Compact Representation of High-Dimensional Feature Vectors for Large-Scale Image Recognition and Retrieval

Yu Zhang, Jianxin Wu, *Member, IEEE*, and Jianfei Cai, *Senior Member, IEEE*

***Abstract*—In large scale visual recognition and image retrieval tasks, feature vectors such as Fisher vector (FV) or the Vector of Locally Aggregated Descriptors (VLAD) have achieved state-of-the-art results. However, the combination of large numbers of examples and high dimensional vectors necessitates dimensionality reduction, in order to reduce its storage and CPU costs to a reasonable range. In spite of the popularity of various feature compression methods, this paper shows that feature (dimension) selection is a better choice for high-dimensional FV/VLAD than feature (dimension) compression methods, e.g., product quantization. We show that strong correlation among feature dimensions in FV and VLAD may not exist, which renders feature selection a natural choice. We also show that many dimensions in FV/VLAD are noise. Throwing them away using feature selection is better than compressing them and useful dimensions altogether using feature compression methods. To choose features, we propose an efficient importance sorting algorithm considering both the supervised and the unsupervised cases, for visual recognition and image retrieval, respectively. Combining with the 1-bit quantization, feature selection has achieved both higher accuracy and less computational cost than feature compression methods such as product quantization on FV and VLAD image representations.**

***Index Terms*—feature selection, large scale, image representation**

I. Introduction

Visual recognition and image retrieval tasks now regularly deal with large-scale datasets. For example, ImageNet [1] contains tens of thousands of object types and millions of images. In the past few years, many powerful, yet very high dimensional features like the Fisher Vector (FV) [2] and VLAD [3], [4] have been proposed. They are often combined with the linear SVM classifier (for recognition) and k-nearest neighbor search (for retrieval), and have achieved state-of-the-art performance on many image classification and image retrieval tasks [5], [6]. However, when these features are applied to large scale datasets, the rendezvous of very high dimensionality and huge number of examples pose serious challenges for both feature vector storage and the subsequent classifier learning or retrieval task. For example, 310 gigabytes are required to store the Fisher Vectors for only a subset of the ImageNet images [2]. Recently, deep learning methods like convolutional neural networks (CNN) [7] become popular in large-scale image recognition. Gong et al. [8] showed that VLAD can be efficiently combined with CNN to further improve its performance, however, which may face the similar memory problem as FV [9], when it is applied on large-scale problems.

Feature[1] compression has been proposed as a remedy for this problem in both the image categorization and the image retrieval domains, which compresses high dimensional feature vectors into manageable length. While "compression cannot significantly reduce accuracy" is an obvious goal for all these methods, some also aim at reducing the computational cost (i.e., reducing classifier training time and increasing testing speed.)

We can roughly divide these methods into three categories. The first is Product Quantization (PQ) [10], [11], which is a state-of-the-art feature compression method, originally proposed for image retrieval [10], and later extended to image categorization [9], [11]. In PQ, a long feature vector is divided into short segments. Each segment is vector quantized into an index using a codebook. The codebooks can be simply acquired by the k-means clustering algorithm, or learned in whole as the Cartesian product of codebooks from all segments [12], [13]. Then, these indices and codebooks are stored, which require much fewer bytes than the original features. The second category of methods are hashing based, which transform a real-valued feature vector into a shorter binary string. In [9], the authors tested hashing kernel on ImageNet, where sparse random projections are generated to reduce feature dimensionality. In [14], an iterative method, iterative quantization (ITQ), is used to find linear projections which map original feature vectors to binary strings through rotation. In [15], a set of bilinear projections (BPBC) are learned to transform a tensor representation of the long vectors into binary codes, which further reduces the storage requirement. BPBC has been used for both image categorization and image retrieval. The third and final category uses dimension reduction techniques to transform long vectors into (real-valued) shorter ones. In [16], rotated sparse regression is proposed to compress the 100K dimensional vector for face verification. [17] learned

J. Wu is supported in part by the National Natural Science Foundation of China under Grant No. 61422203. J. Cai is supported in part by Singapore MoE AcRF Tier-1 Grant RG138/14.

Y. Zhang is with the Bioinformatics Institute, A*STAR, Singapore. Part of this work was finished when he was with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: zhangyu@bii.a-star.edu.sg.

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

J. Cai is with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: asjfcai@ntu.edu.sg.

[1]Note that a feature is a dimension in FV or VLAD throughout this paper, rather than local descriptors like SIFT.

a low-rank and sparse structure for a nonnegative matrix in the Manhattan nonnegative matrix factorization. Besides, many well known technologies can be classified into this category, e.g., PCA, nonnegative matrix factorization (NMF), and k-means.

A common theme of all these compression methods is that linear projection (i.e., dot-product between the long feature vector or part of it with a learned vector) is the key operation; and, the effectiveness of these methods is determined by the quality of the learned linear projections.

Our goal is to reduce the feature dimensionality, and hence the storage cost and computational load of features like FV or VLAD. In this paper, we argue that *to choose is better than to compress, i.e., to select a small subset of dimensions from the original vector rather than compressing it using sophisticated feature compression methods*. Similar feature selection techniques have been used in [18] and the MPEG standard for the compact representation of local descriptors. To our best knowledge, this is the first time feature selection is used for compact image representation of FV/VLAD in large-scale image recognition and retrieval.

We first show that, strong linear correlations may not hold in Fisher vectors or VLAD. Correlation coefficients between randomly chosen dimension pairs almost always reside in a small region around 0, i.e., collinearity (linear correlation between two dimensions) almost does not exist in FV or VLAD in practice.

Second, since only weak correlation exist among dimensions in FV/VLAD, we can process each dimension separately. Considering the fact that noisy dimensions may exist in FV or VLAD, we propose to choose a subset of useful feature dimensions instead to compress all of them without discrimination. We use an importance sorting algorithm, which can deal with both the supervised and the unsupervised cases. In this paper, we evaluate on two representative applications: image classification and image retrieval. For image classification, we use a supervised mutual information (MI) based importance sorting algorithm, which have achieved higher or comparable accuracy than feature compression methods in our experiments. Unlike feature compression which is mostly unsupervised, the proposed MI based feature selection method is supervised, which uses the image labels to guide feature selection. For image retrieval, we use an unsupervised entropy based importance sorting algorithm. With the selected features, we compute two distances: *symmetric* and *asymmetric* distances for different types of queries similar to [10], both of which are computed efficiently with the lookup table technique. In both feature selection methods, we use a 1-BIT quantization to further reduce a selected dimension into one bit.

Feature selection has several advantages compared with the state-of-the-art high-dimensional feature compression method PQ. First, the selected features can improve the classification/retrieval accuracy by removing many original feature dimensions which are noise. Second, even if we want to select multiple feature subsets with different sizes, the importance values for every dimension just need to be computed and sorted once. As a direct comparison, feature compression methods need to run the compression algorithm multiple times if it is required to compress the original features to different compression ratios. Third, because the number of chosen features is much fewer than that of the original features, classifier training/testing in image classification and instance search in image retrieval are now scalable and greatly accelerated. Empirical evaluations in Sec. IV confirm that feature selection has a clear edge over PQ and other compression methods in large scale visual recognition and image retrieval.

Preliminary results of feature selection for image classification have been published as a conference paper [19].

II. Related works

We briefly introduce some feature compression and selection methods for high dimensional vectors, mainly FV and VLAD.

A. Feature compression for high-dimensional feature vectors

For a set of feature vectors with D dimensions (where D is a large number), PQ [10] divides each feature vector into $G = \frac{D}{d}$ segments, where d is the segment length. For each segment, PQ learns a codebook with K words using the k-means clustering method. K is usually a power of 2, and one segment is then represented by the index to its nearest code word. PQ needs to store both the indices (i.e., compressed features) and the G codebooks. Assuming that single precision floating point numbers (4 bytes each) are used, the codebooks in total will require $4GKd = 4KD$ bytes to store, and the D dimensional feature vector are compressed from $4D$ bytes to $\frac{G \log_2 K}{8}$ bytes, with a compression ratio equal to $\frac{32D}{G \log_2 K} = \frac{32d}{\log_2 K}$.

In learning and applying a classifier, the compressed features must be decoded using the real-valued code word associated with the stored indices. Thus, PQ compressed feature has the same training and prediction complexity as the original uncompressed feature. [11] proposes an acceleration technique for SVM training on PQ compressed features. This technique (without special optimization) was shown slower than a plain SVM with optimized BLAS library [11]. Thus, we do not use it in our PQ experiments in this paper.

