D		SCN		TCN		C3D	
	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512	256	512
iFV	36.6	41.8	51.0	56.6	46.1	49.0	75.7	81.0	95.2	96.1	84.7	88.8	67.8	74.1	84.1	85.4	77.7	79.8
VLAD	37.2	40.3	51.1	53.9	46.8	49.1	78.4	80.2	95.5	95.4	86.4	89.0	69.9	73.4	83.7	85.2	78.6	81.4
VCMPF	**44.0**	44.1	55.8	57.2	51.7	52.1	**85.1**	85.1	**97.0**	**97.1**	93.4	**93.7**	78.3	**78.8**	**86.5**	**86.9**	83.5	84.1
VLMPF	43.5	**44.7**	**56.6**	**58.8**	**52.8**	**53.4**	84.7	**85.3**	96.8	96.8	**93.7**	**93.7**	**78.5**	78.6	85.9	**86.9**	**84.5**	**84.7**

5.3 Parameter tuning

We examine the influence on the accuracy of dimensionality reduction using PCA. The left graph from Figure 3 presents the evolution of the accuracy while the number of feature dimensions is changed for the VCMPF encoding approach. By reducing the feature dimensionality by a factor of two, a reasonable trade-off with the accuracy is achieved. In the right graph of Figure 3 we present the situation of decreasing the feature dimensionality in the case of VLMPF encoding approach where the best accuracy is obtained with all 512 dimensions, while the 256 dimensionality for the features represents the trade-off. Following this evaluation we use in the next experiments the feature with full dimensionality (512) and also reducing their dimensionality to 256 with PCA.

5.4 Comparison to other encodings

In this part we compare our proposed encoding approaches with other highly competitive encoding methods, such as improved Fisher Vectors (iFV) and Vector of Locally Aggregated Descriptors (VLAD). The comparison is conducted in both effectiveness and efficiency.

5.4.1 Effectiveness comparison. In Table 1 we present the accuracy comparison of our encoding with iFV and VLAD over all three datasets. For all three considered local deep features (SCN, TCN and C3D) we report the results for full feature dimensionality (512) and also after applying PCA to reduce their dimensionality by a factor of two. We can clearly see that both proposed encoding approaches, VCMPF and VLMPF, outperform by a large margin the state-of-the-art super vector-based encoding approaches iFV and VLAD over all three datasets and feature types. For instance, for C3D features with full dimensionality, VLMPF outperforms iFV by 4.4 percentage points and VLAD by 4.3 percentage points on the challenging HMDB51 dataset.

Very interestingly, in the case of deep features, iFV does not perform always better than VLAD, which is actually the case for hand-crafted features presented in many works, such as [44, 29]. It is well-known by the research community that for hand-crafted features iFV gives a better accuracy than VLAD, one of the reasons is due to the fact that iFV considers high order statistics to build the final representation. However, in the case of deep features, considering high statistical information for the encoding method does not guarantee an improvement for accuracy. However, our encoding methods, specially designed for encoding local deep features, give impressing results for all three datasets.

5.4.2 Efficiency comparison. For the efficiency comparison we randomly sampled 500 videos from HMDB51 dataset and we report the number of frames per second and the number of seconds per video that an encoding method is able to process. The timing measurements are performed on a single core Intel(R) Xeon(R) CPU E5-2690 2.60GHz.

Table 2 presents the efficiency comparison over all three deep features using both full dimensionality of the features (512) and reduced with PCA to 256. The table shows that our approaches are the most efficient. The most demanding for the computational cost is iFV and this is due to the computation of first- and second-order information and also to the use of a soft assignment approach. VLMPF is comparable with VLAD in computational cost, however VLAD is slightly slower due to the computation of the residuals. The fastest approach is our VCMPF, which outperforms including the other proposed encoding, and this is due to the assignment step which avoids the computation of the expensive distances. The last

Table 2: Computational efficiency comparison. We report the number of frames per second (fr/sec) and seconds per video (sec/vid). Last two columns show the dimensionality generated by each encoding method for 256 and 512 feature dimensionality. Best results are in bold.

