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

Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice

Xiaojiang Peng · Limin Wang · Xingxing Wang · Yu Qiao

the date of receipt and acceptance should be inserted later

Abstract Video based action recognition is one of the important and challenging problems in computer vision research. Bag of Visual Words model (BoVW) with local features has become the most popular method and obtained the state-of-the-art performance on several realistic datasets, such as the HMDB51, UCF50, and UCF101. BoVW is a general pipeline to construct a global representation from a set of local features, which is mainly composed of five steps: (i) feature extraction, (ii) feature pre-processing, (iii) codebook generation, (iv) feature encoding, and (v) pooling and normalization. Many efforts have been made in each step independently in different scenarios and their effects on action recognition is still unknown. Meanwhile, video data exhibits different views of visual pattern, such as static appearance and motion dynamics. Multiple descriptors are usually extracted to represent these different views. Fusing these multiple descriptors is crucial for boosting the final performance of action recognition system. Many feature fusion methods have been developed in other areas and their influence on action recognition has never been investigated before. This paper aims to provide a comprehensive study of all steps in BoVW and different fusion methods, and uncover some good practice to produce a state-of-the-art action recognition system. Specifically, we explore two kinds of local features, ten kinds of encoding methods, eight kinds of pooling and normalization strategies, and three kinds of fusion methods. We conclude that every step is crucial for contributing to the final recognition rate and improper choice in one of the steps may counteract the performance improvement of other steps. Furthermore, based on our comprehensive study, we propose a simple yet effective representation, called *hybrid representation*, by exploring the complementarity of different BoVW frameworks and local descriptors. Using this representation, we obtain the state-of-the-art on the three challenging datasets: HMDB51 (61.1%), UCF50 (92.3%), and UCF101 (87.9%).

Keywords Action recognition · Bag of Visual Words · Fusion methods · Survey

Xiaojiang Peng
School of Information Sciences and Technology, Southwest Jiaotong University, Chengdu, China
E-mail: xiaojiangp@gmail.com

Limin Wang
Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong SAR, China.
E-mail: 07wanglimin@gmail.com

Xingxing Wang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.
E-mail: wangxingxing.hz@gmail.com

Yu Qiao
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China.
E-mail: yu.qiao@siat.ac.cn

1 Introduction

Human action recognition [1,43] has become an important area in computer vision research, whose aim is to automatically classify the action ongoing in a video. It is one of the challenging problems in computer vision for serval reasons. Firstly, there are large intra-class variations in the same action class, caused by various motion speeds, viewpoint changes, and background clutter. Secondly, the identification of an action class is related to many other high-level visual clues, such as human pose, interacting objects, and scene class. These related problems are very difficult themselves. Furthermore, the determination of temporal extent for

an actions is more subjective than a static object, which means there is no precise definition about when an action starts and finishes. Finally, the high dimension and low quality of video data usually add difficulty to developing robust and efficient recognition algorithm.

Early approaches interpret an action as a set of space-time trajectories of 2-dimensional or 3-dimensional points of human joints [55,32,7,60]. These methods usually need dedicate techniques to detect body parts or track them at each frame. However, the detection and tracking of body part is still an unsolved problem in realistic videos. Recently, recognition methods using local spatiotemporal features [24,25,46,53] have become the main stream and obtained the state-of-the-art performance on many datasets [47]. These methods do not require algorithms to detect human body, which treat the action volume as a rigid 3D-object and extract appropriate features to describe the patterns of each 3D volume. They are robust to background clutter, illumination changes, and noise.

Bag of Visual Words (BoVW) framework with local features and its variants [48,57,20,29,34] have dominated the research work of action recognition and showed their effectiveness in the recent THUMOS'13 Action Recognition Challenge [18]. As shown in Figure 1, the pipeline of BoVW for video based action recognition consists of five steps: (i) feature extraction, (ii) feature pre-processing, (iii) codebook generation, (iv) feature encoding, and (v) pooling and normalization. In each step, many efforts have been made and several progress has been obtained. Regarding local features, many successful feature extractors (e.g. STIPs [24], Dense Trajectories [46]) and descriptors (e.g. HOG [25], HOF [25], MBH [46]) have been designed for representing the visual patterns of cuboid. Feature pre-processing technique mainly de-correlates these descriptors to make the following representation learning more stable. For codebook generation, it aims to describe the local feature space and provide a partition (e.g. k-means [3]) or generative process (e.g. GMMs [3]) for local descriptor. Feature encoding is a hot topic in image classification and many alternatives have been developed for effective representation and efficient implementation (see good surveys [9,14]). Max pooling [61] and sum pooling [64] are usually used to aggregate information from a spatiotemporal region. For normalization methods, typical choices include ℓ_1-normalization [64], ℓ_2-normalization [50], power normalization [35], and intra normalization [2]. *How to make decision in each step to obtain the best pipeline of BoVW for action recognition* still remains unknown and needs to be extensively explored.

Meanwhile, unlike static image, video data exhibits different views of visual pattern, such as appearance, motion, and motion boundary, and all of them play important roles in action recognition. Therefore, multiple descriptors are usually extracted from a cuboid and each descriptor corresponds to the specific aspect of the visual data [46,25]. BoVW is mainly designed for a single descriptor and ignores the problem of fusing multiple descriptors. Many research works have been devoted to fusing multiple descriptor for boosting performance [11,44,41,47,6]. Typical fusion methods include descriptor level fusion [25,54], representation level fusion [48,46], and score level fusion [41,30]. For descriptor level fusion, multiple descriptors from the same cuboid are concatenated as a whole one and fed into BoVW framework. For representation level fusion, the fusion is conducted in the video level, where each descriptor is firstly fed into BoVW framework independently and the resulting global representations are then concatenated to train a final classifier. For score level fusion, each descriptor is separately input into BoVW framework and used to train a recognition classifier. Then the scores from multiple classifiers are fused using arithmetic mean or geometric mean. In general, these fusion methods are developed in different scenarios and adapted for action recognition by different works. *How these fusion methods influence the final recognition of BoVW framework and whether there exists a best one for action recognition* is an interesting question and well worth of a detailed investigation.

Several related study works have been performed about encoding methods for image classification [9,14] and action recognition [54]. But these study works are with image classification task or lacking full exploration of all steps in BoVW framework. Meanwhile, the study work of action recognition [54] is limited regarding the evaluation dataset and ignores the influence of fusion methods. This article aims to *provide a comprehensive study of all steps in BoVW and different fusion methods, and uncover some good practice to produce a state-of-the-art action recognition system.* Our work is mainly composed of three parts:

Exploration of BoVW. We place an emphasis on extensively explorations about all components in BoVW pipeline and discovery of useful practice tips. Specifically, we investigate two widely-used local features, namely Space Time Interest Points (STIPs) with HOG, HOF [24], and Improved Dense Trajectories (iDTs) with HOG, HOF, MBH [47]. For feature encoding methods, the current approaches can be roughly classified into three categories: (i) voting based encoding methods, (ii) reconstruction based encoding methods, (iii) super vector based encoding

methods. For each type of encoding methods, we choose several representative approaches and totally analyze ten encoding methods. Meanwhile, we explore the relations among these different encoding methods and provide an unified and generative perspective over these encoding methods. We fully explored eight pooling and normalization strategies for each encoding method. From our extensive study of different components in BoVW, server good practice can be concluded:

- Dense features with more descriptors are more informative in capturing the content of video data and suitable for action recognition. Meanwhile, dense features may exhibit different properties with sparse features with respect to variations of BoVW such as codebook size and encoding methods.
- Data pre-processing is an important step in BoVW pipeline and able to greatly improve the final recognition performance.
- Basically, high dimensional representation of super vector is more effective and efficient than the other two types of encoding methods.
- Pooling and normalization is a crucial step in BoVW, whose importance may not be highlighted in previous studies. Sum pooling with power ℓ_2-normalization is the best choice during all the possible combinations.
- In above, every step is crucial for contributing to the final recognition rate. Improper choice in one of the steps may counteract the performance improvement of other steps.

Investigation of Fusion Methods. As combination of multiple descriptors is very crucial for performance improvement, we also investigate the influence of different fusion methods inour designed action recognition system. Specifically, we study three kinds of fusion methods, namely descriptor level fusion, representation level fusion, and descriptor level fusion. We find that the way different descriptors correlate with each other determines the effectiveness of fusion methods. The performance gain obtained from fusing multiple descriptors mainly owns to their complementarity. We observe that this complementarity is not only with multiple descriptors, but also with multiple BoVW models. Based on this view, we propose a new representation, called *hybrid representation*, combining the outputs of multiple BoVW models of different descriptors. This representation utilizes the benefit of each BoVW and fully considers the complementarity among them. In spite of its simplicity, this representation turns out to be effective for improving final recognition rate.

Comparison with the State of the Art. Guided by the practice tips concluded from our insightful analysis of BoVW variants and feature fusion methods, we design an effective action recognition system using our proposed hybrid representation, and demonstrates its performance on three challenging datasets: HMDB51 [23], UCF50 [36], and UCF101 [40]. Specifically, we leverage the richness and effectiveness of low-level features, design a hybrid super vector, a combination of Fisher vector [35] and SVC-k, and resort to representation level fusion to boost final recognition performance. From comparison with other methods, we conclude that our recognition system reaches the state-of-the-art performance on the three datasets, and our hybrid representation acts as a new baseline for further research of action recognition.

The rest of this paper is organized as follows. In Section 2, we give an detailed description of each step in BoVW framework of action recognition system. Meanwhile, we uncover several useful techniques commonly adopted in these encoding methods, and provide a unified generative perspective over these encoding methods. Then, several fusion methods and a new representation are introduced in Section 3. Finally, we empirically evaluate the BoVW frameworks and fusion methods on three challenging datasets. We analyze these experiment results and uncover good practice for constructing a state-of-the-art action recognition system. We conclude the paper in Section 5.

2 Framework of Bag of Visual Words

As shown in Figure 1, the pipeline of Bag of Visual Words (BoVWs) framework consists of five steps: (i) feature extraction, (ii) feature pre-processing, (iii) codebook generation, (iv) feature encoding, and (v) pooling and normalization. Then the global representation is fed into a classifier such as linear SVM for action recognition. In this section, we will give detailed descriptions of the popular technical choices in each step, which are very important for constructing a state-of-the-art recognition system. Furthermore, we summarize several use techniques in these encoding methods and provide a unified generative perspective over these different encoding methods.

2.1 Feature Extraction

Low-level local features have become popular in action recognition due to their robustness to background clutter and independence on detection and tracking techniques. These local features are typically divided into two parts: detecting a local region (detector) and describing the detected region (descriptor) [49]. Many feature detectors have been developed such as

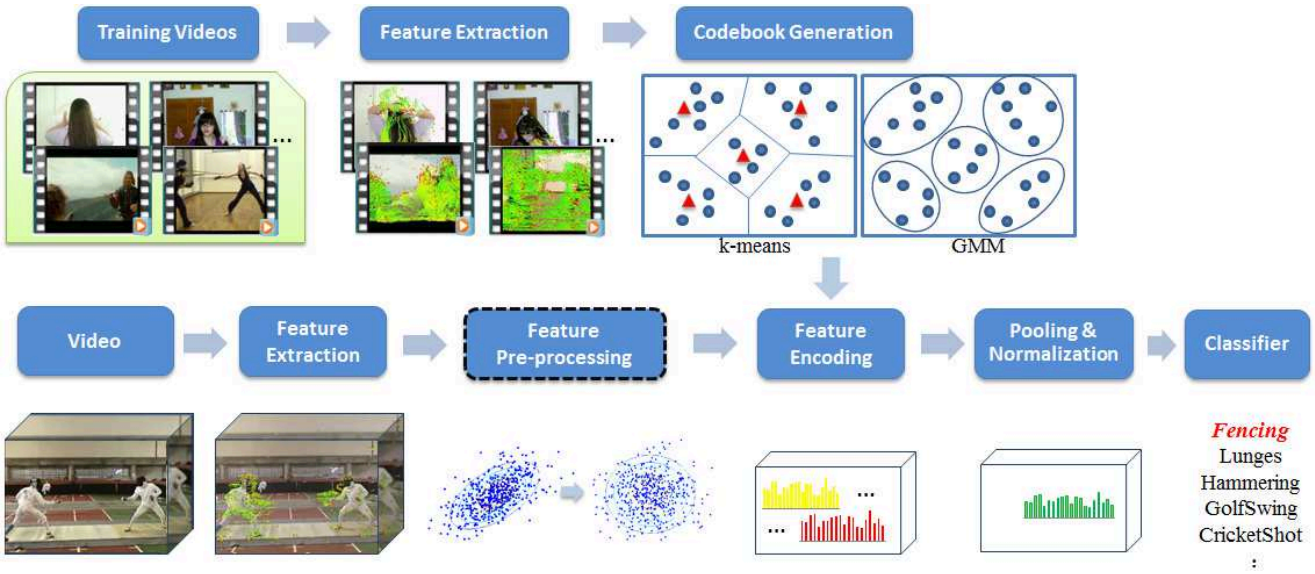


Fig. 1 The pipeline of obtaining Bag of Visual Words (BoVWs) representation for action recognition. It is mainly composed of five steps: (i) feature extraction, (ii) feature pre-processing, (iii) codebook generation, (iv) feature encoding, and (v) pooling and normalization.

