Compact Representation for Image Classification: To Choose or to Compress?

Yu Zhang[1] yzhang4@e.ntu.edu.sg
Jianxin Wu[2,*] wujx2001@nju.edu.cn
Jianfei Cai[1] asjfcai@ntu.edu.sg

[1]School of Computer Engineering, Nanyang Technological University, Singapore
[2]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Abstract

In large scale image classification, features such as Fisher vector or VLAD have achieved state-of-the-art results. However, the combination of large number 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 argues that feature selection is a better choice than feature compression. We show that strong multicollinearity among feature dimensions may not exist, which undermines feature compression's effectiveness and renders feature selection a natural choice. We also show that many dimensions are noise and throwing them away is helpful for classification. We propose a supervised mutual information (MI) based importance sorting algorithm to choose features. Combining with 1-bit quantization, MI feature selection has achieved both higher accuracy and less computational cost than feature compression methods such as product quantization and BPBC.

1. Introduction

Image classification now regularly deals with large-scale datasets. For example, ImageNet [5] 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) [20, 23], VLAD [14, 1], and super vector [30] have been proposed. They are often combined with the linear SVM classifier, and have achieved state-of-the-art performance on many image classification and image retrieval tasks [21, 22]. However, when these features are applied to large scale datasets, the rendezvous of very high dimensionality and huge number of examples poses serious challenges for both feature vector storage and subsequent classifier learning. For example, 310 gigabytes are required to store the Fisher Vectors for a subset of ImageNet images [23].

Feature compression has been proposed as a remedy for this problem, which compresses high dimensional feature vectors into manageable length. While "compression cannot significantly reduce classification 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 divide these methods into three categories. The first is Product Quantization (PQ) [12, 26], which is a state-of-the-art feature compression method. 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 k-means clustering, or learned in whole as the Cartesian product of codebooks from all segments [17, 9]. 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 [22], the authors tested hashing kernel on ImageNet, where sparse random projections are generated to reduce feature dimensionality. In [11], an iterative method, ITQ, is used to find linear projections which map original feature vectors to binary strings through rotation. In [10], 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. The third and final category uses dimension reduction techniques to transform long vectors into (real-valued) shorter ones. In [24], a computationally expensive Partial Least Squares (PLS) analysis is used to reduce the feature length for human detection. In [4], rotated sparse regression is proposed to compress the 100K dimensional vector for face verification.

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; and, the effectiveness of these methods is determined by the quality of the learned linear projections.

*J. Wu is the corresponding author. This research is partially supported by MoE AcRF Tier-1 Grant RG30/11.

Our goal is to reduce the feature dimensionality, and hence the storage cost and computational load. 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*.

Linear (or bilinear) projections generate new features (either real-valued or binary) from the original high dimensional one. One implicit but obvious assumption for all compression methods is that different dimensions in the original vector are not independent and strong linear relations exist among multiple dimensions. Otherwise, linear projections cannot extract informative new features.

We first show that surprisingly, strong linear correlations may not hold in high dimensional vectors. Correlation coefficients between randomly chosen FV dimension pairs almost always reside in a small region around 0, *i.e*., collinearity (linear dependency between two dimensions) almost does not exist in FV in practice. Thus, it is reasonable to conjecture that multicollinearity (linear dependency among more dimensions) does not exist either.

Second, since the probable missing of multicollinearity undermines existing feature compression methods, we propose a simple alternative which chooses a subset of feature dimensions instead of compress them. We use a mutual information (MI) based importance sorting criterion to choose the subset, which has achieved higher or comparable image classification accuracy than feature compression methods in our experiments. Unlike feature compression which is mostly unsupervised, the proposed feature selection method is supervised (using the image labels to guide feature selection), which may partially explain feature selection's effectiveness. Another advantage of the proposed MI method is that even if we require several subsets with different sizes (or different dimensionality compression ratios), the importance values for every dimension just need to be computed and sorted once. We use a 1-BIT quantization to further reduce a selected dimension into one bit.

Because the number of chosen features is much fewer than that of original features, classifier training and testing are now scalable and greatly accelerated, meanwhile still achieve competitive classification accuracy to state-of-the-art compression methods, *e.g*., PQ. Empirical evaluations in Sec. 4 confirm that MI feature selection has a clear edge over PQ and other compression methods in large scale image classification.