	SCN 256		SCN 512		TCN 256		TCN 512		C3D 256		C3D 512		256	512
	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	fr/sec	sec/vid	dim	dim
iFV	253.2	0.357	168.7	0.536	301.4	0.300	197.6	0.457	308.7	0.293	202.3	0.447	131,072	262,144
VLAD	1967.5	0.046	1143.8	0.079	2213.8	0.041	1299.5	0.070	2372.5	0.038	1375.0	0.066	**65,536**	**131,072**
VCMPF	**4435.9**	**0.020**	**3485.1**	**0.026**	**4731.4**	**0.019**	**3612.5**	**0.025**	**5019.5**	**0.018**	**3682.2**	**0.025**	131,072	262,144
VLMPF	2049.4	0.044	1192.6	0.076	2329.2	0.039	1370.9	0.066	2455.0	0.037	1426.0	0.063	**65,536**	**131,072**

Table 3: Fusion strategies for VCMPF and VLMPF. DF (Deep Features) represent all three local deep features (SCN, TCN, C3D), except for UCF50 dataset where we excluded TCN features from the fusion; HMG (Histograms of Motion Gradients) [12]**; and iDT (improved Dense Trajectories)** [47] **is represented with HOG, HOF, MBHx and MBHy. The best performance results are in bold for each fusion type over each feature representation combination. The best result over each dataset is also underlined.**

(a) VCMPF

	HMDB51 (%)			UCF50 (%)			UCF101 (%)		
	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT
Early	64.2	66.6	69.9	94.4	95.3	96.6	**93.1**	**93.8**	**94.1**
sLate	61.5	63.2	66.5	93.0	94.1	95.6	91.3	91.9	92.3
wLate	63.2	63.9	68.2	93.9	94.9	96.5	91.7	92.2	92.9
sDouble	63.2	64.6	68.0	93.5	94.5	95.8	91.9	92.5	92.7
wDouble	**65.2**	**67.1**	**70.5**	**94.6**	**95.4**	**96.9**	**93.1**	**93.8**	**94.1**

(b) VLMPF

	HMDB51 (%)			UCF50 (%)			UCF101 (%)		
	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT
Early	65.3	66.7	70.2	94.5	95.3	96.7	**93.1**	**93.6**	**94.1**
sLate	63.6	64.7	68.0	93.5	94.4	95.6	91.5	92.0	92.3
wLate	65.1	65.7	69.3	93.9	94.9	96.5	91.7	92.2	93.0
sDouble	65.0	65.9	69.0	94.0	95.0	96.0	92.3	92.6	92.8
wDouble	**66.9**	**68.0**	**71.6**	**94.8**	**95.4**	**96.9**	**93.1**	**93.6**	**94.1**

two columns presents the dimensionality of a video representation generated by each encoding approach. VLAD and our proposed encoding approach, VLMPF, generate the lower dimensionality for a video representation.

5.5 Fusion strategies

In the previous experiments we show that our proposed approaches are superior to the other encoding methods. Also from the previous experiments we can notice that our VCMPF is not considerably affected by the dimensionality reduction to 256, while the accuracy for our VLMPF drops more. Considering this observation, for the next experiments we reduce the feature dimensionality to 256 for VCMPF, while for VLMPF we keep full dimensionality of 512. This setting provides also the same size of the video representation generated by VCMPF and VLMPF.

The UCF101 dataset is an extension of UCF50 and the TCN features are extracted using a network trained on UCF101. To avoid the risk of overfitting on UCF50 dataset we exclude the TCN features when performing any fusion. Therefore, for a fair comparison, in the next results, which are used as a comparison with the state-of-the-art, we use as deep features only SCN and C3D for UCF50 dataset.

As feature fusion, we use three combinations. (1) DF stands for deep features represented by fusing SCN, TCN and C3D (note that, as we said before, for UCF50 we excluded TCN features). (2) DF+HMG, we add to deep features a hand-crafted descriptor represented by Histogram of Motion Gradients [12], which is focused on capturing the motion information within the video in an efficient approach. (3) DF+HMG+iDT, in addition, we fuse the improved Dense Trajectories [47] descriptors represented by HOG, HOF, MBHx and MBHy. iDT represents a state-of-the-art approach for extracting hand-crafted descriptors. The hand-crafted descriptors are extracted using the code provided by the authors [12, 47], with the suggested default settings. All hand-crafted descriptors are individually encoded as suggested, using iFV, and then before classification, we individually apply on each resulted representation Power Normalization (with α=0.1) followed by L2 normalization as recommended in [12].

The fusion between features is performed using several strategies. "Early" represents an early fusion by concatenating each

Table 4: Comparison to the state-of-the-art.

