Noname manuscript No.
(will be inserted by the editor)

Efficient Human Action Recognition using Histograms of Motion Gradients and VLAD with Descriptor Shape Information

Ionut C. Duta · Jasper R.R. Uijlings · Bogdan Ionescu · Kiyoharu Aizawa · Alexander G. Hauptmann · Nicu Sebe

Received: date / Accepted: date

Abstract Feature extraction and encoding represent two of the most crucial steps in an action recognition system. For building a powerful action recognition pipeline it is important that both steps are efficient and in the same time provide reliable performance. This work proposes a new approach for feature extraction and encoding that allows us to obtain real-time frame rate processing for an action recognition system. The motion information represents an important source of information within the video. The common approach to extract the motion information is to compute the optical flow. However, the estimation of optical flow is very demanding in terms of computational cost, in many cases being the most significant processing step within the overall pipeline of the target video analysis application. In this work we propose an efficient approach to capture the motion information within the video. Our proposed descriptor, Histograms of Motion Gradients (HMG), is based on a

I.C. Duta, N. Sebe
University of Trento, Italy
E-mail: {ionutcosmin.duta, niculae.sebe}@unitn.it

J.R.R. Uijlings
University of Edinburgh, UK
E-mail: jrr.uijlings@ed.ac.uk

B. Ionescu
University Politehnica of Bucharest, Romania
E-mail: bionescu@imag.pub.ro

K. Aizawa
University of Tokyo, Japan
E-mail: aizawa@hal.t.u-tokyo.ac.jp

A. G. Hauptmann
Carnegie Mellon University, USA
E-mail: alex@cs.cmu.edu

The final publication is available at Springer via:
http://dx.doi.org/10.1007/s11042-017-4795-6

simple temporal and spatial derivation, which captures the changes between two consecutive frames. For the encoding step a widely adopted method is the Vector of Locally Aggregated Descriptors (VLAD), which is an efficient encoding method, however, it considers only the difference between local descriptors and their centroids. In this work we propose Shape Difference VLAD (SD-VLAD), an encoding method which brings complementary information by using the shape information within the encoding process. We validated our proposed pipeline for action recognition on three challenging datasets UCF50, UCF101 and HMDB51, and we propose also a real-time framework for action recognition.

Keywords Video Classification · Action Recognition · Histograms of Motion Gradients (HMG) · Shape Difference VLAD (SD-VLAD) · Computational Efficiency · Real-Time Processing

1 Introduction

Over the recent years an explosive growth in video content has occurred and continues growing. As an example of this fulminant increase, Cisco forecast[1] mentioned that the IP video would account for 80% of all IP traffic by 2019. With this huge amount of multimedia content, computational efficiency has become as important as the accuracy of the techniques.

Even though in the past several years there has been an important progress in video analysis techniques, in particular on improving the accuracy of human action recognition in videos [47, 49, 36, 45, 28, 48], the current methods in terms of computational time are able to run with 1-3 frames per second. For instance, in [45] is reported that the popular approach in [23] runs with 1.4 frames per second. Fast video analysis is important in many applications and this issue of efficiency became very important for large-scale video indexing systems or automatic clustering of large video collections.

The Bag of Visual Words (BoVW) framework with its variations [24, 47, 49] has been widely used and showed its effectiveness in video analysis challenges. The schematic view for a BoVW pipeline is represented in Fig. 1, which contains in general three main steps: feature extraction, feature encoding and classification. In addition to these main steps, the framework contains some pre/post processing techniques, such as PCA, feature decorrelation and normalization, which can influence considerably the performance of the pipeline. The commonly used approach for classification is employing a fast SVM classifier over the resulted video representations. The encoding step creates a final representation of the video and a very widely used approach is counting the frequency of the visual words. However, recently super-vector based encoding methods, such as Vector of Locally Aggregated Descriptors (VLAD) [16] and Fisher Vector (FV) [33], obtained state-of-the-art results for many tasks.

[1] http://newsroom.cisco.com/press-release-content?articleId=1644203

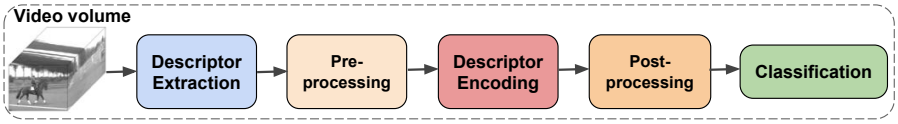

Fig. 1: The general pipeline for video classification.

The video contains two important sources of information: the static information in the frames and the motion between frames. The feature extraction step focuses mainly on these two directions. The first direction has the goal to capture the appearance information in frames, such as Histogram of Oriented Gradients (HOG) [7, 24]. The other direction is based on optical flow fields like Histogram of Optical Flow (HOF) [24] and Motion Boundary Histograms (MBH) [8]. These descriptors are extracted and combined using Space Time Interests Points (STIP) [23], dense sampling [51, 45] or extracting the descriptors along some trajectories [41, 47, 49].

The pipeline in Fig. 1 represents also the common main phases for an action recognition framework. For the classification part, the used approaches are already mature, i.e., most of the existing works, such as [47, 49, 44, 32, 45], use linear SVM, as this is a very fast and effective method. However, for descriptor extraction and encoding there is still room for improvement. For an efficient video classification system it is necessary that both, descriptor extraction and encoding, to be efficient, otherwise if one of them is not competitive regarding the performance, then the target cannot be reached. As one of the goals of this work is to provide a very efficient system for video classification, we propose new solutions for both steps: descriptor extraction and encoding.

Temporal variation within the videos provides an important source of information about its content. Usually, the temporal information is computed with an optical flow method. There is a large number of approaches for extracting the optical flow fields, from relatively classic methods, such as [27, 14] to relatively recent approaches like [13, 5, 54, 6], which use complex algorithms to compute the motion information. The main drawback of those methods is the high computational cost. This shortcoming becomes the bottleneck in many applications. For instance, the authors in [47] report that optical flow takes more than 50% of the total time for feature extraction. We present in this paper a new efficient descriptor, called Histograms of Motion Gradients (HMG), which is based on the motion information. The proposed HMG descriptor captures the motion information using a very fast temporal derivation, which enables us to have similar computational cost as HOG but with a significant improvement in accuracy.

The final representation of the video is one of the key factors for visual recognition such as human action recognition. We can see that in most of the research works in computer vision and multimedia [49, 32, 45] the super vector-based encoding methods are shown to outperform the other encoding methods. Vector of Locally Aggregated Descriptors (VLAD) [16] is one of the most popular and efficient super vector-based encoding methods which

proved its efficiency in creating the final representation of a video for action recognition tasks. Besides its performance, VLAD has several drawbacks. It considers only the mean to represent a cluster of features and also keeps only the first-order statistics and ignores other source of information. The mean is not enough to reflect a distribution, but in general, the mean and the standard deviation can be enough to capture the statistics. To address this issue, this work proposes to improve VLAD by keeping the standard deviation information and incorporating shape information as the difference of standard deviations between the altered standard deviation of local descriptors and the standard deviation of the visual word. This new encoding method, Shape Difference for VLAD (SD-VLAD), captures the distribution shape of the features and brings complementary information to the original VLAD.

The main contributions of this work can be summarized with the following:

- We introduce a new descriptor (HMG), which captures the motion information using a simple temporal derivation, without the need of using the costly optical flow. We make the code for descriptor extraction available[2];
- We propose a new encoding method (SD-VLAD), which captures shape information within the encoding process, providing the best trade-off between accuracy and computational cost. We make the code for descriptor encoding available[2];
- We adopt several speed-ups, such as fast aggregation of gradient responses, reuse subregions of aggregated magnitude responses, and frame subsampling, which make the pipeline more efficient;
- We propose an integration of our descriptor and encoding method in a specifically designed video classification framework which allows for real-time performance while maintaining the high accuracy of the results.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 introduces our new proposed descriptor with the adopted approaches for improving the efficiency. The new encoding method is presented in Section 5.4. The experimental evaluation and the comparison with state-of-the-art are presented in Section 5. Finally, Section 6 concludes this work.

2 Related work

There are mainly two directions to extract features from a video: hand-crafted and deep learning. One of the state-of-the-art approaches in the hand-crafted category is represented by Improved Dense Trajectory (IDT) [49], where the main goal is to track some points through the video and to extract different descriptors along the trajectories of the points. The work in[49] is an extension of [47] by using an algorithm to cancel the camera motion to obtain more reliable features. The work in [23] proposes Space Time Interests Points (STIP), it has successfully adapted interest points from the domain of images to the domain of video by extending the Harris detector to space-time interest points.