不同类型的底层特征

3D-Harris [24], 3D-Hessian [56], Cuboid [10], Dense Trajectories [46], and Improved Dense Trajectories [47]. These detectors try to select locations and scales in video by maximizing certain kind of function or using dense sampling strategy. To describe the extracted region, several hand-crafted features have been designed such as Histogram of Oriented Gradients (HOG) [25, 46], Histogram of Oriented Flow (HOF) [25,46], and Motion Boundary Histogram (MBH) [46,47]. Multiple descriptors are usually adopted to represent the local region, each of which corresponds to a certain aspect of visual pattern such as static appearance, motion, and motion boundary.

Among these local features, Space Time Interest Points (STIPs) [24] and Improved Dense Trajectories (iDTs) [46] are widely used due to their easy usages and good performance. STIPs resort to 3D-Harris to extract regions of high motion salience, which resulting a set of sparse interest points. For each interest point, STIPs extracted two kinds of descriptors, namely HOG and HOF. iDTs features are an improved version from Dense Trajectories (DTs), where a set of dense trajectories are firstly obtained by tracking pixels with median filter, and five kinds of descriptors are extracted, namely trajectory shape, HOG, HOF, MBHx, and MBHy. iDTs improve the performance of DTs by taking into account camera motion correction. Generally speaking, iDTs resort to more sophisticated engineering skills and integrate much richer low-level visual cues compared with STIPs. Therefore, they represent two different kinds of low level features, namely sparse features and dense features, and may exhibit different properties with respect to variants of BoVWs.

2.2 Feature Pre-processing

The low-level local descriptors are usually high dimensional and strong correlated, which results in great challenges in the subsequent unsupervised learning such as k-means clustering and GMM training. Principal component analysis (PCA) [3] is a statistical procedure to pre-process these features, which uses orthogonal transform to map feature into a set of linearly uncorrelated variables called principal components. Typically, the number of used principal components is less than the number of original variables, thus resulting in dimension reduction. Whitening technique usually follows the PCA, which aims to ensure the feature have the same variance through different dimensions. The transform formula of pre-processing is as followings:

$$\mathbf{x} = \Lambda U^{\top}\mathbf{f}, \tag{1}$$

where $\mathbf{f} \in R^M$ is the original feature, $\mathbf{x} \in R^N$ is the PCA-whitened result, $U \in R^{M\times N}$ is the dimension reduction matrix from PCA, Λ is the diagonal whitening matrix $diag(\Lambda) = [1/\sqrt{\lambda_1}, \cdots, 1/\sqrt{\lambda_N}]$, and λ_i is the i^{th} largest eigenvalue of covariance matrix.

It is worth noting that this step is not necessary and many previous encoding approaches skip this step, such as Vector Quantization [38], Sparse Coding [61], and

Table 1 List of encoding methods and their formulations. The detailed descriptions of these encoding methods can be found in the text.

Type	Method	Formulation	Dim.
Voting based	1. Vector Quantization (VQ) / Hard Voting (HV)	$\mathbf{s}(i) = 1,\ \text{if}\ \ i = \arg\min_j ||\mathbf{x} - \mathbf{d}_j||_2^2,$ s.t. $||\mathbf{s}||_0 = 1$	K
	2. Soft-assignment (SA) / Kernel Codebook Coding (KCB)	$\mathbf{s}(i) = \frac{\exp(-\beta\|\|\mathbf{x}-\mathbf{d}_i\|\|_2^2)}{\sum_{i=1}^{K}\exp(-\beta\|\|\mathbf{x}-\mathbf{d}_i\|\|_2^2)}$	K
	3. Localized Soft Assignment (SA-k)	$\mathbf{s}(i) = \frac{\exp(-\beta\|\|\mathbf{x}-\mathbf{d}_i\|\|_2^2)}{\sum_{i=1}^{k}\exp(-\beta\|\|\mathbf{x}-\mathbf{d}_i\|\|_2^2)},\ \text{if}\ \mathbf{d}_i \in N_k(\mathbf{x}),$ s.t. $||\mathbf{s}||_0 = k$	K
	4. Salient Coding (SC)	$\mathbf{s}(i) = \sum_{j=2}^{k}(\frac{\|\|\mathbf{x}-\mathbf{d}_j\|\|_2 - \|\|\mathbf{x}-\mathbf{d}_1\|\|_2}{\|\|\mathbf{x}-\mathbf{d}_j\|\|_2}), \mathbf{d}_j \in N_k(\mathbf{x}),$ if $\ i = \arg\min_j ||\mathbf{x} - \mathbf{d}_j||_2^2,\quad$ s.t. $||\mathbf{s}||_0 = 1$	K
	5. Group Salient Coding (GSC)	$\mathbf{s}(i) = \max\{\mathbf{v}^k(i)\},$ group size $k = 1, ..., M,$ $\mathbf{v}^k(i) = \sum_{j=1}^{M+1-k} ||\mathbf{x} - \mathbf{d}_{k+j}||_2 - ||\mathbf{x} - \mathbf{d}_k||_2,$ if $\mathbf{d}_i \in N_k(\mathbf{x}),\ \text{s.t.}\ \ \|\mathbf{s}\|_0 = 1$	K
Reconstruction based	6. Orthogonal Matching Pursuit (OMP)	$\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{D}\mathbf{s}||_2^2,$ s.t. $||\mathbf{s}||_0 \leq k$	K
	7. Sparse Coding (SPC)	$\mathbf{s} = \arg\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{D}\mathbf{s}||_2^2 + \lambda||\mathbf{s}||_1$	K
	8. Locality-constrained Linear Coding (LLC)	$\mathbf{s} = \arg\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{D}\mathbf{s}||_2^2 + \lambda||\mathbf{e} \odot \mathbf{s}||_2^2,$ s.t. $\ \mathbf{1}^\top\mathbf{s} = 1$	K
	9. Local Coordinate Coding (LCC)	$\mathbf{s} = \arg\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{D}\mathbf{s}||_2^2 + \lambda||\hat{\mathbf{e}} \odot |\mathbf{s}|||_1$	K
Super vector based	10. Local Tangent-based Coding (LTC)	$\mathcal{S} = [\mathbf{s}(i), \mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i)^T\mathbf{U}_i]_{i=1}^K,$ $\mathbf{U}_i \in \mathbb{R}^{D\times C}$ is a projection matrix	$K(1+C)$
	11. Super Vector Coding (SVC)	$\mathcal{S} = [0, \mathbf{0}, \cdots, \frac{\alpha\mathbf{s}(i)}{N\sqrt{p_i}}, \frac{\mathbf{s}(i)}{N\sqrt{p_i}}(\mathbf{x} - \mathbf{d}_i), \cdots, 0, \mathbf{0}],$ where $i = \arg\min_j ||\mathbf{x} - \mathbf{d}_j||_2^2$	$K(1+D)$
	12. Fisher Vector (FV)	$\mathcal{S} = [\mathcal{G}_{\mu,1}^{\mathbf{x}}, ..., \mathcal{G}_{\mu,K}^{\mathbf{x}}, \mathcal{G}_{\sigma,1}^{\mathbf{x}}, ..., \mathcal{G}_{\sigma,K}^{\mathbf{x}}],$ where $\mathcal{G}_{\mu,i}^{\mathbf{x}} = \frac{1}{\sqrt{\pi_i}}\gamma_i(\frac{\mathbf{x}-\mu_i}{\sigma_i}),$ $\mathcal{G}_{\sigma,i}^{\mathbf{x}} = \frac{1}{\sqrt{2\pi_i}}\gamma_i[\frac{(\mathbf{x}-\mu_i)^2}{\sigma_i^2} - 1],$ $\gamma(i) = \frac{\pi_i\mathcal{N}(\mathbf{x};\mu_i,\Sigma_i)}{\sum_{j=1}^{K}\pi_j\mathcal{N}(\mathbf{x};\mu_j,\Sigma_j)}$	$2KD$
	13. Vector of Locally Aggregated Descriptors (VLAD)	$\mathcal{S} = [\mathbf{0}, \cdots, (\mathbf{x} - \mathbf{d}_i), \cdots, \mathbf{0}],$ where $i = \arg\min_j ||\mathbf{x} - \mathbf{d}_j||_2^2$	KD

Vector of Locally Aggregated Descriptor [16]. However, in our evaluation, we found this step is of great importance to improve the recognition performance.

2.3 Codebook Generation

In this section, we present the codebook generation algorithms used for the following feature encoding methods. Generally there are two kinds of approaches: (i) partitioning the feature space into regions, each of which is represented by its center, called codeword, and (ii) using generative model to capture the probability distribution of features. *k*-mean [3] is a typical method for the first type, and Gaussian Mixture Model (GMM) [3] is widely used for the second.

***k*-means.** There are many vector quantization methods such as *k*-means clustering [3], hierarchical clustering [19], and spectral clustering [31]. Among them, *k*-

使用k-means对基本动作进行归类

means is probably the most popular way to construct codebook. Given a set of local features $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, where $\mathbf{x}_m \in \mathbb{R}^D$. Our goal is to partition the feature set into K clusters $\{\mathbf{d}_1, \cdots, \mathbf{d}_K\}$, where $\mathbf{d}_k \in \mathbb{R}^D$ is a prototype associated with the k^{th} cluster. Suppose for each feature $\mathbf{x}_m$, we introduce a corresponding set of binary indicator variables $r_{mk} \in \{0, 1\}$. If descriptor $\mathbf{x}_m$ is assigned to cluster k, then $r_{mk} = 1$ and $r_{mj} = 0$ for $j \neq k$. We can then define an objective function:

$$\min \mathcal{J}(\{r_{mk}, \mathbf{d}_k\}) = \sum_{m=1}^{M} \sum_{k=1}^{K} r_{mk} \|\mathbf{x}_m - \mathbf{d}_k\|_2^2. \tag{2}$$

The problem is to find values for $\{r_{mk}\}$ and $\{\mathbf{d}_k\}$ to minimize the objective function $\mathcal{J}$. Usually, we can optimize it in an iterative procedure where each iteration involves two successive steps corresponding to optimization with respect to the r_{nk} and $\mathbf{d}_k$. The details can be found in [3].

GMM. Gaussian Mixture Model is a generative model to describe the distribution over feature space:

$$p(\mathbf{x}; \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k), \tag{3}$$

where K is mixture number, and $\theta = \{\pi_1, \mu_1, \Sigma_1, \cdots, \pi_K, \mu_K, \Sigma_K\}$ are model parameters. $\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$ is D-dimensional Gaussian distribution.

Given the feature set $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$, the optimal parameters of GMM are learned through maximum likelihood estimation $\arg\max_\theta \ln p(\mathbf{X}; \theta)$. We use the iterative EM algorithm [3] to solve this problem.

k-means algorithm performs a *hard* assignment of feature descriptor to codeword, while the EM algorithm of GMM makes *soft* assignment of feature to each mixture component based on posterior probabilities $p(k|x)$. But unlike k-means, GMM delivers not only the mean information of code words, but also the shape of their distribution.

GMM生成码本

2.4 Encoding Methods

In this section, we provide a detailed description of thirteen feature encoding methods. According to the characteristics of encoding methods, they can be roughly classified into three groups, namely (i) voting based encoding method, (ii) reconstruction based encoding method, and (iv) super vector encoding method, as shown in Table 1.

Let $\mathbf{X}$ be a set of D-dimensional local descriptors extracted from a video, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$. Given a codebook with K codewords, $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K] \in \mathbb{R}^{D \times K}$. The objective of encoding is to compute a code $\mathbf{s}$ (or $\mathcal{S}$) [1] for input $\mathbf{x}$ with $\mathbf{D}$. Table 1 lists all the formulations and dimension of encoding methods, where $\mathbf{s}(i)$ denotes the i^{th} element of $\mathbf{s}$.

X有N个D维特征，码书中有K个D维码字

2.4.1 Voting based encoding methods

Voting based encoding methods [38,12,28,13,59] are designed from the perspective of encoding process and each descriptor directly votes for the codeword using a specific strategy. A K-dimensional (K is the size of codebook) code $\mathbf{s}$ is constructed for each single descriptor to represent the votes of the whole codebook. Methods along this line include Vector Quantization(or Hard Voting) [38], Soft Assignment (or Kernel Codebook Coding) [12], Localized Soft Assignment [28], Salient Coding [13], and Group Salient Coding [59], as shown in Figure 2.

