Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos

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

Abstract

We introduce Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), a super vector-based encoding method specifically designed for local deep features encoding. The proposed method addresses an important problem of video understanding: how to build a video representation that incorporates the CNN features over the entire video. Feature assignment is carried out at two levels, by using the similarity and spatio-temporal information. For each assignment we build a specific encoding, focused on the nature of deep features, with the goal to capture the highest feature responses from the highest neuron activation of the network. Our ST-VLMPF clearly provides a more reliable video representation than some of the most widely used and powerful encoding approaches (Improved Fisher Vectors and Vector of Locally Aggregated Descriptors), while maintaining a low computational complexity. We conduct experiments on three action recognition datasets: HMDB51, UCF50 and UCF101. Our pipeline obtains state-of-the-art results.

1. Introduction

Action recognition is still a very challenging and high computationally demanding task in computer vision, receiving a sustained attention from the research community due to its huge pool of potential applications. Its pipeline can be broken down into three main steps: feature extraction, encoding and classification. While for the classification part, the existing techniques are more mature, for feature extraction and encoding there is still a significant room for improvement. There are two main directions for feature extraction: hand-crafted and learned features (deep features). For hand-crafted category the most popular descriptors are represented by Histogram of Oriented Gradients (HOG) [5, 20], Histogram of Optical Flow (HOF) [20] and Motion Boundary Histograms (MBH) [6]. These descriptors are extracted from a video using different approaches to establish the region of extraction, such as at interest points [19], using a dense sampling [42, 47], along motion trajectories [38, 43, 45]. Recently, the features learned with a deep neural network represent a breakthrough in research, obtaining impressive results [3, 17, 25, 33, 34, 39, 40, 49, 52].

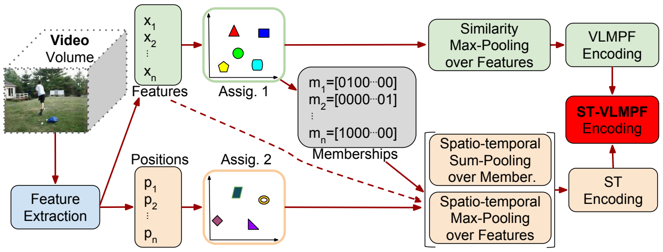

Figure 1: The ST-VLMPF framework for deep features encoding.

The feature encoding is one of the crucial steps for the system performance. Super vector-based encoding methods represent one of the most powerful solutions to build the final representation that servers as an input for a classifier. Improved Fisher Vectors (iFV) [29] and Vector of Locally Aggregated Descriptors (VLAD) [15] proved their superiority over other encoding methods in many works and are presented as state-of-the-art approaches for the encoding step [22, 27, 41, 42, 45]. One of the shortcomings for current standard encodings is the lack of considering the spatio-temporal information, which is crucially important, especially when dealing with videos. These encoding approaches are built around hand-crafted features. However, a new trend is using deep features, as they obtain promising results over the traditional hand-crafted features. Many recent works apply these encoding methods also on deep features. Nevertheless, there is not yet a mature pipeline established for using these new features as their nature and behavior are implicitly different from the hand-designed category.

Deep features are learned throughout a deep neural network, providing high discriminative power on the upper layers of the network, with high level information such as objects, while hand-crafted features are manually designed and usually contain low-level information such as edges. Deep features are characterized also by their high sparsity. For instance, in [34, 49] the feature maps extracted from the upper layers of the networks (which are often used in the works as features), can contain a sparsity level of more than 90%, while for hand-crafted features, as in [42, 47], the level of sparsity is negligible. Most of the current encoding methods, such as iFV and VLAD, are built to capture high statistical information to improve the performance. In general, for hand-crafted features, iFV works better than VLAD [27, 42] due to the fact that iFV captures first- and second-order statistics, while VLAD is based only on first order. While for hand-crafted features using more statistical information improves significantly the performance, considering the difference between them, for deep features using high-order statistics may not guarantee the performance improvement. As a matter of fact, in many recent works, as in [50], VLAD encoding is underlined as outperforming iFV, when using deep features. This aspect is also verified in our experiments, where iFV does not guarantee a better performance than VLAD. This shows that a simpler encoding method can perform better than an approach which rely on high order information. This is a completely opposite behavior in comparison with hand-crafted features. Considering all of these, we argue that a new encoding method, specifically designed for deep features, can provide better results.

