A Comparative Study of Encoding, Pooling and Normalization Methods for Action Recognition

Xingxing Wang[1], LiMin Wang[1,2], and Yu Qiao[1,2]

[1] Shenzhen Key lab of CVPR, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2] Department of Information Engineeing, The Chinese University of Hong Kong
{xx.wang,lm.wang,yu.qiao}@siat.ac.cn

Abstract. Bag of visual words (BoVW) models have been widely and successfully used in video based action recognition. One key step in constructing BoVW representation is to encode feature with a codebook. Recently, a number of new encoding methods have been developed to improve the performance of BoVW based object recognition and scene classification, such as soft assignment encoding [1], sparse encoding [2], locality-constrained linear encoding [3] and Fisher kernel encoding [4]. However, their effects for action recognition are still unknown. The main objective of this paper is to evaluate and compare these new encoding methods in the context of video based action recognition. We also analyze and evaluate the combination of encoding methods with different pooling and normalization strategies. We carry out experiments on KTH dataset [5] and HMDB51 dataset [6]. The results show the new encoding methods can significantly improve the recognition accuracy compared with classical VQ. Among them, Fisher kernel encoding and sparse encoding have the best performance. By properly choosing pooling and normalization methods, we achieve the state-of-the-art performance on HMDB51.[1]

1 Introduction

Human action recognition has become and will continue to be a a highly active research area of research due to its wide applications in video surveillance, human-computer interface, sports video analysis, and content based video retrieval [7,8,5,9,10,11,12,13]. The difficulty of video based action recognition comes from the high dimension of video data, the complexity and intra-class variability of action, etc. In recent years, the computer vision society has witnessed the popularity and success of bag of visual words (BoVW) models in action recognition [5,9,10,11,12]. BoVW, originated from bag of words model in natural language processing, treats visual features as words and images (or videos) as documents [14,15,1,3,2]. The effectiveness and efficiency of BoVW have been exhibited in extensive image and video classification tasks.

[1] The code and supplemental materials are publicly available at
http://mmlab.siat.ac.cn/accv2012

The classical approaches to obtain BoVW representations of action video usually consists of three steps: feature extraction, vector quantilization (VQ) of feature, and sum-pooling with ℓ_1 normalization to construct histogram representation [5,9,10,11,12]. Then the final histogram representation is fed into non-linear kernel SVM for classification. These settings have their root in traditional choice in image classification [14,16,17]. Recent image classification studies yield alterative choices in the second step (encoding) and the third step (pooling and normalization), which lead to better recognition performance. In addition to VQ, several effective encoding methods have been proposed, such as: Fisher Kernel Encoding [4], Soft-assignment Encoding [15,1], Sparse Encoding [2], Locality-constrained Linear Coding [3]. Besides, many novel pooling and normalization methods have also been proposed to further improve the recognition performance. All of these progresses have proved to be effective in image classification tasks such as the PASCAL VOC challenge [18]. Furthermore, most of these new methods use linear SVM, which is more computational efficient than non-linear kernel used in the classical method.

In spite of the success of these new encoding methods and pooling normalization strategies in image tasks, these methods are generally unaware in the context of video-based action classification so far, at least to the best of our knowledge. Similar to image tasks, it can be expected that the new encoding and pooling-normalization methods can affect and improve the action recognition performance.

In this paper, we investigate and evaluate encoding, pooling and normalization methods in BoVW based action recognition. The objectives are twofold. The first is to examine whether and how these methods can improve action recognition performance. The second is to figure out the key factors in constructing high performance action recognition system. For feature encoding, we compare five encoding methods: vector quantization encoding, (localized) soft-assignment encoding, sparse encoding, locality-constrained linear encoding and Fisher encoding. We also investigate other aspects of the encoding methods such as sensitivity to codebook size and computational cost. For pooling and normalization, we analyze max pooling and sum pooling techniques for different encoding methods, which are followed by different normalization choice such as: ℓ_1-normalization, $\ell_2$normalization and power normalization. We carry out experiments on two public datasets: a relative simple and easy dataset KTH [5] and a challenging dataset HMDB51 [6], and systematically analyze the experimental results. We find simple combinations of encoding and pooling get state-of-the-art results on a large and complex dataset. The three main contributions of this paper are summarized as follows:

- To our best knowledge, we first introduce these new encoding methods to action recognition and evaluate their performance with public datasets. Furthermore, we explore the properties of these encoding methods such as sensitivity to codebook size and computational cost.
- We exploit different pooling and normalization strategies, which plays an important role in improving the recognition accuracy from experiment result.