2. Brief introduction of feature compression

We first briefly introduce details of some feature compression methods for high dimensional vectors.

For a set of high dimensional feature vectors with D dimensions, PQ [12] 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 single precision floating numbers (4 bytes) are used, the codebooks in total 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 $\frac{32D}{G \log_2 K} = \frac{32d}{\log_2 K}$.

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. [26] 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 [26]. Thus, we do not use it in our PQ experiments.

[13] recommends randomly rotating the original vector before applying PQ. Hashing based techniques usually have this rotation step, *e.g*., in spectral hashing [27], ITQ [11], BPBC [10], 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 $\text{sgn}(R_1^T \boldsymbol{x}' R_2)$ is the compressed binary features, which requires $\frac{c_1 c_2}{8}$ bytes.

3. Feature selection by mutual information based importance sorting

In this section, we first study properties of high dimensional features and show that selection may be a better choice than compression. We then propose a mutual information based importance sorting feature selection method.

3.1. Feature selection vs. (bi-)linear projection

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 $r \in [-1 \ 1]$ 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 [15]. There are

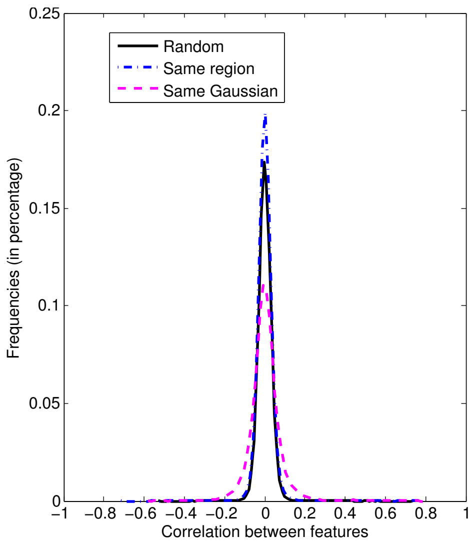

Figure 1. Distribution of Pearson's correlation coefficients between dimensions in Fisher vectors.

128 Gaussian components and 64 dimensional SIFT (after PCA) in FV and 8 spatial regions, leading to 131072 dimensions. Three types of r values are computed for all pairs of dimensions in differently sampled subsets:

- **Random**: randomly sample 1000 dimensions;
- **Same region**: randomly choose a spatial region, then randomly sample 10% dimensions from this region;
- **Same Gaussian**: randomly choose a Gaussian component in FV, then use all dimensions corresponding to it.

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 in the zero point, meaning that feature correlation is not affected by spatial pyramid. Dimensions sampled from the same Gaussian component (*i.e.*, supposedly visually similar) exhibits slightly higher correlations, but the correlation level is still low.

We observed similar results using the VLAD representation [14] and in all datasets we evaluated in Sec. 4. Thus, linear dependency between two dimensions (that is, collinearity) almost does not exist in high dimension visual representations. Although statistical tests for multicollinearity (strong linear relationship among more than 2 features) are too expensive to perform for FV or VLAD, given the missing of collinearity, we have reasons to conjecture that the chance of multicollinearity is also small.

An argument now naturally follows: if we have belief in the missing of multicollinearity, feature compression does not seem to be the best choice for reducing feature dimensionality, because linear and bilinear projections both hinge on strong multicollinearity among dimensions, especially among those features that are contiguous in FV or VLAD. Note that the SAME REGION and SAME GAUSSIAN subset in Fig. 1 both contain such contiguous dimensions.

We will also show in the next section that *most* dimensions are noise and are not useful (if not harmful) for classification. Feature compression use linear projections to include all these noisy features into the new compressed features, which is also sub-optimal. Thus, we propose to choose a subset of "good" features (*i.e.*, feature selection) for dimension reduction.

3.2. MI based importance sorting

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. [25] proposes a Feature Generating Machine, which learns a binary indicator vector to show which dimension is useful or not. [18] considers the dependency between a dimension and labels, and the redundancy between features to select useful features based on mutual information. [7] iteratively selects a new feature based on its mutual information with labels and already selected features. However, these methods are all too expensive to be applied in our large scale image classification problems. Thus, we will use a classic mutual information based importance sorting approach. It is similar to [18] 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.

