Large-scale Web Video Event Classification by use of Fisher Vectors

Anonymous WACV submission

Paper ID 7

Abstract

Event recognition has been an important topic in computer vision research due to its many applications. However, most of the work has focused on videos taken from a fixed camera, known environments and basic events. Here, we focus on classification of unconstrained, web videos into much higher level activities. We follow the approach of constructing fixed length feature vectors from local feature descriptors for classification using an SVM. Our key contribution is the study of the utility of Fisher Vector representation in improving results compared to the conventional Bag-of-Words (BoW) approach. Such coding has shown to be useful for static image classification in the past but not applied to video categorization. We perform tests on the challenging NIST TRECVID Multimedia Event Detection (MED) dataset, which has thousand hours of unconstrained user generated videos; our approach achieves as much as 35% improvement over the BoW baseline. We also offer an analysis of possible causes of such improvements.

1. Introduction

Recognition of events in videos is important for many applications and has been receiving increasing attention in computer vision research in recent years. Most of this work is focused on analysis of videos in a known environment with a fixed camera and the events of interests are the basic ones such as running, jumping and bending, *e.g.* see [15]. There is also another important class of videos which consists largely of user captured content uploaded to the Internet: in such videos, quality is variable, camera is likely in motion and there are large variations in the background environments. Our goal in this research is to provide automatic classification of such videos as belonging to classes defined by the large scale events taking place in them, *e.g.* a *wedding video* or one where someone performs a *board trick*. There are many applications of such classification including sharing and browsing of large video databases.

To promote research in large scale video categorization, the National Institute of Standards and Technology (NIST) has been sponsoring annual TRECVID evaluations [1]. These evaluations provide large scale data collections (courtesy of Linguistics Data Consortium or LDC of University of Pennsylvania), a list of event classes to be annotated and procedures to evaluate the results. We focus our experiments on data provided by these evaluations, in particular, the datasets known as MED11 and MED12 (these datasets also include speech content which we ignore for the study described here).

A variety of approaches have been developed for the constrained environment activity analysis. These include use of statistics of local features as well as analysis of semantic entities by detecting and tracking, actors, their pose and the objects they interact with. At the high level, the web video activities are also characterized by actors and objects; however, detection of such entities in unconstrained environments is extremely difficult and inference of higher level activities is also challenging. Hence, focus has been on use of statistical techniques using local features; some recent results on MED11 data have been reported in [16][12].

The statistical techniques typically have four stages: low-level feature extraction, feature encoding, classification and fusion of results (from multiple channels if available). At the feature extraction stage, local image or video patches are selected, either densely or via some saliency selection process; descriptions are then computed for each patch. Feature detection and description techniques build on ideas developed for object recognition in still images but also incorporate the use of temporal dimension. Interestingly, although it may be expected that sparse salient feature points will be more robust, experiments show that dense features perform better for more complex videos [19]. Feature encoding stage turns sets of local features into fixed length vectors; this is usually accomplished by vector quantization of feature vectors and building histograms of *visual codewords*; this is commonly known as Bag-of-Word (BoW) encoding [4]. Variations include *soft quantization* where distance from a number of codewords is considered [17][20]. Feature vectors are used to train classifiers (typically χ^2 kernel SVMs). Several late fusion strategies can be used to combine the classification results of different

low-level features; such fusion typically shows consistent improvement[12].

Impressive results have been obtained on a large dataset (MED11) that evaluates classification accuracy on thousands of videos with very diverse characteristics for 10 high level events. [16] offers a very systematic analysis of the performance of different features and their combinations. Our aim in this paper is to increase the discriminative power of these features by use of *difference coding* techniques (of which Fisher Vector is one).

Fisher kernel[5] was proposed to utilize the advantages of generative model in a discriminative framework. The basic idea is to represent a set of data by gradient of its loglikelihood with respect to model parameters, and measure the distance between instances with Fisher kernel. The fixed-length representation vector is also called a Fisher Vector. For local features extracted from videos, it's natural to model their distribution as mixtures of Gaussian(GMM), forming a soft codebook. With GMM, the dimension of Fisher Vector is linear in number of mixtures and local feature dimension. Fisher vector has been applied to static image classification[13] and indexing[6], showing significant improvements over BoW methods. In this paper, we show that the Fisher Vector also improves performance greatly over BoW for video activity classification. We also provide an analysis of what may be the cause for observing such improvements.