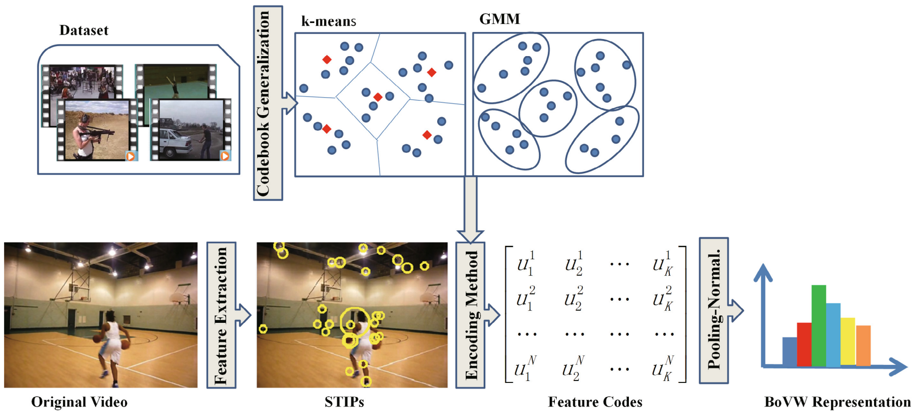

Fig. 1. The pipeline of bag-of-visual-words method for video based action recognition: (i) codebook generation (ii) feature extraction (iii) feature encoding (iv) pooling and normalization

– The proposed methods achieve the state-of-the-art performance on the challenging HMDB51 dataset.

2 Methods

As shown in Figure 1, the typical pipeline to obtain BoVW representation of action video is usually composed of three main steps: (i) spatial temporal feature extraction (e.g. STIP [19], Cuboids [20]); (ii) encoding the local features; (iii) feature pooling and normalization. Then the BoVW representation is input to certain classifier (e.g. SVM) for action recognition. In this paper, we make use of STIP detectors and HOG/HOF features due to its good performance [21]. Previous action recognition methods [5,9,10,11,12] mainly use vector quantization (VQ) for step (ii), and sum pooling and ℓ_1-normalization is used for step (iii). In the next, we describe alternative choices for each of the steps along this pipeline.

2.1 Codebook Generation

To begin with, we need to construct a codebook from a set of input descriptors. Generally, there are two approaches: (i) partitioning the feature space into informative regions which are called visual words (codewords) to construct visual dictionary (codebook), and (ii) using generative model to capture the probability distribution of features. k-mean is a typical method for the first type, while Gaussian Mixture Model (GMM) is widely used for the second.

k-means. There are many vector quantization methods such as k-means clustering [22], hierarchical clustering [23] and spectral clustering [24]. Among them, k-means is probable the most popular way to construct visual dictionary. Given a set of local features $\{\mathbf{x}_1, \cdots, \mathbf{x}_M\}$ $\mathbf{x}_m \in \mathbb{R}^D$. 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 $r_{mk} \in \{0, 1\}$, where if feature $\mathbf{x}_m$ is assigned to cluster k, then $r_{mk} = 1$ and $r_{mj} = 0$ for $j \neq k$. We can then define an objective function:

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

The objective is to find values for $\{r_{mk}\}$ and $\{\mathbf{d}_k\}$ to minimize the objective function $\mathcal{J}$ [2]. Usually, we can optimize it in an iterative procedure where each iteration involves two successive steps corresponding to successive optimization with respect to the r_{nk} and $\mathbf{d}_k$. The details can be found in [22].

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

$$p(\mathbf{x}; \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k), \tag{2}$$

where K is mixture number and $\theta = \{\pi_1, \mu_1, \Sigma_1, \cdots, \pi_K, \mu_K, \Sigma_K\}$ is 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 $\ln p(\mathbf{X}; \theta) = \sum_m \ln p(x_m; \theta)$. We use the iterative EM algorithm [22] to solve this problem.

GMM can be seen as a 'soft' clustering algorithm. 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.

2.2 Encoding Methods

In this section we describe several encoding methods investigated in this paper. Let $\mathbf{X}$ be a set of D-dimensional local descriptors extracted from a video, *i.e.* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$. Given a visual dictionary with K visual words, *i.e.* $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K] \in \mathbb{R}^{D \times K}$. The objective of encoding is to compute a code for input $\mathbf{x}$ with $\mathbf{D}$. We use $\mathbf{u}_n$ to denote the code vector. The dimension of $\mathbf{u}_n$ is the same as the size of $\mathbf{D}$ except Fisher kernel representation.