[2] `https://iduta.github.io/software.html`

The work in [51, 45, 12] uses dense sampling approach. The authors of [51] evaluated several interest point selection methods and several spatio-temporal descriptors. They found that dense sampling methods generally outperform interest points, especially on more difficult datasets.

The previously mentioned methods establish the region of extracting for several standard descriptors, such as Histogram of Oriented Gradients (HOG) [7, 24], Histogram of Optical Flow (HOF) [24] and Motion Boundary Histograms (MBH) [8]. The work in [25, 24] considers Spatial Pyramid (SP) approach to capture the information about features location. The works in [18, 34] focus on improving the efficiency of action recognition by exploring different alternatives for the computation of the standard optical flow.

Recently, the approaches based on Convolutional Neural Networks (CNN) [19, 37, 36, 53, 42, 30, 3] have proven to obtain very competitive results compared to traditional hand-crafted methods. In general, for action recognition tasks, these works use the two-stream approach where one network is trained on the static images and another network is trained on the optical flow fields. In the end there is a fusion over the output of both networks to provide the final result. The work in [9] uses a hybrid representation by combining hand-crafted with deep features and takes advantage of different techniques to boost the performance. The work in [52] is fully based on deep features, modeling long-range temporal structure and using a series of good practices to improve the network performance.

The feature encoding is a very important step for action recognition and influences considerably the performance of the general framework. Vector based approaches showed to be very competitive for this step. The most popular super vector encoding methods are: Fisher Vector (FV) [33], Vector of Locally Aggregated Descriptors (VLAD) [16] and Super Vector Coding (SVC) [55]. FV was initially introduced for large-scale image categorization [33]. This encoding method combines the benefits of generative and discriminative approaches and aggregates the first- and the second-order statistics. FV is performing a soft assignment which in general gives better performance, however, this affects the computational cost. The work in [21] proposes an extension to Spatial Fisher Vector (SFV) which computes per visual word the mean and variance of the 3D spatio-temporal location of the assigned features. VLAD encoding method can be viewed as a simplification of FV which keeps only first-order statistics and performs hard assignment, which makes it much faster than FV. SVC method keeps the zero-order and first-order statistics, thus SVC can be seen as a combination between Vector Quantization (VQ) [38] and VLAD.

There are many precursors who focus on improving VLAD representation, as this is an efficient super vector based encoding method with very competitive results in many tasks. The work in [28] proposes to use Random Forest in a pruned version for the trees to build the vocabulary and then they additionally concatenate second-order information, similar as in FV. Differently from their approach, in this article we keep k-means as clustering method and incorporate second-order information by difference of standard deviations. Another recent work which boosts the performance of VLAD is presented in [31], where the au-

thors suggest improving VLAD by concatenating the second- and third-order statistics, and using supervised dictionary learning. Our approach is different as we consider additionally only the second-order information and build the dictionary in an unsupervised manner. Furthermore, we have a different definition for the first-order statistics by incorporation in the representation the standard deviation, and also our second-order statistics is different in a main key point that we consider an altered standard deviation of local descriptors by counting on the global mean of the cluster instead of local mean of the descriptors. The work in [11] focuses on improving VLAD by using a double assignment approach, and the work in [10] incorporates within the encoding process the spatio-temporal information showing a consistent improvement in accuracy.

The work in [2] proposes to use intra-normalization to improve VLAD performance. The impact of this approach is to suppress the negative effect of the high values within the vector, which can dominate the similarity between vectors. The authors propose to L2 normalize the aggregated residuals within each VLAD block. We consider also intra-normalization in our framework. Furthermore, they use vocabulary adaptation as an efficient approach to extend the vocabulary to another dataset. In [1] it is proposed RootSIFT normalization to improve the performance of the framework for object retrieval. This normalization approach is based on the idea to reduce the influence of large bin values, by computing square root of the values.

Inspired by these previous works, in this paper we propose a new head-crafted descriptor and an extended version for VLAD. The proposed descriptor, Histograms of Motion Gradients (HMG), is computed by initially extracting the motion information by applying a fast temporal derivation between two consecutive frames, then for the resulted "motion image" we compute the horizontal and vertical gradients. From the obtained gradients we compute the magnitude and the angle, and we apply then the quantization and aggregation step to create the final descriptor for a video. For the encoding method, we propose Shape Difference for VLAD (SD-VLAD) where initially a codebook is learnt with k-means and then we compute the final representation with two formulas, one is based on the residual information and the other is focalized on capturing the information regarding the distribution shape by computing the difference between standard deviations. Both source of information are complementary to each other and their combination boosts the performance of the encoding method while achieving a low computational complexity. Our new approach for feature extraction and encoding allows us to build a very efficient pipeline for video classification, being able to run at more than real-time frame rate.

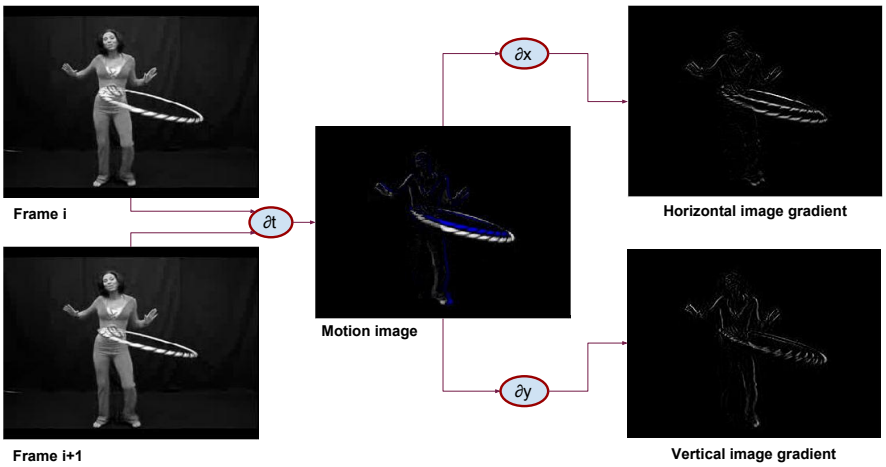

Fig. 2: Visualization of the process for capturing the motion information for the HMG descriptor. We initially perform a fast temporal derivation over each two consecutive frames, which provides us the motion image. Then we compute the horizontal and vertical gradients for the resulted motion image. The pixels depicted in blue color represent the negative values after temporal derivation.

3 Proposed HMG method for descriptor extraction

In this section we introduce the proposed method for capturing motion information from the video. We present several speed-ups that make the framework very efficient, being able to achieve real-time processing.

3.1 Histograms of Motion Gradients (HMG)

Our descriptor, Histograms of Motion Gradients (HMG), is based on a temporal derivation to compute the motion information and it is integrated in the first step of an action recognition framework Fig. 1. The illustration of the process of capturing the temporal information is presented in Fig. 2. For each two consecutive frames we first compute the temporal derivation:

$$T_{(i,i+1)} = \frac{\partial(F_i, F_{i+1})}{\partial t} \tag{1}$$

where F_i is the frame at time index i.

The temporal derivative is computed very effectively by applying a simple and fast filter window [1 -1] for each two consecutive frames (F_i, F_{i+1}). The result of this operation is illustrated in the middle image of Fig. 2, where we can observe that the information about the motion between two frames is kept. We can call the output of the applied temporal derivative "motion image". Obviously, after applying the temporal derivation some values are negative, depending on the result of derivation between the pixels in frame i and frame $i + 1$, we represent the negative values with blue color in Fig. 2.

After the computation of the temporal derivative, we compute the spatial gradients of the resulted motion image, which allows us to compute the magnitude and the angle of the gradient responses. In the right part of Fig. 2 there are represented the horizontal and vertical gradients, computed with:

$$X_{(i,i+1)} = \frac{\partial T_{(i,i+1)}}{\partial x}, \quad Y_{(i,i+1)} = \frac{\partial T_{(i,i+1)}}{\partial y} \tag{2}$$

For the computation of spatial gradients we use also the simple and fast filter window [1 0 -1], similar as for HAAR-features. The gradients with this mask are computed much faster than, for instance, Gaussian derivatives. Basically, the gradients with this filter are obtained by making the difference between a frame and its shifted values with one position, once horizontally and once vertically. This makes the computation of gradients very fast.