This is a supervised method—we use the image labels[1] to estimate how useful a single dimension is. Specifically, we denote image labels as $\boldsymbol{y}$, the i-th dimension FV values as $\boldsymbol{x}_{:i}$, and the mutual information as $I(\boldsymbol{x}_{:i}, \boldsymbol{y})$. In general, the larger the mutual information, the more useful this dimension is for classification. The MI value is our importance score for each dimension, and is computed as:

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

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

We then sort the dimensions according to a decreasing order of their importance values. Then, if we want to reduce to 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: just choose an updated subset of dimensions and no additional computations are needed at all.

[1] A set of values in the set $\{1, \ldots, C\}$ in a problem with C categories.

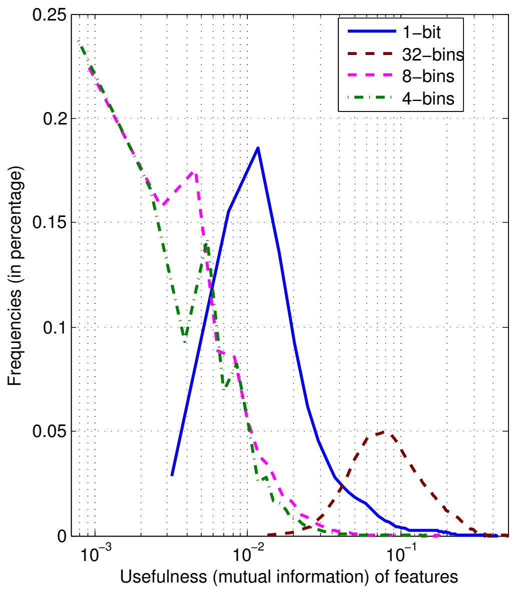

Figure 2. Comparison of dimensions' usefulness / importance values of different quantization methods on the Scene 15 dataset. This figure is best viewed in color.

3.3. 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 ($\boldsymbol{x}_{:i}$) 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. $x \leftarrow 1$ if $x \geq 0$; and $x \leftarrow -1$ if $x < 0$.
- **N-bins**: find the minimum and maximum value in a dimension, and uniformly quantize all values in this dimension into N bins.

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 mutual information between dimensions and the label vector. We show the histogram of mutual information values of different quantization setups in Fig. 2. We use the quantized empirical distributions to compute importance / usefulness scores. Note that this density estimation method just sequentially scan the training vectors once without additional computation, which makes it suitable for large scale dataset.

An observation common to all quantization methods is that most dimensions have small importance values. 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. Feature selection, beyond having the virtue in reducing storage and CPU costs, also has the potential to improve classification accuracy by removing noisy or irrelevant dimensions.

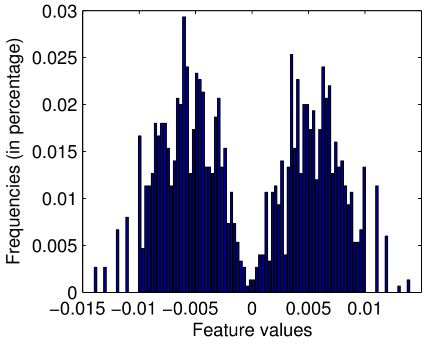

Figure 3. Histogram of values (after power normalization) in one FV dimension in the Scene 15 dataset.

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. 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. In particular, in FV and VLAD, 0 must be a quantization threshold.

Many feature compression methods use the 1-BIT quantization to store features compactly, *e.g.*, BPBC [10] and FV compression [21]. An explanation of its effectiveness is missing, though. From our quantization perspective, Fig. 2 quantitatively shows that 1-BIT is good at preserving useful information. Furthermore, Fig. 3 qualitatively explains why 0 must be the threshold. After power normalization, 9 our of 10 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.

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.

3.4. Classification using quantized features

We then 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) [19] or dual coordinate descent (DCD) [29], 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 $Z \in \mathbb{R}^{256} \times \mathbb{R}^8$:

$$Z = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} . \quad (3)$$

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 is 8) vectors.