Vector Quantization (VQ). VQ is also known as *Hard-assignment coding*, first introduced in object recognition in [14]. For each local feature descriptor $\mathbf{x}_n$, it is represented by its nearest visual word in the dictionary:

$$u_{nk} = \begin{cases} 1. & \text{if } k = \arg\min_k \|\mathbf{x}_n - \mathbf{d}_k\|^2. \\ 0. & \text{otherwise.} \end{cases} \tag{3}$$

[2] In this paper, without any additional specification, $\|\cdot\|$ represents the ℓ_2-norm, i.e. $\|\mathbf{x}\| = \sqrt{\sum_{i=1}^{D} x_i^2}$.

Soft-assignment Encoding (SA). SA is first introduced in [15] for scene classification. For each local feature, the k^{th} coefficient represents the degree of membership of the local feature $\mathbf{x}_n$ being to the k^{th} visual word:

$$u_{nk} = \frac{\exp(-\beta\|\mathbf{x}_n - \mathbf{d}_k\|^2)}{\sum_{j=1}^{K}\exp(-\beta\|\mathbf{x}_n - \mathbf{d}_j\|^2)}, \tag{4}$$

where β is a smoothing factor controlling the softness of the assignment. Note that all the K visual words are used in computing u_{nk}. Recently [1] developed a *localized soft-assignment coding.* They only considered the k nearest visual words into encoding, and conceptually set its distances to the remaining words as infinity:

$$u_{nk} = \frac{\exp(-\beta\hat{d}(\mathbf{x}_n, \mathbf{d}_k))}{\sum_{j=1}^{K}\exp(-\beta\hat{d}(\mathbf{x}_n, \mathbf{d}_j))}, \tag{5}$$

where $\hat{d}(\mathbf{x}_n, \mathbf{d}_k)$ is defined as follows:

$$\hat{d}(\mathbf{x}_n, \mathbf{d}_k) = \begin{cases} \|\mathbf{x}_n - \mathbf{d}_k\|^2 & \text{if } \mathbf{d}_k \in N_k(\mathbf{x}_n), \\ \infty & \text{otherwise}, \end{cases} \tag{6}$$

where $N_k(\mathbf{x}_n)$ denotes the k-nearest neighbors of $\mathbf{x}_n$ defined by the distance $\|\mathbf{x}_n - \mathbf{d}_k\|^2$.

Sparse Encoding (SPC). SPC is proposed for object recognition by [2]. It represents a local feature $\mathbf{x}_n$ by a sparse linear combination of basis vectors. The coefficient vector $\mathbf{u}_n$ is obtained by solving an ℓ_1-norm regularized approximation problem:

$$\mathbf{u}_n = \underset{\mathbf{u}\in\mathbb{R}^K}{\arg\min}\, \|\mathbf{x}_n - \mathbf{D}\mathbf{u}\|^2 + \lambda\|\mathbf{u}\|_1. \tag{7}$$

Locality-constrained Linear Encoding (LLC). It is introduced in [3] for image classification. Unlike the sparse coding, LLC enforces locality instead of sparsity and this leads to smaller coefficient for the basis vectors far away from the local feature $\mathbf{x}_n$. The coding coefficients are obtained by solving the following optimization:

$$\begin{aligned} \mathbf{u}_n &= \underset{\mathbf{u}\in\mathbb{R}^K}{\arg\min}\, \|\mathbf{x}_n - \mathbf{D}\mathbf{u}\|^2 + \lambda\|\mathbf{s}_n \odot \mathbf{u}\|^2. \\ \text{s.t.} &\quad \mathbf{1}^T\mathbf{u}_n = 1. \end{aligned} \tag{8}$$

where $\odot$ denotes the element-wise multiplication and $\mathbf{s}_n$ is the locality adaptor that gives weights for each basis vector proportional to its similarity to the input descriptor $\mathbf{x}_n$:

$$\mathbf{s}_n = \exp\left(\frac{\text{dist}(\mathbf{x}_n, \mathbf{D})}{\sigma}\right), \tag{9}$$

where $\text{dist}(\mathbf{x}_n, \mathbf{D}) = [\text{dist}(\mathbf{x}_n, \mathbf{d}_1), \cdots, \text{dist}(\mathbf{x}_n, \mathbf{d}_K)]^T$ and $\text{dist}(\mathbf{x}_n, \mathbf{d}_k)$ is the Euclidean distance between $\mathbf{x}_n$ and $\mathbf{d}_k$. σ is used for adjusting the weighted decay speed for the locality adaptor. The constraint $\mathbf{1}^T\mathbf{u}_n = 1$ follows the shift-invariant requirements of the LLC code. In practice, an approximation is proposed to improve its computational efficiency. Ignoring the second term in Equation (8), it directly selects the k nearest basis vectors of $\mathbf{x}_n$ to minimize the first term by solving a much smaller linear system. This gives the coding coecients for the selected k basis vectors and other coefficients are simply set to zero.