HMDB51 (%)		UCF50 (%)		UCF101(%)	
Jain et al. [16] (2013)	52.1	Solmaz et al. [38] (2013)	73.7	Wang et al. [48] (2013)	85.9
Zhu et al. [58] (2013)	54.0	Reddy et al. [32] (2013)	76.9	Karpathy et al. [19] (2014)	65.4
Oneata et al. [26] (2013)	54.8	Shi et al. [34] (2013)	83.3	Simonyan et al. [35] (2014)	88.0
Wang et al. [47] (2013)	57.2	Wang et al. [45] (2013)	85.6	Wang et al. [46] (2015)	86.0
Kantorov et al. [18] (2014)	46.7	Wang et al. [47] (2013)	91.2	Sun et al. [41] (2015)	88.1
Simonyan et al. [35] (2014)	59.4	Ballas et al. [3] (2013)	92.8	Ng et al. [54] (2015)	88.6
Peng et al. [30] (2014)	66.8	Everts et al. [13] (2014)	72.9	Tran et al. [42] (2015)	90.4
Sun et al. [41] (2015)	59.1	Uijlings et al. [43] (2014)	80.9	Wang at al. [51] (2015)	91.4
Wang et al. [46] (2015)	60.1	Kantorov et al. [18] (2014)	82.2	Wang et al. [50] (2015)	91.5
Wang et al. [50] (2015)	65.9	Ciptadi et al. [5] (2014)	90.5	Zhang et al. [56] (2016)	86.4
Park et al. [27] (2016)	56.2	Narayan et al. [25] (2014)	92.5	Peng et al. [29] (2016)	87.9
Seo et al. [33] (2016)	58.9	Uijlings et al. [44] (2015)	81.8	Park et al [27] (2016)	89.1
Peng et al. [29] (2016)	61.1	Wang et al. [46] (2015)	91.7	Bilen et al. [4] (2016)	89.1
Yang et al. [53] (2016)	61.8	Peng et al. [29] (2016)	92.3	Diba et al. [9] (2016)	90.2
Bilen et al. [4] (2016)	65.2	Duta et al. [12] (2016)	93.0	Fernando et al. [14] (2016)	91.4
Fernando et al. [14] (2016)	66.9	Seo et al. [33] (2016)	93.7	Yang et al. [53] (2016)	91.6
Our VCMPF(DF)	65.2	Our VCMPF(DF)	**94.6**	Our VCMPF(DF)	**93.1**
Our VCMPF(DF) + iFV(HCF)	**70.5**	Our VCMPF(DF) + iFV(HCF)	**96.9**	Our VCMPF(DF) + iFV(HCF)	**94.1**
Our VLMPF(DF)	**66.9**	Our VLMPF(DF)	**94.8**	Our VLMPF(DF)	**93.1**
Our VLMPF(DF) + iFV(HCF)	**71.6**	Our VLMPF(DF) + iFV(HCF)	**96.9**	Our VLMPF(DF) + iFV(HCF)	**94.1**

resulted individual representation and then performing a final L2 normalization over the resulted final representation. "sLate" represents a late fusion by performing a sum over the classifiers output obtained from each individual feature type. "wLate" represents also a late fusion but using a weighted sum of the classifiers output. The weights combination is tuned taking values in the [0;1] interval with the step 0.1. In addition, we perform a double fusion, where besides the individual classifier output obtained for each feature type we use also the classifier obtained from the early fusion. We have also "sDouble" from sum over all classifiers output and "wDouble" for weighted sum over the classifiers output, with similar strategy as late fusion to establish the weights.

The results for each fusion strategy is presented in Table 3a for VCMPF and in Table 3b for VLMPF. We can notice that fusing hand-crafted descriptors with the deep features boosts the accuracy. In general, early fusion outperforms late fusion and the weighted sum over the classifiers output gives better results than just performing the sum. The double fusion combines the benefits of early and late fusion, achieving the best results overall that serves as comparison to the state-of-the-art.

5.6 Comparison to state-of-the-art

The comparison of our framework for action recognition to the state-of-the-art is presented in Table 4. We compare the state-of-the-art to both proposed solutions for feature encoding: VCMPF and VLMPF. For both solutions we present the results using only deep features (DF) represented by SCN, TCN and C3D (note that, as explained above, in all the cases we excluded TCN features for UCF50 dataset). In addition, we present the results where we combine our representation with the improved Fisher Vector (iFV) applied over hand-crafted features (HCF). The hand-crafted features are represented by Histogram of Motion Gradients (HMG) [12] and improved dense trajectories (iDT) approach (represented by the descriptors: HOG, HOF, MBHx andMBHy) [47]. It is important to mention that our results are obtained without re-training or fine-tuning the ConvNets for the datasets used in this work (except TCN features for UCF101). For instance, the competitive results on HMDB51 datasets are obtained using features extracted from three different ConvNets, none of which had access to any data from HMDB51 during the training. Our results outperform even recent works on action recognition, obtaining state-of-the-art results on all three datasets: HMDB51, UCF50 and UCF101.