Our contribution is thus three-fold: first, we propose a video event classification framework by using Fisher Vector encoding; second, we give an analysis of several desired properties Fisher Vector for this task; and finally, we provide a series of evaluations on the choice of Fisher Vector parameters on a complicated large scale web video dataset.

The remainder of this paper is organized as follows: Related work on video event classification is discussed in section 2. In section 3, we describe our video classification framework with Fisher Vector encoding, and gives an analysis on the benefits of Fisher Vector. Finally, we show experiment results on Fisher kernel, its variation and BoW baseline in section 4, and conclude the paper.

2. Related Work

Since a video is a sequence of image frames, low level features designed for images can all be applied to video classification, such as SIFT[11]. To take motion information into account, there are several innovations in motion related features. STIP[8], for example, extends 2D feature detector to 3D space. An evaluation by Wang *et al*. [19] shows that dense features may have better performance compared to those filtered by detectors. In this category, the dense trajectory features[18] track local points to obtain short tracklets, and describe the volumes around each tracklet. A recent evaluation[16] on a complex web video dataset shows that a late fusion of different features almost always helps.

Besides low level features, some argue that mid-level or high-level features can provide better performance. A common approach is to build pre-trained, fast classifiers for a set of mid-level or high-level concepts, and encode the classifier responses as video representation. Object Bank[10], for example, uses a list of more than 170 object concepts. Its counterpart in event classification, Action Bank[14], has 205 template action detectors.

A number of papers using various combinations of features and classifiers on the MED11 dataset can be found at [2].

Difference coding techniques were first developed in machine learning literature and then applied to image classification tasks as described earlier in Section 1. To the best of our knowledge, these techniques have not been applied to motion features in prior work.

3. Classification Framework

In this section, we will describe the video event classification framework, including feature extraction, Fisher Vector encoding, postprocessing and classification.

3.1. Local Feature Extraction

We perform experiments with both a sparse local feature and a dense local feature. We use Laptev's Space-Time Interest Points (STIP)[8] as the sparse feature, each interest point is described by histograms of gradients (HoG) and optical flow (HoF) of its surrounding volume. For dense feature, we choose Wang *et al*.'s Dense Trajectory (DT)[18] features; the descriptor includes shape of trajectory, HoG, HoF and motion boundaries histograms (MBH). These choices are made based on the good performance of each in their category.

When environment is constrained and camera is fixed, sparse features are likely to select robust features that are highly correlated to events of interest. However, web videos are usually taken in the wild, and in most cases camera motion is unknown. It is likely that camera motion causes many feature points to originate from the static background (See Figure 1). Dense features do not suffer from feature point selection issues, but they treat foreground and background information equally, which can also be a source of distraction.

3.2. Fisher Vector Encoding

One key idea usually associated with local features is that of a visual words codebook, obtained by clustering feature points and quantizing the feature space. A set of feature points can then be represented by a fixed length histogram. However, some feature points may be far from any visual word, to compensate for this, Gemert *et al*. propose a soft

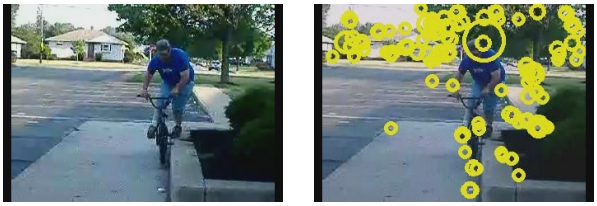

Figure 1. Camera motion can make lots of sparse feature points fall into background. Left is a frame in a bike trick video, right shows the detected feature points by STIP

assignment of visual words[17], but each codeword is still only modeled by its mean.

3.2.1 Fisher Vector under Gaussian Mixture Model

Fisher Vector concepts have been introduced in preivous work [5] and applied to image classification and retrieval by several authors [13][6]. We repeat their formulation below for ease of reading and making this paper self-contained.