For each descriptor $\mathbf{x}$, the voting value for the codeword $\mathbf{d}_i$ can be viewed as a function of $\mathbf{x}$, namely $\mathbf{s}(i) = \phi(\mathbf{x})$. Different encoding methods differ in the formulation of $\phi(\mathbf{x})$. For encoding of Vector Quantization (VQ):

$$\text{VQ:} \quad \phi(\mathbf{x}) = \begin{cases} 1, & \text{if } \ i = \arg\min_j ||\mathbf{x} - \mathbf{d}_j||_2, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where each descriptor $\mathbf{x}$ only votes for its nearest codeword. The VQ encoding method can be viewed as a hard quantization and may cause much information loss. To encounter this problem, Soft Assignment (SA) encoding method votes for all the codewords:

$$\text{SA:} \quad \phi(\mathbf{x}) = \omega_i, \tag{5}$$

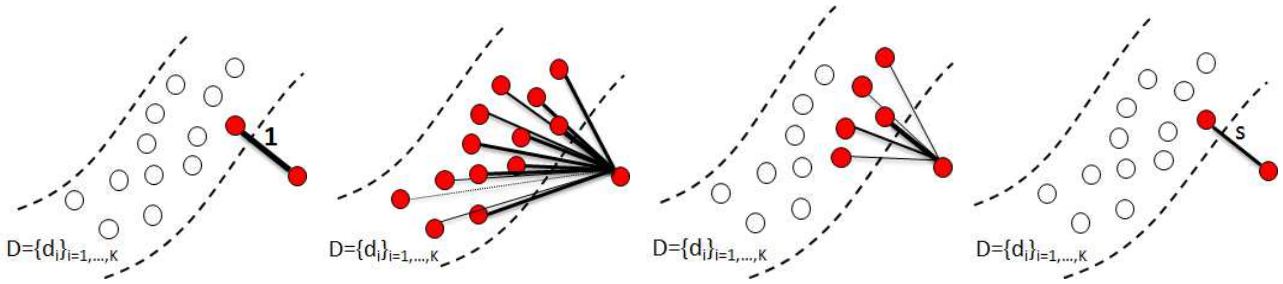
where ω_i is the normalized weight of descriptor $\mathbf{x}$ with respect to codeword $\mathbf{d}_i$:

$$\omega_i = \frac{\exp(-\beta \|\mathbf{x} - \mathbf{d}_i\|_2^2)}{\sum_{j=1}^{K} \exp(-\beta \|\mathbf{x} - \mathbf{d}_j\|_2^2)}, \tag{6}$$

where β is a smoothing factor controlling the softness of the assignment. Considering the manifold structure in the descriptor space, localized Soft Assignment (SA-k) votes for its k-nearest codewords:

$$\text{SA-}k: \quad \phi(\mathbf{x}) = \omega_i' = \frac{I(\mathbf{x}, \mathbf{d}_i) \exp(-\beta \|\mathbf{x} - \mathbf{d}_i\|_2^2)}{\sum_{j=1}^{K} I(\mathbf{x}, \mathbf{d}_j) \exp(-\beta \|\mathbf{x} - \mathbf{d}_j\|_2^2)}, \tag{7}$$

[1] We use $\mathbf{s}$ to denote the code of voting and reconstruction based encoding methods, and $\mathcal{S}$ to represent the one of super vector based encoding methods.



(a) VQ (b) SA (c) SA-k (d) SC/GSC

Fig. 2 Comparison among all the voting based encoding methods.

where $I(\mathbf{x}, \mathbf{d}_i)$ is the indicator function to identify whether $\mathbf{d}_i$ belongs to the k nearest neighbor of $\mathbf{x}$:

$$I(\mathbf{x}, \mathbf{d}_i) = \begin{cases} 1 & \text{if } \mathbf{d}_i \in NN_k(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Note that VQ can be viewed as a special case of SA-k when k is set as 1.

Figure 2 illustrates the difference of these voting based encoding methods. VQ, Salient coding, and Group salient coding are all hard assignment strategies. Unlike VQ, the Salient coding employs the difference between the closest visual word and the other $k-1$ closest ones to obtain the voted weight but not 1. The detailed formulations of Salient coding and Group salient coding can be found in Table 1.

2.4.2 Reconstruction based encoding methods

Reconstruction based encoding methods [61,50,63,42] are designed from the perspective of decoding process, where the codes $\mathbf{s}$ are enforced to reconstruct the input descriptor $\mathbf{x}$. This kind of algorithm includes Orthogonal Matching Pursuit (OMP) [42], Sparse Coding (SPC) [61], Local Coordinate Coding(LCC) [63], and Locality-constrained Linear Coding (LLC) [50]. Typically, these encoding methods are formulated in a least square framework with a regularization term:

$$\arg\min_{\mathbf{s}} ||\mathbf{x} - \mathbf{D}\mathbf{s}||_2^2 + \lambda\psi(\mathbf{s}), \tag{9}$$

where the least square term enforce the small reconstruction error, $\psi(\mathbf{s})$ encourages some properties of codes $\mathbf{s}$, λ is a weight factor to balance this two terms.

Among these methods, OMP and SPC pursue a sparse representation. As for OMP, this constraint is conducted by ℓ_0-norm:

$$\text{OMP:} \quad \psi(\mathbf{s}) = ||\mathbf{s}||_0 \tag{10}$$

where ℓ_0-norm means the number of non-zero elements in $\mathbf{s}$. However, due to the non-convexity of ℓ_0-norm, solution to this problem usually needs some heuristic strategy and obtains an approximate optimal solution. SPC relaxes this non-convex ℓ_0-norm with ℓ_1-norm:

$$\text{SPC:} \quad \psi(\mathbf{s}) = ||\mathbf{s}||_1 \tag{11}$$

where ℓ_1-norm can also encourage the sparsity in code $\mathbf{s}$, and the solution is equal to the solution of ℓ_0-norm under some conditions [5]. The ℓ_1-norm relaxation allows for more efficient optimization algorithm [27] and obtaining the global optimal solution.

OMP and SPC is empirically observed to tend to be local, *i.e.* nonzero coefficients are often assigned to bases nearby to the encoded data [63]. But this locality can not be ensured theoretically and they suggested a modification to SPC, called Local Coordinate Coding (LCC). This encoding method explicitly encourages the coding to be local, and they theoretically pointed out that under certain assumptions locality is more essential than sparsity, for successful nonlinear function learning using the obtained codes. Specifically, the LCC is defined as follows:

$$\text{LCC:} \quad \psi(\mathbf{s}) = \|\hat{\mathbf{e}} \odot |\mathbf{s}|\|_1, \quad \text{s.t.} \quad \mathbf{1}^T\mathbf{s} = 1, \tag{12}$$

where $\odot$ denotes the element-wise multiplication, $\hat{\mathbf{e}}$ is the locality adaptor that give weights for each basis vector proportional to its similarity to the input descriptor $\mathbf{x}$:

$$\hat{\mathbf{e}} = [\text{dist}(\mathbf{x}, \mathbf{d}_1), \cdots, \text{dist}(\mathbf{x}, \mathbf{d}_K)]^\top, \tag{13}$$

where $\text{dist}(\mathbf{x}, \mathbf{d}_k)$ is the Euclidean distance between $\mathbf{x}$ and $\mathbf{d}_k$. Due to the problem of ℓ_1-norm optimization in both SPC and LCC, it is computationally expensive and hard to apply to large scale problem. Then, a practical coding scheme called Locality-constrained Linear Coding (LLC) [50] is designed, which can be viewed as a fast implementation of LCC that utilizes the locality constraint to project each descriptor into its local-coordinate system:

$$\text{LLC:} \quad \psi(\mathbf{s}) = \|\mathbf{e} \odot \mathbf{s}\|_2^2, \quad \text{s.t.} \quad \mathbf{1}^T\mathbf{s} = 1, \tag{14}$$

where $\mathbf{e}$ is the exponentiation of $\hat{\mathbf{e}}$:

$$\mathbf{e} = \exp\left(\frac{\text{dist}(\mathbf{x}, \mathbf{D})}{\sigma}\right), \tag{15}$$

where σ is used for adjusting the weighted decay speed for the locality adaptor. The constraint $\mathbf{1}^T\mathbf{s} = 1$ follows the shift-invariant requirements of the final code vector. In practice, an approximate solution can be used to improve the computational efficiency of LLC. It directly selects the k nearest basis vectors of $\mathbf{x}$ to minimize the first term in Equation (9) by solving a much smaller linear system. This gives the code coefficients for the selected k basis vectors and other code coefficients are simply set to be zero.

2.4.3 Super vector based encoding methods

Super vector based encoding methods yield a very high dimensional representation by aggregating high order statistics. Typical methods include Local Tangent-based Coding (LTC) [62], Super Vector Coding (SVC) [65], Vector of Locally Aggregated Descriptors (VLAD) [16], and Fisher Vector (FV) [35] .

Local Tangent-based Coding [62] assumes that codebook and descriptors are embedded in a smooth manifold. The main contents of LTC are manifold approximation and intrinsic dimensionality estimation. Under the Lipschitz smooth condition, the nonlinear function $f(\mathbf{x})$ can be approximated by a local linear function as:

$$f(\mathbf{x}) \approx \sum_{i=1}^{K} \mathbf{s}(i)\left[f(\mathbf{d}_i) + 0.5\nabla f(\mathbf{d}_i)^T(\mathbf{x} - \mathbf{d}_i)\right], \tag{16}$$

where $\mathbf{s}(i)$ is obtained by LCC [63]. Then, this approximate function can be viewed as a linear function of a coding vector $[\mathbf{s}(i), \mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i)]_{i=1}^K \in \mathbb{R}^{K\times(1+D)}$. LTC argues that there is lower intrinsic dimensionality in the feature manifold. To obtain it, Principal Component Analysis (PCA) is applied to the term of $\mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i)$ using a projection matrix $\mathbf{U}_i = [\mathbf{u}_1^i, \cdots, \mathbf{u}_C^i] \in \mathbb{R}^{D\times C}$ trained from training data, i.e., the local tangent directions of the manifold. Therefore, the final coding vector for LTC is written as follows:

$$\text{LTC:} \qquad \mathcal{S} = \left[\alpha\mathbf{s}(i), \mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i)^T\mathbf{U}_i\right]_{i=1}^K, \tag{17}$$

where α is a positive scaling factor to balance the two types of codes. Super Vector Coding (SVC) [65] is a simple version of LTC. Unlike LTC, SVC yields the $\mathbf{s}(i)$ via VQ and does not apply PCA to the term of $\mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i)$. Consequently, the coding vector of SVC is defined as follows:

$$\text{SVC: } \mathcal{S} = [0, \mathbf{0}, \cdots, \frac{\alpha\mathbf{s}(i)}{N\sqrt{p_i}}, \frac{\mathbf{s}(i)}{N\sqrt{p_i}}(\mathbf{x} - \mathbf{d}_i), \cdots, 0, \mathbf{0}], \tag{18}$$

where $\mathbf{s}(i) = 1$, $\mathbf{d}_i$ is the closest visual word to $\mathbf{x}$, and α is a positive constant.

Fisher vector is another super vector based encoding method derived from fisher kernel [15] and is introduced for large-scale image categorization [35]. The fisher kernel is a generic framework which combines the benefits of generative and discriminative approaches. As it is known, the gradient of the log-likelihood with respect to a parameter can describe how that parameter contributes to the process of generating a particular example. Then the video can be described by the gradient vector of log likelihood with respect to the model parameters [15]:

$$G_\theta^{\mathbf{x}} = \nabla_\theta \log p(\mathbf{x}; \theta). \tag{19}$$

Note that the dimensionality of this vector depends on the number of parameters in θ. Perronnin *et al.* [35] developed an improved fisher vector which is as follows,

$$\mathcal{G}_{\mu,k}^{\mathbf{x}} = \frac{1}{\sqrt{\pi_k}}\gamma_k\left(\frac{\mathbf{x} - \mu_k}{\sigma_k}\right), \tag{20}$$

$$\mathcal{G}_{\sigma,k}^{\mathbf{x}} = \frac{1}{\sqrt{2\pi_k}}\gamma_k\left[\frac{(\mathbf{x} - \mu_k)^2}{\sigma_k^2} - 1\right], \tag{21}$$

where γ_k is the weight of local descriptor $\mathbf{x}$ to k^{th} Gaussian Mixture:

$$\gamma_k = \frac{\pi_k\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)}{\sum_{i=1}^K \pi_i\mathcal{N}(\mathbf{x}; \mu_i, \Sigma_i)}. \tag{22}$$

The final fisher vector is the concatenation this two gradients:

$$\text{FV:} \qquad \mathcal{S} = [\mathcal{G}_{\mu,1}^{\mathbf{x}}, \mathcal{G}_{\sigma,1}^{\mathbf{x}}, \cdots, \mathcal{G}_{\mu,K}^{\mathbf{x}}, \mathcal{G}_{\sigma,K}^{\mathbf{x}}]. \tag{23}$$

Vector of Locally Aggregated Descriptors (VLAD) [16] can be viewed as a hard version of FV and only keeps the 1^{st} order statistics:

$$\text{VLAD:} \qquad \mathcal{S} = [\mathbf{0}, \cdots, \mathbf{s}(i)(\mathbf{x} - \mathbf{d}_i), \cdots, \mathbf{0}], \tag{24}$$

where $\mathbf{s}(i) = 1$, $\mathbf{d}_i$ is the closest visual word to $\mathbf{x}$.