With the current high availability of off-the-shelf pre-trained neural networks, many researchers use them only as feature extraction tools, as re-training or fine tuning is more demanding in many aspects. Thus, it is necessary for a performant deep features encoding approach. One of the major shortcomings of the current ConvNets-based approaches is represented by the fact that the networks take into account one or several staked frames (for instance, 10-staked optical flow fields [33, 49]). Each sampled input for the network is assigned to the overall video label from which it belongs to. The issue is that if we consider such a short number of frames as input to the network, then it may not correctly reflect the overall video label, resulting in a false label input. The current ConvNets approach individually obtains the prediction scores from all sampled inputs from a video. Then, the final prediction for a video is computed by aggregating all these prediction scores resulted from each sampled input. However, this simple aggregation cannot completely solve the aforementioned issue. This work tackles this important problem by learning a new general representation which reflects the overall video label. This approach gives to the classifier access to the deep features extracted over the entire video.

In this paper, we propose the following main contributions: (i) *provide a new encoding approach* specifically designed for working with deep features. We exploit the nature of deep features, with the goal of capturing the highest feature responses from the highest neuron activation of the network. (ii) *efficiently incorporate the spatio-temporal information within the encoding method* by taking into account the features position and specifically encode this aspect. Spatio-temporal information is crucially important when dealing with video classification. Our final proposed encoding method (illustrated in Figure 1), Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), performs two different assignments of the features. One is based on their similarity information, the other on the spatio-temporal information. For each resulted assignment we perform a specific encoding, by performing two max-poolings and one sum-pooling of the information. (iii) *provide an action recognition scheme* to work with deep features, which can be adopted to obtain impressive results with any already trained network, without the need for re-training or fine tuning on a particular dataset. Furthermore, our framework can easily combine different information extracted from different networks. In fact, our pipeline for action recognition provides a reliable representation outperforming the previous state-of-the-art approaches, while maintaining a low complexity. We make the code for our proposed ST-VLMPF encoding publicly available (https://iduta.github.io/software.html).

The rest of the paper is organized as following: Section 2 summarizes the related works. Section 3 introduces our encoding method. 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

There are many works focusing on improving the feature encoding step, as the resulted final representation, which serves as input for a classifier, is a key component for the system performance. Super vector-based encoding methods are among the most powerful representation generators. Improved Fisher Vectors (iFV) [29] 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) [15] is a simplification of iFV capturing only first-order information and performing a hard assignment of the features. Super Vector Coding (SVC) [55] method keeps the zero-order and first-order statistics, thus SVC can be seen as a combination between Vector Quantization (VQ) [35] and VLAD.

Many recent works try to improve the aforementioned methods. The work in [26] proposes to improve VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. The work in [22] proposes to use 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 [20, 21] 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 [1] proposes to use intra-normalization to improve VLAD performance. In [9] is proposed a double assignment for VLAD to boost the accuracy. The work in [28] 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 a method specifically designed for local deep features encoding.

Recently, encouraged by deep learning breakthroughs, many works [3, 17, 25, 33, 34, 39, 40, 49, 52] encapsulate all three main steps: feature extraction, encoding and classification, in an end-to-end framework. The work in [33] uses two streams, to capture both appearance and motion information. The works in [12, 13] are based on rank pooling for encoding; the authors in [3] extend this idea to dynamic images to create a video representation. Over the aforementioned approaches, our proposed method has 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 method can easily combine different networks, with different source of information, to create a competitive video representation.

3. Proposed ST-VLMPF encoding method

In this section we introduce our proposed encoding approach for deep features, Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF). We initially learn a codebook, C, using k-means, from a large subset of randomly selected features extracted from a subset of videos from the dataset. The outcome represents k_1 visual words, $C{=}\{c_1, c_2, ..., c_{k_1}\}$, which are basically the means of each feature cluster learned with k-means. When we extract the features we also retain their location within the video. For each feature we associate a position p:

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

where h, w and $\#fr$ represent the height, width and the number of frames of the video. 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.

In parallel with the first codebook C, we also learn with k-means a second codebook, $PC{=}\{pc_1, pc_2, ..., pc_{k_2}\}$, from the corresponding selected feature locations. The size of PC is k_2 and the outcome represents the positions 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 k_2 spatio-temporal video divisions.

After building the codebooks, we can start creating the final video representation, which serves as input for a classifier. Figure 1 sketches the pipeline that a video traverses to obtain its final representation. The framework starts by extracting the local features from the video (see Section 4). The video is represented by its extracted local features $X{=}\{\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{n}\} \in R^{\mathrm{n}\times\mathrm{d}}$, where d is the feature dimensionality and n is the total number of the local features of the video. Together with the local features, we retain, as explained above, their positions $P{=}\{p_1, p_2, ..., p_n\} \in R^{\mathrm{n}\times 3}$.

