Good Practices for Learning to Recognize Actions using FV and VLAD

Jianxin Wu, *Member, IEEE*, Yu Zhang, and Weiyao Lin

***Abstract*—High dimensional representations such as Fisher Vectors (FV) and Vectors of Locally Aggregated Descriptors (VLAD) have shown state-of-the-art accuracy for action recognition in videos. The high dimensionality, on the other hand, also causes computational difficulties when scaling up to large-scale video data. This paper make three lines of contributions to learning to recognize actions using high dimensional representations. First, we reviewed several existing techniques that improve upon FV or VLAD in image classification, and performed extensive empirical evaluations to assess their applicability for action recognition. Our analyses of these empirical results show that normality and bimodality are essential to achieve high accuracy. Second, we proposed a new pooling strategy for VLAD and three simple, efficient, and effective transformations for both FV and VLAD. Both proposed methods have shown higher accuracy than the original FV / VLAD method in extensive evaluations. Third, we proposed and evaluated new feature selection and compression methods for the FV and VLAD representations. This strategy uses only 4% of the storage of the original representation, but achieves comparable or even higher accuracy. Based on these contributions, we recommend a set of good practices for action recognition in videos for practitioners in this field.**

***Index Terms*—Action Recognition, Empirical Evaluations, Transformations, Feature Selection and Compression**

I. Introduction

The past decade has witnessed increasing interests in recognizing actions in videos (e.g., [1], [2], [3], [4], [5], [6]) based on effective representations and learning methods. Along with the dramatic increase of publicly and commercially available video sharing services like Youtube$^{\text{TM}}$, the need for automatic understanding of video contents has been an attractive research topic in relevant areas, such as computer vision, pattern recognition, signal processing, and multimedia. Action recognition, or recognizing the semantic category of a short video clip that contains an action, has been the most studied topic in this direction, possibly because it forms the basis for successful video understanding.

There has been a vast literature on action recognition. In this paper, we focus on the bag-of-features action recognition pipeline, because it is currently the state-of-the-art action recognition framework. Without loss of generality, we can roughly decompose a bag-of-features based action recognition pipeline into three sequential steps:

Step 1: Extraction of informative raw motion features. The community generally agree that *motion information* is the most important cue for recognizing actions. From the beginning of action recognition research, finding good motion features have been the focal spot. That is, to find where and when the motion is, and to properly represent these raw motion information. Recent examples of popular raw motion features include the space-time interest points (STIP) [1] and dense trajectory features (DTF) with motion boundary histograms (MBH) [6]. STIP and MBH, especially MBH, usually achieve the highest action recognition accuracy in the literature.

Step 2: From raw motion features to action representation. The raw motion features extracted from a single video clip are usually a set of vectors, which are inconvenient for recognizing the action inside it. Conventional classifiers require that a video is represented as a single vector. An important step is to transform a set of raw motion feature vectors into a single vector.

Many transformation methods have been proposed in the literature. The bag-of-features (BOF) representation has been the most popular choice, due to its simplicity and excellent performance. BOF originates from the text processing research, which transforms an article (a set of words) into a single vector (i.e., histogram of word frequencies.) It then becomes popular in image classification, where visual descriptors extracted from various image patches in an image form a set. Then, a k-means clustering process turns this set of visual descriptors into a set of "visual words" [7], [8]. The same BOF framework can thus be adopted. The BOF transformation framework has been popular in action recognition [2], [9], [5], [6].

Step 3: Classification. The third and final step of action recognition is to classify the transformed vectors / videos using machine learning techniques. Various classifiers have been used to learn action recognition models. The support vector machines (SVM) has been the common choice in recent research, partially because of its excellent speed and accuracy, and publicly available high-quality implementations [10].

Among these steps, motion features have enjoyed the most attention in the past. The DTF feature in [6] significantly outperformed previous state-of-the-art results on 9 benchmark datasets. As one example, recognition accuracy on the challenging HMDB51 dataset jumped from 26.9% in [11] to 48.3% in [6]. Finding good motion features may continue to maintain its central role in action recognition research, e.g., the ω-flow features further increased accuracy on HMDB51

This research was supported by the National Natural Science Foundation of China under Grant 61422203, 61471235 and U1201255, the Chinese National 973 under Grant 2013CB329603, and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

J. Wu is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: wujx2001@nju.edu.cn.

Y. Zhang is with the Advanced Digital Sciences Center, Singapore. E-mail: zhang.yu@adsc.com.sg.

W. Lin is with the Institute of Image Communication and Information Processing, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: wylin@sjtu.edu.cn.

to 52.1% [12]. The authors of [6] further improved their DTF features and proposed ITF (improved trajectory features) by taking into account the effect of camera motion and hence achieving better motion estimates [13]. As a consequence of this improved motion information encoded inside ITF, the accuracy on HMDB51 is improved to 57.2%. Hence, the experiments in our evaluations use ITF as the raw motion feature. However, the proposed methods can also be applied to other types of features, such as the optimal set generated by genetic programming [14].

This paper focuses on the second step: how to transform a set of raw motion features into a vector representation of the video. The VLAD framework (Vectors of Locally Aggregated Descriptors) was used in [12] instead of the classic bag-of-features, a fact that contributed a lot to its accuracy gain. One recent trend is that high-dimensional BOF representation frameworks such as Fisher Vector (FV) [15] or VLAD [16] are gaining popularity in video representation—mainly due to their excellent recognition accuracy [17]. For example, VLAD exhibited significant improvements on two difficult datasets (Hollywood2 and HMDB51) over classic bag-of-features [12]. In [18], using FV or VLAD can both improve the event classification accuracy by a large margin.

In this paper, through extensive empirical evaluations on FV and VLAD, we show that *proper video representation, i.e., good practices* in transforming a set of raw motion features to a vector representation, *can further boost action recognition accuracy*, and the gains are achieved with negligible increase of computational cost. We *propose new methods for the transformation*, including a new pooling strategy for VLAD and three per-dimensional transformations for both FV and VLAD. We also present *new analyses and explanations* of various empirical results, which show that normality and bimodality are essential for high recognition accuracy. Finally, we propose a combination of feature selection and compression such that machine learning for large-scale action recognition can be handled. Specifically, the contributions are:

1: Extensive empirical evaluations of various improvement techniques for the FV and VLAD representations.

Many improvements have been proposed for the FV and/or VLAD representation framework in the image classification and retrieval domain, including at least power normalization [19], root descriptors [20], spatio-temporal pyramid (STP) [9] (an extension of SPM to videos, see also [6], [18]) and residue normalization [21]. Most of these improvement techniques have achieved excellent performance in the image domain, but have not yet been applied or evaluated in the video domain. We evaluate these techniques using extensive experiments and a few good practices concerning these techniques are presented based on the evaluation outcomes.

Besides empirical evaluations, we also try to explain the success or failure of these techniques, and propose recommendations on when they are to be used or not to be used.

Empirical evaluation of action recognition frameworks have already been performed previously, e.g., in [17], [22]. We want to note that the aspects evaluated in this paper are mainly improvement techniques proposed in the past few years, which are complementary to those aspects evaluated in previous studies [17], [22].

2. A new pooling strategy for VLAD, and three transformation to improve both FV and VLAD.

We propose a new average pooling strategy to replace the sum pooling strategy in VLAD. We also propose three simple but efficient and effective transformations. The proposed pooling strategy and transformations have excellent performance in terms of both accuracy and running speed, as shown in our evaluations.

3. Combined feature selection and compression for large-scale datasets.

We evaluate the MI-based feature selection method for FV in [23]. We also proposed a new 2-bit compression scheme based on the existing 1-bit compression. The combination of feature selection and compression can achieve higher accuracy than the original FV or VLAD representation, even though only 4% or even less storage is used.

Overall, the goal of this paper is to provide a set of good practices for the representation of an action video which will facilitate the learning of action recognition models. When we are given a type of raw motion feature (or set of motion features that are combined together), good video representation can make the best of the representational power of these raw motion features. Furthermore, if we are to compare different raw motion features, proper video representation can remove the bias of video representations to ensure fair comparisons among the raw motion features. The suggested good practices come from two origins: extensive evaluations of existing techniques, and pooling and post-processing transformation techniques proposed in this paper.

We start by introducing the commonly used representations in Sec. II. The improvement techniques (both existing and proposed) and their evaluations are in Sec. III. Study of distribution of feature values (normality and bimodality) and a new maxent explanation for VLAD is presented in Sec. IV. The three proposed transformations and their evaluations are in Sec. V. Sec. VI concludes this paper. Some preliminary results, including the VLAD-based evaluations on three datasets using the DTF raw motion features, were originally published in [24] as a conference presentation.