2.4.4 Relations of Encoding Methods

In this section, we summarize several practical techniques widely used in these encoding methods, and give a unified generative perspective of these encoding methods. This analysis will uncover the underline relations between these methods and provide insights for developing new encoding methods.

From "hard" to "soft". These encoding methods transform local features from descriptor space to codeword space. There are two typical transformation

rules in these methods, namely *hard assignment* and *soft assignment*. Hard assignment quantizes the feature descriptor into a single codeword, while soft assignment enables the feature descriptor to vote for multiple codewords. In general, soft assignment accounts for the codeword uncertainty and plausibility [12], and reduces the information loss during encoding. This technical skill of soft assignment can be found in several encoding algorithms, such as SA-*all* vs. VQ, and VLAD vs. Fisher Vector. By the same techniques, we can extend the VLAD to VLAD-*all*, SVC to SVC-*all*:

$$\text{VLAD}-all: \ \mathcal{S} = [\omega_1(\mathbf{x}-\mathbf{d}_1), \cdots, \omega_K(\mathbf{x}-\mathbf{d}_K)], \quad (25)$$

$$\text{SVC}-all: \ \mathcal{S} = \left[\frac{\alpha\omega_1}{N\sqrt{p_1}}, \frac{\alpha\omega_1}{N\sqrt{p_1}}(\mathbf{x}-\mathbf{d}_1), \cdots, \frac{\alpha\omega_K}{N\sqrt{p_K}}, \frac{\alpha\omega_K}{N\sqrt{p_K}}(\mathbf{x}-\mathbf{d}_K)\right], \quad (26)$$

where ω_i is the normalized weight of feature descriptor $\mathbf{x}$ with respect to codeword $\mathbf{d}_i$ defined in Equation (6).

From "global" to "local". In several encoding methods, the manifold structure in descriptor space is captured to improve the stability of encoding algorithms. In the traditional soft assignment, each descriptor is assigned with all the codewords, which is called *global assignment*. However, in the high dimensional space of feature descriptor, Euclidian distance may be not reliable especially when the codeword is outside the neighborhood of feature descriptor. Therefore, in the encoding methods such as SA-k and LLC, each descriptor is enforced to only vote for these codewords belonging to its k-nearest neighbors, called *local assignment*. In general, the incorporation of local structure in encoding methods is able to improve the stability and reduce the sensitivity to noise in descriptor. Using the same techniques, we can also extend the VLAD-*all* to VLAD-k, SVC-*all* to SVC-k by replacing the ω_i in Equation (25), (26) with localized ω'_i defined in Equation (7):

$$\text{VLAD}-k: \quad \mathcal{S} = [\omega'_1(\mathbf{x}-\mathbf{d}_1), \cdots, \omega'_K(\mathbf{x}-\mathbf{d}_K)], \quad (27)$$

$$\text{SVC}-k: \quad \mathcal{S} = \left[\frac{\alpha\omega'_1}{N\sqrt{p_1}}, \frac{\alpha\omega'_1}{N\sqrt{p_1}}(\mathbf{x}-\mathbf{d}_1), \cdots, \frac{\alpha\omega'_K}{N\sqrt{p_K}}, \frac{\alpha\omega'_K}{N\sqrt{p_K}}(\mathbf{x}-\mathbf{d}_K)\right], \quad (28)$$

From "zero order statistics" to "high order statistics". In these super vector based encoding methods, they preserve not only the affiliations of descriptors to codewords (zero order statistics), but also the high order information such as the difference between descriptors mean and codeword, thus resulting a high-dimensional super vector representation. As these super vectors keep much richer information for each codeword, the codebook size is usually much smaller than that of voting and reconstruction based encoding methods. Above all, these super vector is with high dimension, storing more information, and is proved to outperform the other two kinds of encoding methods in Section 4. The high dimensional super vector will be a promising representation and designing effective dimension reduction algorithms for super vector will be an interesting problem.

Generative perspective of encoding methods. Although these encoding methods are developed in different scenarios, a unified generative probabilistic model can be used to uncover the underline relations among them. These encoding methods can be interpreted in a latent generative model:

$$\begin{aligned} &p(\mathbf{h}) \in \mathbb{P}, \\ &p(\mathbf{x}|\mathbf{h}) = \mathcal{N}(\mathbf{x}; W\mathbf{h} + \mu_{\mathbf{x}}, \Sigma), \end{aligned} \quad (29)$$

where $\mathbf{x} \in R^D$ represents the descriptor, $\mathbf{h} \in R^K$ denotes the latent factor, $\mathcal{N}(\mathbf{x}; W\mathbf{h} + \mu_{\mathbf{x}}, \Sigma)$ is multivariate Gaussian distribution. Different encoding methods mainly different in two aspects: *How to model the prior distribution* $\mathbb{P}$ *of latent factor* $\mathbf{h}$ and *How to use the probabilistic model to transform the descriptor into codeword space.*

For encoding methods such as VQ, SA-*all*, VLAD-*all*, and Fisher vector, they choose the prior distribution $p(\mathbf{h})$ as follows:

$$p(\mathbf{h}) = \prod_{i=1}^{K} \pi_i^{h_i}, \quad (30)$$

where $\mathbf{h} \in \{0,1\}^K$ is discrete random variable, and the prior distribution is a Multinomial distribution. For SA-*all*, this Mutinomial distribution is specified by uniform distribution, i.e. $\pi_1 = \cdots = \pi_K = \frac{1}{K}$, where for Fisher vector, this Multinomial distribution is learned during GMM training. Meanwhile the SA-*all* choose the latent variable embedding to encodes the descriptor by computing conditional expectation, i.e. $\mathbf{s}(\mathbf{x}) = \mathbb{E}(\mathbf{h}|\mathbf{x})$, while the Fisher vector choose the gradient embedding [15], i.e. $\mathcal{S}(\mathbf{x}) = \nabla_\theta \log p(\mathbf{x}; \theta)$. The VQ encoding can be viewed an extreme case of Soft-*all*, when:

$$p(\mathbf{x}|\mathbf{s}) = \mathcal{N}(\mathbf{x}; W\mathbf{s} + \mu_{\mathbf{x}}, \epsilon I), \quad \epsilon \to 0. \quad (31)$$

VLAD-*all* and SVC-*all* can be viewed as the gradient embedding in this extreme case.

For encoding methods such as sparse coding, the latent variable $\mathbf{h}$ is continuous and its corresponding

prior distribution is specified as:

$$p(\mathbf{h}) = \prod_{i=1}^{K} \frac{\lambda}{2} \exp(-\lambda |h_i|). \quad (32)$$

This prior distribution is called Laplace prior and sparse coding can be viewed as the latent variable embedding of this generative model using the maximum a posteriori value (MAP), i.e. $\mathbf{s}(\mathbf{x}) = \arg\max_{\mathbf{h}} p(\mathbf{h}|\mathbf{x})$.

2.5 Pooling and Normalization Methods

Given the code coefficients of all local descriptors in a video, a pooling operation is often used to obtain a global representation $\mathbf{p}$ for the video. Specifically, there are two common pooling strategies:

- **Sum Pooling.** With sum pooling scheme [26], the k^{th} component of $\mathbf{p}$ is $p_k = \sum_{n=1}^{N} \mathbf{s}_n(k)$.
- **Max Pooling.** With max pooling scheme [61], the k^{th} component of $\mathbf{p}$ is $p_k = \max(\mathbf{s}_1(k), \cdots, \mathbf{s}_N(k))$, where N is the number of extracted local descriptors, $\mathbf{s}_n$ denotes the code of descriptor $\mathbf{x}_n$.

In [4], the authors presented a theoretical analysis of average pooling and max pooling. Their results indicate sparse features may prefer max pooling.

To make this representation invariant to the number of extracted local descriptors, the pooling result $\mathbf{p}$ is further normalized by some methods. Generally, there are three common normalization techniques:

- **ℓ_1-Normalization.** In ℓ_1 normalization [61], the feature $\mathbf{p}$ is divided by its ℓ_1-norm: $\mathbf{p} = \mathbf{p}/\|\mathbf{p}\|_1$.
- **ℓ_2-Normalization.** In ℓ_2 normalization [35], the feature $\mathbf{p}$ is divided by its ℓ_2-norm: $\mathbf{p} = \mathbf{p}/\|\mathbf{p}\|_2$.
- **Power Normalization.** In power normalization [35], we apply in each dimension the following function:

  $$f(p_k) = \text{sign}(p_k)|p_k|^{\alpha}.$$

  where $0 \leq \alpha \leq 1$ is a parameter for normalization. We can combine power normalization with ℓ_1-normalization or ℓ_2-normalization.

Recently, a special normalization strategy is proposed for the VLAD, called **intra-normalization** [2]. In this paper, we extend it to all the super vector based encoding algorithms. This method carries out normalization operation in a block by block manner, where each block denotes the vector related to one codeword. Generally, the intra-normalization can be formulated as follows:

$$\mathbf{p} = \left[\frac{\mathbf{p}^1}{\|\mathbf{p}^1\|}, \cdots, \frac{\mathbf{p}^k}{\|\mathbf{p}^k\|}, \cdots, \frac{\mathbf{p}^K}{\|\mathbf{p}^K\|} \right], \quad (33)$$

where $\mathbf{p}^k$ denotes a vector related to codeword $\mathbf{d}_k$ (or the k^{th} Gaussian), $\|\cdot\|$ may be ℓ_1-norm or ℓ_2-norm.

3 Feature Fusion

Fusing multiple local features has turned out to be an effective method to boost the performance of recognition system in computer vision community [11,44, 41,47,6]. The video data is usually characterized in multiple views, such as static appearance, motion pattern, and motion boundary. The essence of multi-view data requires fusing different features for action recognition. In this section, we present several feature fusion methods for action recognition, and analyze its corresponding properties. Meanwhile, based on the analysis of fusion methods, we propose a simple yet effective representation, called *hybrid representation*.

As shown in Figure 3, the fusion methods are usually conducted in different levels, typically including: descriptor level, representation level, and score level. For descriptor level fusion, it is performed in the cuboid level, where multiple descriptors from the same cuboid are concatenated into a single one, and then it is fed into the BoVW to obtain the global representation. For representation-level fusion, it is performed in the video level, where different descriptors are input into BoVW separately and the resulting global representations are fused as a single one, which is further fed into classifier for recognition. For score-level fusion, it is also performed in the video level, but the representations of different descriptors are used independently for classifier training. The final recognition score is obtained by fusing the scores from multiple classifiers. For fusing the scores, arithmetical mean or geometrical mean is often used.

In general, these fusion methods at different levels owns their pros and cons, and the choice of fusion method should be guided by the dependence of descriptors. If these multiple descriptors from the same cuboid are highly correlated, it will be better to resort to descriptor level feature fusion. Otherwise, the choice of descriptor level fusion is not a good one, as descriptor level fusion usually results in a higher dimension and adds the difficulty for unsupervised feature learning such as k-means and sparse coding. For the case where different views of features are less correlated in cuboid level but highly correlated in video level, representation level fusion is usually a good choice. When these different features are independently with each other, it will be appropriate to choose score level fusion, as this fusion reduce the dimension for classifier training and make the learning faster and more stable.