Fisher Kernel Encoding (FK). Fisher kernel is introduced for large-scale image categorization [4]. Unlike previous coding methods based on a codebook, the fisher kernel is a generic framework which combines the benefits of generative and discriminative approaches. Suppose we has a generative model $p(\mathbf{x};\theta)$ in feature space. Let $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\}$ be the set of T local features extracted from a video. Then the video can be described by the gradient vector of log likelihood with respect to the model parameters. [25]:

$$G_\theta^{\mathbf{X}} = \frac{1}{T}\nabla_\theta \log p(\mathbf{X};\theta). \tag{10}$$

Note that the dimensionality of this vector depends only on the number of parameters in θ, not on the number of local features T. A natural kernel on these gradients is:

$$K(\mathbf{X},\mathbf{Y}) = G_\theta^{\mathbf{X}T} F_\theta^{-1} G_\theta^{\mathbf{Y}}. \tag{11}$$

where F_θ is the Fisher information matrix of $p(\mathbf{x};\theta)$:

$$F_\theta = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x};\theta)}[\nabla_\theta \log p(\mathbf{x};\theta)\nabla_\theta \log p(\mathbf{x};\theta)^T]. \tag{12}$$

As F_θ is symmetric and positive definite, then we can define the *Fisher Vector* as:

$$\mathcal{G}_\theta^{\mathbf{X}} = F_\theta^{-1/2} G_\theta^{\mathbf{X}}. \tag{13}$$

Here we use Gaussian Mixture Model for $p(x;\theta)$, Assume that the covariance matrices Σ_k are diagonal. Fisher coding can be derived as:

$$\mathcal{G}_{\mu,k}^{\mathbf{X}} = \frac{1}{T\sqrt{\pi_k}} \sum_{t=1}^{T} \gamma_t(k) \left(\frac{\mathbf{x}_t - \mu_k}{\sigma_k}\right), \tag{14}$$

$$\mathcal{G}_{\sigma,k}^{\mathbf{X}} = \frac{1}{T\sqrt{\pi_k}} \sum_{t=1}^{T} \gamma_t(k) \left[\frac{(\mathbf{x}_t - \mu_k)^2}{\sigma_k^2} - 1\right]. \tag{15}$$

where $\gamma_t(k)$ is the soft assignment of local feature $\mathbf{x}_t$ to i-th Gaussian i:

$$\gamma_t(k) = \frac{\pi_k \mathcal{N}(\mathbf{x};\mu_k,\Sigma_k)}{\sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x};\mu_i,\Sigma_i)}. \tag{16}$$

The final gradient vector $\mathbf{u}$ is the concatenation of $\mathcal{G}_{\mu,k}^{\mathbf{X}}$ and $\mathcal{G}_{\sigma,k}^{\mathbf{X}}$ and its total dimension is $2KD$.

2.3 Pooling and Normalization

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

- **Sum Pooling.** With sum pooling scheme [16], the k^{th} component of $\mathbf{p}$ is $p_k = \sum_{n=1}^{N} u_{nk}$.
- **Max Pooling.** With max pooling scheme [2], the k^{th} component of $\mathbf{p}$ is $p_k = \max(u_{1k}, u_{2k}, \cdots, u_{nk})$.

In [26], the authors presented a theoretical analysis of average pooling and max pooling. Their results indicate sparse features may prefer max pooling.
Then pooled feature $\mathbf{p}$ is further normalized by some methods. Generally, there are three common normalization techniques:

- **ℓ_1-Normalization.** In ℓ_1 normalization [2], the feature $\mathbf{p}$ is divided by its ℓ_1-norm: $\mathbf{p} = \mathbf{p}/\sum_{k=1}^{K} |p_k|$.
- **ℓ_2-Normalization.** In ℓ_2 normalization [4], the feature $\mathbf{p}$ is divided by its ℓ_2-norm: $\mathbf{p} = \mathbf{p}/\sqrt{(\sum_{k=1}^{K} p_k^2)}$.
- **Power Normalization.** In power normalization [4], 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.

3 Experiments

In this section, we describe the experimental settings and results for different encoding methods and different pooling-normalization strategies. We also analyze and compare different combinations of encoding methods and pooling-normalization strategies. To the best of our knowledge, most of encoding methods and pooling-normalization strategies are introduced to and evaluated on action recognition for the first time.