II. Background: BOF, VLAD and FV

Supposing that we are given a video clip. A set of N motion features are already extracted from it, and a codebook of size K is learned by the k-means clustering algorithm applied to a subset of all raw motion features in the training set. We denote the code words in the codebook as $\boldsymbol{c}_1, \dots, \boldsymbol{c}_K$ (centroids of the K clusters) and the motion features for this video as $\boldsymbol{x}_i^j$, where $i = 1, \dots, K$ and $j = 1, \dots, n_i$. $\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^{n_i}$ are all the motion features that belong to the i-th code word (i.e., whose nearest neighbor in the codebook is $\boldsymbol{c}_i$). We assume that the raw motion features are d-dimensional, that is, $\boldsymbol{c}_i \in \mathbb{R}^d$, $\boldsymbol{x}_i^j \in \mathbb{R}^d$ for all i and j, and that $\sum_{i=1}^K n_i = N$ holds.

The classic bag-of-features representation will represent this video as a vector in $\mathbb{R}^K$, as

$$(n_1, n_2, \cdots, n_K)^T \in \mathbb{R}^K \,, \tag{1}$$

a K-dimensional vector that contains frequencies of different code words. Typical usage is to choose K from the set $\{256, 512, 1024, 2048, 4096\}$.

The VLAD representation extracts more information from the set of raw features than simple BOF. Its representation is based on the residue $\boldsymbol{x}_i^j - \boldsymbol{c}_i$, that is, not only counting how many raw motion features belong to a code word $\boldsymbol{c}_i$, but also taking into account how they distribute with respect to the code word. The VLAD representation, in our notation, is:

$$\boldsymbol{v} = \left(\sum_{j=1}^{n_1} (\boldsymbol{x}_1^j - \boldsymbol{c}_1), \ldots, \sum_{j=1}^{n_K} (\boldsymbol{x}_K^j - \boldsymbol{c}_K) \right) \in \mathbb{R}^{d \times K} . \quad (2)$$

For each code word, VLAD (vector of locally aggregated descriptors) aggregates the residues of raw motion features inside this code word, and concatenates the aggregated vectors for all code words, hence having a dimensionality of Kd. VLAD vectors are usually ℓ_2 normalized by $\boldsymbol{v} \leftarrow \boldsymbol{v}/\|\boldsymbol{v}\|$.

The residues are much higher dimensional than a single counting number (d vs. 1) and contains more useful information. Thus, VLAD has higher recognition rates than BOF in both image and video recognition problems [16], but also requires much higher storage costs and computation times [23]. When we need to deal with large number of video clips, it is important to deal with such computational difficulties in practice.

VLAD encodes the first-order information (the residue) rather than the zero-th order information (counts) and gains in recognition accuracy. It can be viewed as a special case of the Fisher Vector (FV) representation, which further utilizes the second-order information [15].

In FV, let us suppose $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ are used to learn the representation, which is a subset of all training raw motion features. A Gaussian Mixture Model (GMM) is built from this subset, which is used to replace the k-means clustering algorithm. A GMM model has the following form:

$$u_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{k=1}^{K} w_k u_k(\boldsymbol{x}) \,, \quad (3)$$

where

$$u_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right)^T \Sigma_k^{-1} \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right) \right) , \quad (4)$$

is a component Gaussian distribution, $\boldsymbol{x}$ is a raw motion feature vector, and $\boldsymbol{\lambda} = \{w_k, \boldsymbol{\mu}_k, \Sigma_k, k = 1, 2, \ldots, K\}$ is the set of parameters, including the mixing coefficients w_k, the mean $\boldsymbol{\mu}_k$ and the covariance Σ_k (second-order information) for all Gaussian components.

Given any single raw motion feature vector $\boldsymbol{x}$, its posterior probability w.r.t. the Gaussian components are computed as

$$\gamma_k(\boldsymbol{x}) = \frac{w_k u_k(\boldsymbol{x})}{\sum_{j=1}^{K} w_j u_j(\boldsymbol{x})} \,. \quad (5)$$

FV assumes that Σ_k is diagonal and instead denoted by $\boldsymbol{\sigma}_k^2$, which is a vector of d numbers. Then, FV generates

$$\mathcal{G}_{w_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \left(\gamma_k(\boldsymbol{x}) - w_k\right) \,, \quad (6)$$

$$\mathcal{G}_{\boldsymbol{\mu}_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_k(\boldsymbol{x}) \left(\frac{\boldsymbol{x} - \boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k} \right) \,, \quad (7)$$

$$\mathcal{G}_{\boldsymbol{\sigma}_k} = \frac{1}{\sqrt{w_k}} \sum_{t=1}^{T} \gamma_k(\boldsymbol{x}) \frac{1}{\sqrt{2}} \left(\frac{(\boldsymbol{x} - \boldsymbol{\mu}_k)^2}{\boldsymbol{\sigma}_k^2} - 1 \right) . \quad (8)$$

Note that all $\frac{\boldsymbol{x}-\boldsymbol{\mu}_k}{\boldsymbol{\sigma}_k}$ and $(\cdot)^2$ operations that involve vectors are treated as component-wise operations on all dimensions. Thus, $\mathcal{G}_{w_k}$, $\mathcal{G}_{\boldsymbol{\mu}_k}$ and $\mathcal{G}_{\boldsymbol{\sigma}_k}$ are 1-, d- and d-dimensional, respectively. The final FV representation is the concatenation of these three parts for all $k = 1, 2, \ldots, K$, which is $(2K+1)d$ dimensional. In this paper, we ignore $\mathcal{G}_{w_k}$ but keep $\mathcal{G}_{\boldsymbol{\mu}_k}$ and $\mathcal{G}_{\boldsymbol{\sigma}_k}$. Thus, the dimensionality is $2Kd$. An FV vector is also ℓ_2 normalized.

Both FV and VLAD can be considered as variants of the BOF framework. All three methods have been widely used in image classification and retrieval. There are also techniques to improve the original versions of them. Some of these improvement techniques (such as power normalization and STP) have also been used together with BOF, VLAD or FV in action recognition. Some (e.g., root descriptors and residue normalization), however, have not been used in action recognition in videos to the best of our knowledge. We will evaluate them in the context of action recognition in Sec. III.

III. Evaluations of FV and VLAD improvement techniques

We first specify the details of these improvement techniques:

- Power normalization [19]. Apply

  $$v \leftarrow \operatorname{sgn}(v)|v|^{\alpha} (0 \le \alpha \le 1) \quad (9)$$

  to every dimension of the vector $\boldsymbol{v}$ (the FV or VLAD representation) respectively. This operation is applied after Eq. 2 for VLAD or equations 6–8 for FV, but before the ℓ_2 normalization.
- Root descriptors [20]. Apply

  $$\boldsymbol{x} \leftarrow \sqrt{\boldsymbol{x}} \quad (10)$$

  before Eq. 2 or equations 6–8 (also before the k-means clustering or GMM training), assuming any raw motion feature vector $\boldsymbol{x}$ is a non-negative vector. The $\sqrt{\cdot}$ operator is applied to every dimension of $\boldsymbol{x}$ individually.
- Residue normalization [21]. In Eq. 2, replace $\boldsymbol{x}_i^j - \boldsymbol{c}_i$ by

  $$\frac{\boldsymbol{x}_i^j - \boldsymbol{c}_i}{\|\boldsymbol{x}_i^j - \boldsymbol{c}_i\|} . \quad (11)$$

  This operation does not apply to the FV representation.
- Spatio-temporal pyramid (STP) [9]. We use the following setup: split all video frames horizontally into 2 halves, extract VLAD or FV from both the original video and these two halves, leading to a $3Kd$ dimensional video representation if VLAD is used, or $6Kd$ for FV.

In practice, the $\boldsymbol{x}_i^j$ in VLAD or the $\boldsymbol{x}$ in FV is usually *not* a raw motion feature vector. Dimension reduction techniques are usually first applied to the raw motion features. And PCA (Principal Component Analysis) is often used, because PCA has been proved essential for the success of FV and VLAD [15], [16].

Besides these improvement techniques in the literature, *we propose a new variant of the VLAD representation*. In Eq. 2, individual raw motion features are pooled using a sum operation. We found that *using an average pooling can achieve higher accuracy rates*, which is verified in our extensive evaluations. In its precise form, the proposed average pooling VLAD variant is:

$$\boldsymbol{v}' = \left(\frac{\sum_{j=1}^{n_1} (\boldsymbol{x}_1^j - \boldsymbol{c}_1)}{n_1}, \ldots, \frac{\sum_{j=1}^{n_K} (\boldsymbol{x}_K^j - \boldsymbol{c}_K)}{n_K} \right) \in \mathbb{R}^{d \times K} . \tag{12}$$

k-means is used in VLAD to cluster descriptors into groups and aggregate them. However, k-means is known to generate imbalanced grouping results [25], that is, the numbers $n_1, n_2, \ldots, n_K$ will vary a lot. Simply aggregating the residues will cause those large groups (i.e., with a large n_k) to be overly emphasized. Thus, we expect the average pooling in Eq. 12 (which removes this effect by dividing the residues by n_K) will work better than the sum pooling in Eq. 2.