After we obtain the spatial derivatives, similar as for HOG, we compute the magnitude and the angle:

$$mag = \sqrt{X^2 + Y^2}, \quad \theta = \arctan\left(\frac{Y}{X}\right) \tag{3}$$

where each operation from the above formulas is element-wise.

The result of these operations is a 2-dimensional vector field per each new motion frame. We quantize the orientation (θ) in 8 directions/bins and then we accordingly accumulate the magnitude corresponding to each bin. This is similar to how gradient responses are accumulated in SIFT [26]. The next step is to perform the aggregation of those quantized responses over blocks in both spatial and temporal direction. Then we concatenate the responses over several adjacent blocks. We provide in the next subsection the details about the procedure of dividing the video in blocks and volumes. Afterwords, the pipeline in Fig. 1 continues with the next step by applying some pre-processing operations before feature encoding, such as normalization and PCA with decorrelation of features. The next steps after the descriptor extraction are very important for the performance of our descriptor. For instance, the descriptors obtained from the motion image may include noise which can result in high peaks that can dominate the entire vector representation. To reduce the negative influence of this aspect, over the initial representation of the descriptors we apply RootSIFT normalization [1], which penalizes more the high values within the vector, contributing to creating a smoother vector (without large peaks) to represent each local extracted descriptor.

3.2 Speed-up HMG extraction

For our proposed descriptor we use a dense sampling strategy to extract the features. In addition to the presented approach for capturing the motion information very efficiently and using fast filters for derivatives, we describe several speed-ups that improve the efficiency of the descriptor extraction process of HMG. The efficiency improvement is performed by taking the advantage of

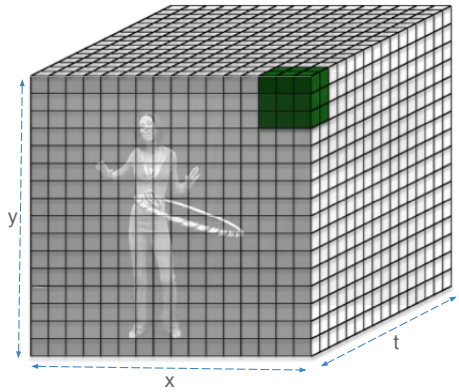

Fig. 3: The process of dividing the video in blocks and volumes. The part depicted in green represents an illustration of a volume created from 3 by 3 by 2 blocks.

the densely sampled approach and by adopting to our new descriptor several speed-ups presented in [45].

1) Reuse of blocks: Our choice to establish the region of the descriptor extraction is the use of dense sampling strategy since this method has a big potential for efficiency. It can be also easily extended to an even faster version using parallelization. Furthermore, in several works, it has been found to be more accurate than keypoint-based sampling in images [17] and videos [51, 32]. We take advantage of the densely sampled descriptor nature in order to speed up the feature extraction time. Fig. 3 illustrates an example for dividing the video into blocks, and how a volume is created of several adjacent blocks. Our HMG descriptor is extracted on a single scale over each block, which consists of 8 by 8 pixels by 6 frames. The size of the blocks is also our dense sampling rate. The green part from the Fig. 3 represents a video volume, where the responses over several adjacent blocks are concatenated for creating the final descriptor. Each video volume consists of 3 by 3 by 2 blocks, corresponding to x, y and t axis. By choosing the sampling rate equally with the block size, then we can reuse the blocks for making the descriptor extraction efficient. Therefore, the representation for a block is computed only once and then use it for the construction of all the volumes around that block. For instance, each block can be reused for 18 times (excepting the blocks on the borders) for the current size of the video volume: 3 by 3 by 2 blocks.

2) Fast aggregation of responses: After we compute the magnitude and the angle, the resulted responses are aggregated for each block. We adopted the approach in [43]. Basically we compute the aggregation of all the frame pixels by doing just a multiplication of three matrices. After the spatial aggregation of 8 by 8 pixels and the temporal aggregation of 6 frames, each block is characterized by 8 values as we consider 8 orientations for quantization of responses. Having 8 bins and a size of 3 by 3 by 2 for video volume, the original dimensionality of our descriptor is therefore 144.

3) Frame subsampling: For efficiency reasons we evaluate HMG by subsampling video frames. Subsequent frames contain redundant information, and the computational cost can be substantially improved by frame subsampling. We evaluate the impact on the accuracy and efficiency of our descriptor by skipping frames. A detailed analysis of the trade-off between accuracy and computational time is presented with the experimental results.

4 Proposed SD-VLAD method for descriptor encoding

In this section we present our encoding method for human action recognition. We first review the original VLAD representation.

4.1 VLAD representation

VLAD is initially proposed in [16] and can be seen as a simplification of the FV. For the VLAD pipeline first a codebook of k visual words is learned with k-means, $M = \{\mu_1, \mu_2, ..., \mu_k\}$, which are the means for each cluster. For each visual word a subset of local descriptors is assigned based on the nearest neighborhood criterion, $X_i = \{x_1, x_2, ..., x_{n_i}\}$, where x is a feature vector and n_i is the number of assigned features to the i-th visual word. The idea of VLAD is to accumulate for each visual word the residuals (the differences between the assigned descriptors and the centroid):

$$v_i = \sum_{j=1}^{n_i} (x_j - \mu_i) \tag{4}$$

The final VLAD representation is a concatenation of all vectors v_i and the final dimensionality of VLAD is $k \times d$, where d is the dimension of the descriptors. The VLAD performance can be boosted by using intra-normalization [2], which normalize independently each VLAD block v_i: $\frac{v_i}{||v_i||_p}$, usually $p = 2$ (i.e., the L2 norm).

4.2 Shape Difference for VLAD

The original VLAD is based only on the mean as statistical information, however, for describing a set of descriptors it is more informative to have at least the mean and standard deviation of them. The residuals computed by VLAD algorithm can provide only partial cluster distribution information. Fig. 4 shows a case when VLAD fails to provide a good discriminative representation. Even if the centroids μ_1 and μ_2 are completely different and the feature distribution assigned to each cluster differs significantly, the standard VLAD returns the same representation, therefore, the sums over the residuals v_1 and v_2 are completely equal ([1 -8]). Also in the case when the features are

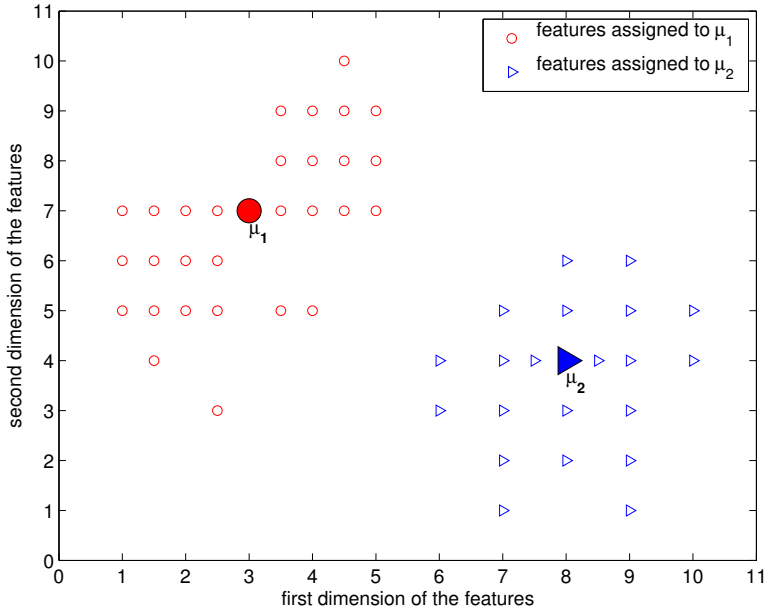

Fig. 4: An illustrative example when VLAD fails to provide a reliable representation. Each descriptor is assigned to its nearest centroid, μ_1 or μ_2. Even though the distribution of the assigned descriptors to each visual word is completely different, the result of VLAD representation (computed with the standard formula (4)) is equal with [1 -8] for both, v_1 and v_2. In this case, only the computation of the residuals is not enough for obtaining a reliable description.

distributed in a symmetrical arrangement around the centroid, then the sum over the residuals is a vector of zeros, this leading to making no difference between the results of a cluster with features distributed symmetrically and a cluster with no features assigned.