Our proposed encoding method performs two hard assignments using the obtained codebooks, the first is based on the features similarity and the second is based on their positions. For the first assignment each local video feature $\mathrm{x_j}$ (j=1, ..., n) is assigned to its nearest visual word from the codebook C. Then, over the groups of features assigned to a cluster c_i (i=1, ..., k_1) we compute a vector representation $v^{c_i}{=}[v_1^{c_i}, v_2^{c_i}, ..., v_d^{c_i}]$, where each $v_s^{c_i}$ (s iterates over each dimension of the vector, s=1, ..., d) is formally computed as following:

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

where $\mathrm{NN}(\mathrm{x_j})$ denotes the nearest neighborhood centroid of the codebook C for the feature $\mathrm{x_j}$, basically it guarantees that we perform separately the pooling over each group of features that are assigned to a visual word; the $sign$ function returns the sign of a number and $|.|$ represents the absolute value.

Basically, Equation 2 obtains the maximum absolute value while keeping the initial sign for the returned final result. In Figure 1 we name this similarity max-pooling over features, as the features are grouped based on their similarity and then perform max-pooling over each resulted group. The concatenation of all vectors $[v^{c_1}, v^{c_2}, ..., v^{c_{k_1}}]$, represents the VLMPF (Vector of Locally Max Pooled Features) encoding, with final vector size $(k_1{\times}d)$.

After the first assignment, we also retain the centroid membership of each feature, with the objective of preserving the associated similarity-based cluster information. For each feature, we represent the membership information by a vector m with the size equal to the number of visual words k_1, 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 visual word of the codebook C.

We perform a second assignment based on the features positions. The bottom part of Figure 1 shows this path. Each feature position p_j from P is assigned to its nearest centroid from codebook PC. After we group the features based on their location we compute another vector representation, by performing two pooling strategies: one max-pooling over the spatio-temporal clustered features and another sum-pooling over the corresponding spatio-temporal clustered features membership. We concatenate the results of these two poolings from each cluster pc_r (r=1, ..., k_2). Therefore, for each spatio-temporal group of features we compute a vector representation $v^{pc_r}{=}[v_1^{pc_r}, v_2^{pc_r}, ..., v_d^{pc_r}]$, where each $v_s^{pc_r}$ is formally computed as following:

$$v_s^{pc_r} = cat\Big[\operatorname{sign}(\mathrm{x_{j,s}}) \max_{p_j:NN(p_j)=pc_r} |\mathrm{x_{j,s}}|, \Big(\sum_{p_j:NN(p_j)=pc_r} \mathrm{m_{j,i}} \Big)^{\alpha} \Big] \tag{3}$$

where cat denotes the concatenation and $NN(p_j)$ denotes the nearest neighborhood visual word of the codebook PC for the feature position p_j. Due to the fact that the sum-pooling over the membership information can create peaks within the vector we normalize the result of sum-pooling similar to power normalization, with standard α=0.5. Basically, in this case we perform square root over the result of the sum-pooling to reduce the peaks within the final vector.

Differently from Equation 2, in Equation 3 we group the features based on the spatio-temporal information and then we compute the maximum absolute value while keeping the original sign over the features. We also concatenate in Equation 3 the membership information regarding the feature similarity obtained from the first assignment with the goal of encapsulating together with spatio-temporal information also the similarity membership of the spatio-temporal grouped features. We concatenate all these vectors $[v^{pc_1}, v^{pc_2}, ..., v^{pc_{k_2}}]$ to create the ST (Spatio-Temporal) encoding, which results in a $(k_2{\times}d + k_2{\times}k_1)$ vector size.

Finally, we concatenate the ST and VLMPF encodings to create the final ST-VLMPF representation, which serves as input the classifier. Therefore, the final size of the vector for ST-VLMPF representation is $(k_1{\times}\ d) + (k_2{\times}\ d + k_2{\times}k_1)$. The goal of ST-VLMPF is to provide a reliable representation which incorporates the deep features over the entire video, providing to the classifier a more complete information for taking the right decision.

4. Local deep features extraction

This section presents the pipeline for local deep features extraction. The approaches based on convolutional networks (ConvNets) [3, 17, 25, 33, 34, 39, 40, 49, 52] have recently obtained very competitive results over traditional hand-crafted features. The videos contain two main sources of information: appearance and motion. In our pipeline for feature extraction we individually use 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. The pipeline for local deep feature extraction is illustrated in Figure 2, where for a given video we extract, independently for each of three networks, the features maps with spatial information.