According to [5], suppose we have a generative probability model $P(X|\theta)$, where $X = \{x_i|i = 1, 2, \ldots, N\}$ is a sample set, and θ is the set of model parameters. We can map X into a vector by computing the gradient vector of its loglikelihood function at the current θ:

$$F_X = \nabla_\theta \log P(X|\theta) \tag{1}$$

F_X is a Fisher Vector, it can be seen as a measurement of the direction to make θ fit better to X. Since $|\theta|$ is fixed, the dimensions of Fisher Vector for different X are the same. This makes F_X a suitable alternative to represent a video with its local features.

GMM has the form

$$P(X|\theta) = \sum_{i=1}^{K} w_i \mathcal{N}(X; \mu_\mathbf{i}, \mathbf{\Sigma_i}) \tag{2}$$

where K is the number of clusters, w_i is the weight of the ith cluster, and $\mu_\mathbf{i}$, $\mathbf{\Sigma_i}$ are the mean and covariance matrix of the ith cluster.

As the dimension of Fisher Vector is the same as the number of parameters, diagonal covariance matrices are usually assumed to simplify the model and thus reduce the size of Fisher Vector. Denote $\mathcal{L}(X|\theta)$ as the loglikelihood function, the dth dimension of $\mu_\mathbf{i}$ as μ_i^d, the dth diagonal element of $\mathbf{\Sigma_i}$ as $(\sigma_i^d)^2$, local feature dimension as D and the total number of feature points as T. By assuming that each local feature is independent, the Fisher Vector F_X of feature points set X is

$$F_X = \left[\frac{\partial \mathcal{L}(X|\theta)}{\partial \mu}, \frac{\partial \mathcal{L}(X|\theta)}{\partial \mathbf{\Sigma}}\right] \tag{3}$$

$$\frac{\partial \mathcal{L}(X|\theta)}{\partial \mu_\mathbf{i}^\mathbf{d}} = \sum_{t=1}^{T} \gamma_t(i) \left\{\frac{x_t^d - \mu_i^d}{(\sigma_i^d)^2}\right\} \tag{4}$$

$$\frac{\partial \mathcal{L}(X|\theta)}{\partial \sigma_\mathbf{i}^\mathbf{d}} = \sum_{t=1}^{T} \gamma_t(i) \left\{\frac{(x_t^d - \mu_i^d)^2}{(\sigma_i^d)^3} - \frac{1}{\sigma_i^d}\right\} \tag{5}$$

Here, $\gamma_t(i)$ is the probability of feature point x_t belongs to the ith cluster, given by

$$\gamma_t(i) = \frac{w_i \mathcal{N}(x_t; \mu_\mathbf{i}, \mathbf{\Sigma_i})}{\sum_{j=1}^{K} w_j \mathcal{N}(x_t; \mu_\mathbf{j}, \mathbf{\Sigma_j})} \tag{6}$$

The dimension of F_X is $2KD$. The first term, $\frac{\partial \mathcal{L}(X|\theta)}{\partial \mu}$, is composed of first order differences of feature points to cluster centers. The second term, $\frac{\partial \mathcal{L}(X|\theta)}{\partial \mathbf{\Sigma}}$, contains second order terms. Both of these are weighted by the covariances and soft assignment terms.

Fisher Vector with GMM can be seen as an extension of BoW[7]. Actually, it accumulates the relative position to each cluster center, and models codeword assignment uncertainty, which has shown to be beneficial for BoW encoding[17].

3.2.2 Non-probabilistic Fisher Vector

In [6], the authors give a non-probabilistic approximation of Fisher Vector, called Vector of Locally Aggregated Descriptors (VLAD). It uses K-Means clustering to get a codebook, each value in VLAD is computed as

$$v_i^d = \sum_{x_t : NN(x_t) = i} x_t^d - \mu_i^d \tag{7}$$

Compared with Fisher Vector, VLAD drops the second order terms, and assumes uniform covariance among all dimensions. It also assigns each feature point to its nearest neighbor in the codebook. The feature dimension is KD.