3.1 Experiment Settings and Dataset

We conduct experiments on two public datasets: KTH [5] and HMDB51 [6]. KTH dataset is one of the earliest datasets for action recognition and is relative simple. It contains 6 action classes: walking, running, jogging, hand-waving, hand-clapping and boxing [3]. Each action was performed by 25 actors in four different scenarios: outdoors, outdoors with scale variation, outdoors with different cloths and indoors. HMDB51 is a new and probable the largest human motion

[3] `http://www.nada.kth.se/cvap/actions/`

recognition dataset so far. It include 51 action classes and each class has more than 100 videos [4]. 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. In both datasets, we use SVM with a linear kernel for classification. Specifically, we use LibSVM [27] and adopt one-versus-other training scheme. We detect spatial-temporal interest points (STIPs) and extract histogram of gradients and flow (HOG/HOF) for both dataset. For the comparison fair, we use the same codebook for all encoding methods.

3.2 Exploration of Encoding Methods

In this subsection, we compare and analyze the performance of different encoding methods. For each encoding method, we choose the standard settings proposed by previous papers such as pooling-normalization strategy, parameter settings. We will investigate how pooling-normalization strategies effect in the next subsection.

Baseline: Vector Quantization Encoding. The baseline encoding method is a histogram of visual word obtained with hard assignment. Following the common settings in object recognition [17], the histogram is obtained by sum pooling and normalized with ℓ_1-norm.

(Localized) Soft-assignment Encoding. The soft-assignment encoding has two forms: assigning each feature to all codeword (SA-*all*) [15] and assigning each feature to k nearest neighborhood (SA-k) [1]. It requires a single parameter β which is the smoothing factor controlling the softness. We set β as 1 for both SA-*all* and SA-k. For SA-k, we set k as 5 and use max-pooling and ℓ_2-normalization due to its good performance in [1].

Locality-constrained Linear Encoding. Following [3], we use approximated LLC for fast Encoding. Instead of solving Equation (8), we simply use k nearest neighbors of features as the local bases. We set k as 5 and use max-pooling and ℓ_2-normalization as [3].

Sparse Encoding. Sparse Encoding (SPC) requires parameter λ, to balance between sparsity regularization term and loss term. Following [2], we choose the default λ as 0.15 and use max pooling and ℓ_2-normalization.

Fisher Kernel Encoding. Fisher kernel(FK) requires to train a GMM for input features. Since the HOG/HOF feature has a high dimension of 162 and includes redundancy, directly using EM algorithm suffers from the singular problem of covariance matrix. Therefore, we first use PCA to reduce its dimensionality to 100. In our experiments, PCA proves be effective to avoid the singular problem and leads to better recognition results. Another advantage of using PCA

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

Table 1. Comparison of different encoding methods on KTH: Vector Quantization(VQ), Soft-assignment Encoding(SA-k), Fisher Kernel Encoding(FK), Local-constrain Linear Encoding(LLC) and Sparse Encoding(SPC)

Encoding Methods	VQ	SA-*all*	SA-*k*	LLC	SPC	FK
Codebook Size	1k	1k	1k	1k	1k	256
Pooling-Normalization	sum-ℓ_1	max-ℓ_2	max-ℓ_2	max-ℓ_2	max-ℓ_2	P-ℓ_2
Classification Accuracy	86.11	89.81	88.89	89.81	90.74	**92.13**

Table 2. Comparison of different encoding methods on HMDB51: Vector Quantization(VQ), Localized Soft-assignment(SA-k), Fisher Encoding(FK), Local-constrain Linear Encoding(LLC) and Sparse Encoding(SPC)

Method	Codebook size	Pooling	Normalization	Parameter Setting	Accuracy
FK	64	-	P+ℓ_2	PCA=100	26.58
FK	128	-	P+ℓ_2	PCA=100	28.23
FK	256	-	P+ℓ_2	PCA=100	**29.22**
VQ	1k	sum	ℓ_1	-	16.84
SA-*all*	1k	max	ℓ_2	$\beta = 1$	18.69
SA-*k*	1k	max	ℓ_2	$k = 5, \beta = 1$	22.68
LLC	1k	max	ℓ_2	$k = 5$	20.89
SPC	1k	max	ℓ_2	$\lambda = 0.15$	**22.92**
VQ	6k	sum	ℓ_1	-	19.67
SA-*all*	6k	max	ℓ_2	$\beta = 1$	20.59
SA-*k*	6k	max	ℓ_2	$k = 5, \beta = 1$	28.82
LLC	6k	max	ℓ_2	$k = 5$	27.76
SPC	6k	max	ℓ_2	$\lambda = 0.15$	**29.20**
VQ	8k	sum	ℓ_1	-	18.89
SA-*all*	8k	max	ℓ_2	$\beta = 1$	20.94
SA-*k*	8k	max	ℓ_2	$k = 5, \beta = 1$	28.98
LLC	8k	max	ℓ_2	$k = 5$	28.68
SPC	8k	max	ℓ_2	$\lambda = 0.15$	**29.97**

is to reduce the computational cost in encoding and classification. Following [4], we choose power normalization and ℓ_2-normalization for the final feature codes.