For providing a more discriminative representation, it is necessarily to introduce more statistical information. We propose Shape Difference for VLAD (SD-VLAD), which captures information related to the distribution shape of the descriptors. Our final representation is computed with two formulas. Similar to FV, for the first formula, the residuals are divided by standard deviation, and our first part of the final representation is represented as:

$$v_i^{\mu} = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{x_j - \mu_i}{\sigma_i} \tag{5}$$

where n_i is the number of descriptors assigned to the cluster μ_i and σ_i is the standard deviation of the cluster with the mean μ_i.

The division by the number of descriptors, that switches sum pooling to average pooling, is a very simple technique to deal with the problem of bursti-

ness when some parts of VLAD can dominate the entire representation. In the end, these components will have a greater weight for the classifier and influence negatively the performance. This normalization, with the number of descriptors assigned, becomes more important if no intra-normalization technique is used. Intra-normalization is a better strategy to deal with the problem of burstiness, but there are some cases when intra-normalization is not recommended. For instance, in [32] it is underlined that intra-normalization may have a negative effect for sparse features like STIPs.

By considering the normalization (with the number of descriptors assigned) for VLAD, the Equation (4) becomes:

$$\bar{v}_i = \frac{1}{n_i}\sum_{j=1}^{n_i}(x_j - \mu_i) = \frac{1}{n_i}(\sum_{j=1}^{n_i}(x_j) - n_i\mu_i) = \frac{1}{n_i}\sum_{j=1}^{n_i}(x_j) - \mu_i = \hat{\mu}_i - \mu_i \quad (6)$$

where $\hat{\mu}_i$ is the mean of the local descriptors assigned to the cluster μ_i. In this way, VLAD can be seen as the difference between the mean of the local descriptors and its assigned visual word.

To address the shortcoming of VLAD considering only the mean as statistical information, we consider in our representation the shape information. Starting from the Equation (6) we can go further with the analogy (the difference between standard deviations) and build the shape difference representation as following:

$$v_i^{\sigma} = \hat{\sigma}_i - \sigma_i = \left(\frac{1}{n_i}\sum_{j=1}^{n_i}(x_j - \mu_i)^2\right)^{\frac{1}{2}} - \sigma_i \quad (7)$$

where $\hat{\sigma}_i$ is the altered standard deviation of the local descriptors assigned to cluster μ_i and σ_i is the standard deviation of the cluster μ_i; the power of a vector is the element-wise power. We compute the standard deviation for the local descriptors by using the mean of the cluster and not the local mean of assigned descriptors due to the fact that the local mean of the assigned descriptors may not contain statistical information, as there are many cases when too few descriptors (even one or two descriptors) are assigned to a cluster, especially when the number of clusters is increased. Making the difference of descriptors and their local mean can lead to cases with no information. Instead, by considering the mean of the cluster, which is computed on a large number of descriptors, the difference is more stable, especially for the cases with less descriptors assigned to a cluster.

The shape difference brings complementary information related to the mean, in the experimental part of the paper we show that it is beneficial for the classifier. For our final SD-VLAD representation we concatenate the resulting vectors from v^{μ} and v^{σ} (Equation (5) and Equation (7)). We apply also intra-normalization L2 for each v_i^{μ} and v_i^{σ}.

5 Experimental Evaluation

The general pipeline used for evaluation is the one presented in Fig. 1, and more details for each step are presented in the remaining part of the paper.

In the following we present: the datasets used for evaluation (Section 5.1); experimental setup (Section 5.2); comparison of the proposed descriptor with other dense methods (Section 5.3); evaluation of the proposed encoding method together with different standard approaches (Section 5.4); comparison of our proposed descriptor with Improved Dense Trajectories approach (Section 5.5); the impact of the frame subsampling on the accuracy and on the computational cost (Section 5.6); the proposed pipeline for real-time video classification (Section 5.7); comparison with the state-of-the-art approaches (Section 5.8).

5.1 Datasets

We evaluate our framework on three of the most popular and challenging datasets for action recognition: UCF50 [35], UCF101 [40] and HMDB51 [22].

The UCF50 dataset [35] contains 6,618 realistic videos taken from YouTube. There are 50 human action categories mutually exclusive, which range from general sports to daily life exercises. 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 [40] is a widely adopted benchmark for action recognition, consisting in 13,320 realistic videos, which are divided into 25 groups for each action category. This dataset contains 101 action classes and there are at least 100 video clips for each class. We follow for evaluation the recommended default three training/testing splits. We report the average recognition accuracy over these three splits.

The HMDB51 dataset [22] contains 51 action categories, with a total of 6,766 video clips extracted from various sources, such as Movies, the Prelinger archive, Internet, Youtube and Google videos. 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 [22]. We report average accuracy over the three splits as performance measure.

5.2 Experimental setup

For evaluation of our proposed HMG descriptor the baseline is to use dense sampling with 8 by 8 pixels by 6 frames as in [44, 45] and the gradient magnitude quantized in 8 orientations. The final descriptor is a concatenation of 3 by 3 by 2 blocks. For the pre-processing step we perform RootSIFT [1] normalization and then we apply PCA to reduce the dimensionality by a factor

Table 1: Accuracy and efficiency comparison of various dense descriptors (results reported on the UCF50 dataset, best result is in bold).

descriptor	HOG	HOF	MBHx	MBHy	**HMG**
accuracy	0.762	0.799	0.784	0.792	**0.814**
seconds/video	2.67	4.03	4.37	4.37	2.73
frames/second	73.50	48.61	44.80	44.84	71.73

of two and decorrelate the features. This yields a final descriptor dimension of 72. We use spatial pyramid in all our experiments, we divide all the frames of the video into three horizontal parts which roughly correspond to a ground, object, and sky division.

The codebook for each experiment needed for feature encoding is built from randomly sampled 500K features of the training set for the specific tested dataset. For the resulted vectors after descriptor encoding we apply power normalization followed by L2 for the super-vector based encoding methods and power normalization followed by L1 for all other visual word assignment methods. The parameter α for power normalization is initially fixed to 0.5. We perform the classification with SVMs, with a linear kernel for super-vector based encoding methods and histogram intersection kernel for all other encoding methods, with $C = 100$.

We initially compare our descriptor with dense HOG, HOF, MBHx and MBHy using the available code from [44, 45]. For these descriptors we use the same settings and speed-ups as presented for HMG, see Section 3. The optical flow for HOF, MBHx and MBHy is computed with Horn-Schunck method [14] using the Matlab Computer Vision System Toolbox as recommended in [45]. In [45] is presented a detailed evaluation of using different optical flow approaches with the conclusion that Horn-Schunck method provides the best trade-off between the accuracy and computational efficiency. The timing measurements are performed on a single core Intel(R) Xeon(R) CPU E5-2690 2.60GHz, using 500 randomly sampled videos (10 videos for each class) from the UCF50 dataset. We report the average of the number of seconds per video and the number of frames per second that the system can process. We perform the parameter tuning on the UCF50 dataset.

5.3 Comparison to dense descriptors

In this part we present a first comparison between the proposed HMG descriptor and popular dense descriptors for action recognition: HOG, HOF, MBHx and MBHy [45]. The comparison is conducted in terms of accuracy and computational cost. All the descriptors benefit of the same settings and the same speed-up approaches presented above for the HMG descriptor. All dense descriptors are extracted using only the intensity information. All the computational time measurements for descriptor computation include also the loading time of the video and converting the frames to graylevel. For this set of experiments we use Fisher Vector (FV) [33] as encoding method, with the

common setting of 256 clusters. We choose FV for this set of experiments as this is a standard widely used encoding method for action recognition task, thus, the direct comparison with other approaches is straight forward.

The comparative results are presented in Table 1. Our approach of computing the motion information by applying a simple and efficient temporal filter does not affect significantly the computational cost as compared with the fast HOG descriptor. While the efficiency is preserved, in terms of accuracy our HMG descriptor outperforms with a large margin HOG, by 5.2 percentage points. This significant performance improvement while preserving the efficiency shows that the motion information captured by our descriptor is very discriminative for videos and can be considered as a good option for the applications based on video analysis, especially for those where the computational cost is crucially important. Remarkably, HMG outperforms even descriptors based on classical optical flow which are more demanding for computational cost. For instance, HMG outperforms HOF by 1.5 percentage points in terms of accuracy, moreover, the descriptor extraction for HMG runs with approximately 72 frames/second while HOF runs only at around 49 frames/second. This big difference in efficiency is due to the optical flow computation, which can take up to 50% of the cost for HOF extraction.