For capturing the appearance information in our spatial stream we use the VGG ConvNet in [34], which is a network with 19 layers. The local deep feature extraction pipeline for this network is depicted in the upper part of Figure 2. The input of VGG19 ConvNet is an image with 224×224 resolution and three channels for the color information. After we extract the individual frames from a video, 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, pool5. Our choice for the convolutional layer is motivated by the fact that the 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, containing also the details about spatial membership of the features. The output of pool5 is a feature map with a spatial size of 7×7 and 512 channels. For extracting local deep features from a feature map we individually take each spatial location and concatenate the values along all 512 channels, obtaining local deep features with 512 dimensions. Thus, from each frame we obtain 7×7=49 local deep features and each feature is a 512 dimensional vector. Therefore, for each video we obtain in total $\#frames{\times}49$ local deep features. SCN refers to the features extracted with this Spatial Convolutional Network.

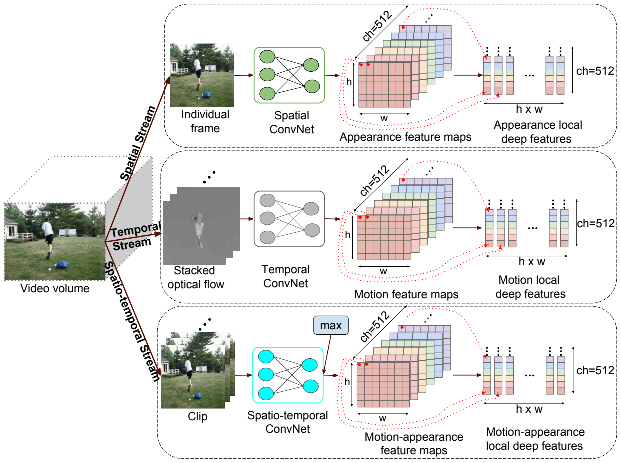

Figure 2: The framework for deep local feature extraction pipeline.

For the motion information we use the re-trained network in [49]. This deep network, also VGG, is initially proposed in [34] and contains 16 layers. The authors in [49] 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 [37]. The input to the temporal ConvNet is 10-stacked optical flow fields, each of them with one image for vertical and one image for horizontal motion. Therefore, in total there are 20-staked optical flow images as one input to the network. To extract optical flow fields we use the OpenCV implementation of the TVL1 algorithm [53]. 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. This results in $(\#frames-9)\times 49$ local deep features for a video using the temporal ConvNet. TCN refers to the features extracted with this Temporal Convolutional Network.

For the spatio-temporal stream, represented at the bottom part of Figure 2, we use the 3D ConvNet [40]. This network is trained on Sports-1M dataset [17] and contains 16 layers. 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. As the last layer of this network with spatial information has the size of the feature maps of only 4×4, we consider in our pipeline one layer before, which is called conv5b. The conv5b layer has a similar spatial size of the feature maps as the previous two networks i.e., 7×7 and similar number of channels i.e., 512. However, the conv5b layer contains two features maps, each of them $7\times 7\times 512$. In our pipeline, for this network, for an input, we build only one feature map of $7\times 7\times 512$ by taking the maximum value for each position of the both feature maps from conv5b. Then, we can extract the local deep feature similar to the previous two networks. For the 3D network, the total number of local deep features is $(\#frames-15)\times 7\times 7$ for an input video. Each resulted local deep feature is a vector with also 512 dimensions. We refer to the features extracted with this Convolutional 3D network as C3D. For all resulted local deep features from these three networks, the normalized positions, needed for ST-VLMPF, are extracted based on the localization on the feature maps.

5. Experimental Evaluation

This section presents the experimental part, where we test our proposed framework in the context of action recognition.

5.1. Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: HMDB51 [18], UCF50 [30], and UCF101 [37].

The HMDB51 dataset [18] contains 51 action categories, with a total of 6,766 video clips. We use the original non-stabilized videos, and we follow the original protocol using three train-test splits [18]. We report average accuracy over the three splits as performance measure.

The UCF50 dataset [30] 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.

	SCN		TCN		C3D	
	256	512	256	512	256	512
VLMPF	43.5	44.7	56.6	58.8	52.8	53.4
ST-VLMPF	**47.0**	**49.8**	**58.9**	**61.3**	**55.1**	**56.3**

Table 1: The final accuracy on HMDB51 using 32 spatio-temporal video divisions, with 256 and 512 feature dimensionality. We report also the results when the spatio-temporal information is not used (VLMPF).

The UCF101 dataset [37] 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 [49] 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 ST-VLMPF encoding method with two state-of-the-art approaches for feature encoding: improved Fisher Vectors (iFV) [29] and Vector of Locally Aggregated Descriptors (VLAD) [15]. We create the codebooks from 500K random selected features extracted from a subset of videos. We set the size of the codebook to 256 visual words, which is the standard adopted size, widely used by the community when using super vector-based encoding methods. Setting also the size of codebook C (k_1=256) for ST-VLMPF the same as for the other encoding methods makes easier to compare them and also it is a fair comparison having a similar number of visual words for all super vector-based encoding methods.