3.2.3 Comparison with BoW

The basic idea of both BoW and Fisher Vector is to map feature point set X into a fixed dimension vector, from which the distribution in the original feature space can be reconstructed approximately. However, there are also several key differences discussed below:

First, BoW uses a hard quantization of feature space by KMeans, where each cluster has the same importance and is described by its centroid only. Meanwhile, Fisher Vector assumes GMM is the underlying generative model for local features. Although modifications to BoW can help

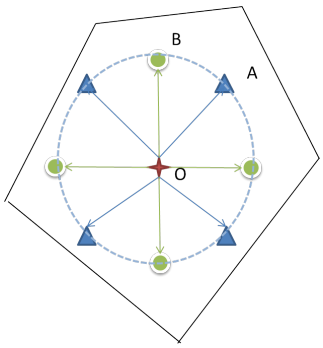

Figure 2. Suppose O is both the mean of a Gaussian mixture and a centroid of KMeans. A and B contribute the same under BoW but differently under Fisher Vector.

it capture more information, such as assigning different weights to codewords (term frequencyCinverse document frequency) and soft assignment of codewords[17], GMM incorporates them naturally.

Secondly, in Fisher Vector, local features' contribution to a Gaussian mixture depends on their relative position to the mixture center. In Figure 2, suppose we have a trained GMM for Fisher Vector as well as a trained visual codebook for BoW, and O happens to be both the mean of a mixture in GMM and the centroid of a codeword. Given two points A and B to be coded, as their distances to O are the same but $\vec{AO}$ and $\vec{BO}$ are different, they contribute the same to the codeword in BoW but differently to the mixture in Fisher Vector.

Finally, let X be separated into two sets X_r and X_b, where X_b contains the points that fit the GMM model well, we have

$$\frac{\partial \mathcal{L}(X|\theta)}{\partial \theta} = \frac{\partial \mathcal{L}(X_r|\theta)}{\partial \theta} + \frac{\partial \mathcal{L}(X_b|\theta)}{\partial \theta} \tag{8}$$

By definition, $\frac{\partial \mathcal{L}(X_b|\theta)}{\partial \theta} \approx 0$, so

$$\frac{\partial \mathcal{L}(X|\theta)}{\partial \theta} \approx \frac{\partial \mathcal{L}(X_r|\theta)}{\partial \theta} \tag{9}$$

Since θ is inferred by training on general data, which is likely to be dominated by background features, above implies that Fisher Vector can suppress the part of data that fit the general model well. Figure 2 also gives an illustration, the triangles and circles are feature points taken from two different videos, though their positions are different, they fit the model perfectly thus have no overall influence on the Fisher Vector.

3.3. Postprocessing and Classification

Both BoW and Fisher Vector drop spatial and temporal information of the feature points. However, sometimes

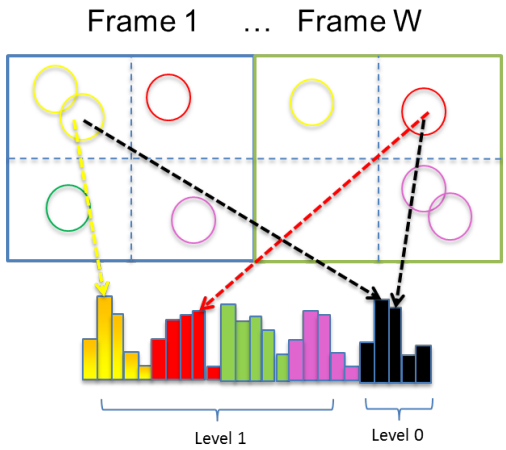

Figure 3. Spatial-Temporal Pyramids. For pyramid level i, video is divided into 2^i slices along timeline, and 2^i by 2^i blocks for each frame.

spatial and temporal structures can be useful for classification. Lazebnik *et al*. proposed to build spatial pyramids to preserve approximate location information for image classification task [9]. Here, we use a similar approach, but taking temporal information into account. At pyramid level video i, video is divided into 2^i slices along timeline, and 2^i by 2^i blocks for each frame. Suppose there are P_s spatial pyramid levels, and P_t temporal pyramid levels, the total number of sub-volumes is $(4^{P_s}-1)(2^{P_t}-1)/3$. Encoding is performed for local features in each sub-volume, the final representation is a concatenation of all vectors.