5.4 Feature Encoding

In this set of experiments we make the first comparison of our proposed encoding method, SD-VLAD, with the other standard approaches for creating a final representation that serves as input for a classifier.

In this part we compare our dense HMG descriptor with dense HOG, HOF, MBHx and MBHy for Bag-of-Visual-Words (BoVW) using three approaches for visual word assignment: k-means, hierarchical k-means (hk-means) and Random Forests (RF) [4]. In addition we use other three variations of BoVW: Fisher Vectors (FV) [33] and Vector of Locally Aggregated Descriptors (VLAD) [16], and the proposed method Shape Difference VLAD (SD-VLAD). For k-means and hk-means we use the implementation made available with VLFeat [46]. For both we create a codebook of 4,096 visual words. For hk-means we learn a hierarchical tree of depth 2 with 64 branches per node. RF are well-known for their speed, they are binary decision trees, learned in a supervised manner by randomly picking several descriptor dimensions at each node with several random thresholds. The split with the highest Entropy Gain is selected. We follow the recommendations of [43, 44, 45], using 4 binary decision trees of depth 10, which create a codebook of 4,096 visual words.

For FV we keep the codebook size of 256 clusters. We test VLAD representation with 256 and also with 512 visual words for making the comparison between super vector-based encoding methods more fair. For SD-VLAD we fix the codebook size to the standard 256 words. For VLAD and SD-VLAD we apply also intra-normalization L2 as explained in the previous section.

Table 2: Accuracy vs. processing time for different encoding methods and using several dense descriptors (results reported on the UCF50 dataset, best results are in bold).

	k-means	hk-means	RF	FV	VLAD (256)	VLAD (512)	**SD-VLAD**
HOG	0.731	0.720	0.718	0.762	0.712	0.731	**0.768**
HOF	0.789	0.779	0.738	0.799	0.810	0.815	**0.833**
MBHx	0.772	0.760	0.731	0.784	0.782	0.795	**0.800**
MBHy	0.783	0.774	0.750	0.792	0.799	0.814	**0.820**
HMG	0.781	0.759	0.735	0.814	0.805	0.822	**0.834**
sec/video	8.42	0.37	**0.05**	2.09	0.37	0.56	0.38
frame/sec	24	526	**3788**	94	532	352	513

The results for different encoding methods are presented in Table 2, which confirm that super-vector encoding methods give a better video representation than the other encoding approaches. The superiority of super-vector encoding methods is due to the fact that they capture information related to the mean and variance/standard deviation of the features and not only the membership information of the features to the clusters. Our HMG descriptor is very competitive for all the encoding methods, especially for super-vector encoding methods, which outperforms all the other descriptors, with an accuracy of 0.834 accuracy for SD-VLAD.

The computational cost for the encoding step is not dependent on the type of features, it depends on the number of visual words and the dimensionality of descriptors. As all our descriptors have the same dimensionality, we reported the computational cost for encoding a descriptor (can be any) with 72 dimensions. The RF approach for encoding step is by far the fastest and takes 0.05 seconds per video, however, the accuracy drops significantly for all descriptors related to the best encoding method for the performance. The results for hk-means represents a good trade-off between accuracy and computational efficiency. It can process the video at a frame rate of 526. When the speed is crucially important then RF is the best choice, encoding the features with 3,788 frames per second.

After these experiments we can take the conclusion that super-vector based encoding methods give the best performance. For SD-VLAD the most demanding part for the computational cost is the the codebook assignment and this is the reason that the efficiency of SD-VLAD is similar to the baseline VLAD256. The computation of an extra representation for SD-VLAD does not increase significantly the computational cost. For feature encoding, SD-VLAD represents the best trade-off between accuracy and computational efficiency, running at a frame rate of 513 frames/second. Regarding the encoding method, our goal is to find the best approach for the accuracy of the system and the method with best trade-off between computational cost and accuracy of the pipeline. Considering this, for the further experiments we use only super vector-based encoding methods and we perform some supplementary tests to establish the best approach for the encoding choice.

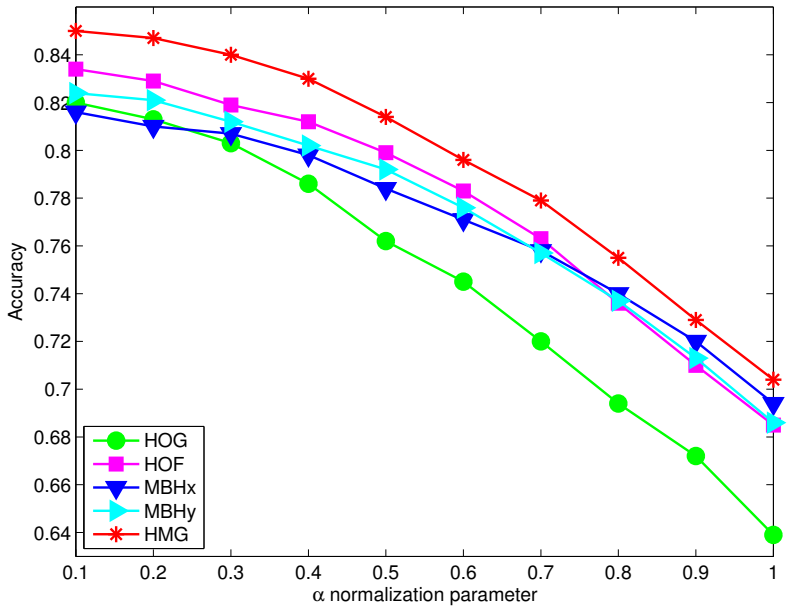

Fig. 5: Impact of the normalization parameter on the Fisher Vector performance for the UCF50 dataset.

5.4.1 Improving the performance

The post-processing step after the encoding method can boost the performance of the system by preparing the input for the classifier. After feature encoding, for the resulted vector of the video representation we apply before classification Power Normalization followed by L2-normalization ($||sign(x)|x|^{\alpha}||$, where $0 \leq \alpha \leq 1$ is the normalization parameter), we call this PNL2. The effect of this normalization is reducing the peaks within the vector. It is very important for the performance of the system that the values within the vector are not spread on a large interval, since high peaks will dominate the distances between the vectors. As the classifier receives as input the distances between the vectors, the peaks receive a higher weight and the other components of the vector will contribute less, in the end this may influence negatively the classifier output. The α parameter from PNL2 controls the level of penalization, by giving a smaller α, the large values are shrinked more and reduce the peaks within the vector.

We perform the α parameter tuning within the interval [0.1; 1], with the step 0.1. Fig. 5 presents the graphs with the impact of the normalization for FV. We can see that the accuracy is drastically affected by the α normalization parameter for PNL2. There is a continuous increase in performance of all the descriptors when choosing a smaller α. The $\alpha = 1$ actually means that only

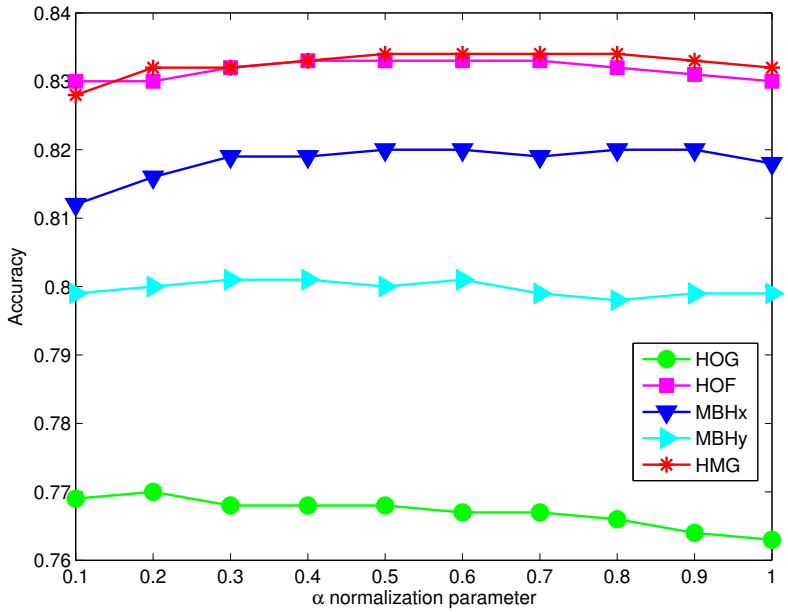

Fig. 6: Impact of the normalization parameter on the SD-VLAD performance for the UCF50 dataset.