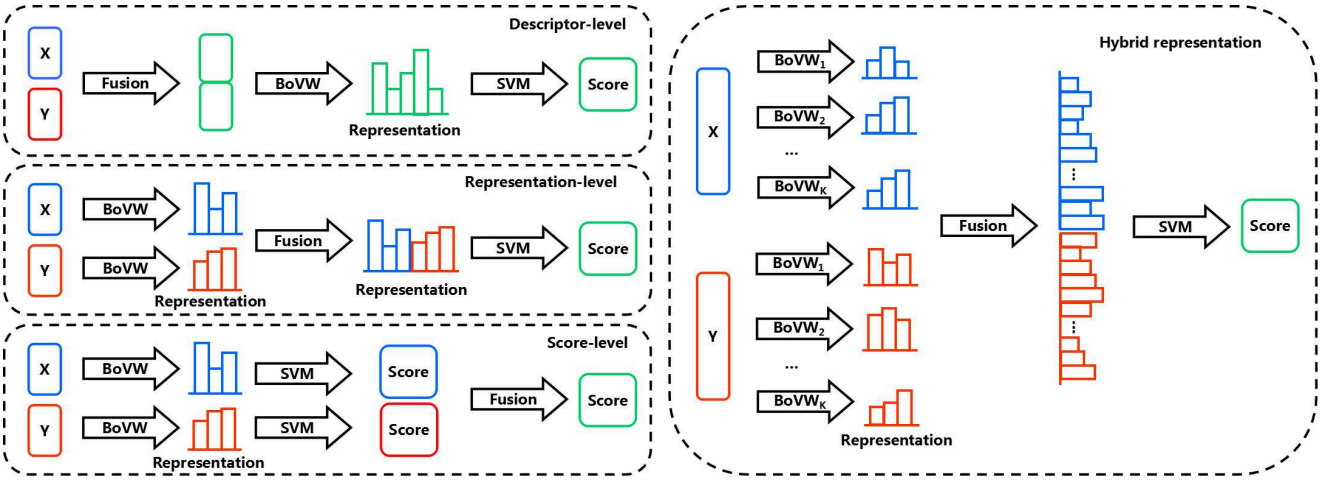


Fig. 3 Feature fusion is performed in different levels: descriptor level, representation level, and score level. The complementary effect of varied BoVW models can also be taken into account and a hybrid representation is obtained by fusing outputs from different BoVW models.

The performance boosting of fusing multiple features mainly owns the complementarity of these features. However, the complementarity can be explored not only for different features, but also for different types of BoVW methods. As shown in Figure 3, we propose a simple yet effective representation, called hybrid representation, which combines the outputs from multiple variants of BoVW and multiple descriptors. The resulting hybrid representation effectively explores the complementarity of different encoding methods and greatly enhances the descriptive power for action recognition. As we shall see in Section 4.7, this representation will improve the recognition rate of a single BoVW model and obtain the state-of-the-art results on the three challenging datasets.

4 Empirical Study

In this section, we describe the detailed experimental settings and the empirical study of variants of BoVW and different fusion methods. We first introduce the datasets used for evaluation and their corresponding experimental setup. We then extensively study different aspects of BoVW, including pre-processing techniques, encoding methods, pooling strategies, and normalization approaches. After that, we explore the different choices of fusion methods for multiple features. Finally, we compare the performance of our hybrid representation with that of the state-of-the-art methods on three challenging datasets.

4.1 Datasets and Evaluation Protocols

We conduct experiments on three public datasets: HMDB51 [23], UCF50 [36], and UCF101 [40]. Some examples of video frames are illustrated in Figure 4. Totally, we work with 26,704 videos in this paper.

The **HMDB51 dataset** has 51 action classes with total 6,766 videos and each class has more than 100 videos [2]. All the videos are obtained from real world scenarios such as: movies, youtube. The intra-class variation is very high due to many factors, such as viewpoint, scale, background, illumination etc. Thus, HMDB51 is a very difficult benchmark for action recognition. There are three training and testing splits released on the website of this dataset. We conduct experiments based on these splits and report average accuracy for evaluation.

The **UCF50 dataset** has 50 action classes with total 6,618 videos, and each action class is divided into 25 groups with at least 100 videos for each class. The video clips in the same group are usually with similar background. We choose the suggested evaluation protocols of Leave One Group Out cross validation (LOGO) and report the average accuracy [36].

The **UCF101 dataset** is an extension of the UCF50 dataset and has 101 action classes. The action classes can be divided into five types: human-object interaction, body-motion only, human-human interaction, playing musical instruments, and sports. Totally, it has 13,320 video clips, with fixed frame rate and resolution 25 FPS and 320 × 240 respectively. To our best knowledge, this dataset has been the largest dataset so far. We perform evaluation according to the

[2] http://serre-lab.clps.brown.edu/resources/HMDB/index.htm

(a) HMDB51 (b) UCF 50 and UCF101

Fig. 4 Sample frames from the HMDB51, UCF50 and UCF101 datasets. Note that UCF50 is a subset of UCF101.

three train/test splits released in Thumos'13 challenge [3] and report the mean average accuracy of these splits.

In our evaluation experiment, we choose linear Support Vector Machine (SVM) as our recognition classifier. Specifically, we use the implementation of LIBSVM [8]. For multiclass classification, we adopt one-vs-all training scheme and choose the prediction with highest score as our predicted label.

多类SVM

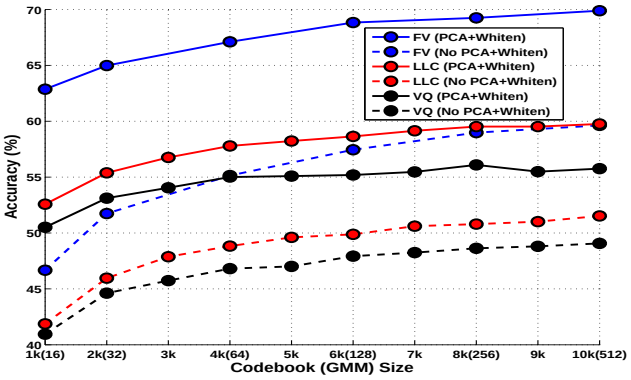


Fig. 5 Comparison the results with PCA+Whiten and without PCA-Whiten of different encoding methods on the UCF101 dataset, where STIPs are chosen as the local features.

4.2 Local Features and Codebook Generation

In our evaluation, we choose two widely-used local features, namely Space Time Interest Points (STIPs) [24] with HOG, HOF descriptors [25], and improved Dense Trajectories (iDTs) with HOG, HOF, MBHx, MBHy descriptors [46]. Specifically, we use the implementation released on the website of Laptev [4] for STIPs and Wang [5] for iDTs. We choose the default parameter settings for both local features. STIPs and iDTs represent two types of local features: sparse interest points and densely-sampled trajectories. They may exhibit different properties with varying BoVW settings, and thus it is well worth exploring both STIPs and iDTs.

Regarding codebook generation, we randomly sample $100,000$ features to conduct k-means, where codebook size range from 1,000 to 10,000 for STIPs, and from 1,000 to 20,000 for iDTs. For GMM training, we randomly sample 256,000 features to learn GMMs with mixture number ranging from 16 to 512 for both STIPs and iDTs.

4.3 Importance of Pre-processing

In this section, we explore the importance of pre-processing in BoVW framework. Specifically, we use STIPs as local features and choose a representative method for each type of encoding, namely FV, LLC, and VQ . For pooling and normalization strategy, we use sum pooling and power ℓ_2-normalization. We use the descriptor-level fusion method to combine HOG and HOF descriptors.

We conduct experiments on the UCF101 dataset and investigate the importance of pre-processing for these encoding methods. With pre-processing step, the descriptors of STIPs are firstly reduced to 100-dimension and then whitened to have unit variance. The results are shown in Figure 5. We observe that the pre-processing technique of PCA-Whiten is very important to boost the performance of encoding methods.

[3] http://crcv.ucf.edu/ICCV13-Action-Workshop/
[4] http://www.di.ens.fr/ laptev/download.html
[5] https://lear.inrialpes.fr/people/wang/improved_trajectories

Surprisingly, the performance of FV (state-of-the-art) without PCA-Whiten is lower than or comparable to VQ and LLC with PCA-Whiten. In previous research work, PCA-Whiten is often done for FV encoding methods but seldom used for other encoding methods. Our study suggests that using PCA-Whiten techniques enable us to greatly improve final recognition rate for all encoding methods. We obtain the recognition rate 56.1% for VQ, which significantly outperform over the result 43.9% reported in [40], where the same local feature and encoding method is used.

In the remaining part of evaluation, we will use PCA-Whiten to de-correlate the descriptor, reduce the dimension, and normalize the variance. For descriptor level fusion of STIP, the dimension of concatenated descriptor is reduced from 162 to 100. For HOG and HOF, the dimension is reduced from 72 to 40, and from 90 to 60, respectively. For descriptor level fusion of iDT, the dimension of concatenated descriptor is reduced from 396 to 200. For separate descriptor, the dimensions of HOG, MBHx, and MBHy are all reduced from 96 to 48. HOF descriptor is reduced from 108 to 54.

4.4 Exploration of Encoding Methods

In this section, we compare and analyze the performance of different encoding methods. For each encoding method, we fix other settings, such as parameter setting, pooling and normalization strategy, the same with previous papers. We explore these encoding methods with descriptor level fusion, for both STIPs and iDTs. The influence of different pooling and normalization strategy, and fusion methods will be investigated in the following sections.

Encoding methods selection and setting. We select six popular encoding methods according to categorization in Table 1. For voting based encoding methods, we choose VQ as a baseline and SA-k as a representative method. LLC is selected as the representative of reconstruction-based encoding methods due to its computational efficiency and performance [54]. Super vector based encoding methods have shown the state-of-the-art performance on several datasets [47] . We choose three super vector based encoding methods for evaluation, namely FV, VLAD, and SVC.

Baseline: Vector Quantization Encoding (VQ). In the baseline method, each descriptor is quantized into a single codeword. Following the suggested settings in object recognition [64], the final histogram is obtained by sum pooling and normalized with ℓ_1 norm.

Localized Soft Assignment Encoding (SA-k). In the localized soft assignment, each descriptor is assigned to its corresponding k nearest neighborhood. It requires a single parameter β, which is the smoothing factor controlling the softness. According to [28], we set β as 1 and k as 5 in our evaluation. We use max pooling and ℓ_2-normalization.

Locality-constrained Linear Encoding (LLC). Following [50], we use approximated LLC for fast encoding, where we simply use k nearest neighborhood of descriptor as the local bases. The parameter of k is set as 5, and we choose max pooling and ℓ_2-normalization strategy.

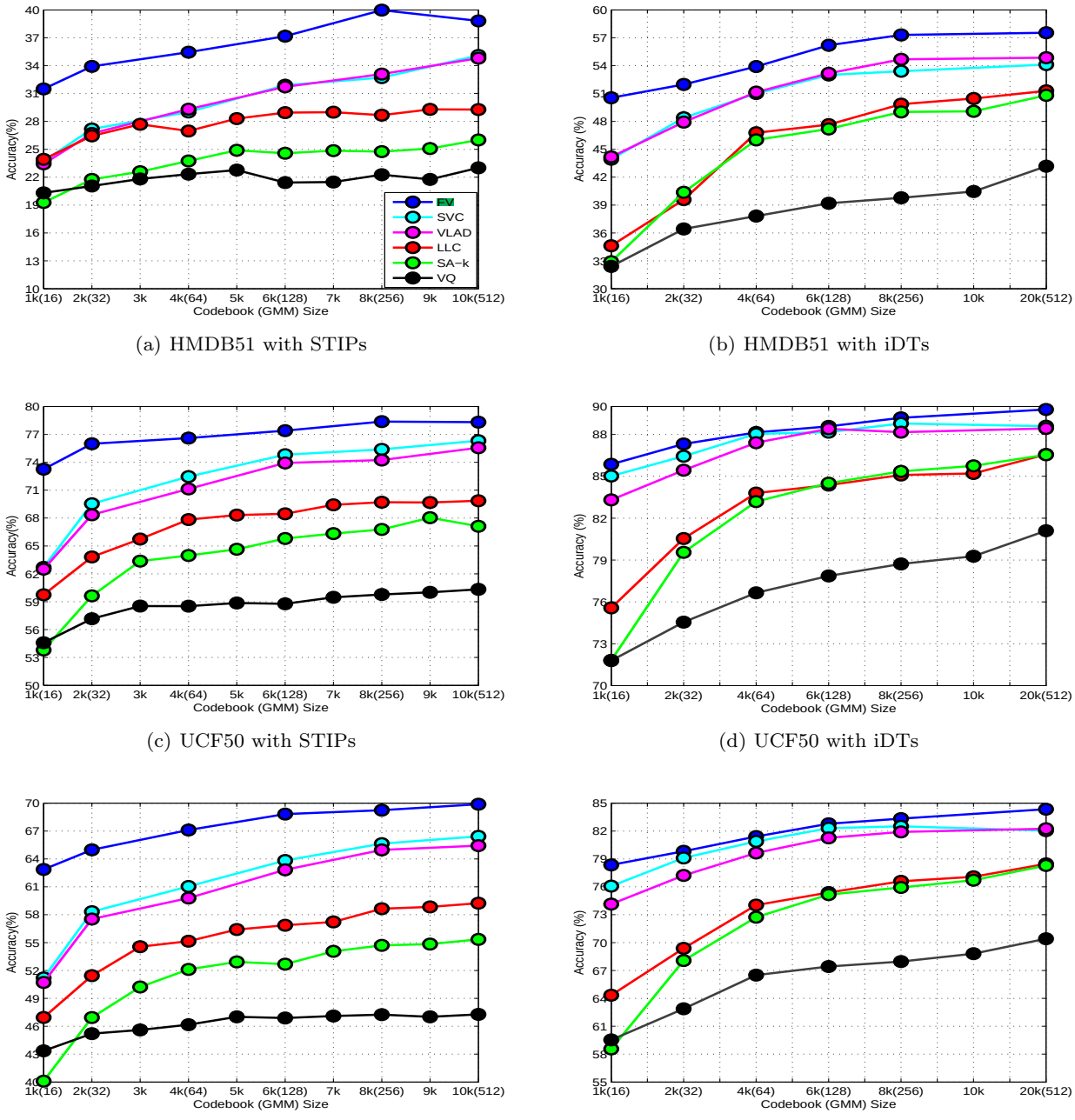
Fisher Vector (FV). For GMM training, we use the k-means result to initialize iteration and the covariance matrix of each mixture is set as a diagonal one. Following [35], we use sum pooling and power ℓ_2-normalization.