Results. The experimental results are summarized in Table (1) and Table (2). For KTH dataset, the number of action category and the number of videos are relative small. We follow the normal settings in previous works and set the codebook size as 1k. For the complex HMDB51 dataset, we conduct experiments with different codebook sizes (e.g., 1k, 4k, 6k, 8k). For FK, we use GMM with different mixtures (e.g. 128, 256). As shown in Table (1) and Table (2), the new encoding methods SA-*k*, SA-*all*, LLC, SPC and FK always outperform the baseline VQ in both datasets. Among the new encoding methods, FK and SPC seem to receive better performance. Compared with VQ, FK can improve the recognition rate about 6% for KTH and about 9% for HMDB51. We also noticed that local soft assignment (SA-*k*) has better results than classical soft assignment (SA-*all*). For all the methods compared, the accuracy increases as codebook size enlarges As shown in Figure 2.

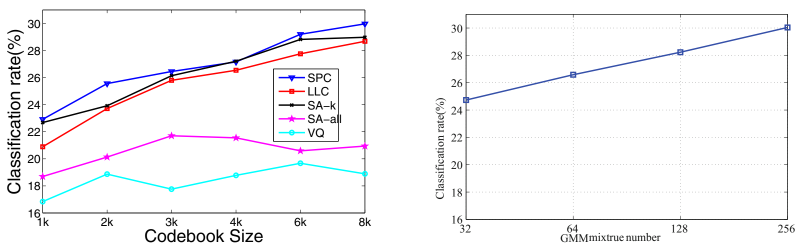

Fig. 2. Exploration of performance of different encoding methods with changing codebook size. Left: Codebook based encoding methods: Right: GMM based encoding methods.

Analysis. From the results on both datasets, we observe that classical VQ encoding receives the lowest recognition results among the other encoding methods compared. In VQ, each feature descriptor is represented by a single basis (codeword) in the codebook. If a feature locates at the middle between two or more code words, a slight change of the feature can lead to different coding words. This ambiguity makes the encoding unstable, and can lead to lower recognition rates. In soft assignment (SA-all), each feature descriptor is represented by the degree of membership to all the code words. This can overcome the codeword ambiguity to certain degree. These facts may explain why the performance of soft assignment is better than VQ. However, most of code words are far from the input feature vector and should have no contribution to the coding.

SA-all ignores the sparsity and locality in the encoding process. Sparsity can help to choose more discriminative code words and reduce the redundancy of data representation. Locality can help to handle the underlying manifold structure of the feature space. Thus, the localized soft-assignment encoding (SA-k) takes account of only the k-nearest code words into encoding and significantly improves the recognition performance. In essence, LLC is similar to SA-k for that they both consider locality. The only difference is that coefficients of SA-k can be explained as the posterior probability that the feature belong to each code words while the coefficients of LLC are related to the projections of feature to the code words. As shown in Table 1 and Table 2, the results of LLC and SA-k are similar to each other. SPC has the best performance among the codebook based encoding methods. This may be ascribed to the fact that SPC can adaptively determine the number of code words used in coding with sparse constraint, while LLC and SA-k fix the number of code words. The adaptive sparsity of each codes may be very important for encoding. We also notice that Fisher kernel (FK) yields good performance in the experiments. Different from codebook based encoding methods, FK characterizes input feature descriptor with a gradient vector derived from a generative probability model e.g. GMM. This mechanism allows it to combine the strengths of generative and discriminative approaches.