L2 normalization is applied, and the performance boost between $\alpha = 1$ and $\alpha = 0.1$ is 18.1 percentage points on HOG, 14.9 on HOF, 12.2 on MBHx, 13.8 on MBHy and 14.5 for HMG. This considerable increase in performance for FV when PNL2 is applied with a small α is due to the fact that the resulted final vector after applying the encoding contains large peaks, having a large interval for the values. This is caused by the FV formulation which is built using two different formulas that provide in the end different intervals for the values. PNL2 with a small α helps in bringing the values in a smaller interval, reducing the peaks, and therefore, the distances between vectors are more reliable. For all the next experiments we set $\alpha = 0.1$ for PNL2 when using FV as encoding method.

Fig. 6 shows the parameter tuning for PNL2 when using SD-VLAD as encoding method. In this case the influence of α normalization parameter is not that radical. This is due to the fact that we apply intra-normalization L2 when encoding the features with SD-VLAD and therefore, the peaks within the vector are already reduced. However, if intra-normalization is not used during the encoding process then we recommend to use PNL2 with a very small α for the resulted vector, similar as for FV. For all the next experiments we keep the initial setting of PNL2 with $\alpha = 0.5$ for VLAD and SD-VLAD.

Table 3: Performance comparison between Euclidean distance (E. dist.) and inner product (inner p.) used for the assignment step during the encoding process (results reported on the UCF50 dataset, best results are in bold).

	VLAD256		VLAD512		**SD-VLAD**	
	E. dist.	inner p.	E. dist.	inner p.	E. dist.	inner p.
HOG	0.712	**0.717**	0.731	**0.735**	0.768	**0.772**
HOF	0.810	**0.812**	0.815	**0.823**	0.833	**0.835**
MBHx	0.782	**0.789**	0.795	**0.801**	0.800	**0.802**
MBHy	0.799	**0.802**	0.814	**0.816**	0.820	**0.822**
HMG	0.805	**0.809**	0.822	**0.823**	0.834	**0.834**
sec/video	0.37	**0.20**	0.56	**0.32**	0.38	**0.23**
frame/sec	532	**971**	352	**620**	513	**848**

5.4.2 Improving the efficiency

From the previous experiments we can take the conclusion that FV and SD-VLAD provide the best results. SD-VLAD is the choice for a trade-off between the computational cost and accuracy. One of the reasons for which VLAD and SD-VLAD are more efficient than FV is the assignment approach. The VLAD based encoding methods use hard assignment while FV uses soft assignment making this step more demanding for the computational cost.

VLAD and SD-VLAD are the choices for a trade-off between accuracy and computational cost. In this set of experiments we investigate how we can improve the computational cost for VLAD and SD-VLAD methods without affecting negatively the accuracy. The assignment step is the most demanding part for the computational time of an encoding method. To decide to which centroid a feature belongs, it is necessary to compute the distance to all centroids of the codebook and to assign the feature to the closer visual word. We evaluated two approaches to compute the distances. First approach is to use the standard Euclidean distance. After we compute the distances we take the minimum value to decide to which visual word the feature belongs. The second approach is to use inner product to compute the distances. By making unit length for both vectors for which we compute the distance we can apply inner product operation as a measurement for the distance. In this case, to decide to which centroid a feature belongs we take the maximum value among the computed inner products. If all feature vectors have the same length (e.g. unit length), then taking the maximum inner product is equivalent to taking the minimum Euclidean distance.

Table 3 presents the comparison results between Euclidean distance and the inner product used for the assignment step within the encoding process. We can see that the encoding method is much faster when using the inner product than in the case of Euclidean distance, being able to improve the computational cost from 513 to 848 frames per second for SD-VLAD. The slight improvement of the accuracy is the effect of applying L2 norm for making unit length when we compute the inner product. For all next experiments we use the inner product for the assignment step when VLAD-based methods are used for encoding.

Table 4: A closer comparison of the computational cost and the accuracy with the direct competitors on the encoding approach (results reported on the UCF50 dataset, best results are in bold).

	HOG	HOF	MBHx	MBHy	HMG	sec/video	frame/sec
VLAD256	0.717	0.812	0.789	0.802	0.809	0.20	971
VLAD512	0.735	0.823	0.801	0.816	0.823	0.32	620
H-VLAD [31]	0.755	0.825	0.800	0.809	0.820	0.38	520
SD-VLAD	**0.772**	**0.835**	**0.802**	**0.822**	**0.834**	0.23	848

5.4.3 Closer comparison of SD-VLAD with direct competitors

Our proposed encoding method, SD-VLAD, provides the best trade-off between the accuracy and computational cost. We provide a direct comparison of our method with the closest approaches for the encoding step. The closest approaches to ours are represented by VLAD [16] and the work in [31]. The authors in [31] use high-order statistics for VLAD and dictionary learning to boost the performance. They use three order statistics, which makes the final vector three times bigger than VLAD.

The goal of this comparison is to discover which formulas are better to compute the final representation that will serve as input for the classifier. Of course, in this comparison all the methods benefit of the same settings including using intra-normalization L2 and inner product for the assignment step. As in the previous experiments, the computational time reported includes also the time for making the unit length for the vectors needed for inner product and also the time for intra-normalization. Furthermore, besides the fact that the vocabulary is build in the same way for all approaches, for VLAD256, H-VLAD [31] and for SD-VLAD we use also the same codebook (this is straight forward as we use for all 256 visual words for the codebook size), which makes the comparison more reliable as the randomness of constructing the codebook is excluded.

Table 4 presents the efficiency and performance comparison for all five dense descriptors. In terms of accuracy our approach outperforms the other methods for all descriptors by a large margin. Furthermore, the dimensionality of our final vector is 33% less than the representation in [31], as we concatenate two order information and they concatenate three order statistics. In general, high order information leads to a good improvement compared to the original VLAD. In terms of computational cost, VLAD256 is the fastest. VLAD512 and H-VLAD are almost twice as slow with moderate accuracy improvements. In contrast, SD-VLAD is almost as fast as VLAD256 yet gives the highest accuracies of all encoding methods.

For a better understanding of the performance contribution, Table 5 presents the accuracy comparison of each representation component of SD-VLAD with the similar approach [31]. We report the comparison for first- and second-order statistics, and as we do not consider the third-order statistics we just report the performance of [31] in this case. We can see that our formulation for first-order statistics gives slightly better results than the approach in [31], which

Table 5: A deeper comparison: for each component. We report the performance comparison for each part of our formulation of the encoding method with the approach [31] (results reported on the UCF50 dataset, best results are in bold, in italic are represented the results for the third-order statistics of [31], as we do not use the third-order information, it is not possible a direct comparison).

	HOG	HOF	MBHx	MBHy	HMG
H-VLAD [31] first-order (=VLAD256)	0.717	**0.812**	0.789	0.802	0.809
SD-VLAD first-order	**0.721**	**0.812**	**0.793**	**0.805**	**0.816**
H-VLAD [31] second-order	0.724	0.809	0.774	0.782	0.786
SD-VLAD second-order	**0.760**	**0.822**	**0.785**	**0.796**	**0.810**
H-VLAD [31] third-order	*0.606*	*0.709*	*0.708*	*0.705*	*0.700*

Table 6: Comparison to IDT [49] in terms of accuracy and computational cost on the UCF50 dataset.

	HOG	HOF	MBH	HMG	sec/video	frames/sec
IDT [49]	0.826	0.851	0.889	-	50.5	3.9
dense	0.820	0.834	0.832	0.850	10.9	18.0

for their first order statistics represents the original VLAD formulation. This slightly improvement is mainly due to the standard deviation integration from Equation (5).

The main difference in performance is reflected for the second-order statistics, where SD-VLAD outperforms [31] consistently for all five descriptors. This consistent improvement for the second-order of VLAD-DV is mainly due to the consideration of the global mean of the cluster instead of the local mean of the descriptors when computing the standard deviation for the local descriptors. Remarkably that the shape difference formulation of our SD-VLAD provides only by itself a better representation than VLAD. For example, VLAD obtains for HOG an accuracy of 0.717, while the shape difference for SD-VLAD (from Equation (7)) obtains an accuracy of 0.760, and in this case the vector lengths are equal and the computational costs are similar. Therefore, the shape difference for SD-VLAD can replace directly VLAD in many situations. After this set of experiments we can take the conclusion that SD-VLAD is a proper method for a trade-off between the efficiency and the performance, thus, for the next experiments we will continue with FV as the choice when accuracy is crucially important and with SD-VLAD for the trade-off.