Vector of Locally Aggregated Vector (VLAD). VLAD was originally designed for image retrieval in [16] and can be viewed as a simplified version FV for fast implementation. Just like FV, we choose sum pooling and power ℓ_2-normalization.

Super Vector Coding (SVC). From the view of statistics, SVC can be viewed as a combination of VQ and VLAD. It contains the zeros and first-order statistics, and the parameter α keep balance between these two components. Following [65], we set α as 0.1. Like other super vector based encoding methods, we choose sum pooling and power ℓ_2-normalization.

Results and analysis. The experimental results of STIPs and iDTs on the three datasets are shown in Figure 6. Several rules can be found from these experimental results:

– Basically, the recognition performance of all selected encoding methods increases as the size of codebook (GMM) becomes larger and will reach a plateau when the size exceeds a threshold. For super vector based encoding methods, the performances reach a saturation when size of codebook (GMM) becomes 256 for both STIPs and iDTs. There is a slight change of the recognition rate when GMM size grows from 256 to 512. For the other two types of encoding methods, the performances are saturated as the size of codebook reaches 8,000. We also notice that these encoding methods using iDTs have slight improvements when the codebook size varies from 8,000 to 20,000 , while the performances using STIPs start shaking when the codebook size becomes larger than 8,000 due to the over-fitting effect. This difference may be ascribed to the dimension of local descriptors and sampling strategy. The descriptors dimension of iDTs is twice of STIPs and requires more codewords to divide the feature space. Meanwhile, STIPs is a set of interest points and the extracted descriptors distribute sparsely in



(a) HMDB51 with STIPs

(b) HMDB51 with iDTs

(c) UCF50 with STIPs

(d) UCF50 with iDTs

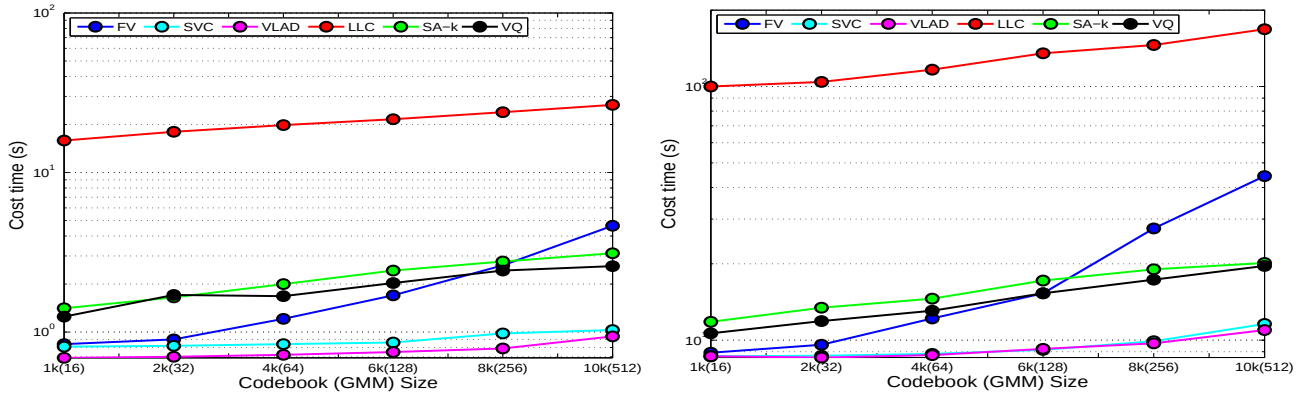
(e) UCF101 with STIPs

(f) UCF101 with iDTs

Fig. 6 Performance of different encoding methods with varying codebook (GMM) sizes on the HMDB51, UCF50, and UCF101 datasets for STIPs and iDTs features using descriptor-level fusion.

the feature space. The codebook with large size will result in an over-partition of feature space, which means for a specific video, there may be no descriptors falling into the corresponding regions for some codewords. iDTs are more densely sampled features and codebook with large size is more suitable to divide the space of dense features. **Above all, for a good balance between performance and efficiency, sizes of** 256 **and** 8,000 **are good choices for super vector based encoding and other encoding respectively**.

– For local features of both SITPs and iDTs, super vector based encoding methods outperform the other types of encoding methods on the three datasets. According to previous introduction, these super vector encoding methods not only preserve the affiliations of descriptors to codewords, but also keep high order information such as the difference of



(a) Time of STIPs (b) Time of iDTs

Fig. 7 Average time of different encoding methods with varying codebook (GMM) sizes on the UCF101 datasets for STIPs and iDTs features using descriptor-level fusion.

means and variances. These high order information enables the encoding methods to better capture the distribution shape of descriptor in feature space. In these super vector based methods, FV is typically better than VLAD and SVC, whose performance is quite similar. This can be own to two facts: (i) FV keeps both 1^{st} and 2^{nd} statistics, which is more informative than VLAD (only 1^{st} statistics) and SVC (0^{th} statistics and 1^{st} statistics). (ii) FV is based on GMM and each descriptor is softly assigned to codewords using posterior probability, while VLAD and SVC are based on k-means results and use hard assignment. We also notice that the difference between FV and the other two methods (VLAD, SVC) for iDTs seems smaller than STIPs. The more dense descriptors may make the learned codebook more stable for SVC and VLAD, and reduce the influence of soft assignment in FV. Meanwhile, the information contained in 2^{nd} statistics may be less complementary to 1^{st} statistics for iDTs. **In conclusion, super vector based representation, aggregating high order information, is a more suitable choice for good performance, when the high dimension of representation is acceptable**.

– For reconstruction based and voting based encoding methods, VQ reaches the lowest recognition rate for STIPs and iDTs on the three datasets. This can be ascribed to the hard assignment and descriptor ambiguity in the VQ method. In essence, the LLC and SA-k are quite similar in spirt, for that they both consider locality when mapping descriptor into codeword. The performance of LLC is better than SA-k for STIPs, while the performances of them are almost the same for iDTs. This can be explained by the mapping strategy in LLC and SA-k. The mappings of descriptor to the nearest codewords in LLC are determined jointly according to their effect in minimizing the reconstruction error, while the mappings in SA-k are calculated independently for each individual codeword according to the Euclidean distance. The mapping method in LLC may be more effective to deal with manifold structure than just considering Euclidean distance in SA-k. For sparse features such as STIPs, the descriptors distribute sparsely around each codeword, and using Euclidean distance may introduce noise and instability for SA-k. For dense features such as iDTs, the descriptors are usually sampled densely and more compact around codewords, reducing the influence caused by the usage of Euclidean distance. **In a word, compared with hard assignment, locality and soft assignment is an effective strategy to improve the performance of encoding methods**.

– STIPs and iDTs represents two types of local features, namely sparsely-sampled and densely-sampled features. In general, they exhibit consistent performance trends for different encoding methods, for example, super vector encoding methods outperforms others, soft-assignment is better than hard-assignment. However, there is a slight difference between them in some aspects, such as sensitivity to codebook size and encoding methods, performance gaps among super vector based methods, difference between LLC and SA-k, as previously observed. From the perspective of data manifold, the more densely-sampled features can help us more accurately describe the data structure in the feature space. We can obtain a more compact clustering result using k-means, and the local Euclidean distance is more stable. **Thus, when choosing**

codebook size and encoding method, the type of local feature can be a factor needed to be considered.

Computational costs. We also compare the efficiency of different encoding methods and the running time is shown in Figure 7. Our codes are all implemented in Matlab, and running on a workstation with 2x Intel Xeon 5560 2.8GHz CPU and 32G RAM. We randomly sample 50 videos from the UCF101 dataset and report the total time for these videos. For super vector based methods, FV is much slower due to the calculation of posterior probability during encoding, and the time of VLAD and SVC is almost the same. For the other types of encoding methods, LLC is less efficient as it solves a least square problem. The computational cost of super vector encoding methods are usually lower than that of the other types of encoding methods, due to their smaller codebook sizes.

Based on the above analysis, super vector based encoding methods are more promising for high performance and fast implementation, especially for SVC, VLAD. However, the feature dimension of super vector methods is much higher than the other two kinds of encoding methods, for example, when the codebook size is 256, the dimension of FV and VLAD is 102,400 and 51,200 respectively for iDT features. The effective dimension reduction may be a future research direction for super vector encoding methods.

4.5 Exploration of Pooling and Normalization

In this section, we mainly investigate the influence on recognition rate for different pooling and normalization strategies on the UCF101 dataset. Based on the performances of different encoding methods on the UCF101 dataset in previous section, we choose the codebook (GMM) size as 512 for super vector based methods and codebook size as $8,000$ for the other two types of encoding approaches. Meanwhile, according to our conclusion that super vector based encoding is a promising method and soft assignment is an effective way to improve the encoding methods, we extend VLAD to VLAD-k and VLAD-*all*, SVC to SVC-k and SVC-*all* as described in Section 2.4.4. Thus, there are totally 10 kinds of encoding methods.

For super vector based encoding methods, we evaluate eight normalization methods, specified by with or without intra-normalization, with or without power operation, final ℓ_1 or ℓ_2 normalization. For the other two types of encoding methods, we choose two pooling methods, namely max pooling and sum pooling, and four normalization methods, namely ℓ_1-normalization, ℓ_2-normalization, power ℓ_1-normalization, and power ℓ_2-normalization. In our evaluation, the parameter α in power normalization is set as 0.5.

The experimental results are shown in Figure 8 and Figure 9. Several observation can be concluded from these results:

- For super vector based encoding methods, intra-normalization is an effective way to balance the weight of different codewords and suppress the burst of features corresponding to background. We found this technique works very well when dense features are chosen. A large number of features in iDTs are irrelevant with the action class and intra-normalization can suppress this influence. However, for sparse features, the effect of intra normalization is not so evident, and even cause performance degradation in the case of hard assignment such as VLAD, SVC. We ascribe this phenomenon to the fact that the STIPs features are usually located in the moving foreground and related with action class. Thus, these descriptors only vote for a subset of codewords, that are highly related with action class. In this case, intra-normalization can decrease the discriminative power of action-related codewords and increase the influence of irrelevant codewords. **In conclusion, intra-normalization is effective in handling burst of irrelevant features in the case of dense-sampling strategy**.
- For different encoding methods and local features, we observe that the final ℓ_2-normalization outperforms ℓ_1-normalization. In fact, the normalization method is related to kernel used in final classifier. In our case of linear SVM, the kernel is $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$. The choice of ℓ_2-normalization can ensure two things: (i) $k(\mathbf{x}, \mathbf{x}) = \text{const.}$; (ii) $k(\mathbf{x}, \mathbf{x}) \geq k(\mathbf{x}, \mathbf{y})$. This can guarantee a simple consistency criterion: by interpreting $k(\mathbf{x}, \mathbf{y})$ as a similarity score, $\mathbf{x}$ should be the most similar point to itself [45]. However, the choice of ℓ_1-normalization can not make sure that the point is most similar to self and may cause the instability during SVM training. **Above all, ℓ_2-normalization generally outperforms ℓ_1-normalization when using linear SVM**.
- The influence of power operation in normalization is highly related with pooling method. We observe that power normalization is an effective approach to boost the performance of representation obtained from sum pooling, such as super vector based representation, LLC, SA-k with sum pooling. However, power normalization have little effect for max pooling and sometimes even cause the performance degradation for LLC, SA-k. The operation of power usually reduces the difference between different

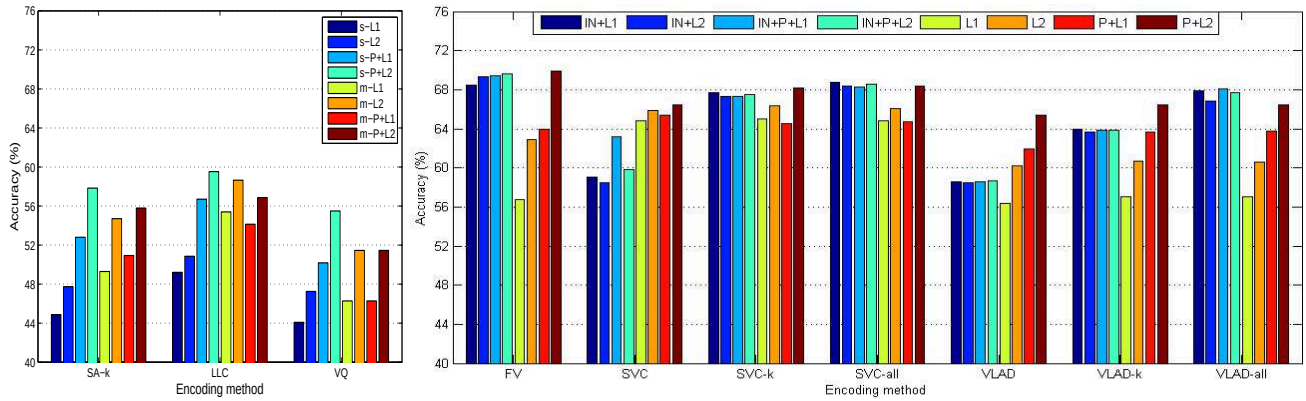


Fig. 8 Comparison of different pooling-normalization strategies with **STIPs** features using descriptor level fusion on the UCF101 dataset. Note that there is only max pooling for voting and reconstruction based encoding methods, and there is only intra normalization for super vector based encoding methods.