A. Datasets and experimental setup

We evaluate the performance of these techniques using seven datasets. They cover different scales and difficulty levels for action recognition:

UCF101 [26]: This is a large scale dataset with 101 action categories and 13320 clips. We follow the evaluation protocol in [27]. Three sets of training / testing videos are defined in the protocol. For every experimental setup, we train 3 classification models using the training sets, and evaluate the models' performance using their respective testing sets. The average recognition accuracy is reported.

HMDB51 [5]: This is a medium scale dataset with 51 actions in 6766 clips. We use the original (not stabilized) videos and follow [5] to report average accuracy of its 3 predefined splits of training / testing videos.

UCF50 [28]: This is a medium scale dataset with 50 action categories. For each action type, there are 25 groups and the first 4 videos inside each group are used in our experiments, following the protocol of [28]. A leave-one-out (LOO) cross-validation (CV) type of evaluation method is used, but each group is considered as an atomic unit in the evaluation (rather than a video). The average accuracy of the 25 LOO-CV runs is averaged and reported.

Youtube [4]: This is a small scale dataset with 11 action types. Similar to the UCF50 dataset, there are 25 groups in each action category and 4 videos are used in each group. Following the original protocol, we again report the average of the 25-fold LOO-CV accuracy rates.

UCF sports [29]: This is a small scale action recognition dataset with 10 classes, totaling only 150 videos. Following [6], we flip every video horizontally to obtain more training videos. The LOO-CV strategy is used in the evaluation. We run the experiments 150 times, each time using one original video as the testing video, while the remaining 149 original videos and their 149 flipped versions are used during training. The average accuracy of 150 runs is reported.

Hollywood2 [30]: This is a small scale action recognition dataset with 12 action types. The video clips are extracted from Hollywood movies and difficult to classify. There are 823 training and 884 testing videos, totaling 1707 clips. Note that the training and testing clips are extracted from different videos. The average precision (AP) for each action type is first computed, and the mean AP (mAP) of the 12 types is reported, following [30].

Olympic sports [31]: This is a small scale dataset with 783 videos (649 training + 134 testing) and 16 sports actions. Following commonly used protocol for this dataset, we report the mean average precision (mAP) over all action types.

Because our focus is video representation, we fix the raw motion feature in our experiments to the improved trajectory features (ITF) [13], which has further improved over the DTF features. We also fix the classifier in the third step to linear SVM, which is the default choice to use together with FV and VLAD representations. Note that evaluations using DTF features on three datasets (UCF101, HMDB51 and Youtube) are presented in our preliminary work [24].

In ITF, we use all its five raw motion feature types: trajectory, HOG, HOF, mbhX and mbhY. We first sample roughly one million (10^6) raw motion features from 10% videos. Then, for each feature type, we perform the Principal Component Analysis (PCA). The threshold of PCA is picked such that 90% energy is kept. The PCA operations are performed separately for each of the 5 feature types. In generating the ITF features, we use the default parameters, which results in 426 dimensions in total. After the PCA operations, PCA reduce the 426 dimensions to different number of dimensions in different datasets, but are all around approximately 200 dimensions in different runs of all 7 datasets. For the third step, we use the LIBLINEAR software package [10] with its default parameters for classification, except that an additional parameter `-B 1` is used in all our experiments. When performing power normalization, we set $\alpha = 0.5$.

We want to present some notes about the features and classifiers. ITF is a complex and time-consuming feature which suits complex video data. There are other lighter features such as the spatio-temporal Laplacian pyramid feature [32]. This feature is lightweight and achieves high accuracy on the UCF sports dataset, but has not been tested on the recently proposed complex dataset such as UCF 101.

Very recently, deep learning has also been applied to action recognition and has achieved excellent recognition accuracy, e.g., using 3D convolution directly on several consecutive video frames [33], [34]. As aforementioned, motion information is the key information for action recognition, but such information is only implicitly included in the 3D convolutions. The two-stream framework applies deep learning methods to both the video frames and optical flow fields (which encodes motion information explicitly) computed from these frames [35], which yields higher accuracy rates than ITF features (e.g., 88.0% for UCF101). Motion information is further considered in the temporal direction using LSTM (Long-Short Term Memory) in [36], which further boosts recognition accuracy (e.g., 91.3% for UCF101). Since the focus of this paper is on the FV and VLAD based on local descriptors, we will not cover and compare with the approaches using deep learning based features.

As for classifiers, although we only use linear SVM in this paper, there are many other choices beyond SVM. For example, in scenarios where reporting posterior probabilities for all classes is more important than reporting the predicted label, we can resort to the Gaussian processes regression technique in [37].

B. Evaluation results and discussions

The evaluation results are summarized in Tables I and II, for FV and VLAD based techniques, respectively. Now we analyze the results for each technique one by one. In these tables, K stands for the codebook size, STP for spatio-temporal pyramid (3 means original + two horizontal splits), RN for residue normalization, RD for root descriptors, and PN for power normalization. Specific to VLAD, the POOL column denotes whether a sum (Eq. 2) or average pooling ('avg', Eq. 12) is used, respectively. Both POOL and RN are specific to VLAD, which are not evaluated for FV.

- **Baselines**. The row 2 in Table I and row 3 in Table II are baselines in the evaluations, respectively. Except for power normalization, none of the improvements is used. $K = 128$ is used for FV and $K = 256$ for VLAD, which makes the FV and VLAD vectors having the same number of dimensions. When a different setting is compared with the baselines, a $\uparrow$, $\downarrow$, or $=$ sign indicates that the alternative method is better than, worse than, or same as the baseline. The following evaluations alter one condition to the baseline at a time.
- **K and STP**. One observation consistent with previous evaluations (e.g., [17]) is that: in general the accuracy improves when more information is used. When K increases or STP is used, accuracy generally increases. It is also obvious that increasing K is more effective in VLAD than in FV. The STP operation is also more effective in VLAD than in FV. One possible explanation is that FV already contains enough relevant information for action recognition when $K = 128$, and further increasing K to 256 or using STP may not increase effective information.
- **Power normalization**. PN seems equally effective as increasing information. Row 6 in Table I and row 8 in Table II clearly shows that when PN is removed, the accuracy is consistently lower than the baselines in all 14 experiments. We will provide an explanation for its effectiveness in Sec. IV.
- **Average pooling**. As shown by row 7 in Table II, the proposed average pooling variant outperforms original VLAD in almost all datasets.
- **Residue normalization**. It only applies to VLAD, and is slightly effective (in 3 out of 7 datasets).
- **Root descriptors**. It incurs losses in all VLAD results. However, it is slightly effective in the FV experiments (2 out of 7). We will further examine power normalization and root descriptors in Sec. IV.
- **Combined improvements**. The last row in Table I or II combines these techniques that can increase accuracy and evaluate again, i.e., combining all techniques that exhibits a $\uparrow$ sign in a column. For FV, we compare `K=256` and `K=128, STP` and choose the one with higher accuracy for each column. The last row accuracy is shown in boldface if it is the highest in its column. The combined result achieves the highest accuracy in all datasets for FV, and 5 out of 7 in VLAD experiments. That is, combining effective individual techniques usually leads to even higher accuracy, although it is not guaranteed.

In short, we make the following suggestions of good practices based on evidences from the evaluations of existing and the proposed improvement techniques:

- *Complexity is an important factor.* Techniques that improve performance by introducing longer features (i.e., increasing K or using STP) is usually effective. It is good to start with a small K, and gradually increase the codebook size or the number of divisions in STP, until the performance is saturated. As a rule of thumb, we recommend $K = 128$ plus STP if FV is used, and $K = 256$ if VLAD is used;
- *FV is in general better than VLAD.* The gap, however, is not significant in many cases. Given the simplicity of its implementation, VLAD is also an excellent choice for action recognition representations;
- *Average pooling and power normalization are good practices.* PN is a necessary step for both FV and VLAD. For VLAD, the proposed average pooling is a better choice than the sum pooling in the original VLAD; and,
- *Try other techniques and combine them.* It is also worthwhile to try techniques such as root descriptors and residue normalization. It is also important to combine those individual techniques that show improvements on a specific dataset.

Important questions remain though: 1) Why root descriptors have mixed results and when will it help? 2) Why is power normalization always effective in video action recognition? We will handle both questions in the next section.