We then normalize each dimension of F_X by a power normalization:

$$f(x_i) = sign(x_i)|x_i|^\alpha, 0 \leq \alpha \leq 1 \tag{10}$$

Power normalization step is important when a few dimensions have large values and dominate the vector, the normalized vector will become flatter as α decreases. It is suggested by [7] for image retrieval task.

We train Support Vector Machines (SVM) classifiers. Though the similarity of Fisher Vector is usually measured by an inner product weighted by the inverse of Fisher information matrix, Fisher Vector itself can be used with nonlinear kernels. Moreover, to combine first and second order terms in Fisher Vector, we can either directly concatenate them or build classifiers separately and do a late fusion. For the previous approach, we use

$$K(F_{X_i}, F_{X_j}) = \exp\left\{-\sum_{f\in\mathcal{F}} \frac{1}{A_f} D(F^f_{X_i}, F^f_{X_j})\right\} \tag{11}$$

where

$$D(F^f_{X_i}, F^f_{X_j}) = \|F^f_{X_i} - F^f_{X_j}\|_2^2 \tag{12}$$

$\mathcal{F}$ is the set of different feature vector types, $D()$ is the distance function, and A_f is the average of distances for feature type f in training data. This kernel function is a special case of RBF kernel, where features are concatenated with different weights, and sigma is set based on average distances.

Late fusion is a way to combine decision confidences from different classifiers, it has shown superior performance than early fusion in some tasks. In this paper, we use a geometric average of individual scores.

4. Experiments

This section describes the dataset for evaluation, and provides experimental results.

4.1. Dataset

We use videos selected from the entire TRECVID MED11 video corpus and from MED12 Event Kit data [1] for evaluation. The datasets contain more than 40000 diverse, user-generated videos vary in length, quality and resolution. The total length of videos is more than 1400 hours. There are 25 positive event classes defined in this data set, along with a big collection of samples that belong to none of these. The event concepts can be basically categorized as:

- Distinct by objects: Board trick, bike trick, making a sandwich, etc.
- Distinct by motion patterns: Changing vehicle tire, getting vehicle unstuck, etc.
- Distinct by high-level motives: Birthday party, wedding ceremony, marriage proposal, etc.

A complete list of event definitions can be found in [1]. [1]

For our evaluation, we utilize two different data partitions. A smaller set, called Event Kit (EK), has 2062 positive samples of events 1 to 15. It is used for fast selection of classifier independent parameters. A larger set with 13274 videos is separated into 2 partitions, a training set (Train) and a test set (Test), its goal is to evaluate the framework's performance on a large scale dataset. The number of videos in each partition is shown in Table 1. We sampled randomly from the large set to create these partitions.

4.2. Experiment Setup

In this section, we describe the proposed classification framework as well as the baseline system.

[1]There are five events not belonging to test set, they are: Attempting a board trick, feeding an animal, landing a fish, wedding ceremony and working on a woodworking project.

Partition	#Events	#Pos	#Neg	#Total
Event Kit	15	2062	0	2062
Train	25	1763	7077	8840
Test	25	889	3545	4434

Table 1. Number of positive and negative samples in each partition

4.2.1 Fisher Vector and VLAD Generation

We use Laptev's STIP implementation[2], with default parameters and sparse feature detection mode. For Dense Trajectory[3], we resize the video's width to 320 first and set sampling stride to 10 pixels. Both descriptors have several components, we concatenate them directly to form a 162 dimension feature vector for STIP features and a 426 dimension feature vector for DT features. Since the length of Fisher Vector is linear in the dimension of local features, PCA is used to project the features onto a lower dimension space; we project STIP features to 64 dimensions and DT features to 128 dimensions.

We randomly select about 1500000 feature descriptors from the Train set. These descriptors are used to train the PCA projection matrix and get codebooks with GMM and K-Means clustering.

With spatio-temporal pyramid, each sub-volume has its own Fisher Vector or VLAD. According to our experiments, increasing the number of spatial pyramid layers boosts the performance, while increasing temporal pyramid layers has little influence or even hampers the performance. We set number of spatial pyramid layers (#SP) as 2 and temporal pyramid layers (#TP) as 1, to balance the classification performance and speed.