[20] recommends randomly rotating the original vector before applying PQ. Hashing based techniques usually have this rotation step, e.g., in spectral hashing [21], ITQ [14], BPBC [15], etc. Among these methods, BPBC is more suitable for large scale problems. It uses bilinear instead of linear projections to reduce the memory requirement for the codebooks. In BPBC, each high dimensional vector $\boldsymbol{x} \in \mathbb{R}^D$ is reshaped into a matrix (or 2nd order tensor) $\boldsymbol{x}' \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2}$, $D = d_1 d_2$. Then, two orthogonal projection matrices $R_1 \in \mathbb{R}^{d_1} \times \mathbb{R}^{c_1}$ and $R_2 \in \mathbb{R}^{d_2} \times \mathbb{R}^{c_2}$ can be learned or randomly generated. Finally, a vectorized form of $\text{sign}(R_1^T \boldsymbol{x}' R_2)$ is the compressed binary feature vector, which requires $\frac{c_1 c_2}{8}$ bytes to store.

PCA has been widely used for dimension reduction and noise removal. However, it is not suitable for high-dimensional FV/VLAD. It needs to compute a $D \times D$ matrix ($D = 262144$ in the experiment, leading to 256 gigabytes of memory), which can not fit in the RAM of most PC (256G memory).

B. Feature selection

There is a vast literature on feature selection. For example, given a decision function $\boldsymbol{w}^T\boldsymbol{x}$, if $\boldsymbol{w}$ is learned with the ℓ_1 norm regularizer, the nonzero elements in $\boldsymbol{w}$ correspond to the selected features. [22] proposes a Feature Generating Machine, which learns a binary indicator vector to show whether one dimension is useful or not. [23] considers the dependency between a dimension and labels, and the redundancy between features to select useful features based on mutual information. [24] iteratively selects a new feature based on its mutual information with labels and already selected features. More feature selection methods [25], [26], [27] are proposed for different types of specific applications.

However, most of these existing feature selection methods are too expensive to be applied on high-dimensional FV/VLAD in our large scale image classification/retrieval problems. Some methods require to compute an affinity matrix among instances, e.g., Relieff [28], Laplacian score [29], NDFS [30], RUFS [31]. Computing the affinity matrix usually involves the nearest neighbor step, which will encounter severe curse of dimensionality on very high-dimensional FV/VLAD ($D = 262144$). Some methods need to compute a $D \times D$ matrix, which cannot fit into the memory of most computers. These methods include: MTFS [32], UDFS [33], LUFS [34]. Thus, we need to use some efficient feature selection methods to solve our problem.

III. Information Theory Based Importance Sorting

In this section, we first study properties of Fisher vectors and VLAD, and show that selection may be a better choice than compression. We then propose a supervised mutual information based importance sorting feature selection method for image classification and an unsupervised entropy based method for image retrieval.

A. FV's and VLAD's property

We start by studying the correlation between any two dimensions in FV vectors. Pearson's correlation coefficient measures correlation between dimensions i and j:

$$r = \frac{\boldsymbol{x}_{:i}^T \boldsymbol{x}_{:j}}{\|\boldsymbol{x}_{:i}\| \|\boldsymbol{x}_{:j}\|}, \tag{1}$$

where $\boldsymbol{x}_{:i}$ is the vector formed by the i-th dimension of FV of all images in a dataset, and $r \in [-1\ 1]$ is Pearson's correlation coefficient, which measures the linear dependency between the two dimensions (± 1 for total correlation and 0 for no correlation).

We show the histograms of r values in Fig. 1 computed from Fisher vectors of the Scene 15 dataset [35]. There are 128 Gaussian components and 64 dimensional SIFT (after PCA) in FV and 8 spatial regions, leading to 131072 dimensions. More than 90% values in FV are non-zero. Three types of r values are computed for all pairs of dimensions in differently sampled subsets:

1) **Random**: randomly sample 1000 dimensions;
2) **Same region**: randomly choose a spatial region, then randomly sample 10% dimensions from this region;

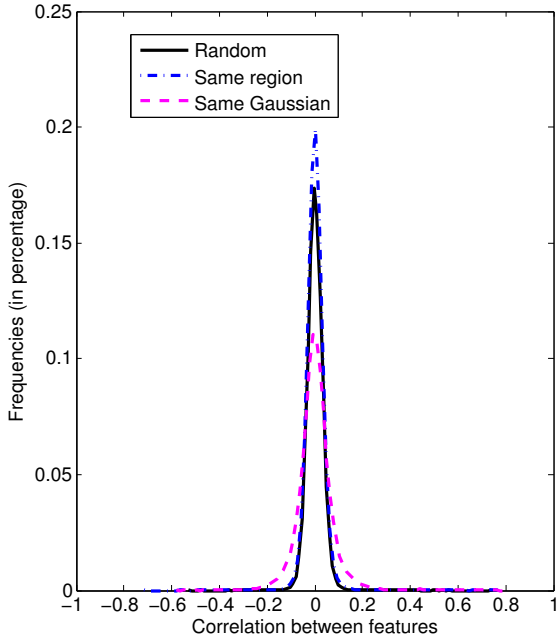


Fig. 1. Distribution of Pearson's correlation coefficients between dimensions in Fisher vectors.

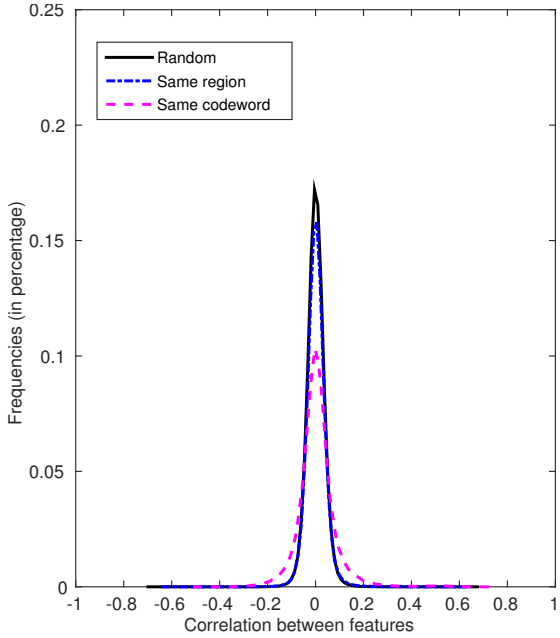


Fig. 2. Distribution of Pearson's correlation coefficients between dimensions in VLAD.

3) **Same Gaussian**: randomly choose one Gaussian component from GMM (Gaussian Mixture Model) in FV, then use all dimensions computed from it.[2]

Fig. 1 shows that in all subsets, almost all r values are very close to zero—more than 99.9% satisfy that $|r| < 0.2$. The RANDOM and SAME REGION curves are almost identical except at the zero point, suggesting that feature correlation is not affected by the spatial pyramid. Dimensions sampled

[2]A GMM is a weighted average of several Gaussian components, and every Gaussian component is a multivariate normal distribution.

from the same Gaussian component (i.e., supposedly visually similar) exhibit slightly higher correlations than the other two subsets, but the overall correlation level is still low.

We observed similar results using the VLAD representation in Fig. 2. Thus, we have good reasons to believe that strong linear correlation between two dimensions (that is, collinearity) almost does not exist in FV or VLAD.

Since FV has the low correlation between different dimensions, it advocates to process each dimension separately. We will show in the next section that *most* dimensions are noise and are not useful (if not harmful) by computing the importance of each dimension. Feature compression use linear projections to combine all these noisy features and form the new compressed features, which is sub-optimal. Thus, we propose to choose a subset of "good" features (i.e., *feature selection*) for dimension reduction, when FV or VLAD are used to represent images.

B. Feature importance sorting

There is a vast literature on feature selection. Most of these methods are too expensive to be applied on high dimensional FV/VLAD in our large scale image classification/retrieval problems.

Thus, we will use a classic mutual information/entropy based importance sorting approach. It is similar to [23] in that mutual information (MI) between a dimension and the labels is used to compute the importance of it, but dependencies among different dimensions are ignored such that we can afford the computational cost.

In supervised tasks like image classification, we use the image labels to estimate how useful a single dimension is. The label for an image in a problem with C categories is an integer in the set $\{1, \dots, C\}$. Specifically, we denote image labels as $\boldsymbol{y}$, the FV values in the i-th dimension as $\boldsymbol{x}_{:i}$, and the mutual information as $I(\boldsymbol{x}_{:i}, \boldsymbol{y})$. Based on information theory, in general, the larger the mutual information, the more useful this dimension is for predicting the label (or, for classification). The MI value is our importance score for each dimension when image labels are available (i.e., supervised importance score), and is computed as:

$$I(\boldsymbol{x}_{:i}, \boldsymbol{y}) = H(\boldsymbol{y}) + H(\boldsymbol{x}_{:i}) - H(\boldsymbol{x}_{:i}, \boldsymbol{y}), \tag{2}$$

where H is the entropy of a random variable. Since $\boldsymbol{y}$ remains unchanged for different i, we just need to compute $H(\boldsymbol{x}_{:i}) - H(\boldsymbol{x}_{:i}, \boldsymbol{y})$.