5.5 Comparison with Improved Dense Trajectories

The Improved Dense Trajectories (IDT) [49] represents a state-of-the-art video representation approach. We compare our descriptor extraction approach with IDT in terms of accuracy and of computational efficiency. As in [49] where there are reported the results for FV with 256 clusters, we perform the comparison with our dense approach using the same encoding method. As the code in [49] provides four main descriptors (HOG, HOF, MBHx and MBHy), for a fair comparison we compare its extraction time with dense extraction time

Table 7: Trade-off between frame sampling rate and accuracy for the UCF50 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates. †Frames/second is measured in terms of the total number of frames of the original video, not in terms of how many frames are actually processed during descriptor extraction.

$\binom{\text{frames/block}}{\text{sample rate}}$		$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.772	0.768	0.770	0.775
	FV	0.820	0.817	0.814	0.820
	sec/video	2.67	1.54	1.15	0.78
	frame/sec†	73.50	127.07	170.99	250.79
HOF	SD-VLAD	0.835	0.823	0.812	0.798
	FV	0.834	0.820	0.817	0.799
	sec/video	4.03	2.28	1.68	1.06
	frame/sec†	48.61	86.04	116.27	184.73
MBHx	SD-VLAD	0.802	0.793	0.792	0.778
	FV	0.816	0.806	0.797	0.779
	sec/video	4.37	2.45	1.80	1.12
	frame/sec†	44.80	80.03	108.96	174.60
MBHy	SD-VLAD	0.822	0.816	0.811	0.796
	FV	0.824	0.819	0.814	0.794
	sec/video	4.37	2.44	1.80	1.12
	frame/sec†	44.84	80.27	108.67	174.37
HMG	SD-VLAD	0.834	0.835	0.835	0.822
	FV	0.850	0.845	0.843	0.829
	sec/video	2.73	1.59	1.19	0.80
	frame/sec†	71.73	123.47	164.17	245.45

also for four descriptors: HOF, MBHx, MBHy and HMG. Notice that dense HOG and HMG have similar computational time, so it is not relevant for time measurement which one is selected. The comparison with IDT is presented in Table 6. For the computational efficiency the dense approach outperforms by a large margin IDT, being 4.6 times faster. The dense approach is able to process a video with 18 frames per second while IDT can process only 3.9 frames per second. Even though [49] provides a fast code in C++, the Matlab implementation for dense descriptors is considerably less demanding for the computational cost due to several factors. First, IDT uses a more complicate algorithm to extract the descriptors and furthermore, their approach improves the accuracy by canceling the camera motion. For doing this it is necessary to compute two times the optical flow, which makes the algorithm more demanding for computational efficiency. Another reason is that the dense descriptors are computed more efficiently, being able to reuse the blocks for many times and without the need to compute any trajectories. Very interesting is the fact that our HMG descriptor is able to compete even with HOF from IDT, with almost similar performance of 0.85.

Table 8: Trade-off between frame sampling rate and accuracy for the UCF101 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates.

$\binom{\text{frames/block}}{\text{sample rate}}$		$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.653	0.658	0.662	0.664
	FV	0.708	0.719	0.721	0.722
HOF	SD-VLAD	0.740	0.725	0.719	0.697
	FV	0.741	0.729	0.719	0.700
MBHx	SD-VLAD	0.709	0.703	0.696	0.681
	FV	0.729	0.718	0.708	0.688
MBHy	SD-VLAD	0.728	0.723	0.717	0.701
	FV	0.742	0.731	0.723	0.707
HMG	SD-VLAD	0.747	0.753	0.745	0.743
	FV	0.771	0.780	0.771	0.757

Table 9: Trade-off between frame sampling rate and accuracy for the HMDB51 dataset. We keep video volumes from which descriptors are extracted the same for all sampling rates.

$\binom{\text{frames/block}}{\text{sample rate}}$		$\binom{6}{1}$	$\binom{3}{2}$	$\binom{2}{3}$	$\binom{1}{6}$
HOG	SD-VLAD	0.367	0.380	0.378	0.380
	FV	0.406	0.399	0.390	0.399
HOF	SD-VLAD	0.433	0.420	0.411	0.392
	FV	0.433	0.424	0.408	0.388
MBHx	SD-VLAD	0.399	0.395	0.393	0.376
	FV	0.407	0.400	0.397	0.371
MBHy	SD-VLAD	0.395	0.405	0.397	0.384
	FV	0.405	0.402	0.401	0.377
HMG	SD-VLAD	0.418	0.417	0.408	0.409
	FV	0.440	0.438	0.435	0.414

5.6 Frame subsampling

Subsequent video frames contain similar information. In this set of experiments we investigate the impact on the accuracy results when frames are skipped, with the goal of speeding up the feature extraction process. We evaluate when skipping 2, 3 and 6 frames. The modality of frame subsampling is similar as in [45]. For a fair comparison, the features describe the same video volume for the process of subsampling frames. For instance, if we sample every 2 frames, our baseline for the size of the block of 8 by 8 pixels by 6 frames is changing to 8 by 8 by 3 frames; for skipping 3 frames we have only 8 by 8 pixels by 2 frames; and when sampling every 6 frames the block size became 8 by 8 pixels by 1 frame.

The results for frame subsampling are presented in Table 7 for UCF50 dataset, in Table 8 for UCF101 and in Table 9 for HMDB51. In Table 7 we present the computational cost for the descriptor extraction. We reported the computational cost measurements only for UCF50 dataset in Table 7, as the video resolution is similar for all tree datasets, thus, the numbers reported for

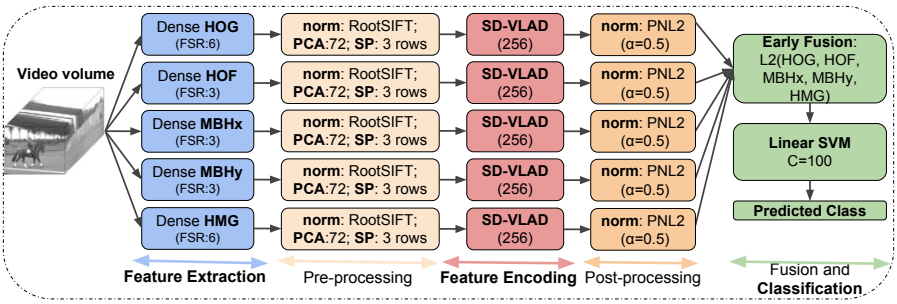

Fig. 7: The pipeline for real-time video classification. This framework can process the video at a speed of 39 frames/second and yields an accuracy of 0.845 on UCF50 dataset, 0.767 on UCF101 and 0.470 on HMDB51.

the measurements for the frames per second are valid also for UCF101 and HMDB51 datasets.

By subsampling frames the computational cost is significantly improved, making the pipeline more efficient. HOG descriptor is not negatively affected by skipping frames because this descriptor captures the appearance information and subsequent video frames contain similar information. Therefore, for HOG descriptor we can skip frames with a step of 6 without loosing accuracy for both FV and SD-VLAD, being able to process more than 250 frames from the video per second. For the descriptors based on optical flow a frame sampling rate of 3 gives a good trade-off, improving considerably the computational cost. HMG with SD-VLAD can have a frame sampling rate of 6 without decreasing significantly the accuracy, and a frame sampling rate of 2 almost without affecting accuracy.

5.7 Real-time video classification

For a real-time video classification system we propose the framework illustrated in Fig. 7. We use dense descriptors due to their efficiency, additionally we speed-up the pipeline by using a frame sampling rate (FSR) of 6 (thus, frames/block=1) for HOG and HMG and a FSR of 3 (thus, frames/block=2) for HOF, MBHx add MBHy. It is recommended that the FSR to be equal for all descriptors based on optical flow for the reason of computing the optical flow only once and use it for all of them. After we extract the descriptors, each of them follows its separate path through the pipeline.