Input : Local feature point set X from a single video
Output: Fisher Vector / VLAD F_X
Build spatial-temporal pyramid $V = \{V_1, ..., V_k\}$
Project X to X' by PCA
Set F_X as an empty vector
for V_i *in* V **do**
  Select point set X'_i that lie in V_i
  Encode X'_i to Fisher Vector or VLAD $F_{X'_i}$
  Power normalize $F_{X'_i}$
  $l2$-normalize $F_{X'_i}$
  Concatenate $F_{X'_i}$ at the end of F_X
end
$l2$-normalize F_X

Algorithm 1: Fisher Vector/VLAD generation

Before concatenation, we normalize each vector by two

[2]http://www.di.ens.fr/~laptev/download.html#stip

[3]http://lear.inrialpes.fr/people/wang/dense_trajectories

steps. First, a power normalization is conducted on each dimension. Then, all vectors are $l2$-normalized, concatenated together and $l2$-normalized again, this is different from the traditional spatial pyramid, where histograms from larger cells are penalized and normalization is after concatenation[9]. We use $l2$-normalization since it is natural with linear kernel, which is evaluated later. A full algorithm is shown in Algorithm 1.

In the following experiments, we will call event classification framework with VLAD as VLAD, with first order components of Fisher Vector as FV 1 and with second order components of Fisher Vector as FV 2.

4.2.2 BoW Baseline

We use the same local features with no dimension reduction, and a standard BoW approach with the following modifications:

First, instead of hard assigning each local feature to its nearest neighbor, soft assignment to nearest #K neighbors is used[17].

Secondly, we use spatio-temporal pyramid to encode spatial and temporal structures.

Based on experimental results, we set codebook size as 1000, $K = 4$, $\#SP = 3$ and $\#TP = 1$, the final representation has 21000 dimensions and is $l1$-normalized to form a histogram.

4.2.3 Classification Scheme

Classifiers are built with SVM[3] in a one over rest approach. We use the probability output produced by classifier as confidence values.

For parameter selection, we use 5-fold cross validation: Training data are separated into 5 parts randomly, and the ratio of positive over negative samples is approximately kept, the parameter set with the highest average performance is selected. Because the dataset is highly unbalanced, traditional accuracy based parameter search is quite likely to produce a trivial classifier predicting all queries as negative. We choose to optimize mAP instead.

Kernel function also plays a key role in SVM classification. For Fisher Vector and VLAD, we compare the kernel mentioned in Section 3.3 and linear kernel. For BoW histogram, χ^2 kernel is used.

4.3. Results on Event Kit

We use Event Kit data and STIP features to study how to choose non-classifier related parameters for Fisher Vector and VLAD, including power normalization factor α, PCA projected dimension D' and number of clusters K, their default values are 0.5, 64 and 64, respectively. Gaussian kernel is used for SVM, and mAPs are calculated based on average cross-validation results.

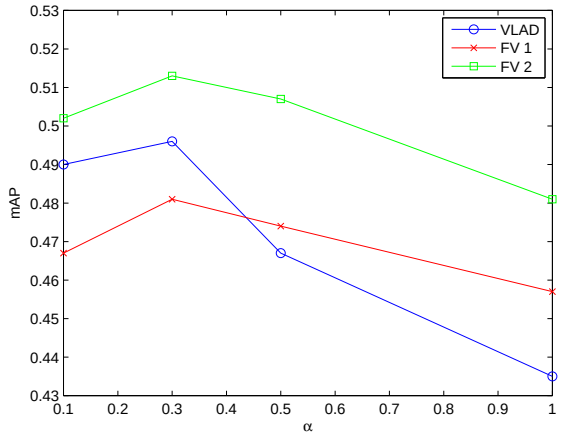

Figure 4. mAPs of power normalization factor α with VLAD, FV 1 and FV 2

Encoding	no PCA (162 dim)	PCA (64 dim)
VLAD	**0.478**	0.467
FV 1	0.448	**0.474**
FV 2	0.471	**0.507**

Table 2. mAP of VLAD, FV 1 and FV 2, with and without PCA

4.3.1 Effect of Power Normalization

As discussed above, when $0 \leq \alpha < 1$, power normalization can smooth the spikes in the feature vector. We set α as 0.1, 0.3, 0.5 and 1.0, their performance are shown in Figure 4. From the figure, it's easy to see that power normalization step improves mAP. Meanwhile, VLAD is more susceptible to change of α than the Fisher Vector. One possible reason is that VLAD treats all cluster centers as equally important, and ignores covariance information.