In unsupervised tasks like image retrieval, because there is no label information, we only compute the entropy $H(\boldsymbol{x}_{:i})$ as the importance of each dimension. This choice to find dimensions with maximum entropy values are widely used in the information retrieval research [36]. It is also an intuitive importance score. If a dimension has a small entropy value (e.g., all feature values in this dimension concentrate in a small range), it is unlikely that it will be useful for the retrieval purpose.

We then sort the dimensions according to the decreasing order of their importance values. Then, if we want to reduce to D' dimensions from the original D dimensions, the top D' in the sorted list can be directly selected.

When the compression ratio changes in feature compression methods, they must update codebooks, and update compressed features too. Both steps are very time consuming. In feature selection, we enjoy the simplicity when D' changes: simply choose an updated subset of dimensions and no additional computations are needed at all.

We want to add a note about the use of MI-based feature selection. This is a classic algorithm and is not invented by us. Our contribution is the novel pipeline to choose a subset of dimensions for very high dimensional image representations such as FV and VLAD, instead of employing the current feature compression methods. This pipeline reveals properties of such features, and pave grounds for using feature selection to replace feature compression. It neatly solves the serious storage and computational problems in large scale image classification or retrieval tasks, and achieves higher accuracies and lower computational costs than existing feature compression methods.

As for which specific feature selection algorithm to adopt, we can use any feature selection method that can select a high quality subset of feature dimensions, subject to the following constraints:

- from hundreds of thousands of dense features (i.e., with few zero entries),
- using millions of examples, and,
- in a reasonable amount of time (e.g., seconds or minutes).

The MI-based feature selection algorithm satisfies these constraints. It is also (possibly) the easiest-to-implement and simplest algorithm that meet all such constraints. Thus, it is adopted in our pipeline and has shown excellent empirical results, as will be presented in Sec. IV.

C. Entropy calculation and further quantization

In Eq. 2 we need to compute the entropy values. A typical method to compute the differential entropy of a real-valued random variable (e.g., $\boldsymbol{x}_{:i}$) is to estimate its probability distribution function by kernel density estimation, which is again computationally too heavy in large scale problems. Rather, we use quantized discrete variables to compute entropy and further reduce storage costs.

We use two kinds of quantization:

- **1-bit**: quantize a real number x into 2 discrete bins according to

$$x \leftarrow \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}. \tag{3}$$

- **N-bins**: find the minimum and maximum value in a dimension, and uniformly quantize all values in this dimension into N bins. That is,

$$x \leftarrow \left\lfloor \frac{x - \underline{v}}{\bar{v} - \underline{v} + \epsilon} N \right\rfloor, \tag{4}$$

where $\underline{v}$ and $\bar{v}$ are the minimum and maximum value in this dimension, respectively; $\epsilon > 0$ is a small number, so that each value can be quantized into one bin in the set $\{0, ..., N-1\}$. The inclusion of ϵ also avoids errors when $\underline{v} = \bar{v}$.

The discrete entropy is $H(x) = -\sum_j p_j \log_2(p_j)$ where p_j is the probability that x falls in the j-th bin.

A good quantization should maximize the importance score of features. We show the histogram of mutual information and entropy values of different quantization setups in Fig. 3. We use the quantized empirical distributions to compute importance/usefulness scores. Note that this method just scans sequentially on the training vectors once without additional computation, which makes it suitable for large scale datasets.

One observation common to all quantization methods is that most dimensions have small importance values. In Fig. 3a, except in 32-BINS, more than 90% dimensions have their MI values smaller than 0.1. This observation shows that most dimensions may not be useful for image classification. In Fig. 3b, which shows the usefulness value distribution in unsupervised problems, this observation also holds true for quantization using 4-BINS and 8-BINS. Thus, feature selection, beyond having the virtue in reducing storage and CPU costs, also has the potential to improve classification/retrieval accuracy by removing noisy or irrelevant dimensions.

Another important observation is that the 1-BIT quantization is better than 4-BINS and 8-BINS. Two points are worth noting. First, 1-BIT has 2 quantization bins. However, it is different from 2-BINS: in 1-BIT the quantization threshold is fixed to 0 *a priori*, rather than determined empirically as in 2-BINS. Second, in general for the same x, a bigger N in N-BINS will lead to higher entropy value. For example, if x is uniform in $[0\ 1]$, 2-, 4- and 8-BINS quantization will have discrete entropy values 1, 2 and 3, respectively. Thus, this fact (1-BIT is better than 8-BINS) proves that proper quantization is essential. We want to clarify MI against MI ratio. Consider 1-bit and 2-bits quantization methods. In these two cases, the largest possible MI values are 1 and 2, respectively. We calculate the MI of two dimensions. One uses 1-bit, with MI value being 0.6. Then, its MI ratio is 0.6/1 = 0.6 (ratio between this 0.6 MI value and the largest possible MI in the 1-bit method). The other dimension is calculated using 2-bits, with MI value being 1.1. Of course 1.1 is bigger than 0.6, but its MI ratio is 1.1/2 = 0.55, which is smaller than the 0.6 MI ratio. Thus, it is natural to see that the 0.6 MI value in 1-bit is more useful than the 1.1 MI value in 2-bits. In particular, in FV and VLAD, 0 must be a quantization threshold.

Many feature compression methods already use the 1-BIT quantization to store features compactly, e.g., BPBC [15] and FV compression [37]. An explanation of its effectiveness is missing, though. From our quantization perspective, Fig. 3a quantitatively shows that 1-BIT is good at preserving useful information. Furthermore, Fig. 4 qualitatively explains why 0 must be the threshold, which is revealed in [38], [39]. After power normalization[3], most randomly chosen dimensions follow bimodal distributions, where the two modes are separated exactly at the zero point. Thus, in order to keep the discriminative power of a feature, we must use 0 as a quantization threshold.

[3]Power normalization is a postprocessing technique that transform a feature value x to $\text{sign}(x)\sqrt{(|x|)}$. This operation improves the classification accuracy on FV features [2] and is widely adopted.

Based on the above results, we use 1-BIT to compute importance scores and choose a subset of dimensions. Finally, we further quantize the chosen dimensions using the 1-BIT quantization to further reduce storage cost.

If D' dimensions are chosen, we just need $\frac{D'}{8}$ bytes to store an image. The first bin ($x \geq 0$) is stored as a value 1 in the bit, while the second bin ($x < 0$) is stored as a value 0.

D. Classification using quantized features

In image classification, we learn linear SVM classifiers on top of the quantized features. A simple table lookup trick can accelerate linear SVM training and testing.

Efficient linear SVM learners, such as stochastic gradient descent (SGD) [40] or dual coordinate descent (DCD) [41], spend most of their training and almost all testing time in the following two types of computations:

- **Update**: $\boldsymbol{w} \leftarrow \lambda_1 \boldsymbol{w} + \lambda_2 \boldsymbol{x}$,
- **Dot-product**: $\boldsymbol{w}^T \boldsymbol{x}$,

where $\boldsymbol{w}$ and $\boldsymbol{x}$ are the classifier boundary and one example, respectively.

When we use a selected and quantized feature vector $\widehat{\boldsymbol{x}} \in \mathbb{R}^{D'/8}$, both steps can be made efficient by using a lookup table. Note that each byte within $\widehat{\boldsymbol{x}}$ corresponds to 8 chosen dimensions, and we define a table $Z \in \mathbb{R}^{256} \times \mathbb{R}^8$:

$$Z = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & +1 \\ -1 & -1 & -1 & -1 & -1 & -1 & +1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix}. \quad (5)$$