IV. Distribution of values & A MaxEnt Interpretation

In this section, we will have a closer look of the properties of feature vectors generated by FV and VLAD, in order to understand root descriptors and power normalization better. Given a raw motion feature $\tilde{\boldsymbol{x}} \in \mathbb{R}^{\tilde{d}}$, we first perform the PCA to get our motion feature $\boldsymbol{x}$

$$\boldsymbol{x} \leftarrow P\left(\tilde{\boldsymbol{x}} - \mathrm{E}(\tilde{\boldsymbol{x}})\right) , \tag{13}$$

where $P \in \mathbb{R}^{d\times\tilde{d}}$ and $\mathrm{E}(\tilde{\boldsymbol{x}})$ are eigenvectors of the covariance matrix of $\tilde{\boldsymbol{x}}$ and its mean, respectively. Further assuming we center the raw motion features (i.e., minus $\mathrm{E}(\tilde{\boldsymbol{x}})$ from them), Eq. 13 reduces to $\boldsymbol{x} \leftarrow P\tilde{\boldsymbol{x}}$.

In order to simplify the notations, in this section we assume $K = 1$ without loss of generality. All the raw features from

TABLE I
EVALUATIONS OF THE FV IMPROVEMENT TECHNIQUES. ROW 2 IS THE BASELINE METHOD. BEST IF VIEWED IN COLOR.

#	K	STP	RD	PN	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
1	64	1		√	82.20↓	55.38↓	89.52↓	89.73↓	88.00=	61.39↓	90.89↓
2	128	1		√	84.22	56.27	89.90	90.82	88.00	62.62	91.37
3	256	1		√	84.56↑	56.93↑	90.28↑	90.64↓	86.00↓	62.71↑	90.43↓
4	128	3		√	84.56↑	55.53↓	90.16↑	90.64↓	91.33↑	61.91↓	89.43↓
5	128	1	√	√	83.56↓	56.64↑	89.90=	90.09↓	89.33↑	62.30↓	90.64↓
6	128	1			80.64↓	51.55↓	86.70↓	87.91↓	85.33↓	59.98↓	86.92↓
7	combine results				**84.56**↑	**57.21**↑	**90.28**↑	**90.82**=	**92.00**↑	**62.71**↑	**91.37**=

TABLE II
EVALUATIONS OF THE VLAD IMPROVEMENT TECHNIQUES. ROW 3 IS THE BASELINE METHOD. BEST IF VIEWED IN COLOR.

#	K	STP	RN	RD	PN	POOL	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
1	64	1			√	sum	80.46↓	51.72↓	88.24↓	88.82↓	86.67↓	58.70↓	88.55↑
2	128	1			√	sum	81.72↓	53.94↓	89.06↑	89.45↓	80.33↓	59.28↓	89.21↑
3	256	1			√	sum	83.13	55.32	89.04	89.73	88.00	60.96	88.20
4	256	3			√	sum	83.68↑	55.38↑	89.26↑	90.18↑	88.67↑	60.99↑	87.47↓
5	256	1	√		√	sum	83.15↑	55.27↓	88.90↓	89.55↓	90.00↑	61.15↑	87.49↓
6	256	1		√	√	sum	82.86↓	54.84↓	88.92↓	88.91↓	86.67↓	60.80↓	88.06↓
7	256	1			√	avg	83.36↑	55.84↑	89.56↑	90.27↑	89.33↑	59.43↓	88.71↑
8	256	1				sum	79.25↓	50.89↓	86.36↓	88.27↓	87.33↓	58.01↓	84.45↓
9	**256**	combine results					**84.10**↑	55.56↑	**89.86**↑	**90.45**↑	**92.67**↑	59.41↓	**88.73**↑

a video are denoted as $\tilde{\boldsymbol{x}}^j$, $j = 1, \ldots, N$; and after PCA they become $\boldsymbol{x}^j = P\tilde{\boldsymbol{x}}^j$. There is only one code word, which we call $\boldsymbol{c}$. Since $K = 1$, we will drop the subscripts from $\boldsymbol{c}$ and $\boldsymbol{x}$. Then, the VLAD vector is

$$\boldsymbol{v} = \textstyle\sum_{j=1}^N (\boldsymbol{x}^j - \boldsymbol{c}) = P\left(\sum_{j=1}^N \tilde{\boldsymbol{x}}^j\right) - N\boldsymbol{c}\,. \qquad (14)$$

Eq. 14 states that given the input (sum of raw motion features belonging to a given code word), the VLAD vector $\boldsymbol{v}$ contains values of several linear feature functions, whose projection directions are determined by the PCA eigenvectors.

In other words, given an input video, VLAD is in effect a two step process: 1) reduce the set of raw features belonging to a code word into a single vector (sum of raw features); and, 2) compute values of linear feature functions. Here we ignore the effect of ℓ_2 normalization, which can be viewed as a subsequent postprocessing step.

As for the FV vectors, it is clear that the $\boldsymbol{\mu}$ components (Eq. 7) can also be interpreted as this two-step process. Although the $\boldsymbol{\sigma}$ components in FV (Eq. 8) involves higher-order operations, we believe that the linear two-step process are important in understanding both FV and VLAD, and will study it in detail in this section.

A. *VLAD as MaxEnt linear features*

Linear feature functions have been widely used, e.g., in natural language processing. In the literature, maximum-entropy (maxent) is a widely used method for supervised and unsupervised linear feature function learning [38]. The maxent principle indicates that in an unsupervised case (i.e., without prior knowledge or constraint), we should seek linear projections that lead to maximum entropy of the projected data ($P\sum_j \tilde{\boldsymbol{x}}^j$ in our case).

It is well known that if the input data ($\tilde{\boldsymbol{x}}^j$) follows a normal distribution, then the PCA eigenvectors with largest eigenvalues are the optimal maxent solutions [39]. For completeness, we also provide a sketch of its proof. Suppose $\boldsymbol{x} \sim N(\boldsymbol{0}, \Sigma)$, that is, the random vector $\boldsymbol{x}$ follows a multivariate zero-mean normal distribution. We want to find a maximum entropy (*maxent*) linear projection $\boldsymbol{w}$ such that $\|\boldsymbol{w}\| = 1$ and the projected random variable $y = \boldsymbol{x}^T\boldsymbol{w}$ will have the maximum differential entropy [40]

$$H(y) = -\int p(y) \log p(y)\, dy \qquad (15)$$

among all possible projection directions.

Using properties of the normal distribution, it is easy to derive that $y \sim N(0, \boldsymbol{w}^T\Sigma\boldsymbol{w})$. Since the entropy of a standard normal distribution $N(0, \sigma^2)$ is $\frac{1}{2}\ln\left(2\pi e\sigma^2\right)$ nats, we have

$$H(y) = \frac{1}{2}\ln\left(2\pi e\boldsymbol{w}^T\Sigma\boldsymbol{w}\right)\,. \qquad (16)$$

Thus, finding maxent linear projections for zero-mean Gaussian data is equivalent to solving the following PCA problem:

$$\max_{\boldsymbol{w}} \boldsymbol{w}^T\Sigma\boldsymbol{w} \quad \text{s.t.} \quad \|\boldsymbol{w}\| = 1. \qquad (17)$$

The generalization to multiple linear projections is trivial if we require the linear projections are perpendicular to each other—which is exactly the PCA requirement.

Thus, we arrive at the conclusion that *VLAD can be viewed as a maximum-entropy linear feature learning process, which is optimal when the raw motion features are Gaussian.* This interpretation also applies to the $\boldsymbol{\mu}$ components in Fisher Vectors.

B. *Normality and root descriptors*

Since normality is the assumption for optimal maxent solutions and for PCA to produce statistically independent dimensions (which is suitable for linear classifiers), it is thus desirable to have Gaussian raw features. However, normality rarely holds for either image or motion features, as will be shown by the examples in Fig. 1. These facts lead to a conjecture that *the success or failure of root descriptors hinges on whether it can improve the normality.*

Although there is no intuitive way to check normality of multivariate distributions (in our case $\sum_j \tilde{\boldsymbol{x}}^j$), we can check normality of a single linear feature function (i.e., one dimension in $P \sum_j \tilde{\boldsymbol{x}}^j$). If normality holds for raw features, the extracted features must also be normal, because linear combination of Gaussians is once again a Gaussian.

The normality of a one-dimensional random variable can be tested intuitively using the normal probability plot [41]. In Fig. 1, we show example normal probability plots generated from the UCF sports dataset. The plots are using data from one FV dimension (dimension 7984, $K = 128$ in FV, with power normalization). A normal probability plot has the data value in the x-axis, and the data percentiles in the y-axis in log-scale. When the empirical distribution (blue '+' signs) fits well to the groundtruth normal probability line (red dashed line), the data follows a normal distribution. It is easy to observe that this FV dimension (i.e., one linear projection feature) deviates from the red line significantly in Fig. 1b, but gets closer to Gaussian in Fig. 1a when the root descriptor technique is used.