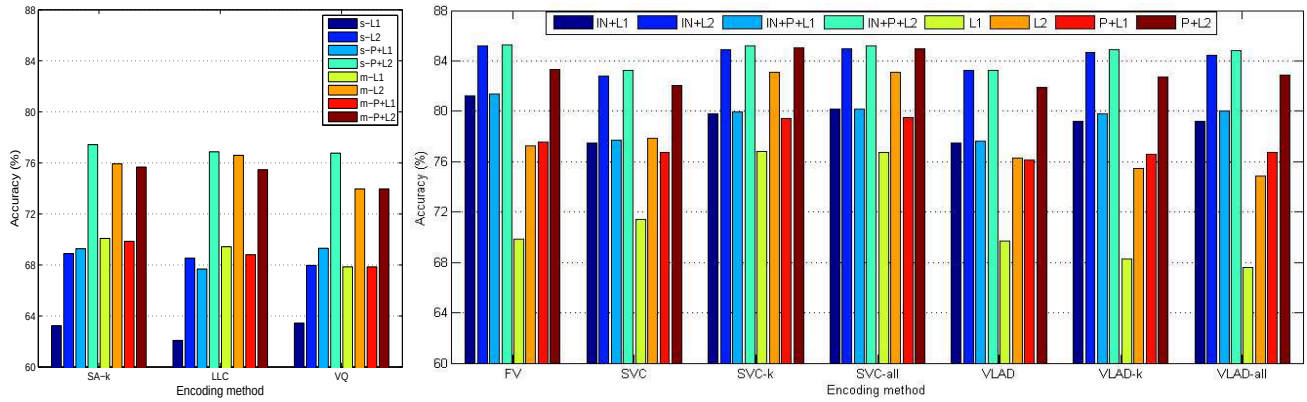


Fig. 9 Comparison of different pooling-normalization strategies with **iDTs** features using descriptor level fusion on the UCF101 dataset. Note that there is only max pooling for voting and reconstruction based encoding methods, and there is only intra normalization for super vector based encoding methods.

codewords, which means smoothing the histogram. This smooth effect can reduce the influence of high frequent codeword on the kernel calculation and improve the influence of less frequent codeword. For sum pooling, the resulting histogram is usually very sharp and unbalanced due to feature burst, and the smooth operation has a positive effect for suppress the high frequent codeword. However, for max pooling, the histogram is itself not so sharp as sum pooling, and thus the power normalization may have a side effect. **In above, power operation is an effective strategy to smooth the resulting histogram and can greatly improve the performance of sum pooling representation**.

– Among different choices of pooling operations and normalization methods, we conclude that sum pooling and power ℓ_2-normalization is the best combination. For dense features, intra normalization is an extra bonus for performance boosting. For sparse features, intra normalization sometimes may have a negative effect. The success of sum pooling and power ℓ_2-normalization can be explained by the Hellinger's kernel [45], which has turned out to be an effective kernel to calculate the similarity between two histograms. The linear kernel calculation in the feature space resulting from power ℓ_2-normalization is equivalent to the Hellinger's kernel calculation in the original space just using ℓ_1 normalization:

$$< \frac{\sqrt{\mathbf{x}}}{\|\sqrt{\mathbf{x}}\|_2}, \frac{\sqrt{\mathbf{y}}}{\|\sqrt{\mathbf{y}}\|_2} >=< \sqrt{\frac{\mathbf{x}}{\|\mathbf{x}\|_1}}, \sqrt{\frac{\mathbf{y}}{\|\mathbf{y}\|_1}} > \qquad (34)$$

which means power ℓ_2-normalization explicitly introduces non-linear kernel in the final classifier. **In a word, sum pooling and power ℓ_2-normalization is effective and efficient way to enable linear SVM to have the power of non-linear classifier and boost final recognition rate**.

In conclusion, pooling and normalization is a crucial step in the pipeline of BoVW framework, whose importance may not be highlighted in previous research work. Proper choice of pooling and normalization strategy may largely reduce the performance gap of different

encoding methods. For sum pooling and power ℓ_2-normalization, which is the best combination in all these possible choices, the performances of LLC, SA-k, and VQ are comparable to each other for iDT features. Thus, in the remaining evaluation for fusion methods, we fix the pooling and normalization strategy as sum pooling and power ℓ_2-normalization.

4.6 Exploration of Fusion Methods

The local features usually have multiple descriptors, such as HOG, HOF, MBHx, and MBHy, each of which corresponds to a specific view of video data. For the empirical study in previous section, we choose a simple method to combine these multiple descriptors, where we just concatenate them into a single one, namely descriptor level fusion. In this section, we mainly analyze the influence of different fusion methods on final recognition performance.

For encoding methods, we choose the same ten approaches as in previous section. The codebook size of super vector based methods is set as 512 and the one of other encoding methods is set as $8,000$. For pooling and normalization methods, we use sum pooling and power ℓ_2-normalization, according to the observations in Section 4.5. We also use intra normalization for super vector based encoding methods of iDTs features. For fusion methods, we evaluate three kinds of methods, namely descriptor level fusion, representation level fusion, and score level fusion, as described in Section 3. For score level fusion, we use the geometrical mean to combine the scores from multiple SVMs.

The experimental results on three datasets are shown in Table 2, Table 3, and Table 4. From these results, we observe serval trends:

- **For iDTs features, representation level fusion is the best choice for all of the selected encoding methods on the three datasets**. This result indicates that these multiple descriptors are most correlated in the video level. Descriptor level fusion emphasizes the dependance in cuboid and results in high dimension features for codebook training and encoding. This may make these unsupervised learning algorithm unstable.
- **For STIPs features, representation level fusion is more effective for reconstruction based and voting based encoding methods**. For super vector based encoding methods, the performance of representative level fusion is comparable to that of descriptor level fusion. This trend is consistent with the finds with iDTs features.
- **For both features, SA-k, LLC, and VQ encoding methods are much sensitive to fusion methods than those super vector based encoding methods**. Great improvement can be obtained for SA-k, LLC, and VQ when using representation level fusion, but slight improvements happen to those super vector methods. We analyze this is due to two facts. Firstly, for reconstruction and voting based encoding methods, the final dimension of representation level fusion is M (the number of descriptors) times of the dimension of descriptor level fusion. However, for super vector based encoding methods, the dimension of descriptor level fusion is the same with representation level fusion. The higher dimension of final representation may enable SVM to classify more easily. Secondly, the codebook size K of super vector methods is much smaller than that of other types of encoding methods, where clustering algorithm may be more stable for high dimensionality in descriptor level fusion method.

Based on the observation and analysis above, we conclude that fusion method is a very important component for handling combination of multiple descriptors in the action recognition system. Representation level fusion method is a suitable choice for different kinds of encoding methods due to its good performance. From our analysis, we know that the performance boosting of fusing multiple features mainly owns the complementarity of these features. This complementarity may be not limited to the exploration of different descriptors, but also can be extended to the different BoVWs. From the perspective of statistics, FV aggregates information using 1^{st} and 2^{nd} order statistics, while SVC is about zero and 1^{st} order statistics. Intuitively, these two kinds of super vector encoding methods are complementary to each other. Thus, we present a new feature representation, called hybrid representation, combining the outputs FV and soft version SVC of multiple descriptors, including HOG, HOF, MBHx, and MBHy. This representation is simple but proved to be effective in next section.

4.7 Comparison to the State-of-the-Art Results

In this section, we demonstrate the effectiveness of our proposed hybrid representation according to our previous insightful analysis. Specifically, we choose two super vector based encoding methods, namely SVC-k and FV, for iDTs features. We use the power operation and then intra ℓ_2-normalization. For feature fusion, we adopt the representation level fusion method.

Table 2 Comparison of different fusion methods for the encoding methods on the **HMDB51** dataset.

Methods	FV	SVC	SVC-*k*	SVC-*all*	VLAD	VLAD-*k*	VLAD-*all*	LLC	SA-*k*	VQ
Space Time Interest Points (STIPs)										
HOG	22.81	17.76	21.09	21.87	18.13	19.87	20.04	20.46	18.39	16.10
HOF	31.96	30.44	32.68	33.36	30.46	31.53	31.55	27.19	26.27	24.49
d-Fusion	**38.82**	**35.12**	36.64	**37.19**	**34.81**	**36.18**	**36.23**	29.87	28.13	25.66
r-Fusion	37.32	34.36	**36.73**	**37.19**	34.23	35.84	35.88	**33.44**	**32.59**	**30.35**
s-Fusion	36.71	32.14	34.51	34.99	32.11	33.90	34.01	32.52	30.96	27.54
Improved Dense Trajectories (iDTs)										
HOG	45.12	36.93	39.32	38.10	36.93	39.30	37.08	37.08	35.45	34.81
HOF	50.70	47.70	49.00	48.00	47.70	49.00	45.80	42.20	42.70	42.10
MBHx	44.14	39.35	43.01	41.68	39.43	43.03	41.55	35.51	35.51	34.6
MBHy	50.04	44.25	47.02	46.51	44.27	47.02	44.68	40.39	40.35	39.78
d-Fusion	58.37	54.12	56.82	56.86	54.2	56.88	54.73	48.25	48.58	47.93
r-Fusion	**60.22**	**58.19**	**60.09**	**60.07**	**58.26**	**60.09**	**58.58**	**55.45**	**55.8**	**55.27**
s-Fusion	59.62	57.27	59.11	58.78	57.14	59.17	57.54	53.68	53.94	53.27

Table 3 Comparison of different fusion methods for the encoding methods on the **UCF50** dataset.

Methods	FV	SVC	SVC-*k*	SVC-*all*	VLAD	VLAD-*k*	VLAD-*all*	LLC	SA-*k*	VQ
Space Time Interest Points (STIPs)										
HOG	66.20	60.76	63.98	63.94	60.22	62.32	62.22	60.42	59.11	56.21
HOF	73.10	71.93	74.14	74.56	71.30	72.36	72.51	64.72	63.80	61.55
d-Fusion	**78.32**	**76.33**	77.60	77.59	**75.57**	**76.06**	**76.13**	70.13	68.66	67.16
r-Fusion	77.21	76.07	**78.42**	**78.91**	75.36	75.95	75.99	**74.05**	**73.67**	**71.95**
s-Fusion	76.33	76.19	77.76	77.25	73.79	74.91	74.98	72.95	71.66	69.16
Improved Dense Trajectories (iDTs)										
HOG	84.39	78.22	80.29	79.97	78.19	80.20	78.33	72.73	73.76	74.27
HOF	86.33	85.18	85.92	84.94	85.15	85.87	83.48	80.23	80.58	80.29
MBHx	84.03	81.33	83.19	82.46	81.28	83.12	81.16	77.77	77.91	77.04
MBHy	87.02	84.64	86.38	85.29	84.60	86.32	84.04	80.36	80.6	80.3
d-Fusion	90.84	89.39	90.72	90.62	89.43	90.64	90.18	84.18	84.76	84.67
r-Fusion	**92.07**	**90.87**	**91.89**	**91.50**	**90.82**	**91.80**	**90.56**	**87.56**	**87.92**	**88.12**
s-Fusion	91.03	90.08	90.71	90.36	90.11	90.63	89.67	87.37	87.86	87.41

Table 4 Comparison of different fusion methods for the encoding methods on the **UCF101** dataset.