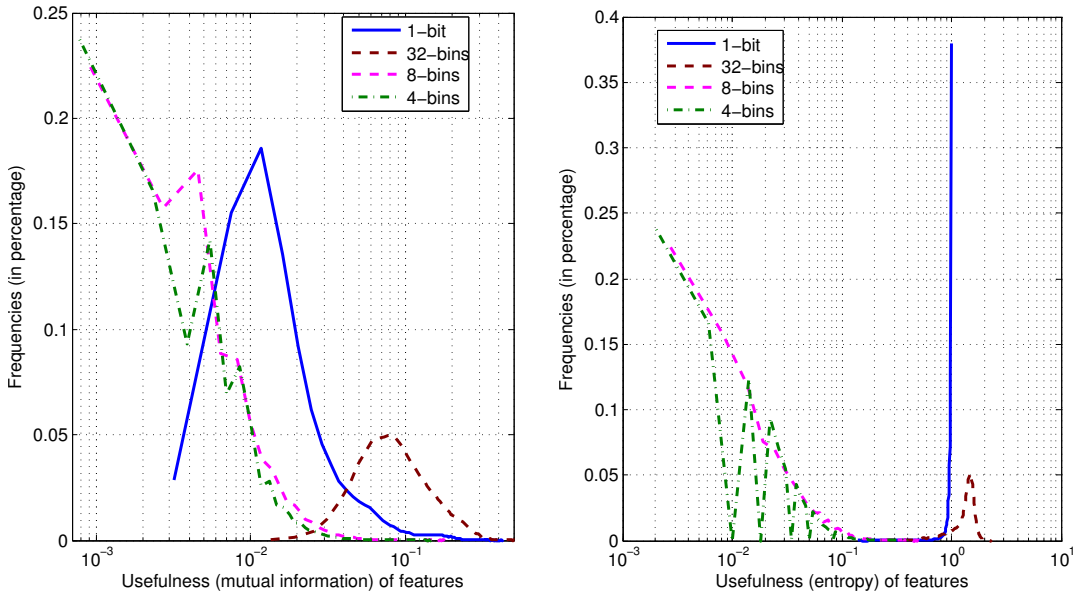
A byte i ($0 \leq i \leq 255$) corresponds to the i-th row in Z, which expands the byte into its 8 quantized dimensions.

We assume the length of $\boldsymbol{w}$ can be divided by 8, which is easy to satisfy. Then, when we perform either the update or the dot-product computation, we process 8 dimensions in a batch processing style: read one byte from $\widehat{\boldsymbol{x}}$, then find its corresponding row in Z, finally compute the summation or dot-product of two small (length 8) vectors.

E. Fast distance computation using quantized features

In image retrieval, distance computations can also be efficiently performed on the quantized features. Given a query, we search the image database for images whose feature vectors are the most similar to the query (i.e., having smallest distance values.) There are two possible cases: the query is represented by a quantized feature vector or a real value feature vector. Corresponding to them, there are symmetric distance (SD) and asymmetric distance (AD) [10], respectively. We devise two fast ways to compute these two distance measures for 1-BIT representation with the lookup table technique.

First of all, the database images are always represented using the quantized feature vector format. When a query is a quantized feature vector, we use a symmetric distance to compute its distance with the quantized feature vectors in the image database. We first precompute a lookup table



(a) (b)

Fig. 3. Comparison of dimensions' usefulness / importance values of different quantization methods on the Scene 15 dataset. This figure is best viewed in color.

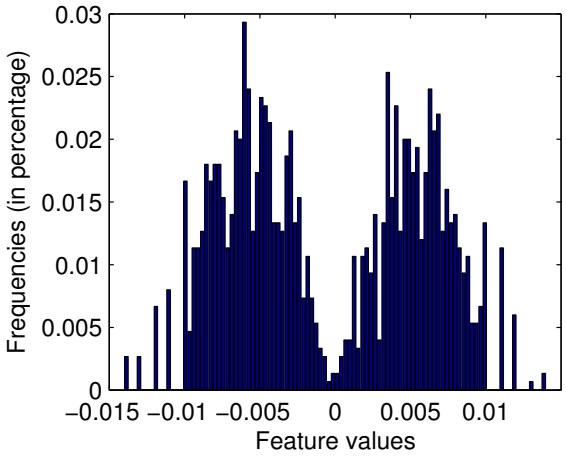


Fig. 4. Histogram of values (after power normalization) in one FV dimension in the Scene 15 dataset.

$T \in \mathbb{R}^{256} \times \mathbb{R}^{256}$ which contains pairwise distances of rows in the matrix Z (cf. Eq. 5):

$$T_{ij} = \|Z_{i:} - Z_{j:}\|^2, \tag{6}$$

where $Z_{i:}$ and $Z_{j:}$ are the i-th and j-th row of Z, respectively. Then, given two quantized feature vectors $\widehat{\boldsymbol{x}}_1$ and $\widehat{\boldsymbol{x}}_2$, their symmetric distance can be computed as:

$$d_{SD}(\widehat{\boldsymbol{x}}_1, \widehat{\boldsymbol{x}}_2) = \sqrt{\sum_{i=1}^{D'/8} T_{\widehat{\boldsymbol{x}}_{1i}, \widehat{\boldsymbol{x}}_{2i}}}\,. \tag{7}$$

When the query is a real valued vector (real values after feature selection, whose length is D'), we use an asymmetric distance to compute its distance with the quantized feature vectors in the image database. Given a selected real value vector $\boldsymbol{x}' \in \mathbb{R}^{D'}$, we also use the lookup table technique for efficient computation. We first divide it into segments of 8 dimensions. Then, for each segment, we precompute its distance with each row in Z, which are stored into a lookup table $T' \in \mathbb{R}^{256} \times \mathbb{R}^{D'/8}$. Each column of T' corresponds to a segment in $\boldsymbol{x}'$. Finally, the asymmetric distance between the selected real valued vector $\boldsymbol{x}'$ and a quantized database vector $\widehat{\boldsymbol{x}}_i$ is:

$$d_{AD}(\boldsymbol{x}', \widehat{\boldsymbol{x}}_i) = \sqrt{\sum_{j=1}^{D'/8} T'_{\widehat{\boldsymbol{x}}_{ij}, j}}\,. \tag{8}$$

The asymmetric distance can provide better retrieval accuracy than the symmetric distance. However, it needs to compute a new lookup table for every new query, so it is not as efficient as the symmetric distance, which only need to precompute one lookup table for all queries.

IV. Experiment

We evaluate the proposed feature selection method on image classification and retrieval with several large scale benchmarks. It is denoted by FS, and is mainly compared with PQ and other related compression methods. Since the datasets are large scale and time consuming to evaluate, we use PQ results from the literature whenever they are available for a specific dataset, otherwise we report PQ results from our own implementation. We use the Fisher Vector to represent all images.

In image classification, SIFT is densely sampled in every 4 pixels from each image. Following the setup in [9], only the mean and variance part in FV are used. SIFT is reduced from 128 to 64 dimensions using PCA. The number of Gaussian components is 256. We use the spatial pyramid matching structure in [42], [43] which extracts 8 spatial regions from an image. Its structure is: the whole image, three horizontal regions, and two by two split regions. The total number of

dimensions in FV is $D = 64 \times 2 \times 256 \times 8 = 262144$. We revise the dual coordinate descent algorithm [41] to learn a linear SVM classifier from our selected and quantized features or BPBC; and use the LIBLINEAR software package [44] in our PQ experiments. The following benchmark datasets are used for image classification:

- **VOC 2007** [45]. It has 20 object classes. Each image may contain more than one object. We use all the training and validation images (5000) for training and the testing images (5000) for testing.
- **ILSVRC 2010** [46]. It has 1000 classes and 1.2 million training images. We use all the provided training and testing images for training and testing, respectively.
- **SUN 397** [47]. It has 397 classes. In each class, we use 50 training images and 50 testing images, which amounts to 19850 training and 19850 testing images.
- **Scene 15** [35]. It has 15 classes. In each class, 100 images are used for training, and the rest images are used for testing.

We will compare our methods with PQ and BPBC under different compression ratios, i.e., the ratio of the memory of the original feature to that of the feature after compression/selection. In PQ, we use the segment length $d = 8$, which has the overall best performance in [9] under different compression ratios and is also used in BPBC [15]. We use k-means to generate codebooks. Then, we change the codebook size K to achieve different compression ratios in PQ. In BPBC, we reshaped FV into a 128×2048 matrix, and learn bilinear projections to achieve different compression ratios. BPBC parameters need iterative updates, for which a maximum of 10 iterations is used in our experiments.

The results are averaged on 5 random train/test splits in Scene 15 and SUN 397. In VOC 2007, we use the predefined split, but run 5 times to get different GMM models and report the average mAP (mean average precision). For ILSVRC 2010, we run one time using the given split.

In image retrieval, we extract SIFT from interesting points detected by the Hessian-affine detector [48]. 64 GMMs are used in FV and its dimension is $D = 64 \times 2 \times 64 = 8192$. We evaluate the proposed method on three image retrieval datasets:

- **Holiday** [49]. It contains 1491 images in total. 500 images are used as query and 991 are used as the dataset images for training. Mean average precision (mAP) is used to evaluate the retrieval performance. We use the independent dataset Flickr60k provided with Holiday to learn GMM for FV.
- **Oxford5k** [50]. It contains 5062 images of various buildings. 55 images from 11 distinct buildings are used as queries. mAP is used to evaluate its retrieval performance. We use the Paris6k [51] dataset to learn GMM for FV.
- **Holiday+Flickr1M** [49]. This is a large scale experiment. We use the Flickr1M dataset as the distractor together with Holiday. The retrieval performance is evaluated in terms of top R retrieved images as in [37], [3], e.g., recall@R.