Based on Fig. 1, we conjecture that the application of root descriptors can sometimes improve normality. Since Fig. 1 only visualizes one dimension in the FV vector, we use statistical tests to examine the normality of a large number of dimensions. We group all dimensions in the FV / VLAD vector into groups, with each group containing 32 contiguous dimensions. Then we use the Jarque-Bera test to examine the first dimension in each group. In the Jarque-Bera test, any p-value less than 0.001 is set to 0.001, and we call dimensions with such p-values as trivial dimensions. We count the number of non-trivial p-values (bigger than 0.001) as the indicator as normality. It is found that when FV is used, the number of non-trivial p-values almost quadrupled in the HMDB51 and UCF sports dataset. However, in other datasets or when VLAD is used, there is no significant increase in this indicator. Furthermore, the trend of this normality indicator coincides well with the rates in Tables I and II. These observations, although far from being strong evidences, suggests that *when the root descriptor technique increases normality of a dataset, we can expect that it will also increase the action recognition accuracy in this dataset.* We want to add that our preliminary work [24] performed experiments using VLAD vectors and the DTF raw motion features, whose results exhibited the same trend between normality and recognition accuracy.

In short, putting these observations together, we have reasons to conjecture that *root descriptors are improving action recognition accuracy through improving normality of the raw motion features—that is, making the normality assumption of maxent feature learning more realistic.*

The normality of root descriptors, however, are still weak, as shown in Fig. 1a. It does not seem easy to find a simple data transformation that could make motion features more Gaussian-like. Thus, it might be advantageous to directly explore the maxent linear projections to replace the PCA eigenvectors. We leave this topic to future research.

TABLE III
PERCENTAGES OF ZERO ENTRIES IN FV AND VLAD VECTORS. DATASETS ARE GENERATED USING THE SETUP OF ROW 2 IN TABLE I AND ROW 3 IN TABLE II FOR FV AND VLAD, RESPECTIVELY.

	FV	VLAD
UCF101	0.05%	12.81%
HMDB51	0.24%	14.85%
UCF50	0.13%	12.29%
Youtube	0.50%	12.14%
UCF sports	0.69%	10.07%
Hollywood2	0.01%	2.81%
Olympic	0.14%	5.55%

C. Side information, power normalization and bimodal distribution

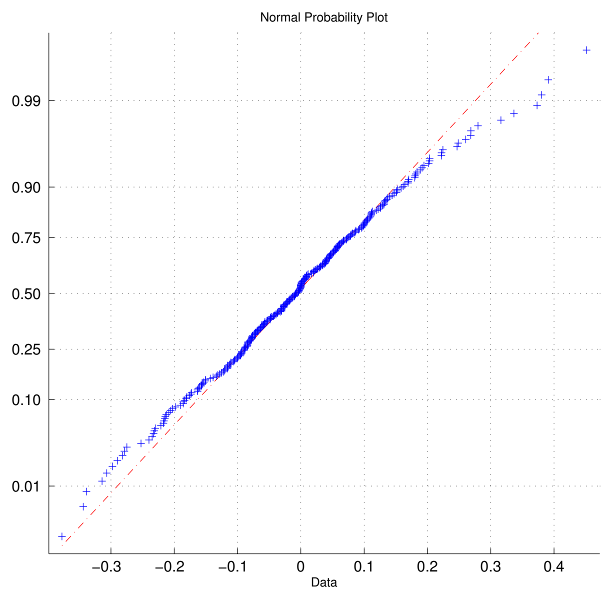
We also visualize histograms before & after power normalization in Fig. 2. These plots reveal some interesting properties about both FV and VLAD.

1) Missing code words as side information: One obvious common pattern in Fig. 2 is the existence of a tall bar at the x-axis' zero point, i.e., many values are *exactly 0 or extremely close to 0*. Table III shows the percentages of zero entries in VLAD vectors. From Table III, when VLAD is used, the percentage of zero entries is high (more than 10% entries are zero in 5 datasets). When FV is used, the percentage of zero entries is low (below 1% in all datasets). However, the FV feature values are concentrated closer to 0 than VLAD feature values, as shown in Fig. 2.

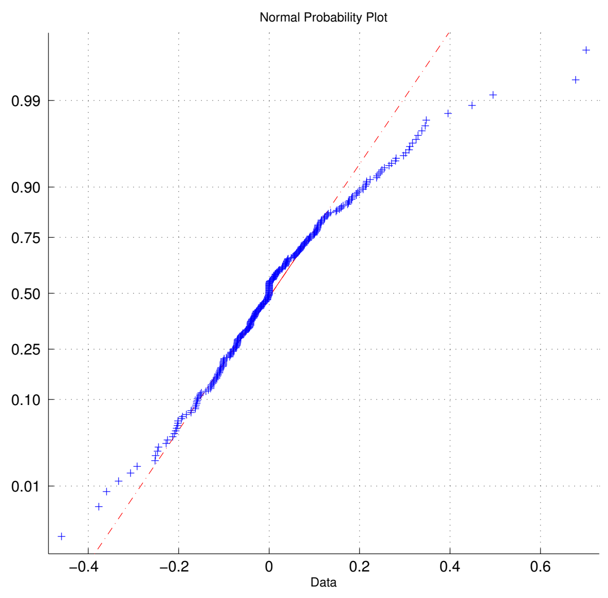
In Eq. 2, if $n_j > 0$ (i.e., number of raw motion features belonging to a code word is non-zero), it is rarely possible that its corresponding entires in $\boldsymbol{v}$ will be zero. Thus, Table III indicates that on average, more than 10% code words are missing in any single video in most datasets—a fact that carries useful side information for categorizing a video's action type.

A common VLAD practice is to encode a missing code word as $\mathbf{0} \in \mathbb{R}^d$. However, this practice completely ignores the useful side information. Let us take image classification as an example. If a visual code word corresponds to tree leafs and it does not appear in the VLAD representation of an image, it will cause zero entries in the VLAD vector. These zero entries, currently not useful for classification, are useful in ruling out certain categories. Since leafs always appear in forests, this image cannot belong to the `forest` category. Similarly, if a visual code word corresponds to a jump action in action recognition, zero entries corresponding to this specific visual word indicates that a `jump` action is not an option for this video. Thus, we need to find ways to utilize this useful side information (zero entries).

2) Bimodality and Normality from power normalization: If we ignore the tall bars at zero, the histograms after power normalization in Fig. 2b clearly show bimodal distributions for VLAD values. Although Fig. 2 only visualizes one dimension, we observe the same phenomenon in all 10 other randomly chosen VLAD dimensions in all datasets. The FV values show different patterns in Fig. 2a. After power normalization, the distribution of FV values still has one mode (i.e., unimodal). Fig. 2 shows results for the UCF101 dataset, however, we

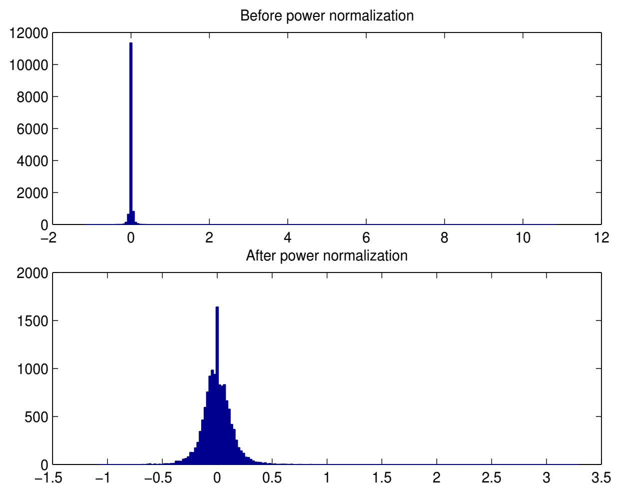


(a) With root descriptors

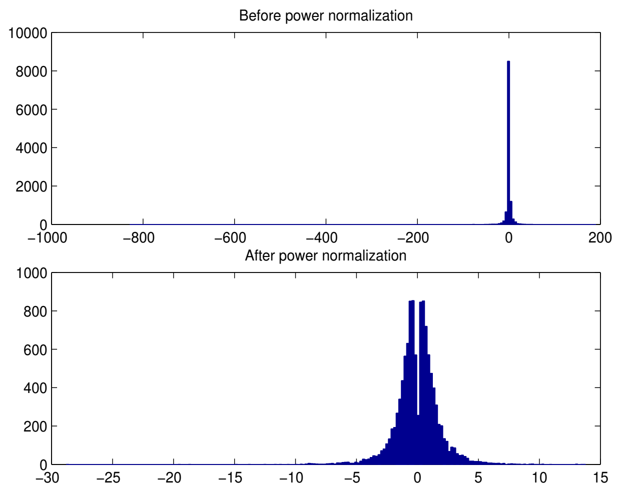


(b) Without root descriptors

Fig. 1. Normal probability plots of one dimension (7984-th) of all videos in the UCF sports dataset. Features are generated using FV, with $K = 128$ and power normalization. (1a) uses root descriptors, but (1b) does not. Best if viewed in color.