4.3.2 Effect of PCA

Next we show how dimension reduction influences classification performance. The mAPs are displayed in Table 2. Interestingly, PCA has different effects on VLAD and Fisher Vector. For VLAD, the performance drops slightly, it is understandable since some information is lost during PCA. However, for Fisher Vector, PCA helps improve the performance. One possible explanation, as described in [7], is PCA's impact on GMM: It decorrelates different dimensions during the projection. Since we assume diagonal covariance matrix for Gaussian distribution, it may be more advantageous to do so. Besides, clustering methods can become unstable in higher dimension space, which is often a trade-off with information preserved.

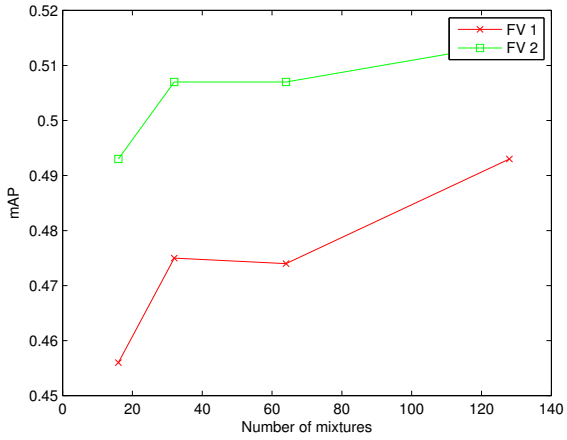

Figure 5. mAPs of cluster size K with VLAD, FV 1 and FV 2

4.3.3 Effect of Number of Clusters

Finally, Figure 5 shows how the performance changes with the number of clusters K. Based on the figure, mAP grows monotonically as K becomes larger. However, relative improvement is small after $K \geq 32$. Since computational cost is high after K gets large, we try K only as large as 128.

It is worth noting that, second order components of Fisher Vector performs the best when used alone. VLAD and first order components of Fisher Vector have very similar performance. However, since VLAD is more sensitive to the change of α, Fisher Vector might be still preferred when only first order information is used.

4.4. Results on Test Set

To compare our framework with BoW baseline, and have a better idea of how our method works on larger dataset, evaluation on our Test set is given below. All the classifiers are trained by videos in the Train partition. For both VLAD and Fisher Vector, we set $K = 64$, the total dimension is 20480 for STIP and 40960 for DT. For BoW, we use $K = 1000$, the total dimension is 21000.

4.4.1 Gaussian Kernel vs Linear Kernel

In this section, we study the influence of different kernel functions. STIP features are used and α is set to 0.5. According to Table 3, Gaussian kernel gives higher mAP in all three encodings, the difference can be as big as 13%. This may be explained by the non-linear nature of representations. Note that is in variance with observations of work in image classification [13] that focus on use of linear SVMs only for Fisher Vector features.

In the following experiments, we use Gaussian kernel to build SVM classifiers.

Encoding	Linear Kernel	Gaussian Kernel
VLAD	0.236	**0.259**
FV 1	0.259	**0.265**
FV 2	0.244	**0.275**

Table 3. mAP Performance comparison on Test set with linear kernel and Gaussian kernel

Combination	Direct	MK	Late Fusion
FV 1 + FV 2	0.298	0.295	**0.305**

Table 4. mAP comparison on Test set among different fusion methods

4.4.2 Different fusion methods

There are several ways to combine the first and second order terms of Fisher Vector. Besides the multi-kernel (MK) approach discussed above, we also try to concatenate the two directly (Direct). All methods give similar results, with late fusion being slightly better. The mAPs are shown in Table 4.

4.4.3 Comparison with Baseline

We compare the performance of BoW, VLAD and FV with STIP and DT features. Late fusion is used for Fisher Vector (FV), α is set to 0.3 for VLAD and 0.5 for FV. The results are shown in Table 5.