We compare FS with PQ in image retrieval.

All experiments are tested on a computer with one Intel i7-3930K CPU and 32G main memory. All CPU cores are used during feature compression. In classifier learning and testing of image classification and distance computations of image retrieval, only one core is used.

TABLE I
MEAN AVERAGE PRECISION (MAP) ON VOC 2007. THE LOSS OF MAP RELATIVE TO ORIGINAL DENSE FEATURE (RATIO 1) IS ALSO COMPUTED.

Method	Compression ratio	mAP (%)	Loss (%)
FS	1	58.57 ± 0.19	0
	32	60.09 ± 0.09	-1.52
	64	60.05 ± 0.16	-1.48
	128	58.97 ± 0.23	-0.40
	256	56.82 ± 0.49	1.75
	512	52.70 ± 0.44	5.87
	1024	46.52 ± 0.40	12.05
PQ [11]	1	58.8	0
	$32(d = 6)$	58.2	0.6
	$64(d = 8)$	56.6	2.2
	$128(d = 8)$	54.0	4.8
	$256(d = 8)$	50.3	8.5
PQ [9]	1	58.3	0
	$32(d = 8)$	57.3	1.0
	$64(d = 8)$	55.9	2.4
	$64(d = 16)$	56.2	2.1

A. Image classification results

In this section, we evaluate the proposed mutual information based feature selection (FS) for image classification. We report the absolute classification performance (top 1 accuracy, top 5 accuracy, or mAP). Where space permits, we also report the loss of performance (delta between the performance obtained from uncompressed and compressed data) for easier comparisons. Finally, we compare the efficiency of feature selection or feature compression, classifier learning and testing for FS, PQ, and BPBC. Note that when compression ratio is 1, it means that the original feature values are directly used without performing feature selection or compression.

1) VOC 2007: Mean average precisions (mAP) of various methods are shown in Table I. We use the code from [42][4] to generate FV with the same length as [9], thus it is fair to compare FS's performance with the PQ result from [9].

Under the same compression ratio, FS's mAP is higher than that of PQ on VOC 2007. The uncompressed result in our experiment and two cited PQ methods are close, but the accuracy loss of FS is much less than that of PQ. For example, when the ratio is 256, FS only loses 1.75% mAP, while PQ in [11] loses 8.5%.

In FS, a compression ratio 32 means that all dimensions are kept but quantized. Similarly, ratio 128 means that a quarter dimensions are selected and quantized. Ratio 32 provides the best discriminative ability in classification, which confirms yet another time that the 1-BIT quantization is effective in keeping useful information in features.

Another important observation is that when the compression ratio is smaller than 128, FS's mAP is higher than that of the uncompressed one. For example, ratio 64 (half dimensions used) has almost the same mAP as ratio 32 (all dimensions

[4]The FV code is available at http://www.robots.ox.ac.uk/~vgg/software/enceval_toolkit/, version 1.1.

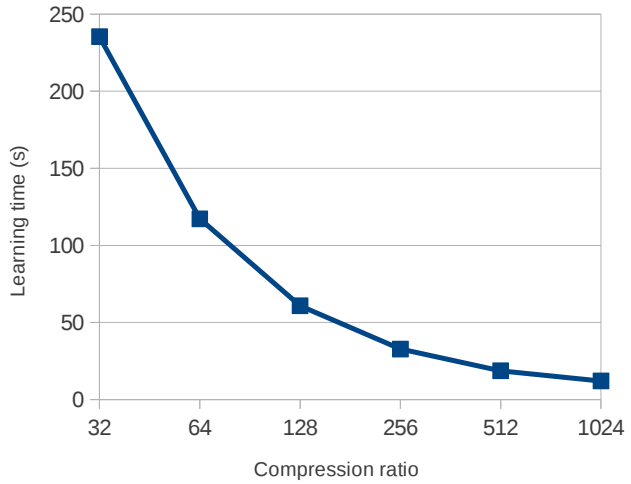


Fig. 5. Classifier learning time using the proposed (FS) compressed features on VOC 2007 under different compression ratios.

TABLE II
TOP-5 ACCURACY ON THE ILSVRC 2010 DATASET.

Method	Compression ratio	Accuracy (%)
FS	64	61.06
	128	56.64
	256	50.15
PQ [9]	$32(d = 8)$	56.2
	$64(d = 8)$	54.2
	$64(d = 16)$	54.9

used). This observation corroborates that removing (a large portion of) noisy features will not hurt classification.

In contrast, PQ's accuracy decreases quickly and monotonically when the compression ratio increases. The classification results of FS include those higher compression ratios, up to 1024. When the compression ratio is 256, FS is comparable to that of PQ with compression ratio 32. Even with compression ratio 1024, FS's mAP (46.52%) is still acceptable—remember that only 1024 bytes are needed to store the FV for an image at this compression level!

FS's classifier training time on VOC 2007 is shown in Fig. IV-A1. When the compression ratio changes from 32 to 1024 (doubling each time), the training time approximately halves each time. In other words, training time is roughly linearly proportional to storage size.

2) ILSVRC 2010: We report top-5 accuracy on the ILSVRC 2010 dataset in Table II. Limited by our computer's memory capacity, we need to start from compression ratio 64 in FS. As shown in Table II, FS's result is better than PQ's [9] with the same FV setup and the same compression ratio. FS with compression ratio 128 has similar result as PQ at ratio 32.

In terms of absolute accuracy rates reported in literature, [40] reported that PQ with a well-tailored SGD classifier achieves 66.5% top-5 accuracy. When combining more visual descriptors like color descriptors [9], [2] and LBP [52], higher accuracy can be achieved on this dataset. We conjecture that the proposed feature selection framework can also achieve better results than PQ in these richer representations.

3) SUN 397: We show the accuracy of FS and PQ on the SUN 397 dataset in Table III. Limited by our main memory

TABLE III
TOP-1 ACCURACY ON THE SUN 397 DATASET.

Method	Compression ratio	Accuracy (%)
dense FV [2]	1	43.3
multiple features [47]	1	38.0
spatial HOG [53]	1	26.8
FS	32	41.88±0.31
	64	42.05±0.36
	128	40.42±0.40
	256	37.36±0.34
PQ	32	42.72±0.45
	64	41.74±0.38
	128	40.13±0.33
	256	37.84±0.33

size, we do not evaluate accuracy of the uncompressed dataset. Comparing with uncompressed FV [2], FS is inferior yet close to its accuracy when the compression is small, e.g., 32 and 64. Note that when color descriptors are combined with SIFT, higher absolute accuracy is achieved [2] on this dataset.

Because we did not find any PQ compression results on this dataset, we implemented and evaluated our own PQ feature compression, and learned classifiers using a DCD linear SVM classifier. Comparing with PQ, FS is 0.8% inferior to PQ when the compression ratio is 32, but better than or comparable with it when the compression ratio gets larger. However, as we will show in Sec. IV-A4, the required time in feature compression/selection, classifier learning and testing of FS is much less than that of PQ. Thus, overall FS is more efficient and effective than PQ.

One final note for Table III is that FS exhibits non-monotone accuracy again: ratio 64 accuracy is slightly higher than ratio 32.

4) Scene 15: On the Scene 15 dataset, since it is small scale, besides comparing the accuracy of different methods, we also perform additional experiments to compare the efficiency of the proposed FS method with PQ and BPBC, in terms of feature compression/selection time, the classifier learning time, and the testing time.

The minimum compression ratio that FS and BPBC can reach is 32, and the maximum compression ratio of PQ can reach is 256 when group size $d = 8$. So, we restrict the compression ratio to the set $\{32, 64, 128, 256\}$. The class average accuracy is reported in Fig. 6. Overall all three methods have similar accuracies. PQ has an edge at ratio 32, while FS wins at all other ratios.