(a) FV



(b) VLAD

Fig. 2. Distribution of values in one dimension (13579-th) of all videos in the UCF101 dataset. Features are generated with $K = 128$ and power normalization. (2a) uses FV, (2b) uses VLAD. In each subfigure, the top and bottom figures are generated without and with power normalization, respectively.

observe similar results for other datasets.

In the unimodal histograms, almost all values squeeze in a small range around the zero point, meaning that the features have small discriminative power, because a zero feature value will have no effect to SVM classification. In other words, most of the feature values will have no or very small impact during classification. The bimodal distributions are more discriminative because these very small feature values are pushed away from the origin, hence will have larger impact in classification. This phenomenon might explain why the simple power normalization technique has significantly improved action recognition results for VLAD.

We also noticed that the power normalization converted highly peaked distributions into histograms that are more spread out, in other words, more Gaussian like in the Fisher Vector case. Tracing back to the discussions in Sec. IV-B, we conjecture that power normalization helps FV by making its unimodal distribution closer to the Gaussian distribution.

3) Scale = Importance?: One final important fact we observe is that the scales of different dimensions in the FV or VLAD vector are diverse. Fig. 3a shows the histograms of maximum values of all VLAD dimension for the HMDB51 dataset. It is clear that the scales of different dimensions vary significantly.

A dimension with larger scale will (in practice) be more important in classification than a dimension with smaller

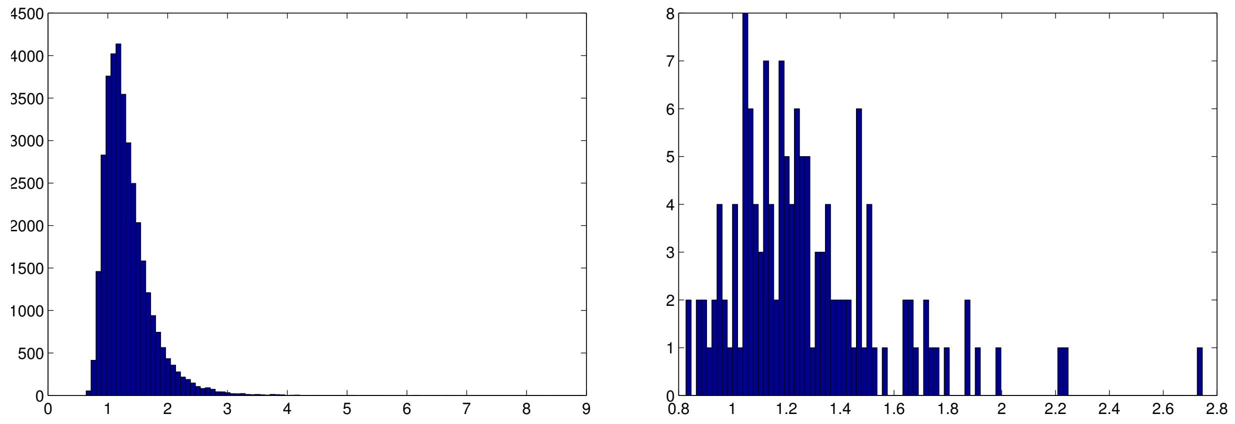


Fig. 3. Distribution of maximum values of all dimensions in the HMDB51 dataset. Values are generated using Fisher Vectors, with $K = 128$, root descriptors, and power normalization. (3a) shows the distributions of all FV dimensions, while (3b) shows only those dimensions correspond to the largest eigenvalue in PCA.

scales. However, there is no evidence that dimension scale variations are directly related to their importance levels for categorizing actions. For example, we examined the scales of only those dimensions corresponding to the eigenvector with the largest eigenvalue, which is shown in Fig. 3b. This subset is considered as containing the most information in PCA, but the resulting histogram has similar shape and range as those shown in Fig. 3a. In both figures, the values are concentrated in the range $[0.8\ 2]$ and both modes are around 1.3. Thus, we believe that $scale \neq importance$, and it is a good practice to remove the scale differences in different dimensions.

In short, based on careful analyses of VLAD's data distribution properties and the maxent interpretation, we suggest the following good practices:

- Use root descriptor if it can improve normality;
- Make good use of the side information (i.e., which code words are missing in a video?);
- Take advantage of the bimodal distribution; and,
- Remove the scale differences.

V. Bimodal encoding and compression

In this section, we propose three transformation methods that incorporate various practices suggested in Sec. IV, and empirically evaluate their performance. We show that what we name as the *bimodal encoding* is the most effective in experiments. We will also propose a combination of feature selection and compression to handle large-scale datasets.

A. Encoding transformations

These encodings are all per-dimension transformations. Suppose x a variable representing one dimension in FV or VLAD (after power and ℓ_2 normalization), we denote $\overline{x}(>0)$ and $\underline{x}(<0)$ as the maximum and minimum value of this dimension observed in the training set, respectively. The encoding transformations are defined as:

- *Even-Scale*. x is transformed as follows:
$$x \leftarrow \frac{x}{\overline{x} - \underline{x}}. \tag{18}$$
The new x has varying minimum and maximum values in each dimension, but the difference between minimum and maximum in each dimension is always 1. Note that 0 remain unchanged.
- *Side-Info*. x is transformed as follows:
$$x \leftarrow \frac{x - \underline{x}}{\overline{x} - \underline{x}}. \tag{19}$$
Note that 0 will be transformed to $\frac{-\underline{x}}{\overline{x}-\underline{x}} \neq 0$, thus it is a simple way to incorporate the side information. The new range is $[0\ 1]$ in all dimensions.
- *Bimodal*. x is transformed as follows:
$$x \leftarrow \begin{cases} x/\overline{x} & \text{if } x \geq 0 \\ x/\underline{x} & \text{if } x < 0. \end{cases} \tag{20}$$
This encoding handles the bimodality explicitly and fix scales to 1 in both sides of 0. The new range is $[-1\ 1]$. Note that 0 remain unchanged.

All three transformations are very computational efficient. In order to get the parameters $\overline{x}$ and $\underline{x}$, we just need 1 comparison and 1 assignment operation for each feature value x. In order to apply the transformations, only 1 or 2 minus and 1 division operation are needed for each x.

B. Comparison of encodings and with the state-of-the-art

Evaluation results for different encoding transformations are presented in Table IV. The baselines for FV and VLAD are the `combined results` row in Tables I and II, respectively. Then, the three proposed transformations are applied and their results listed as three rows in Table IV.

We observe different outcomes for FV and VLAD once again, and different trends are observed for large and small datasets. In detail,

- In the FV results, all three proposed simple transformations successfully improved over the results in the `none` row. The improvements are consistent in the first 4 datasets, which are larger one (with more than 1000

TABLE IV
ACCURACY OF DIFFERENT ENCODING CHOICES AND COMPARISON WITH STATE-OF-THE-ART RESULTS. THE ROWS LABELED AS **NONE** ARE BASELINE RESULTS FOR FV AND VLAD, WHICH ARE COPIED FROM THE **COMBINED RESULTS** ROW IN TABLES I AND II, RESPECTIVELY.

FV results							
	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
none	84.56	57.21	90.28	90.82	92.00	62.71	**91.37**
even-scale	85.35↑	57.89↑	90.96↑	**91.45**↑	91.33↓	54.06↓	76.52↓
side-info	84.79↑	**58.45**↑	90.58↑	91.18↑	**92.67**↑	62.80↑	91.08↓
bimodal	85.17↑	58.43↑	90.94↑	91.27↑	90.00↓	57.41↓	87.82↓
VLAD results							
	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
none	84.10	55.56	89.86	90.45	92.67	59.41	88.73
even-scale	83.74↓	56.03↑	89.86=	91.09↑	92.67=	49.35↓	86.01↓
side-info	83.76↓	57.04↑	89.60↓	90.18↓	92.00↓	59.52↑	88.72↓
bimodal	84.13↑	56.86↑	90.22↑	91.36↑	92.67=	52.65↓	86.96↓
Previous state-of-the-art							
	85.90[27]	57.2[13]	**91.2**[13]	85.4[6]	89.1[6]	**64.3**[13]	91.1[13]

training videos). However, in the rest three small datasets (with fewer than 1000 training videos), these transformations mostly lead to lower accuracy. Results on small datasets, however, are less reliable than those on large-scale ones;

- In the VLAD results, the proposed transformations are effective in half cases, but leads to worse results in the other half. Similar to the FV results, they are more effective in the larger datasets;
- *Bimodal is the most effective transformation*. It consistently improves recognition accuracy in the 4 larger datasets; and,
- Side-info and Even-scale are effective when FV is used, but not for VLAD. However, Side-info improves VLAD's result on HMDB51 by 1.5%, which suggests that when the recognition problem is difficult, it is still effective. The fact that Even-scale is effective for FV also confirms our observations: $scale \neq importance$.