When using our encoding method, ST-VLMPF, we L2 normalize the final video representation vector before classification. Many works, such as [42, 45], indicate that iFV and VLAD perform better if after feature encoding the Power Normalization (PN) is applied followed by L2-normalization ($||sign(\mathrm{x})|\mathrm{x}|^{\alpha}||$). We follow this line 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, in our application, ST-VLMPF does 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.

5.3. Parameter tuning

We present the parameter tuning regarding the number of divisions of a video and the features dimensionality. All the

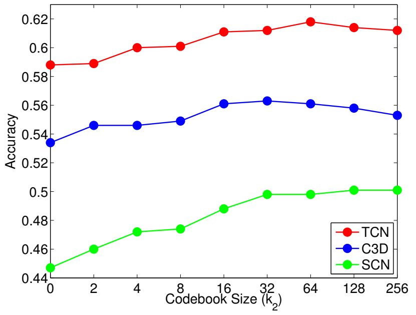

Figure 3: Evaluation of the spatio-temporal divisions of the video.

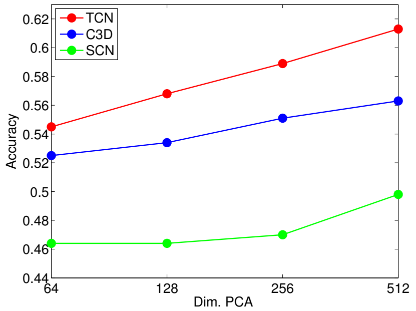

Figure 4: Evaluation of the dimensionality reduction with PCA.

tuning experiments are reported on the HMDB51 dataset.

Figure 3 presents the evaluation of parameter k_2, which denotes the size of codebook PC. The k_2 parameter represents the number of video divisions used for our ST-VLMPF encoding approach. We report the evaluation on all three local deep features considered: SCN, TCN and C3D; keeping all 512 dimensions of the original local deep features. The 0 value illustrated in Figure 3 represents the case when the spatio-temporal information is not considered, which refers to the VLMPF encoding from Figure 1. Remarkably, the performance of ST-VLMPF for all three features has a continuous significant boost in accuracy when increasing the video divisions, until k_2=32. This graph clearly shows that our approach to incorporate spatio-tempral information within the encoding process brings significant gain on the final accuracy for an action recognition system. While for the C3D features the increase in the accuracy stops around the value of k_2=32, for the SCN and TCN the accuracy still continue to have a slight increase. However, we set k_2=32 for our ST-VLMPF encoding in all remaining experiments in this paper, as this value provides a good trade-off between accuracy and computational cost and the size of the final video representation.

Figure 4 illustrates the evaluation when using PCA to reduce the feature dimensionality and decorrelate the data. From the graph we can see that for all features the accuracy is drastically influenced by the number of dimensions kept. Decreasing the features dimensionality from the original size of 512 to 64 causes a considerable drop in accuracy for SCN from 0.498 to 0.464, for TCN from 0.613 to 0.545 and for C3D from 0.563 to 0.525. For the next experiments we will consider the original features dimensionality of 512 and also when the dimensionality is decreased to 256.

Table 1 summarizes the performance numbers obtained for all three features. This table includes the results with two settings for feature dimensionality: 256 and 512. We report also the results when the spatio-temporal information is not used within the encoding process (VLMPF). In this way we can directly observe the benefit of incorporating the spatio-temporal information in the encoding method over for the performance of the system.

5.4. Comparison to other encoding approaches

In this part we present the comparison of our ST-VLMPF encoding method, with VLAD and iFV, in terms of accuracy and computational efficiency.

Accuracy comparison. We present the comparison of ST-VLMPF with VLAD and iFV in terms of accuracy over three datasets: HMDB51, UCF50 and UCF101. We report the comparison results with the features dimensionality of 256 and the 512. Table 2 shows the comparison accuracy results for all three datasets. On the challenging HMDB51 dataset ST-VLMPF clearly outperforms by a large margin iFV and VLAD for all three features. For instance, for SCN with 256 dimensionality, ST-VLMPF obtains with 9.8 percentage points more than VLAD and with 10.4 percentage points more than iFV. Similar results are reported for UCF50 and UCF101 respectively, where we can see that our proposed encoding method, ST-VLMPF, outperforms also by a large margin iFV and VLAD in all the cases, showing the effectiveness of our representation. We can also see from Table 1 that our method without spatio-temporal information, still outperforms iFV and VLAD.