We further evaluate the feature generation time of these three methods, including extraction of visual descriptors (SIFT), FV generation, and feature compression/selection. Note that the first two parts of time is the same for all the methods. The feature selection time in FS is negligible because it only involves keeping a small subset of values from the FV vector, while mutual information based selection of the subset is also very efficient. In PQ, when the compression ratio is small, e.g., 32, feature compression (including the codebook generation time) takes about 30% of the total feature generation time in our experiments. When the compression ratio gets larger, PQ's feature compression time decreases significantly till less than 5% of the whole feature generation

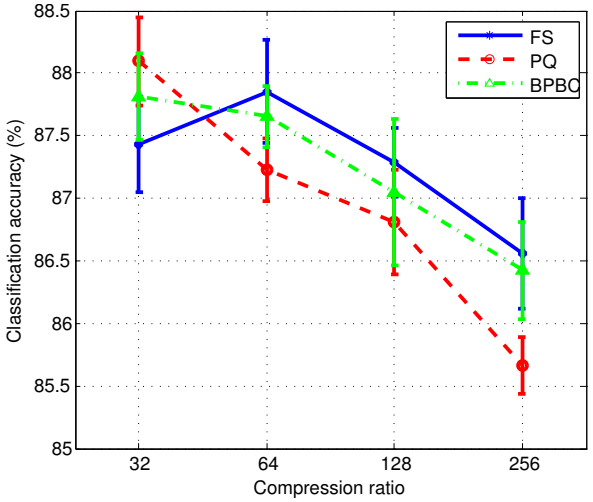


Fig. 6. Classification accuracy of FV on Scene 15 using three methods: FS (proposed), PQ, and BPBC.

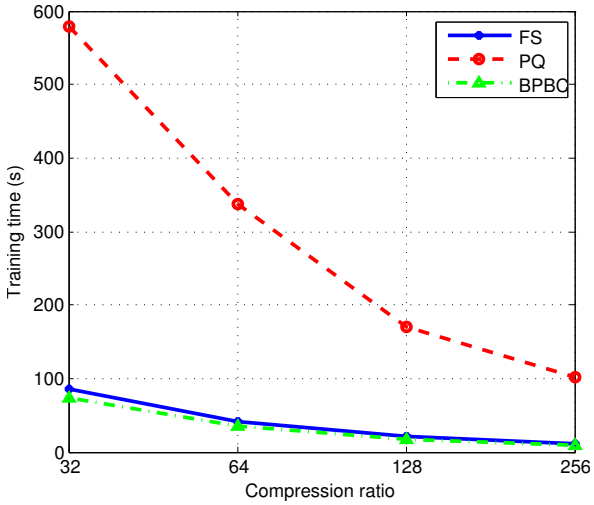


Fig. 7. Classifier training time of FV on Scene 15 using three methods: FS (proposed), PQ, and BPBC.

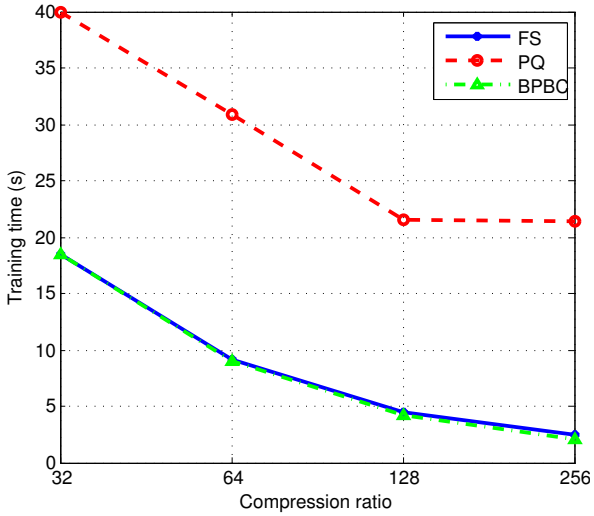


Fig. 8. Classifier testing time of FV on Scene 15 using three methods: FS (proposed), PQ, and BPBC.

time. BPBC costs most time in the feature compression step. At least 50% of the entire feature generation time is spent on feature compression, which includes feature vector reshaping, learning of projection matrices, and feature quantization. Another issue is that BPBC is not suitable for learning two high dimensional matrices. In our experiments, the second dimension of the reshaped matrix is 2048, which makes the learning process very lengthy. Our evaluations find that the most efficient dimension range of the reshaped matrix is on the order of hundreds. That is, BPBC highly relies on the intrinsic structure of a feature vector, which is not as flexible as FS or PQ in dimensionality reduction for high dimensional vectors.

The classifier training and testing time of these three methods are shown in Fig. 7 and Fig. 8, respectively. The training and testing times of FS and BPBC are similar to each other at all compression levels, and both are much less than that of PQ. This is because FS and BPBC only use the compressed/selected dimensions, while PQ has to decode the short compressed vector into a vector as long as the original features on-the-fly. A similar pattern is observed in Fig. 8 for testing time.

Finally, we compare our feature selection method (MI) with various feature selection methods on FV in Fig. 9. We first compare with N-bins methods. We find using more bins does not improve the classification accuracy. This may attribute to the limited number of instances. Fisher score, RFS, and mRMR achieve close classification accuracy to ours. However, these methods cost more time in computing the feature scores than ours, and they are 105s (Fisher score), 300s (RFS), 4 days (mRMR), and 20s (MI), respectively. Random projection, LASSO, and intra encoding [18] all lead to worse accuracy than ours.

5) Feature selection using FS on VLAD: We evaluate FS on image classification using VLAD, which is also a common high-dimensional representation. We evaluate it on two datasets: VOC 2007 and Scene 15, for which we report mAP and average class accuracy, respectively. 256 clusters are used to learn the visual codebook in VLAD. It is shown that FS is also applicable on VLAD in Table IV. On both datasets, classification results achieve the peak at compression ratio 64, which use half of VLAD features. Overall, VLAD have inferior results than FV.

B. Image retrieval results

In this section, we evaluate entropy based feature selection for image retrieval tasks. We implemented PQ code by ourselves and compared it with FS. For PQ, we fix the segment length to 8, the same as that in image classification. The number of GMM in FV generation is set to be 64, the same as [55]. Both FS and PQ are evaluated using two distance measures: symmetric (SD) and asymmetric (AD) distances.

The results of FS and PQ under different compression ratios are shown in Table V for the Holiday and Oxford5k datasets. The mAP of FS is better than PQ at the same compression ratio, no matter which of the two distance measures are used.

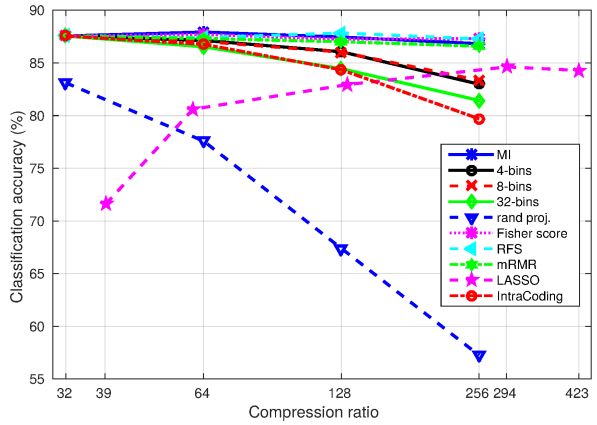


Fig. 9. Compare with different feature selection methods on Scene 15 using FV. These methods are: MI (the proposed), N-bins (N=4,8,32), random projection, Fisher score, RFS [54], mRMR [23], LASSO, Intra Coding [18]. This figure needs to be viewed in color.

TABLE IV
CLASSIFICATION ACCURACY (%) OF VLAD USING FS AT DIFFERENT COMPRESSION RATIOS.

Ratio	VOC 2007 (mAP)	Scene 15 (acc.)
32	56.76±0.24	86.36±0.33
64	56.87±0.15	86.86±0.30
128	54.49±0.29	86.58±0.09
256	49.96±0.36	85.67±0.49

This fact clearly shows that the selected features are better than PQ compressed ones.

In both methods, mAP of the asymmetric distance is better than that of the symmetric distance. However, at the same compression level, the loss of mAP in PQ is greater than that in FS. This fact also shows that the information loss in compressed features using PQ is greater than that using FS.

In FS, the small subset of selected features can achieve higher accuracy than the original feature vectors. As shown in Table V, selected features after 1-BIT quantization can provide higher mAP than that of the original features when we retain 75% feature dimensions (ratio 43). In contrast, mAP of PQ almost decreases monotonically, which is the same as that in image classification.

The selected features (plus 1-bit quantization) have better retrieval results than the uncompressed feature vectors in [55] when they use the same amount of bytes per image. PCA is used to reduce the length of FV in [55], which is the same strategy (linear projection) as PQ.

Large scale image retrieval is evaluated on the Holiday+Flickr1M. The recall@R with different R is shown in Table VI. For PQ, we learn the PQ codebooks using 40% FVs of database images (about 400000), because all the dense FVs cannot fit into the memory. PQ has much worse accuracy than FS on this dataset. Also, PQ's results do not decrease monotonically as previous experiments when the compression ratio increases. We conjecture it is due to the bad quality of the learned PQ codebooks, which deteriorates the quality of compressed PQ codes. In the binarized FV [37], only the mean component in FV is used. Its recall is lower than that of binarized FV (generated with the same number of GMMs) using both mean and variance components (FS: SD, ratio 32). Its recall is also lower than that of the selected features with the same memory (FS: SD, ratio 64).