6 CONCLUSION

In this work we present two competitive solutions specifically designed for encoding the local deep features. The provided solutions are useful in many practical scenarios for working with deep features to generate a robust final representation. In this work we show that our powerful encodings for deep features outperform by a large margin the existing state-of-the-art encoding approaches, which are built around hand-crafted features, such as VLAD and improved Fisher Vectors, this motivating the necessity of an encoding approach adapted for local deep features. Furthermore, our proposed solutions are highly efficient, being able to run faster than real time. Our encoding approaches provide a solution to encode the features extracted with a ConvNet over the entire video. Our proposed pipeline for action recognition is pushing the state-of-the-art further by 4.7, 3.2 and 2.5 percentage points on the HMDB51, UCF50 and UCF101 datasets respectively.

Acknowledgments. This work was partially supported by EIT Digital. Part of this work was funded under UEFISCDI research grant PN-III-P2-2.1-PED-2016-1065, agreement 30PED/2017, project SPOTTER.

REFERENCES

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] N. Ballas, Y. Yang, Z.-Z. Lan, B. Delezoide, F. Prêteux, and A. Hauptmann. Space-time robust representation for action recognition. In *ICCV*, 2013.

[4] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

[5] A. Ciptadi, M. S. Goodwin, and J. M. Rehg. Movement pattern histogram for action recognition and retrieval. In *ECCV*, 2014.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[7] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*. 2006.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[9] A. Diba, A. M. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. In *ECCV ws*, 2016.

[10] I. C. Duta, B. Ionescu, K. Aizawa, and N. Sebe. Spatio-temporal vlad encoding for human action recognition in videos. In *MMM*, 2017.

[11] I. C. Duta, T. A. Nguyen, K. Aizawa, B. Ionescu, and N. Sebe. Boosting VLAD with double assignment using deep features for action recognition in videos. In *ICPR*, 2016.

[12] I. C. Duta, J. R. R. Uijlings, T. A. Nguyen, K. Aizawa, A. G. Hauptmann, B. Ionescu, and N. Sebe. Histograms of motion gradients for real-time video classification. In *CBMI*, 2016.

[13] I. Everts, J. C. Van Gemert, and T. Gevers. Evaluation of color spatio-temporal interest points for human action recognition. *TIP*, 2014.

[14] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016.

[15] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *TPAMI*, 2016.

[16] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[17] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 34(9):1704–1716, 2012.

[18] V. Kantorov and I. Laptev. Efficient feature extraction, encoding and classification for action recognition. In *CVPR*, 2014.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[20] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

[21] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.

[22] I. Laptev, M. Marsza lek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[23] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[24] I. Mironică, I. C. Duță, B. Ionescu, and N. Sebe. A modified vector of locally aggregated descriptors approach for fast video classification. *Multimedia Tools and Applications*, pages 1–28, 2016.

[25] S. Narayan and K. R. Ramakrishnan. A cause and effect analysis of motion trajectories for modeling actions. In *CVPR*, 2014.

[26] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.

[27] E. Park, X. Han, T. L. Berg, and A. C. Berg. Combining multiple sources of knowledge in deep cnns for action recognition. In *WACV*, 2016.

[28] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *ECCV*. 2014.

[29] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CVIU*, 150:109–125, 2016.

[30] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014.

[31] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010.

[32] K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.

[33] J.-J. Seo, H.-I. Kim, W. De Neve, and Y. M. Ro. Effective and efficient human action recognition using dynamic frame skipping and trajectory rejection. *IVC*, 2016.

[34] F. Shi, E. Petriu, and R. Laganiere. Sampling strategies for real-time action recognition. In *CVPR*, 2013.

[35] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

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

[37] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth International Conference on*, pages 1470–1477, 2003.

[38] B. Solmaz, S. M. Assari, and M. Shah. Classifying web videos using a global video descriptor. *Machine vision and applications*, 2013.

[39] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[40] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li. Hierarchical spatio-temporal context modeling for action recognition. In *CVPR*, 2009.

[41] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[43] J. R. R. Uijlings, I. C. Duta, N. Rostamzadeh, and N. Sebe. Realtime video classification using dense hof/hog. In *ICMR*, 2014.

[44] J. R. R. Uijlings, I. C. Duta, E. Sangineto, and N. Sebe. Video classification with densely extracted hog/hof/mbh features: an evaluation of the accuracy/computational efficiency trade-off. *International Journal of Multimedia Information Retrieval*, 4(1):33–44, 2015.

[45] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 103(1):60–79, 2013.

[46] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015.

[47] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[48] H. Wang and C. Schmid. Lear-inria submission for the thumos workshop. In *ICCV Workshop*, 2013.

[49] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.

[50] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015.

[51] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015.

[52] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015.

[53] X. Yang, P. Molchanov, and J. Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACMMM*, 2016.

[54] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[55] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. 2007.

[56] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *CVPR*, 2016.

[57] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *Computer Vision–ECCV 2010*, pages 141–154. 2010.

[58] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013.