Efficiency comparison. Table 3 presents an efficiency comparison of our ST-VLMPF with iFV and VLAD. The timing measurements are performed on a single core Intel(R) Xeon(R) CPU E5-2690 2.60GHz, using 500 randomly sampled videos from HMDB51 dataset.

We report the average number of frames per second and number of seconds per video that an encoding method can process for creating a video representation. For our encoding method we report also the results without using the spatio-temporal information (VLMPF) for directly observing the cost of adding the spatio-temporal encoding. We can see that by far the most expensive method for the computational cost is iFV. This is due to the fact that the method uses soft assignment and high order statistics to create the final representation. The VLAD encoding is slightly slower than VLMPF and this is due to the computation of the residuals. The computational cost for our ST-VLMPF is comparable with VLAD, however, it is more efficient than iFV, being more than 5 times faster.

The last two columns of Table 3 present the dimensional-

	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
ST-VLMPF	**47.0**	**49.8**	**58.9**	**61.3**	**55.1**	**56.3**	**86.3**	**87.7**	**97.1**	**97.2**	**94.1**	**94.7**	**80.4**	**81.8**	**86.6**	**87.3**	**85.5**	**86.2**

Table 2: Accuracy comparison on all three datasets. 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**
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**
ST-VLMPF	1531.1	0.059	964.7	0.094	1741.0	0.052	1062.0	0.085	1769.6	0.051	1086.5	0.083	81,920	155,648

Table 3: 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.

ity of the generated video representations for each encoding method. We can see that iFV is more demanding, generating a large dimensionality, while ST-VLMPF is comparable to VLAD. Even though the generated dimensionality is relatively high, in the case of a linear SVM (as we use in this paper) for ST-VLMPF with 512 feature dimensionality, the classification time to get the predicted class for a given video representation is less than 0.001 seconds, therefore, this is a negligible cost.

5.5. Fusion strategies

The previous results show that our ST-VLMPF approach obtains the best accuracy on all datasets and for all feature types. Also we show that the accuracy drops significantly when the features dimensionality decreases, therefore, to obtain the final score we use all 512 feature dimensions. Combining deep features with hand-crafted features can boost the performance of the system. Therefore, in this paper we report three feature combinations: DF, DF+HMG and DF+HMG+iDT. DF (Deep Features) is represented by SCN, TCN and C3D, all deep features are encoded with our ST-VLMPF method. As previously pointed, to extract the TCN features we use a ConvNet, which is trained on the split1 of UCF101. As the UCF101 is an extension of the UCF50 dataset, to avoid the risk of overfitting, for any further fusion and for the comparison with the state-of-the-art, we excluded TCN features for the UCF50 dataset results. HMG (Histograms of Motion Gradients) [10] is a hand-crafted descriptor which efficiently captures motion information. We used the code provided by the authors with default settings for descriptor extraction, and we encode the descriptors accordingly as recommended in the paper, using iFV. iDT (improved Dense Trajectories) [45] is a state-of-the-art hand-crafted approach, and is represented in our paper by four individual hand-crafted descriptors (HOG, HOF, MBHx, MBHy). We also use the authors provided code to extract the descriptors with default settings, and create the final representation as recommended also using iFV. For all hand-crafted features we individually apply before classification PN (α=0.1) and then L2 as recommended in [10].

For these four feature combinations we evaluate different fusion strategies: Early, where after we individually build the final representation for each feature type and normalize it accordingly, we concatenate all resulted representations in a final vector, we apply L2 normalization for making unit length and then perform the classification part; sLate, where we make late fusion by making sum between the classifiers output from each representation; wLate, where we give different weights for each feature representation classifier output, and then we perform the sum. The weight combinations are tuned by taking values between 0 and 1 with the step 0.05; sDouble, where besides summing the classifier output from the individual feature representations, we also add the classifier output resulted from the early fusion; wDouble, where we tune the weight combinations for the sum, similar to wLate.

Table 4 shows that early fusion performs better than late fusion. Double fusion combines the benefit of both, early and late fusion, and boosts further the accuracy. For more challenging datasets such as HMDB51, combining deep features with hand-crafted features improves considerably the accuracy, while for less challenging datasets such as UCF50, the hand-crafted features do not bring significant contribution over deep features. With this framework, we obtain outstanding final results of 73.1% on HMDB51, 97.0% on UCF50 and 94.3% on UCF101.