TABLE V
MAP (%) ON HOLIDAY AND OXFORD5K USING FV (GMM 64). ENTROPY BASED SELECTION (FS) IS COMPARED WITH PQ AT DIFFERENT COMPRESSION RATIOS (RATIO) WITH TWO DISTANCE MEASURES: SYMMETRIC DISTANCE (SD) AND ASYMMETRIC DISTANCE (AD). THE MEMORY (BYTES) FOR A COMPRESSED FEATURE VECTOR IS REPORTED.

Methods		Ratio	Bytes	Holiday	Oxford5K
dense		1	32768	64.21	50.80
FS	SD	32	1024	62.85	**50.30**
		≈43	768	**65.52**	47.12
		64	512	64.90	42.58
		128	256	58.81	36.49
		256	128	54.00	28.42
	AD	32	1024	67.06	**51.59**
		≈43	768	**68.86**	49.10
		64	512	66.90	45.41
		128	256	63.75	39.75
		256	128	57.98	34.20
PQ	SD	32	1024	62.38	44.59
		64	512	36.86	39.12
		128	256	44.86	33.47
		256	128	43.57	29.11
	AD	32	1024	66.42	47.05
		64	512	49.47	44.27
		128	256	51.04	38.74
		256	128	48.86	33.57
Triangulation embedding [55]			1024	65.7	47.2
			512	61.5	40.0

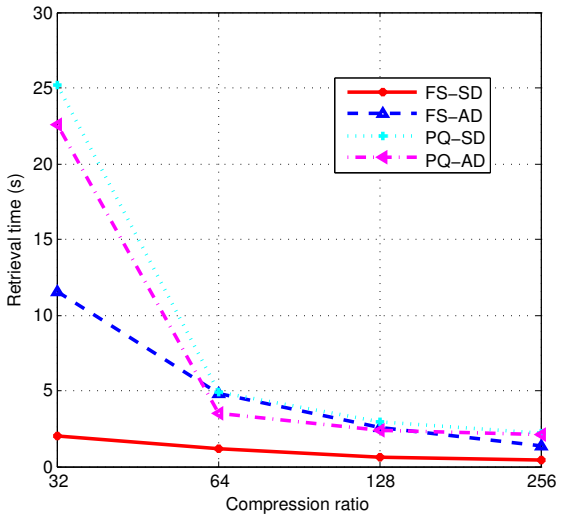
TABLE VI
RECALL@R (R=10, 100, AND 1000) ON HOLIDAY+FLICKR1M USING FV (GMM 64). ENTROPY BASED SELECTION (FS) IS COMPARED WITH PQ AT DIFFERENT COMPRESSION RATIOS (RATIO) WITH TWO DISTANCE MEASURES: SYMMETRIC DISTANCE (SD) AND ASYMMETRIC DISTANCE (AD). THE MEMORY (BYTES) FOR A COMPRESSED FEATURE VECTOR IS REPORTED.

Methods		Ratio	Bytes	R10	R100	R1000
FS	SD	32	1024	**51.70**	59.02	65.97
		≈43	768	50.13	**59.21**	**68.59**
		64	512	48.56	57.42	67.77
		128	256	42.83	50.71	62.53
		256	128	33.41	42.36	54.86
	AD	32	1024	**54.00**	**63.16**	**71.89**
		≈43	768	52.54	62.17	71.11
		64	512	50.24	59.33	70.14
		128	256	45.64	55.00	66.15
		256	128	39.20	48.66	59.54
PQ	SD	32	1024	6.26	7.85	14.22
		64	512	4.35	5.56	8.74
		128	256	26.57	33.33	42.80
		256	128	30.28	40.43	52.68
	AD	32	1024	14.87	18.45	26.32
		64	512	18.23	22.32	30.74
		128	256	36.92	45.41	55.03
		256	128	35.66	45.84	58.21
Binarized FV [37]			512	43.00	53.00	65.00

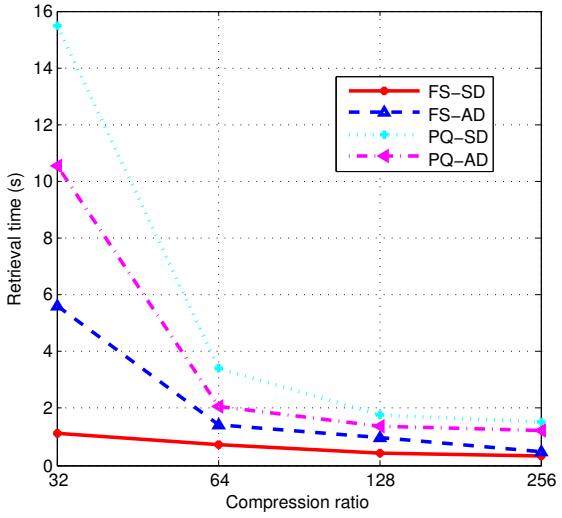
The retrieval time of FS and PQ using two distance measures is shown in Fig. 10. In most cases, FS costs less time than PQ. The main reason is the effective dimension of feature vector after selection is fewer than PQ.

C. Discussions

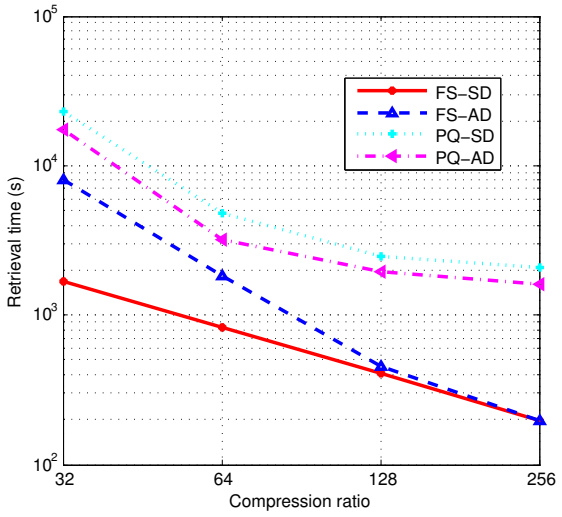
After presenting experimental results and related observations, we now provide some more subjective comments



(a) Retrieval time (seconds) on Holiday



(b) Retrieval time (seconds) on Oxford5k



(c) Retrieval time (seconds) on Holiday+Flickr1M

Fig. 10. Retrieval time (seconds) at different compression ratios. The entropy based feature selection (FS) is compared with PQ using two distance measures.

to summarize the findings of this paper, and to discuss the limitations of the proposed methods.

The major argument of this paper is that feature selection is better than compression in finding representations for high-dimensional feature vectors like FV, in both efficiency or accuracy. Particularly, we find that:

- Strong collinearity almost does not exist in Fisher vectors (cf. Fig. 1). Based on this observation, there is no need to compress features through linear transformations. Instead, selection is a better way.
- Feature selection computes the importance of features once. With the importance value, it can reach any compression ratio for feature vectors without incurring expensive reprocessing time.
- Selected features can be more discriminative by discarding redundant / noisy features in the original high dimensional feature vectors.
- Selected feature vectors have much shorter effective length than original feature vectors to reduce image classification / retrieval time.
- The 1-BIT quantization is a suitable quantization method for feature values in Fisher vector and VLAD, which exhibit bimodal distributions.

Based on the above findings, we provide two strategies to compute feature importance for the two typical applications, respectively:

- In image classification, mutual information based feature selection is a supervised method. It is more accurate than the existing feature compression methods, most of which are unsupervised.
- In image retrieval, entropy based feature selection can also provide better accuracy and efficiency than the existing compression methods.

In our experience, there is one major concern: the 1-BIT quantization is only suitable for dense features (ratio of nonzero entries $> 90\%$) that have bimodal distributions. When too many zeros exist in feature vectors, the 1-BIT quantization will not be able to characterize such a distribution accurately. Instead, a different quantization should be used which can take into account the zero value.

V. CONCLUSION

In this paper, we propose that in order to reduce the dimensionality of powerful yet very high dimensional features (such as Fisher vector or VLAD), feature selection (i.e., choosing a subset of dimensions from the original feature vector) is a better approach than existing feature compression methods (which perform linear projections on the original feature to extract new lower dimensional ones).

We first empirically show that collinearity almost does not exist in FV or VLAD, which advocates to process each dimension separately. Considering noisy dimensions may exist in FV or VLAD, selecting useful dimensions is better than compressing all dimensions (including useful and noisy dimensions) using exiting feature compression methods. We propose to use mutual information / entropy to choose a subset of dimensions for dimensionality reduction.