Methods	FV	SVC	SVC-*k*	SVC-*all*	VLAD	VLAD-*k*	VLAD-*all*	LLC	SA-*k*	VQ
Space Time Interest Points (STIPs)										
HOG	53.74	47.56	50.07	50.31	47.15	49.21	49.35	46.70	45.79	42.85
HOF	62.89	60.57	63.81	64.02	60.04	61.73	61.60	54.16	52.78	50.04
d-Fusion	**69.90**	**66.43**	68.22	68.40	**65.42**	**66.42**	**66.46**	59.52	57.83	56.09
r-Fusion	68.21	65.39	**69.00**	**69.18**	65.39	66.13	66.19	**63.04**	**62.13**	**59.31**
s-Fusion	66.77	62.50	65.81	65.98	62.17	63.97	64.15	60.94	59.48	56.69
Improved Dense Trajectories (iDTs)										
HOG	74.79	69.74	72.14	72.36	69.66	71.65	71.39	65.46	65.81	65.40
HOF	78.63	76.26	77.70	77.12	76.28	77.76	76.35	71.03	71.14	70.57
MBHx	76.82	71.63	74.24	73.92	71.62	74.11	71.84	67.00	67.55	66.43
MBHy	79.15	74.53	77.46	76.82	74.54	76.78	74.21	69.6	69.67	68.50
d-Fusion	85.32	83.36	85.19	85.17	83.39	85.14	85.45	77.65	77.96	76.76
r-Fusion	**87.11**	**84.87**	**86.54**	**86.19**	**84.90**	**86.16**	**85.59**	**81.43**	**81.65**	**81.37**
s-Fusion	85.49	83.34	84.84	84.57	83.29	85.04	83.83	80.11	80.39	79.81

Table 5 Comparison our hybrid representation with the sate-of-the-art methods.

HMDB51	Year	%	UCF50	Year	%	UCF101	Year	%
Kuehne *et al.* [23]	2011	23.0	Sadanand *et al.* [37]	2012	57.9	Soomro *et al.* [40]	2012	43.9
Sadanand *et al.* [37]	2012	26.9	Kliper-Gross *et al.* [22]	2012	72.7	Karpthy *et al.* [21]	2014	63.3
Kliper-Gross *et al.* [22]	2012	29.2	Solmaz *et al.* [39]	2012	73.7	Cai *et al.* [6]	2014	83.5
Jiang *et al.* [17]	2012	40.7	Reddy *et al.* [36]	2012	76.9	Wu *et al.* [58]	2014	84.2
Wang *et al.* [52]	2013	42.1	Wang *et al.* [52]	2013	78.4	Peng *et al.* [34]	2013	84.2
Wang *et al.* [46]	2013	46.6	Wang *et al.* [46]	2013	84.5	Murthy *et al.* [29]	2013	85.4
Peng *et al.* [33]	2013	49.2	Wang *et al.* [51]	2013	85.7	Karaman *et al.* [20]	2013	85.7
Wang *et al.* [47]	2013	57.2	Wang *et al.* [47]	2013	91.1	Wang *et al.* [48]	2013	85.9
Hybrid representation	-	**61.1**	Hybrid representation	-	**92.3**	Hybrid representation	-	**87.9**

Table 5 shows our final recognition rates and compare our results to that of state-of-the-art approaches. For the HMDB51 dataset, we obtain a recognition rate of 61.1%, which is superior to the best result [47] by 3.9%. Our system reaches classification accuracy of 92.3% on the dataset of UCF50 and 87.9% on the dataset of UCF101, which outperform the best results by 1.2% and 2.0% respectively. It is worth noting that UCF101 is newest and largest dataset, so few published papers have reported results on this dataset. We mainly compare with those top performers in the Thumos'13 Action Recognition Challenge [18]. We also compare with three latest papers in CVPR 2014. Karpathy *et al.* [21] resorts to a large deep Convolutional Neural Network trained with an extra 1-M training dataset. Cai *et al.* [6] propose a complex and less efficient

encoding method by considering the correlation of different descriptors. Wu *et al.* [58] propose a simple, lightweight, but powerful bimodal encoding method. Our results outperform these top performer and latest papers on the UCF101. From these comparisons, our hybrid representation is an efficient and effective method and obtains the state-of-the-art performance on the three challenging datasets.

5 Conclusion

In this paper, we have comprehensively studied each step in the BoVW pipeline and tried to uncover good practice to build a more accurate and efficient action recognition system. Specifically, we mainly explore five aspects, namely local features, pre-processing techniques, encoding methods, pooling and normalization strategy, fusion methods. We conclude that every step is crucial for contributing to the final recognition rate and improper choice in one of the steps may counteract the performance improvement of other steps. Meanwhile, based on the insights from our comprehensive study, we propose a simple yet effective representation, called *hybrid representation*. Using this representation, our action recognition system obtains the state-of-the-art performance on the three challenging datasets.

References

1. Aggarwal, J.K., Ryoo, M.S.: Human activity analysis: A review. ACM Comput. Surv. **43**(3), 16 (2011)
2. Arandjelovic, R., Zisserman, A.: All about VLAD. In: CVPR, pp. 1578–1585 (2013)
3. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
4. Boureau, Y.L., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in visual recognition. In: ICML, pp. 111–118 (2010)
5. Bruckstein, A.M., Donoho, D.L., Elad, M.: From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Review **51**(1), 34–81 (2009)
6. Cai, Z., Wang, L., Peng, X., Qiao, Y.: Multi-view super vector for action recognition. In: CVPR (2014)
7. Campbell, L.W., Bobick, A.F.: Recognition of human body motion using phase space constraints. In: ICCV, pp. 624–630 (1995)
8. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM TIST **2**(3), 27 (2011)
9. Chatfield, K., Lempitsky, V.S., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: BMVC, pp. 1–12 (2011)
10. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: VS-PETS (2005)
11. Gehler, P.V., Nowozin, S.: On feature combination for multiclass object classification. In: ICCV, pp. 221–228 (2009)
12. van Gemert, J., Veenman, C.J., Smeulders, A.W.M., Geusebroek, J.M.: Visual word ambiguity. IEEE Trans. Pattern Anal. Mach. Intell. **32**(7), 1271–1283 (2010)
13. Huang, Y., Huang, K., Yu, Y., Tan, T.: Salient coding for image classification. In: CVPR, pp. 1753–1760 (2011)
14. Huang, Y., Wu, Z., Wang, L., Tan, T.: Feature coding in image classification: A comprehensive study. IEEE Trans. Pattern Anal. Mach. Intell. **36**(3), 493–506 (2014)
15. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: NIPS, pp. 487–493 (1998)
16. Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C.: Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. **34**(9), 1704–1716 (2012)
17. Jiang, Y.G., Dai, Q., Xue, X., Liu, W., Ngo, C.W.: Trajectory-based modeling of human actions with motion reference points. In: ECCV, pp. 425–438 (2012)
18. Jiang, Y.G., Liu, J., Roshan Zamir, A., Laptev, I., Piccardi, M., Shah, M., Sukthankar, R.: THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/ICCV13-Action-Workshop/` (2013)
19. Johnson, S.C.: Hierarchical clustering schemes. Psychometrika **32**(3), 241–254 (1967)
20. Karaman, S., Seidenari, L., Bagdanov, A.D., Bimbo, A.D.: L1-regularized logistic regression stacking and transductive CRF smoothing for action recognition in video. In: ICCV Workshop on Action Recognition with a Large Number of Classes (2013)
21. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
22. Kliper-Gross, O., Gurovich, Y., Hassner, T., Wolf, L.: Motion interchange patterns for action recognition in unconstrained videos. In: ECCV (2012)
23. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: A large video database for human motion recognition. In: ICCV, pp. 2556–2563 (2011)
24. Laptev, I.: On space-time interest points. International Journal of Computer Vision **64**(2-3), 107–123 (2005)
25. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR, pp. 1–8 (2008)
26. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, pp. 2169–2178 (2006)
27. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS, pp. 801–808 (2006)
28. Liu, L., Wang, L., Liu, X.: In defense of soft-assignment coding. In: ICCV, pp. 2486–2493 (2011)
29. Murthy, O.V.R., Goecke, R.: Combined ordered and improved trajectories for large scale human action recognition. In: ICCV Workshop on Action Recognition with a Large Number of Classes (2013)
30. Myers, G.K., Nallapati, R., van Hout, J., Pancoast, S., Nevatia, R., Sun, C., Habibian, A., Koelma, D., van de Sande, K.E.A., Smeulders, A.W.M., Snoek, C.G.M.: Evaluating multimedia features and fusion for example-based event detection. Mach. Vis. Appl. **25**(1), 17–32 (2014)
31. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS, pp. 849–856 (2001)
32. Niyogi, S.A., Adelson, E.H.: Analyzing and recognizing walking figures in XYT. In: CVPR, pp. 469–474 (1994)
33. Peng, X., Qiao, Y., Peng, Q., Qi, X.: Exploring motion boundary based sampling and spatial-temporal context descriptors for action recognition. In: BMVC, pp. 1–11

34. Peng, X., Wang, L., Cai, Z., Qiao, Y., Peng, Q.: Hybrid super vector with improved dense trajectories for action recognition. In: ICCV Workshop on Action Recognition with a Large Number of Classes (2013)
35. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV, pp. 143–156 (2010)
36. Reddy, K.K., Shah, M.: Recognizing 50 human action categories of web videos. Mach. Vis. Appl. **24**(5), 971–981 (2013)
37. Sadanand, S., Corso, J.J.: Action bank: A high-level representation of activity in video. In: CVPR (2012)
38. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: ICCV, pp. 1470–1477 (2003)
39. Solmaz, B., Assari, S.M., Shah, M.: Classifying web videos using a global video descriptor. Mach. Vis. Appl. **24**(7), 1473–1485 (2013)
40. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR **abs/1212.0402** (2012)
41. Tang, K.D., Yao, B., Li, F.F., Koller, D.: Combining the right features for complex event recognition. In: ICCV, pp. 2696–2703 (2013)
42. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. IEEE Transactions on Information Theory **53**(12), 4655–4666 (2007)
43. Turaga, P.K., Chellappa, R., Subrahmanian, V.S., Udrea, O.: Machine recognition of human activities: A survey. IEEE Trans. Circuits Syst. Video Techn. **18**(11), 1473–1488 (2008)
44. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: ICCV, pp. 606–613 (2009)
45. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 480–492 (2012)
46. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision **103**(1), 60–79 (2013)
47. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV, pp. 3551–3558 (2013)
48. Wang, H., Schmid, C.: Lear-inria submission for the thumos workshop. In: ICCV Workshop on Action Recognition with a Large Number of Classes (2013)
49. Wang, H., Ullah, M.M., Kläser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC, pp. 1–11 (2009)
50. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR, pp. 3360–3367 (2010)
51. Wang, L., Qiao, Y., Tang, X.: Mining motion atoms and phrases for complex action recognition. In: ICCV, pp. 2680–2687 (2013)
52. Wang, L., Qiao, Y., Tang, X.: Motionlets: Mid-level 3D parts for human motion recognition. In: CVPR, pp. 2674–2681 (2013)
53. Wang, L., Qiao, Y., Tang, X.: Latent hierarchical model of temporal structure for complex activity classification. IEEE Transactions on Image Processing **23**(2), 810–822 (2014)
54. Wang, X., Wang, L., Qiao, Y.: A comparative study of encoding, pooling and normalization methods for action recognition. In: ACCV, pp. 572–585 (2012)
55. Webb, J.A., Aggarwal, J.K.: Structure from motion of rigid and jointed objects. In: IJCAI, pp. 686–691 (1981)
56. Willems, G., Tuytelaars, T., Gool, L.J.V.: An efficient dense and scale-invariant spatio-temporal interest point detector. In: ECCV, pp. 650–663 (2008)
57. Wu, J.: Towards good practices for action video encoding. In: ICCV Workshop on Action Recognition with a Large Number of Classes (2013)
58. Wu, J., Zhang, Y., Lin, W.: Towards good practices for action video encoding. In: CVPR (2014)
59. Wu, Z., Huang, Y., Wang, L., Tan, T.: Group encoding of local features in image classification. In: ICPR, pp. 1505–1508 (2012)
60. Yacoob, Y., Black, M.J.: Parameterized modeling and recognition of activities. Computer Vision and Image Understanding **73**(2), 232–247 (1999)
61. Yang, J., Yu, K., Gong, Y., Huang, T.S.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR, pp. 1794–1801 (2009)
62. Yu, K., Zhang, T.: Improved local coordinate coding using local tangents. In: ICML, pp. 1215–1222 (2010)
63. Yu, K., Zhang, T., Gong, Y.: Nonlinear learning using local coordinate coding. In: NIPS, pp. 2223–2231 (2009)
64. Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. International Journal of Computer Vision **73**(2), 213–238 (2007)
65. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image classification using super-vector coding of local image descriptors. In: ECCV, pp. 141–154 (2010)