We can see that Fisher Vector gives the best mAP for both STIP and DT, it has about 35% improvement for STIP and 17% improvement for DT over the baseline. VLAD improves mAP by 19% for STIP, less than Fisher Vector does. Considering AP is the percentage of positive samples when assigning labels randomly, our best framework is 43 times better than random performance ($35.5/4434 \approx 0.008$).

It is difficult to account precisely for the reasons of superior results of Fisher Vector coding. It captures more of the feature point distributions and hence likely to be more discriminative. We believe that our hypothesis, given in Section 3.2.3, that Fisher coding can suppress the contribution of background features when they fit the model well, is also part of the answer. From the table, we can also see that DT outperforms STIP in most events, but the relative improvement from BoW to Fisher Vector is smaller. Since DT features are dense, the impact of background may be less than for sparse features.

5. Conclusion

We have presented a technique for classification of unconstrained videos by using Fisher Vector coding of sparse and dense local features. Significant improvements (35% and 19% improvement for sparse STIP features and dense DT features respectively) over standard Bag-of-Words have

	BoW+ STIP	VLAD+ STIP	FV+ STIP	BoW+ DT	FV+ DT
E001	0.399	0.412	0.450	0.468	**0.490**
E002	0.053	0.084	**0.141**	0.059	0.127
E003	0.174	0.259	0.273	0.384	**0.419**
E004	0.517	0.583	0.581	**0.624**	0.554
E005	0.193	0.229	0.270	0.194	**0.319**
E006	0.217	0.217	0.192	0.225	**0.279**
E007	0.064	0.165	0.130	0.190	**0.231**
E008	0.535	**0.579**	0.576	0.564	0.567
E009	0.284	0.316	0.368	0.403	**0.442**
E010	0.093	0.116	0.154	0.216	**0.272**
E011	0.154	0.193	0.224	0.198	**0.235**
E012	0.260	0.364	**0.459**	0.446	0.443
E013	0.366	0.404	0.381	0.413	**0.457**
E014	0.357	0.370	0.403	0.417	**0.445**
E015	0.292	0.346	0.393	0.352	**0.450**
E021	0.104	0.234	0.240	0.245	**0.446**
E022	0.058	0.088	0.069	0.066	**0.094**
E023	0.361	0.489	0.550	0.600	**0.645**
E024	0.194	0.148	**0.202**	0.069	0.090
E025	0.040	0.107	**0.175**	0.059	0.142
E026	0.182	0.201	0.252	0.277	**0.345**
E027	0.156	0.326	0.364	0.470	**0.516**
E028	0.285	0.286	**0.438**	0.317	0.361
E029	0.187	0.174	**0.271**	0.179	0.256
E030	**0.148**	0.064	0.068	0.072	0.118
mAP	0.227	0.270	0.305	0.300	**0.350**

Table 5. mAP Performance comparison on Test set with different features and encodings

been demonstrated on a rather large test set which contains highly diverse videos of varying quality. The improvement is consistent across the event classes indicating robustness of the process. While use of similar techniques for image classification has been demonstrated before, we are not aware of use of such methods for video classification in previous work. We also find that use of the full Fisher vector gives significant improvements over the simpler VLAD representation for video classification.

6. Acknowledgement

Omitted to maintain anonymity, will be provided in the final version.

References

[1] http://www.nist.gov/itl/iad/mig/med12.cfm. 1, 5

[2] http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.11.org.html. 2

[3] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2:27:1–27:27, 2011. 6

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop*, 2004. 1

[5] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999. 2, 3

[6] H. Jegou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 2, 3

[7] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *PAMI*, 2011. 3, 4, 6

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

[9] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 4, 6

[10] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010. 2

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2

[12] P. Natarajan, S. Vitaladevuni, U. Park, S. Wu, V. Manohar, X. Zhuang, S. Tsakalidis, R. Prasad, and P. Natarajan. Multimodel feature fusion for robust event detection in web videos. In *CVPR*, 2012. 1, 2

[13] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 2, 3, 7

[14] S. Sadanand and J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012. 2

[15] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004. 1

[16] A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. S. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In *CVPR*, 2012. 1, 2

[17] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. In *ECCV*, 2008. 1, 3, 4, 6

[18] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011. 2

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

[20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 1