5.6. Comparison to the state-of-the-art

Table 5 presents the comparison of our final results with the state-of-the-art approaches on HMDB51, UCF50 and UCF101. For this comparison we report two final results. First result, represents only our ST-VLMPF(DF), which is obtained by using our proposed encoding method over all three deep features (SCN, TCN and C3D). The second one,

	HMDB51 (%)			UCF50* (%)			UCF101 (%)		
	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT	DF	DF+HMG	DF+HMG+iDT
Early	68.6	69.5	71.7	95.0	95.3	96.7	93.5	**94.0**	**94.3**
sLate	66.4	66.5	68.8	94.2	94.4	95.6	92.0	92.5	92.4
wLate	67.6	67.8	70.9	94.8	95.1	96.6	92.2	92.7	93.4
sDouble	68.3	68.4	70.3	94.6	94.9	96.1	92.6	93.1	92.8
wDouble	**69.5**	**70.3**	**73.1**	**95.1**	**95.4**	**97.0**	**93.6**	**94.0**	**94.3**

Table 4: Fusion strategies. DF (Deep Features) represent all three local deep features (SCN, TCN, C3D), HMG (Histograms of Motion Gradients) [10] and iDT (improved Dense Trajectories) [45] 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. (*TCN features are not considered for UCF50 dataset as explained above.)

HMDB51 (%)		UCF50* (%)		UCF101(%)	
Jain et al. [14] (2013)	52.1	Solmaz et al. [36] (2013)	73.7	Wang et al. [46] (2013)	85.9
Zhu et al. [56] (2013)	54.0	Reddy et al. [30] (2013)	76.9	Karpathy et al. [17] (2014)	65.4
Oneata et al. [24] (2013)	54.8	Shi et al. [32] (2013)	83.3	Simonyan et al. [33] (2014)	88.0
Wang et al. [45] (2013)	57.2	Wang et al. [43] (2013)	85.6	Wang et al. [44] (2015)	86.0
Kantorov et al. [16] (2014)	46.7	Wang et al. [45] (2013)	91.2	Sun et al. [39] (2015)	88.1
Simonyan et al. [33] (2014)	59.4	Ballas et al. [2] (2013)	92.8	Ng et al. [52] (2015)	88.6
Peng et al. [28] (2014)	66.8	Everts et al. [11] (2014)	72.9	Tran et al. [40] (2015)	90.4
Sun et al. [39] (2015)	59.1	Uijlings et al. [41] (2014)	80.9	Wang at al. [49] (2015)	91.4
Wang et al. [44] (2015)	60.1	Kantorov et al. [16] (2014)	82.2	Wang et al. [48] (2015)	91.5
Wang et al. [48] (2015)	65.9	Ciptadi et al. [4] (2014)	90.5	Zhang et al. [54] (2016)	86.4
Park et al. [25] (2016)	56.2	Narayan et al. [23] (2014)	92.5	Peng et al. [27] (2016)	87.9
Seo et al. [31] (2016)	58.9	Uijlings et al. [42] (2015)	81.8	Park et al [25] (2016)	89.1
Peng et al. [27] (2016)	61.1	Wang et al. [44] (2015)	91.7	Bilen et al. [3] (2016)	89.1
Yang et al. [51] (2016)	61.8	Peng et al. [27] (2016)	92.3	Diba et al. [8] (2016)	90.2
Bilen et al. [3] (2016)	65.2	Duta et al. [10] (2016)	93.0	Fernando et al. [12] (2016)	91.4
Fernando et al. [12] (2016)	66.9	Seo et al. [31] (2016)	93.7	Yang et al. [51] (2016)	91.6
Our ST-VLMPF(DF)	**69.5**	Our ST-VLMPF(DF)	**95.1**	Our ST-VLMPF(DF)	**93.6**
Our best	**73.1**	Our best	**97.0**	Our best	**94.3**

Table 5: Comparison to the state-of-the-art. Our ST-VLMPF(DF) represents the results obtained with only our representation over deep features (SCN, TCN and C3D). Our best is the final result from the best combination of our ST-VLMPF with hand-crafted features HMG [10] and iDT (HOG, HOF, MBHx, MBHy) [45]. (*TCN features are not considered for UCF50 dataset as explained above.)

is our best result reported in this paper, obtained using ST-VLMPF(DF) + HMG + iDT.

Our ST-VLMPF representation outperforms the state-of-the-art approaches by a large margin on all three datasets, which demonstrates that our method provides a powerful video representation with very competitive results. Furthermore, with our best results, which use also hand-crafted features, we improve the state-of-the-art by 6.2 percentage points on the challenging HMDB51 dataset, by 3.3 percentage points on UCF50 and by 2.7 percentage points on UCF101. It is important to highlight that these results are obtained using pre-trained networks which are not re-trained or fine-tuned on our particular datasets (except TCN features for UCF101 dataset). For instance, for HMDB51 dataset all three networks did not see any training example from this dataset, and we still obtain impressive results. Therefore, our approach is also suitable in various practical scenarios when re-training or fine-tuning is more difficult to accomplish.