We initially normalize each descriptor using RootSIFT [1], then we apply PCA to reduce their dimension by a factor of 2 (from 144 dimensions to 72). Before encoding we apply a spatial pyramid representation (SP), dividing (in 3 rows) the descriptors based on their location in the frame, in bottom, middle and top part of the frame. Then for each group of the descriptors we apply our efficient encoding method, SD-VLAD, with 256 visual words. The output vector of the encoding method is normalized using PNL2 with $\alpha = 0.5$. In

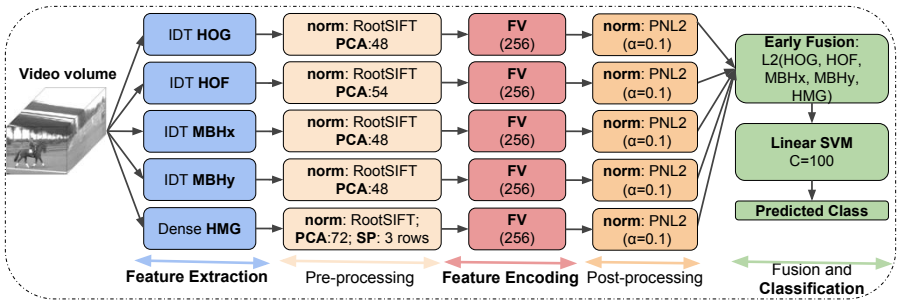

Fig. 8: The pipeline for accurate video classification. This framework yields an accuracy of 0.930 on UCF50 dataset, 0.881 on UCF101 and 0.610 on HMDB51, but can process only 3 frames/second.

Table 10: Comparison to the state-of-the-art.

UCF50 (Acc.)		UCF101 (Acc.)		HMDB51 (Acc.)	
Kliper-Gross et al. [20]	0.727	Karpathy et al. [19]	0.654	Jain et al. [15]	0.521
Solmaz et al. [39]	0.737	Wang et al. [50]	0.859	Oneata et al. [29]	0.548
Reddy et al. [35]	0.769	Wang et al. [48]	0.860	Park et al. [30]	0.562
Uijlings et al. [44]	0.809	Peng et al. [31]	0.877	Wang et al. [49]	0.572
Uijlings et al. [45]	0.818	Peng et al. [32]	0.879	Sun et al. [42]	0.591
Wang et al. [47]	0.856	Simonyan et al. [36]	0.880	Simonyan et al. [36]	0.594
Wang et al. [49]	0.912	Sun et al. [42]	0.881	Peng et al. [31]	0.598
Wang et al. [48]	0.917	Park et al. [30]	**0.891**	Wang et al. [48]	0.601
Peng et al. [32]	0.923	Bilen et al. [3]	**0.891**	Bilen et al. [3]	**0.652**
HMG + iDT	**0.930**	HMG + iDT	0.881	HMG + iDT	0.610

the end we perform an early fusion by concatenating all five descriptors, then we apply L2 normalization on the final representation for making unit length. After we compute the distances we feed them to a linear SVM ($C = 100$) to get the final result, the predicted class for a video.

The pipeline from Fig. 7 is able to obtain more than real-time processing rate, being capable to run with 39 frames per second and to obtain an accuracy of 0.845 for UCF50 dataset, 0.767 on UCF101 and 0.470 on HMDB51.

5.8 Comparison to state-of-the-art

When accuracy is crucially important for the application we recommend using Fisher Vector (with 256 clusters) for feature encoding and combining our HMG descriptor with IDT descriptors. The HMG descriptor is used in this case without skipping frames. Our pipeline for accurate action recognition can be visualized in Fig. 8. We extract all the descriptors of IDT (HOG, HOF, MBHx and MBHy) with the default settings provided in [49]. We perform early fusion between HMG and IDT by concatenating all features. For all features we apply separately, before early fusion, PNL2 normalization with $\alpha = 0.1$. This combination improves the accuracy from 0.912 reported in [49] (for IDT) to 0.930 for UCF50 dataset, from 0.859 [50] to 0.881 for UCF101 and from 0.572

[49] to 0.610 for HMDB51 . This significant improvement in performance shows that our HMG descriptor brings complementary information for IDT and can be used to boost the performance of the system. However, using IDT and FV makes the pipeline more demanding for the computational cost, enabling to process only around 3 frames per second, making it not suitable for real-time applications.

Table 10 presents the performance comparison of our accurate pipeline from Fig. 8 with state-of-the-art approaches. The proposed combination for accuracy between HMG and IDT obtains state-of the art results on UCF50 and competitive results on UCF101 and HMDB51. Our framework outperforms all the methods based on hand-crafted features, including the recent work of [48], which considers as encoding method the spatial FV [21] together with spatio-temporal pyramid [24]. Our results are better than [48] which considers a hybrid representation by combining two different representations. While the approaches based on learned features (deep learning) such as [3, 30] obtain state-of-the-art results, remarkably that our pipeline is able to outperform many other well-known approaches based on deep learning such as [19, 36].

6 Conclusion

In this work we propose an efficient pipeline for action recognition. As two critical factors for a powerful action recognition pipeline are represented by descriptor extraction and descriptor encoding steps, we propose a new solution for both of them. We introduce in this paper a new descriptor, Histograms of Motion Gradients (HMG), that captures motion information without the need of computing the optical flow, which obtains very competitive results while achieving a low computational complexity. Regarding the descriptor encoding we propose Shape Difference for VLAD (SD-VLAD). This approach captures information regarding the distribution shape of the descriptors, providing the best trade-off between computational cost and accuracy.

Based on our solutions for descriptor extraction and encoding, we propose an accurate and a real-time video classification pipeline. We test our approach on the challenging datasets: UCF50, UCF101 and HMDB51, being able to outperform well-know competitive approaches on this task. For the future work we will focus on further improvement of computational efficiency by using parallel computation. Furthermore, for new features we consider the idea of learning a new representation by training a convolutional neural network with the resulted motion frames obtained after temporal derivation, instead of using an optical flow method which is very expensive for the computational cost.

Acknowledgments. Part of this work was funded under research grant PN-III-P2-2.1-PED-2016-1065, agreement 30PED/2017, project SPOTTER. This material is based in part on work supported by the National Science Foundation (NSF) under grant number IIS-1251187.

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. H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.
4. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
5. T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. 2004.
6. T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *TPAMI*, 33(3):500–513, 2011.
7. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
8. N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. In *ECCV*. 2006.
9. C. R. de Souza, A. Gaidon, E. Vig, and A. M. López. Sympathy for the details: Dense trajectories and hybrid classification architectures for action recognition. In *ECCV*, 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. G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. 2003.
14. B. K. Horn and B. G. Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.
15. M. Jain, H. Jégou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.
16. 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.
17. F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.
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. O. Kliper-Gross, Y. Gurovich, T. Hassner, and L. Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*,

2012.
21. J. Krapac, J. Verbeek, and F. Jurie. Modeling spatial layout with fisher vectors for image categorization. In *ICCV*, 2011.
22. H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.
23. I. Laptev. On space-time interest points. *IJCV*, 64(2-3):107–123, 2005.
24. I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.
25. S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
26. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
27. B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.
28. 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*, 2016.
29. D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.
30. 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.
31. X. Peng, L. Wang, Y. Qiao, and Q. Peng. Boosting vlad with supervised dictionary learning and high-order statistics. In *ECCV*. 2014.
32. X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv:1405.4506*, 2014.
33. F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*. 2010.
34. S. Poularakis, K. Avgerinakis, A. Briassouli, and I. Kompatsiaris. Computationally efficient recognition of activities of daily living. In *ICIP*, 2015.
35. K. K. Reddy and M. Shah. Recognizing 50 human action categories of web videos. *Machine Vision and Applications*, 24(5):971–981, 2013.
36. K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.
37. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
38. 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.
39. B. Solmaz, S. M. Assari, and M. Shah. Classifying web videos using a global video descriptor. *Machine vision and applications*, 2013.
40. 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.

41. 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.
42. L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi. Human action recognition using factorized spatio-temporal convolutional networks. In *ICCV*, 2015.
43. J. R. Uijlings, A. W. Smeulders, and R. J. Scha. Real-time visual concept classification. *TMR*, 12(7):665–681, 2010.
44. J. R. R. Uijlings, I. C. Duta, N. Rostamzadeh, and N. Sebe. Realtime video classification using dense hof/hog. In *ICMR*, 2014.
45. 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*, 2015.
46. A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *ACM Multimedia*, 2010.
47. 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.
48. H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A robust and efficient video representation for action recognition. *IJCV*, 2015.
49. H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.
50. H. Wang and C. Schmid. Lear-inria submission for the thumos workshop. In *ICCV Workshop*, 2013.
51. H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009.
52. L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*, 2016.
53. 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.
54. C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. 2007.
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.