We use the 1-BIT quantization to compute the density of feature values in FV and VLAD. We show that when quantizing dimensions, the zero value must be a threshold because it aligns well with bimodal distributions. Later, we also quantize the chosen dimensions into 1-BIT for compact storage. During the learning stage, for image classification, we design a linear SVM classifier to learn directly on the quantized values. For image retrieval, we devise fast ways to compute two distance measures for 1-BIT representation. Experiments show that the proposed feature selection and quantization method achieves better accuracy and efficiency than feature compression methods.

There are a few issues worth further investigations. First, it is meaningful to evaluate our method on more high dimensional feature vectors, which can be extracted from multiple descriptors like [9], [2], [52]. Second, it is more valuable to develop more effective feature selection methods for higher accuracy. In addition, more efficient classifier learning / distance computing techniques can be investigated for the compressed or quantized features.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. 1

[2] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the Fisher Vector: Theory and practice," *International Journal of Computer Vision*, vol. 105, no. 3, pp. 222–245, 2013. 1, 5, 8, 12

[3] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2011. 1, 7

[4] R. Arandjelovic and A. Zisserman, "All about VLAD," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585. 1

[5] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2015. 1

[6] D. Tao, X. Tang, X. Li, and X. Wu, "Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1088–1099, 2006. 1

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105. 1

[8] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activations features," in *Proc. European Conf. Computer Vision*, 2014, pp. 392–407. 1

[9] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2011, pp. 1665–1672. 1, 6, 7, 8, 12

[10] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011. 1, 2, 5

[11] A. Vedaldi and A. Zisserman, "Sparse kernel approximations for efficient classification and detection," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 2320–2327. 1, 2, 7

[12] M. Norouzi and D. J. Fleet, "Cartesian k-means," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 3017–3024. 1

[13] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 2946–2953. 1

[14] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2011, pp. 817–824. 1, 2

[15] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 484–491. 1, 2, 5, 7

[16] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2013, pp. 3025–3032. 1

[17] T. Liu and D. Tao, "On the performance of manhattan nonnegative matrix factorization," in *IEEE Trans. on Neural Networks and Learning Systems*, 2015. 1

[18] P. Monteiro and J. Ascenso, "Coding mode decision algorithm for binary descriptor coding," in *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO)*, 2014, pp. 541–545. 2, 9, 10

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

[20] H. Jégou, M. Douze, C. Schmid, and P. Perez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311. 2

[21] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Advances in Neural Information Processing Systems*, 2008, pp. 1753–1760. 2

[22] M. Tan, L. Wang, and I. W. Tsang, "Learning sparse SVM for feature selection on very high dimensional datasets," in *Proc. Int'l Conf. on Machine Learning*, 2010, pp. 1047–1054. 3

[23] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27(8), pp. 1226–1238, 2005. 3, 4, 10

[24] F. Fleuret, "Fast binary feature selection with conditional mutual information," *Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004. 3

[25] K. Huang and S. Aviyente, "Wavelet feature selection for image classification," *IEEE Trans. on Image Processing*, vol. 17, pp. 1709–1720, 2008. 3

[26] J. Y. Choi, Y. M. Ro, and K. N. Plataniotis, "Boosting color feature selection for color face recognition," *IEEE Trans. on Image Processing*, vol. 20, pp. 1425–1434, 2011. 3

[27] Z. Sun, L. Wang, and T. Tan, "Ordinal feature selection for iris and palmprint recognition," *IEEE Trans. on Image Processing*, vol. 23, pp. 3922–3934, 2014. 3

[28] M. Robnik-Sikonja and I. Kononenko, "Theoretical and Empirical Analysis of ReliefF and RReliefF," *Machine Learning*, vol. 53, no. 1, pp. 23–69, 2003. 3

[29] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Proc. Advances in Neural Information Processing Systems*, 2005. 3

[30] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using non-negative spectral analysis," in *Proc. AAAI Conference on Artificial Intelligence*, 2012. 3

[31] M. Qian and C. Zhai, "Robust unsupervised feature selection," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2013. 3

[32] G. Obozinski, B. Taskar, and M. Jordan, "Multi-task feature selection," in *Proc. Advances in Neural Information Processing Systems*, 2006. 3

[33] Y. Yang, H. T. Shen, Z. Ma, Z. Huang, and X. Zhou, "l2,1-norm regularized discriminative feature selection for unsupervised learning," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2011. 3

[34] J. Tang and H. Liu, "An unsupervised feature selection framework for social media data," *IEEE Transactions on Knowledge and Data Engineering*, 2012. 3

[35] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178. 3, 7

[36] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008. 4

[37] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 3384–3391. 5, 7, 10

[38] J. Wu, Y. Zhang, and W. Lin, "Towards good practices for action video encoding," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 2577–2584. 5

[39] Y. Zhang, X.-S. Wei, J. Wu, J. Cai, J. Lu, V.-A. Nguyen, and M. N. Do, "Weakly supervised fine-grained categorization with part-based image

representation," *IEEE Trans. on Image Processing*, vol. 25, pp. 1713–1725, 2016. 5

[40] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2012, pp. 3482–3489. 5, 8

[41] G.-X. Yuan, C.-H. Ho, and C.-J. Lin, "Recent advances of large-scale linear classification," *Proceedings of the IEEE*, vol. 100, pp. 2584–2603, 2012. 5, 7

[42] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *Proc. British Machine Vision Conference*, 2010, pp. 76.1–76.12. 6, 7

[43] Y. Zhang, J. Wu, J. Cai, and W. Lin, "Flexible image similarity computation using hyper-spatial matching," *IEEE Trans. on Image Processing*, vol. 23, pp. 4112–4125, 2014. 6

[44] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, Aug. 2008. 7

[45] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL VOC 2007 results. http://pascallin.ecs.soton.ac.uk/challenges/voc/voc2007/." 7

[46] A. Berg, J. Deng, and F.-F. Li, "ILSVRC 2010. http://www.imagenet.org/challenges/lsvrc/2010/index." 7

[47] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2010, pp. 3485–3492. 7, 8

[48] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60(1), pp. 63–86, 2004. 7

[49] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale iamge search," *International Journal of Computer Vision*, vol. 87, pp. 316–336, 2010. 7

[50] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2007, pp. 1703–1710. 7

[51] J. Philbin, O. Chum, M. Isard, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2008, pp. 1–8. 7

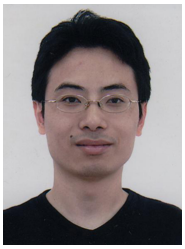
[52] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, and K. Yu, "Large-scale image classification: Fast feature extraction and SVM training," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2011, pp. 1689–1696. 8, 12

[53] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *Proc. IEEE Int'l Conf. on Computer Vision*, 2011, pp. 2072–2079. 8

[54] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2,1-norms minimization," in *Proc. Advances in Neural Information Processing Systems*, 2010, pp. 1813–1821. 10

[55] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *Proc. IEEE Int'l Conf. on Computer Vision and Pattern Recognition*, 2014, pp. 3482–3489. 9, 10

Yu Zhang received his BS and MS degrees in telecommunications engineering from Xidian University, China, and his PhD degree in computer engineering from Nanyang Technological University, Singapore. He is currently a postdoctoral fellow in the Bioinformatics Institute, A*STAR, Singapore. His research interest is computer vision.



Jianxin Wu (M'09) received his BS and MS degrees in computer science from Nanjing University, and his PhD degree in computer science from the Georgia Institute of Technology. He is currently a professor in the Department of Computer Science and Technology at Nanjing University, China, and is associated with the National Key Laboratory for Novel Software Technology, China. He was an assistant professor in the Nanyang Technological University, Singapore, and has served as an area chair for ICCV 2015 and senior PC member for AAAI 2016. His research interests are computer vision and machine learning. He is a member of the IEEE.

Jianfei Cai (S'98-M'02-SM'07) received his PhD degree from the University of Missouri-Columbia. He is currently an Associate Professor and has served as the Head of Visual & Interactive Computing Division and the Head of Computer Communication Division at the School of Computer Engineering, Nanyang Technological University, Singapore. His major research interests include computer vision, visual computing and multimedia networking. He has published more than 170 technical papers in international journals and conferences. He has been actively participating in program committees of various conferences. He has served as the leading Technical Program Chair for IEEE International Conference on Multimedia & Expo (ICME) 2012 and the leading General Chair for Pacific-rim Conference on Multimedia (PCM) 2012. Since 2013, he has been serving as an Associate Editor for IEEE Trans on Image Processing (T-IP). He has also served as an Associate Editor for IEEE Trans on Circuits and Systems for Video Technology (T-CSVT) from 2006 to 2013.