We also compare the proposed simple transformations with the state-of-the-art results in the literature, which forms the last row in Table IV. The transformations achieved best results in 4 datasets. The best result of the proposed transformations is only 0.55% worse in the UCF101 dataset, 0.24% worse in the UCF50 dataset than the results in [27] and [13], respectively. In the Hollywood2 dataset, it is 1.5% worse than the result in [13]. However, we want to point out that: in addition to the same ITF raw motion features, [13] also uses extra human detection results to improve action recognition.

In short, on one hand the proposed transformations are extremely simple and computationally inexpensive. On the other hand, their recognition accuracy compare favorably with state-of-the-art methods which require additional expensive computations.

C. Compress high-dimensional vectors

FV and VLAD vectors are usually very high dimensional, e.g., when $K = 128(256)$, FV (VLAD) is around 51200 dimensions in this paper. When a large number of videos are available (e.g., UCF101 with 13320 videos), memory storage and CPU time requirements might make FV / VLAD become too costly for practical use. Many feature compression methods have been proposed to handle this problem (e.g., product quantization [15]) in the image categorization domain.

However, since there are far more images than videos, we probably do not need a very large compression ratio or sophisticated feature compression methods in the video domain. Two encodings on top of bimodal can compress FV or VLAD with satisfactory results:

$$\text{1-bit}: x \leftarrow \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}, \tag{21}$$

$$\text{2-bit}: x \leftarrow \begin{cases} 1 & x > 0.5 \\ 0.5 & 0 \leq x \leq 0.5 \\ -0.5 & -0.5 \leq x < 0 \\ -1 & x < -0.5 \end{cases}. \tag{22}$$

They quantize a real-value into 2 and 4 discrete values (i.e., 1 and 2 bits to store), respectively, corresponding to 32 and 16 times compression, because a `float` requires 32 bits. The 1-bit encoding is the sign code in [42], and 2-bit is a simple extension of 1-bit we propose in this paper.[1]

Recently, it is shown that for the Fisher Vector representation, we can select a subset of feature dimensions using mutual information (MI) as the selection score, which is not only much more efficient than compression methods, but achieves higher image classification accuracy [23]. In this section, we also show that for video action recognition, the MI-based feature selection is also efficient and effective. The feature selection results with different selection ratios ($c = \frac{4}{5}, \frac{2}{3}, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$) are shown in Table V.

In Table V, we use the bimodal transformation as the baseline algorithm, whose results are same as the two rows labeled as `bimodal` for FV and VLAD in Table V, respectively. When FV is concerned, the MI-based feature selection strategy almost consistently improves the baseline algorithm, although they only use 50% to 80% of the original feature dimensions in the baseline. These observations corroborates the statement in [23] in the video recognition domain: *many dimensions in the FV representation are noise, and removing them will even improve recognition accuracy.* The MI-based selection strategy, however, is not as effective in the VLAD case. Another observation is that similar to image classification [23], when c reduces gradually to a small value ($\frac{1}{16}$, or choosing only 6% of original dimensions), the recognition accuracy degradation is also gradual (for both FV and VLAD).

Next we combine MI-based feature selection with 1- or 2-bit compression. On top of the bimodal encoding, we fix the feature selection ratio to $\frac{2}{3}$, and then followed by the 2-bit compression. These two steps results in an overall compression ratio $\frac{1}{24}(= \frac{2}{3} \times \frac{1}{16})$. Similarly, if 1-bit is used, the overall compression ratio is $\frac{1}{48}$. These results are also shown in Table V.

[1]The two parameter values (0.5 and 0.5) in Eq. 22 are chosen in an ad-hoc manner. Since they work reasonably well in practice, we did not use more sophisticated methods to adjust these parameters.

TABLE V
ACCURACY OF FEATURE SELECTION AND 1- OR 2-BIT COMPRESSION METHODS. c MEANS THE SELECTION OR COMPRESSION RATIO. THE BASELINES (**BIMODAL**) ARE THE TWO BIMODAL ROWS IN TABLE IV.

FV results							
	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
bimodal	85.17	58.43	90.94	91.27	90.00	57.41	87.82
$c = 4/5$	85.36↑	58.14↓	91.28↑	91.82↑	89.33↓	58.18↑	88.76↑
$c = 2/3$	85.39↑	58.41↓	91.14↑	91.82↑	90.67↑	57.75↑	89.19↑
$c = 1/2$	85.21↑	57.76↓	91.22↑	91.45↑	92.00↑	57.63↑	88.83↑
$c = 1/4$	84.13↓	57.12↓	90.54↓	90.45↓	92.67↑	56.36↓	89.47↑
$c = 1/8$	83.10↓	54.27↓	89.48↓	89.00↓	90.00=	54.50↓	88.79↑
$c = 1/16$	81.05↓	51.00↓	88.10↓	88.55↓	88.67↓	52.80↓	87.50↓
1-bit ($c = \frac{1}{48}$)	86.14↑	58.63↑	91.60↑	90.73↓	92.00↑	54.29↓	87.30↓
2-bit ($c = \frac{1}{24}$)	86.09↑	58.45↑	91.60↑	91.18↓	92.67↑	56.23↓	87.75↓
VLAD results							
	UCF101	HMDB51	UCF50	Youtbue	UCF sports	Hollywood2	Olympic sports
bimodal	84.13	56.86	90.22	91.36	92.67	52.65	86.96
$c = 4/5$	84.09↓	56.38↓	90.36↑	91.36=	92.67=	53.04↑	87.57↑
$c = 2/3$	83.93↓	55.71↓	90.36↑	91.45↑	92.00↓	52.19↓	88.24↑
$c = 1/2$	83.26↓	54.77↓	89.92↓	91.45↑	91.33↓	51.35↓	87.40↑
$c = 1/4$	81.58↓	51.70↓	88.78↓	91.18↓	92.67=	49.33↓	86.81↓
$c = 1/8$	79.51↓	48.45↓	87.14↓	89.64↓	91.33↓	47.43↓	86.33↓
$c = 1/16$	76.53↓	45.10↓	84.56↓	87.18↓	89.33↓	44.87↓	86.21↓
1-bit ($c = \frac{1}{48}$)	81.36↓	51.92↓	88.24↓	88.36↓	92.67=	48.03↓	84.55↓
2-bit ($c = \frac{1}{24}$)	81.90↓	53.36↓	89.14↓	89.45↓	92.00↓	49.88↓	86.56↓

We observe that in the FV case, 2-bit usually achieves higher or comparable accuracy than the original bimodal encoding. It is worth noting that although in Table IV, bimodal is worse than state-of-the-art in the UCF101 and UCF50 datasets, the 2-bit accuracy in Table V are in fact higher than them, in spite of the fact that we only use 4% memory as the original bimodal encoding! The 1-bit encoding compresses the vector even shorter, and achieves comparable results as 2-bit.

In the VLAD case, feature selection and 1- or 2-bit compression is not as effective as in the FV case. However, it is clear that when storage is a serious issue, both strategies are very useful to reduce storage requirements with only small reduction in accuracy.

One more fact that is clear from Table V is that combining 1- or 2-bit compression with MI-based feature selection is more effective than using feature selection along (e.g., by comparing the row $c = 1/16$ with the row `1-bit`.)

Finally, based on the observations in this section, we suggest the following good practices:

- Use the proposed transformations when Fisher Vector is used; try them if VLAD is used. Given the fact that all three transformations incur only negligible computational costs and are very easy to implement, they are convenient to apply;
- Bimodal encoding can be the default choice, which usually gives the best result among the three proposed transformation; and,
- Use the combination of MI-based feature selection and the 2-bit (or 1-bit) compression if FV is used. If VLAD is used, use this combination if storage becomes an issue.

VI. Conclusions

In this paper, we focused on action recognition in videos using two types of high dimensional representation (FV and VLAD). Three lines of contributions were presented. First, we empirically evaluated several improvement techniques to FV and/or VLAD, including power normalization, root descriptors, residue normalization, and spatio-temporal pyramid. We also proposed a new average pooling strategy for VLAD. Extensive evaluations on 7 video datasets ranging from small- to large-scale lead to a few good practices for action recognition in Sec. III. A new maxent interpretation for VLAD was also proposed. This interpretation, together the scrutiny of the distribution of FV or VLAD vectors, show that normality and bimodality are essential for high recognition accuracy. These analyses are beneficial for inventing better recognition algorithms.

Second, based on the data distribution properties, we proposed three very simple, yet efficient and effective transformation to further improve FV or VLAD. Our evaluations verify that they can improve action recognition accuracy in medium and large scale datasets, with only negligible computational overhead.

Third, we proposed a combination of MI-based feature selection and 2-bit (or 1-bit) compression to reduce the length of a video representation to only 2%–4% of its original FV or VLAD vector, yet achieving comparable or even higher accuracy than using the original representation.