6. Conclusion

In this paper we introduced the Spatio-Temporal Vector of Locally Max Pooled Features (ST-VLMPF), a super vector-based encoding method specifically designed for encoding local deep features. We also efficiently incorporate the spatio-temporal information within the encoding method, providing a significant boost in accuracy. ST-VLMPF outperforms two of the most powerful encoding methods by a large margin (Improved Fisher Vectors and Vector of Locally Aggregated Descriptors), while maintaining a low computational complexity. Our approach provides a solution for incorporating deep features over the entire video, helping to solve the issue with the false label assigned to the network input. The comparison of our action recognition pipeline with the state-of-the-art approaches over three challenging datasets proves the superiority and robustness of our video representation.

Acknowledgments. Part of this work was funded under research grant PN-III-P2-2.1-PED-2016-1065, agreement 30PED/2017, project SPOTTER.

References

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

[2] N. Ballas, Y. Yang, Z.-Z. Lan, B. Delezoide, F. Prêteux, and A. Hauptmann. Space-time robust representation for action recognition. In *ICCV*, 2013. 8

[3] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 1, 3, 4, 8

[4] A. Ciptadi, M. S. Goodwin, and J. M. Rehg. Movement pattern histogram for action recognition and retrieval. In *ECCV*, 2014. 8

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1

[6] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*. 2006. 1

[7] 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.

[8] A. Diba, A. M. Pazandeh, and L. Van Gool. Efficient two-stream motion and appearance 3d cnns for video classification. In *ECCV ws*, 2016. 8

[9] 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. 2

[10] 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. 7, 8

[11] I. Everts, J. C. Van Gemert, and T. Gevers. Evaluation of color spatio-temporal interest points for human action recognition. *TIP*, 2014. 8

[12] B. Fernando, P. Anderson, M. Hutter, and S. Gould. Discriminative hierarchical rank pooling for activity recognition. In *CVPR*, 2016. 3, 8

[13] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars. Rank pooling for action recognition. *TPAMI*, 2016. 3

[14] M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013. 8

[15] 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. 1, 2, 5

[16] V. Kantorov and I. Laptev. Efficient feature extraction, encoding and classification for action recognition. In *CVPR*, 2014. 8

[17] 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. 1, 3, 4, 5, 8

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011. 5

[19] I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005. 1

[20] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 1, 2

[21] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2

[22] I. Mironică, I. C. Duţă, 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. 1, 2

[23] S. Narayan and K. R. Ramakrishnan. A cause and effect analysis of motion trajectories for modeling actions. In *CVPR*, 2014. 8

[24] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 8

[25] 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. 1, 3, 4, 8

[26] X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *ECCV*. 2014. 2

[27] 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. 1, 2, 8

[28] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 2, 8

[29] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010. 1, 2, 5

[30] K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013. 5, 8

[31] 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. 8

[32] F. Shi, E. Petriu, and R. Laganiere. Sampling strategies for real-time action recognition. In *CVPR*, 2013. 8

[33] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 3, 4, 8

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 3, 4

[35] 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. 2

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

and applications, 2013. 8

[37] 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. 4, 5

[38] 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. 1

[39] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015. 1, 3, 4, 8

[40] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 1, 3, 4, 5, 8

[41] J. R. R. Uijlings, I. C. Duta, N. Rostamzadeh, and N. Sebe. Realtime video classification using dense hof/hog. In *ICMR*, 2014. 1, 8

[42] 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. 1, 2, 5, 8

[43] 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. 1, 8

[44] H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015. 8

[45] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 5, 7, 8

[46] H. Wang and C. Schmid. Lear-inria submission for the thumos workshop. In *ICCV Workshop*, 2013. 8

[47] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. 1, 2

[48] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *CVPR*, 2015. 8

[49] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 1, 2, 3, 4, 5, 8

[50] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, 2015. 2

[51] X. Yang, P. Molchanov, and J. Kautz. Multilayer and multimodal fusion of deep neural networks for video classification. In *ACMMM*, 2016. 8

[52] 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. 1, 3, 4, 8

[53] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. 2007. 4

[54] B. Zhang, L. Wang, Z. Wang, Y. Qiao, and H. Wang. Real-time action recognition with enhanced motion vector cnns. In *CVPR*, 2016. 8

[55] 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. 2

[56] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu. Action recognition with actons. In *ICCV*, 2013. 8