Computational Cost. In addition to the classification accuracy, we also compare the computational efficiency of the encoding methods. Our codes are all

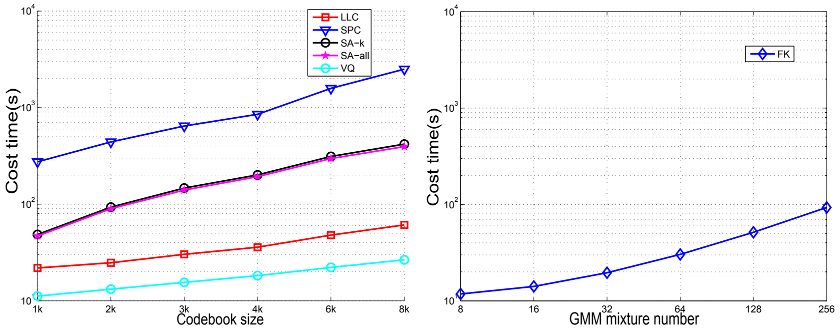

Fig. 3. Exploration of the computational cost of different encoding methods. Left: Codebook based encoding methods: Right: GMM based encoding methods.

implemented in Matlab, running on computer with i5 2400 CPU (3.7 GHZ). We randomly choose 100 videos and run different encoding methods for the videos. The running time of different encoding methods are shown in Figure 3 (Note that the scale of Y-axis is log scale). We observe that LLC and VQ are more computational efficient among the methods compared. The efficiency of SA-*all* and SA-k are near. SPC needs to solve the ℓ_1-minimization problem and requires heavier computational cost than other methods. For example, when codebook size is 8k, the time-cost of SPC is almost 100 times of VQ and 50 times of LLC. FK is also computationally efficient compared with SA and SPC. Note that the final dimension of FK is $2DK$ (D is the dimension of descriptor and K is mixture number) , much larger than the size of codebook. Thus the computational cost of training SVM for FK is much larger than the other encoding methods.

3.3 Exploration of Pooling-Normalization

In this subsection, we mainly analyse different pooling-normalization strategies. Note the pooling strategies only work for the new codebook based methods such as: SA-k, LLC, and SPC. We try two pooling methods: max pooling (m) and sum-pooling (s), which is followed by four normalization strategies: ℓ_1-normalization (ℓ_1), ℓ_2-normalization (ℓ_2), power normalization+ℓ_1-normalization (P+ℓ_1) and power normalization+ℓ_2-normalization (P+ℓ_2). For VQ and FK, we test four normalization strategies. The result is shown in Figure (4), where α of power normalization is set to 0.5, codebook size is set as 8k and GMM mixture number is 128.

For LLC, SA-k and SPC, ℓ_2-normalization is better than ℓ_1-normalization no matter using or without using normalization. In general, when choosing max pooling, the result of with power normalization is worse than without power normalization. When using sum pooling, the result with power normalization is better than without power normalization. The effect of power normalization is to smooth the histogram which narrow the difference between large value and small value. The histogram of max pooling itself is very sharp, thus the power

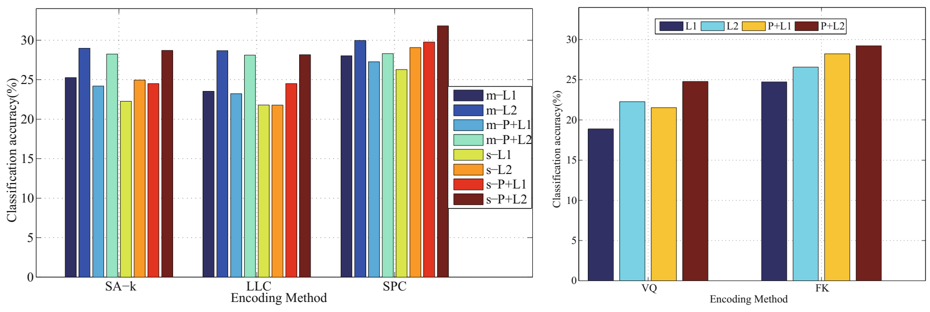

Fig. 4. Comparison of different pooling-normalization strategies on HMDB51. Note that there is only sum pooling for VQ and no pooling for FK.

Table 3. Comparison our proposed methods with state of the art on KTH

Methods	Ours (FK)	Schüldt [5]	Laptev [21]	Ryoo [29]	Liu [30]	Sada [28]
Accuracy(%)	92.1	71.7	91.8	91.1	91.6	**98.2**

normalization can reduce the performance. While, the histogram of sum pooling is usually very smooth and the power normalization can help sharp the histogram, improving the representation. We can conclude that the best choice for max-pooling is ℓ_2-normalization and for sum-pooling is power normalization+ℓ_2-normalization. The best performance for HMDB51 dataset is achieved by SPC with sum pooling and power+ℓ_2 normalization.