As a summary of our evaluations, proposals and findings, we suggest the following good practices for action video encoding:

- Use FV or VLAD instead of bag-of-features. Default parameter could be $K = 128$ in FV and $K = 256$ in VLAD. Use STP if the dataset size is not small;
- Use the proposed average pooling for VLAD. Always use power normalization;
- Try residue normalization and root descriptors. If they are useful, combine them with other techniques;
- Use the three proposed transformations, especially the bimodal transformation;, and,
- Use the combination of 1- or 2-bit compression with the MI-based feature selection ($c = \frac{1}{2}$ or $\frac{2}{3}$), especially if the dataset at hand is large-scale.

Some future research topics are already discussed. For example, instead of using PCA for dimension reduction, we want to find maximum entropy linear feature function, which may improve the feature quality and consequently recognition accuracy. The bimodal encoding does not use the side information ($x = 0$), which can be further improved.

One promising future work is to include information from multiple views [43], which is not currently utilized at all in our framework. Beyond action recognition which work with pre-segmented video clips, it is also very important to detect actions in long videos [44], and the findings in this paper might be useful for video action detection.

References

[1] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2/3, pp. 107–123, 2005. 1

[2] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional SIFT descriptor and its application to action recognition," in *ACM Multimedia*, 2007, pp. 357–360. 1

[3] K. Schindler and L. van Gool, “Action snippets: How many frames does human action recognition require?” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2008. 1

[4] J. Liu, J. Luo, and M. Shah, “Recognizing realistic actions from videos “in the wild”,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 1996–2003. 1, 4

[5] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: a large video database for human motion recognition,” in *Proc. IEEE Int’l Conf. on Computer Vision*, 2011, pp. 2556–2563. 1, 4

[6] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Dense trajectories and motion boundary descriptors for action recognition,” *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, May 2013. 1, 2, 4, 10

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004. 1

[8] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *Proc. IEEE Int’l Conf. on Computer Vision*, vol. 2, 2003, pp. 1470–1477. 1

[9] I. Laptev, M. Marszaek, C. Schmid, and B. Rozenfeld, “Learning realistic human actions from movies,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2008. 1, 2, 3

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, Aug 2008. 1, 4

[11] S. Sadanand and J. J. Corso, “Action bank: A high-level representation of activity in video,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 1234–1241. 1

[12] M. Jain, H. Jégou, and P. Bouthemy, “Better exploiting motion for better action recognition,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 2555–2562. 2

[13] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proc. IEEE Int’l Conf. on Computer Vision*, 2013, pp. 3551–3558. 2, 4, 10

[14] L. Liu, L. Shao, X. Li, and K. Lu, “Learning spatio-temporal representations for action recognition: A genetic programming approach,” *IEEE Trans. Cybernetics*, 2015. 2

[15] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, “Image classification with the fisher vector: Theory and practice,” *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013. 2, 3, 4, 10

[16] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, “Aggregating local images descriptors into compact codes,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012. 2, 3, 4

[17] X. Wang, L. Wang, and Y. Qiao, “A comparative study of encoding, pooling and normalization methods for action recognition,” in *Proc. Asia Conf. Computer Vision*, 2012. 2, 5

[18] C. Sun and R. Nevatia, “Large-scale web video event classification by use of fisher vectors,” in *IEEE Workshop on the Applications of Computer Vision*, 2013, pp. 15–22. 2

[19] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the Fisher kernel for large-scale image classification,” in *Proc. European Conf. Computer Vision*, ser. LNCS 6314, 2010, pp. 143–156. 2, 3

[20] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 2911–2918. 2, 3

[21] J. Delhumeau, P.-H. Gosselin, H. Jégou, and P. Pérez, “Revisiting the VLAD image representation,” in *ACM Multimedia*, 2013. 2, 3

[22] D. Oneata, J. Verbeek, and C. Schmid, “Action and event recognition with Fisher Vectors on a compact feature set,” in *Proc. IEEE Int’l Conf. on Computer Vision*, 2013, pp. 1817–1824. 2

[23] Y. Zhang, J. Wu, and J. Cai, “Compact representation for image classification: To choose or to compress?” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 907–914. 2, 3, 10

[24] J. Wu, Y. Zhang, and W. Lin, “Towards good practices for action video encoding,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 2577–2584. 2, 4, 7

[25] J. Wu, W.-C. Tan, and J. M. Rehg, “Efficient and effective visual codebook generation using additive kernels,” *Journal of Machine Learning Research*, vol. 12, pp. 3097–3118, Nov 2011. 4

[26] K. Soomro, A. R. Zamir, and M. Shah, “UCF101: A dataset of 101 human action classes from videos in the wild,” CRCV-TR-12-01, University of Central Florida, Tech. Rep., 2012. 4

[27] Y.-G. Jiang, J. Liu, A. R. Zamir, I. Laptev, M. Piccardi, M. Shah, and R. Sukthankar, “THUMOS: The first international workshop on action recognition with a large number of classes,” 2013. 4, 10

[28] K. K. Reddy and M. Shah, “Recognizing 50 human action categories of web videos,” *Machine Vision and Applications*, vol. 24, no. 5, pp. 971–981, 2013. 4

[29] M. Rodriguez, J. Ahmed, and M. Shah, “Action MACH: A spatio-temporal maximum average correlation height filter for action recognition,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2008, pp. 1–8. 4

[30] M. Marszalek, I. Laptev, and C. Schmid, “Actions in context,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 2929–2936. 4

[31] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, “Modeling temporal structure of decomposable motion segments for activity classification,” in *Proc. European Conf. Computer Vision*, ser. LNCS 6312, 2010, pp. 392–405. 4

[32] L. Shao, X. Zhen, D. Tao, and X. Li, “Spatio-temporal Laplacian pyramid coding for action recognition,” *IEEE Trans. Cybernetics*, vol. 44, no. 6, pp. 817–827, 2014. 4

[33] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013. 4

[34] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732. 4

[35] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proc. Advances in Neural Information Processing Systems 27*, 2014, pp. 568–576. 4

[36] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, “Modeling spatial-temporal clues in a hybrid deep learning framework for video classification,” *arXiv:1504.01561*, 2015. 5

[37] L. Liu, L. Shao, F. Zheng, and X. Li, “Realistic action recognition via sparsely-constructed Gaussian processes,” *Pattern Recognition*, vol. 47, no. 12, pp. 3819–3827, 2014. 5

[38] C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 6

[39] R. He, B. Hu, X. Yuan, and W.-S. Zheng, “Principal component analysis based on non-parametric maximum entropy,” *Neurocomputing*, vol. 73, no. 10-12, pp. 1840–1852, 2010. 6

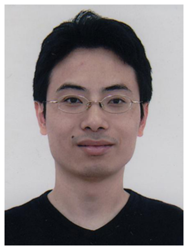
[40] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006. 6

[41] J. M. Chambers, W. S. Cleveland, P. A. Tukey, and B. Kleiner, *Graphical Methods for Data Analysis*. Duxbury Press, 1983. 7

[42] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, “Large-scale image retrieval with compressed fisher vectors,” in *Proc. IEEE Int’l Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 3384–3391. 10

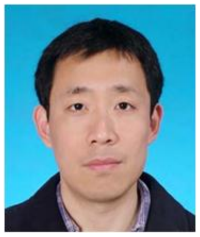
[43] A.-A. Liu, Y.-T. Su, P.-P. Jia, Z. Gao, T. Hao, and Z.-X. Yang, “Multipe/single-view human action recognition via part-induced multitask structural learning,” *IEEE Trans. Cybernetics*, vol. 45, no. 6, pp. 1194–1208, 2015. 11

[44] G. Yu, J. Yuan, and Z. Liu, “Action search by example using randomized visual vocabularies,” *IEEE Trans. on Image Processing*, vol. 22, no. 1, pp. 377–390, 2013. 11



Jianxin Wu received his BS and MS degrees in computer science from Nanjing University, and his PhD degree in computer science from the Georgia Institute of Technology. He is currently a professor in the Department of Computer Science and Technology at Nanjing University, China, and is associated with the National Key Laboratory for Novel Software Technology, China. He was an assistant professor in the Nanyang Technological University, Singapore, and has served as an area chair for ICCV 2015 and senior PC member for AAAI 2016. His research interests are computer vision and machine learning. He is a member of the IEEE.

Yu Zhang received his BS and MS degrees in telecommunications engineering from Xidian University, and his PhD degree in computer engineering from Nanyang Technological University. He is currently a senior research engineer in the Advanced Digital Sciences Center, Singapore. His research interest is computer vision.



Weiyao Lin received the B.E. degree from Shanghai Jiao Tong University, China, in 2003, the M.E. degree from Shanghai Jiao Tong University, China, in 2005, and the Ph.D degree from the University of Washington, Seattle, USA, in 2010, all in electrical engineering. Currently, he is an associate professor at the Department of Electronic Engineering, Shanghai Jiao Tong University. His research interests include video surveillance, computer vision and video coding & compression.