3.4 Comparison with the State of the Art

We compare our results with the state-of-the-art method action recognition methods [28]. The results are sumarized in Table (3) and Table (4). For KTH dataset, we see our result is comparable to most of the recently published methods. In [28], the authors proposed action bank for action recognition and achieved good performance on KTH dataset. However, this method requires to use a number of action template as detectors, which is compositional expensive in practice. HMDB51 includes complex actions with large variations. There are not too many papers reporting the experimental result on this dataset. From the result, we see that our SPC with sum-pooling and power-ℓ_2 normalization achieves the best among those compared. Compared with the action bank method, our method can improve the recognition accuracy nearly 5% with less computational cost. .

4 Conclusion

Encoding, pooling and normalization are necessary steps in constructing Bag of Visual Words (BoVW) based image and video representations. In this paper,

Table 4. Comparison our proposed methods with state of the art on HMDB51

Methods	Ours (SPC-s-P+ℓ_2)	HOG/HOF [6]	C2 [6]	Action Bank [28]
Accuracy(%)	**31.82**	20.44	22.83	26.9

we introduce these encoding methods. i.e. soft assignment, locality-constrained linear coding, sparse coding, and Fisher kernel encoding, to video based action recognition. We also investigate the affection of pooling-normalization strategies and their combination with encoding methods. We conduct experiments on a relative simple KTH dataset and a more challenging HMDB51 dataset. The results show the new encoding methods and pooling-normalization strategies can significantly improve the recognition accuracy on both datasets. Sparse coding and Fisher kernel achieve higher recognition accuracy than other methods in most of the experiments. For pooling-normalization strategies, we find that max-pooling favors ℓ_2-normalization, while sum-pooling favors power normalization+ℓ_2-normalization. With SPC-s-P+ℓ_2, we achieves the state of the art performance on HMDB51.

Acknowledgement. This work is partly supported by Shenzhen Basic Research Program for Distinguished Young Scholar (JC201005270350A), 100 Talents Programme of Chinese Academy of Sciences, National Natural Science Foundation of China (61002042), Foshan-CAS collaboration project (2011BY100082) and Guangdong Innovative Research Team Program(No. 201001D0104648280).

References

1. Liu, L., Wang, L., Liu, X.: In defense of soft-assignment coding. In: ICCV, pp. 2486–2493 (2011)
2. 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)
3. 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)
4. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for Large-Scale Image Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
5. Schüldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: ICPR, vol. 3, pp. 32–36 (2004)
6. 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)
7. Aggarwal, J., Ryoo, M.: Human activity analysis: A review. ACM Comput. Surv. 43, 1–43 (2011)
8. Turaga, P.K., Chellappa, R., Subrahmanian, V.S., Udrea, O.: Machine recognition of human activities: A survey. TCSVT, 1473 –1488 (2008)

9. Niebles, J.C., Chen, C.-W., Fei-Fei, L.: Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 392–405. Springer, Heidelberg (2010)
10. Tang, K., Fei-Fei, L., Koller, D.: Learning latent temporal structure for complex event detection. In: CVPR, pp. 1250–1257 (2012)
11. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR, pp. 2929–2936 (2009)
12. Kovashka, A., Grauman, K.: Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In: CVPR, pp. 2046–2053 (2010)
13. Brendel, W., Todorovic, S.: Learning spatiotemporal graphs of human activities. In: ICCV, pp. 778–785 (2011)
14. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV Workshop on Statistical Learning in Computer Vision, pp. 1–22 (2004)
15. van Gemert, J.C., Geusebroek, J.-M., Veenman, C.J., Smeulders, A.W.M.: Kernel Codebooks for Scene Categorization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 696–709. Springer, Heidelberg (2008)
16. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
17. Zhang, J., Marszalek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: A comprehensive study. IJCV 73, 213–238 (2007)
18. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. IJCV 88, 303–338 (2010)
19. Laptev, I.: On space-time interest points. IJCV 64, 107–123 (2005)
20. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: 2005 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 65–72 (2005)
21. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)
22. Bishop, C.M.: Pattern Recognition and Machiner Learning. Springer (2006)
23. Johnson, S.: Hierarchical clustering schemes. Psychometrika 32, 241–254 (1967)
24. Ng, A., Jordan, M., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS, vol. (2), pp. 849–856.
25. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: NIPS, pp. 487–493 (1998)
26. Boureau, Y., Ponce, J., LeCun, Y.: A theoretical analysis of feature pooling in visual recognition. In: ICML (2010)
27. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011)
28. Sadanand, S., Corso, J.: Action bank: A high-level representation of activity in video. In: CVPR, pp. 1234–1241 (2012)
29. Ryoo, M.S., Aggarwal, J.K.: Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In: ICCV, pp. 1593–1600 (2009)
30. Liu, J., Kuipers, B., Savarese, S.: Recognizing human actions by attributes. In: CVPR, pp. 3337–3344 (2011)