4. Experiment

We evaluate the proposed mutual information based importance sorting feature selection method on several large scale benchmarks. It is compared with PQ (product quantization) and BPBC. Since the datasets are large scale and time consuming to evaluate, we use PQ results from the literature when they are available for a dataset, otherwise we report PQ results from our own implementation.

We use the Fisher Vector to represent all images, following the setup in [22]. Only the mean and variance part in FV are used. The base visual descriptor is SIFT, which is reduced from 128 to 64 dimensional using PCA. The number of Gaussian components is 256. We use the spatial pyramid matching structure in [3] 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 to learn a linear SVM classifier from our selected and quantized features or BPBC; and use the LIBLINEAR software package in our PQ experiments.

The following benchmark datasets are used:

- **VOC 2007** [6]. It has 20 object classes. Each image may contain more than one object. We use all the training and validation images (5K) for training and the testing images (5K) for testing.
- **ILSVRC 2010** [2]. It has 1000 classes and 1.2M training images. We use all provided training and testing images for training and testing, respectively.
- **SUN 397** [28]. It has 397 classes. In each class, we use 50 training images and 50 testing images.
- **Scene 15** [15]. It has 15 classes. In each class, 100 images are used for training, and the rest images are used for testing.

In PQ, we use the segment length $d = 8$, which has the overall best performance in [22] under different compression ratios and also used in BPBC [10]. 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. For ILSVRC 2010, we run one time using the given split. All experiments are tested on a computer with Intel i7-3930K CPU and 32G main memory. All CPU cores are used during feature compression. In classifier learning and testing, only one core is used.

We first 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 these three methods.

4.1. VOC 2007

Mean average precisions (mAP) of various methods are shown in Table 1. We use the code from [3][2] to generate FV with the same length as [22], thus it is fair to compare MI's performance with the PQ result from [22].

Under the same compression ratio, MI'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 MI is less than that of PQ. For example, when the ratio is 256, MI only loses 1.75% mAP, while PQ in [26] lost 8.5%.

In MI, 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,

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

Table 1. 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 (%)
MI	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 [26]	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 [22]	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

which confirms yet another time that the 1-BIT quantization is effective in keeping useful information in features.

Another important fact is that when compression ratio is smaller than 128, MI's mAP is higher than the uncompressed one. For example, ratio 64 (half dimensions used) has almost the same mAP as ratio 32 (all dimensions 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 MI with more compression ratios are shown from 32 to 1024. When the compression ratio is 256, MI is comparable to that of PQ with compression ratio 32. Even with compression ratio 1024, MI'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!

MI's classifier training time on VOC 2007 is shown in Fig. 4. 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.

4.2. ILSVRC 2010

We report top-5 accuracy on the ILSVRC 2010 dataset in Table 2. Limited by our computer's memory capacity, we need to start from compression ratio 64 in MI. As shown in Table 2, MI's result is better than PQ's [22] with the same FV setup and the same compression ratio. MI with compression ratio 128 has similar result as PQ at ratio 32.

If absolute accuracy rates are concerned, [19] reported that PQ with a well-tailored SGD classifier achieves 66.5% top-5 accuracy. When combining more visual descriptors like color descriptors [22, 23] and LBP [16], 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.

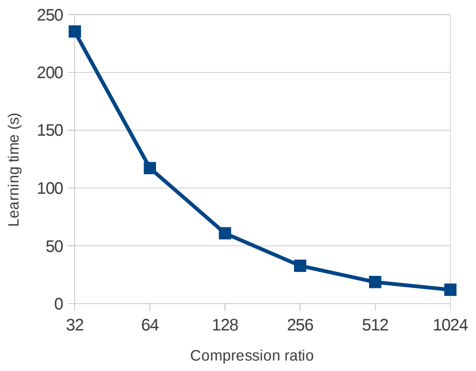

Figure 4. Classifier learning time using the proposed (MI) compressed features on VOC 2007 under different compression ratios.

Table 2. Top-5 accuracy on the ILSVRC 2010 dataset.

Method	Compression ratio	Accuracy (%)
MI	64	61.06
	128	56.64
	256	50.15
PQ [22]	$32(d=8)$	56.2
	$64(d=8)$	54.2
	$64(d=16)$	54.9

Table 3. Top-1 accuracy on the SUN 397 dataset.

Method	Compression ratio	Accuracy (%)
dense FV [23]	1	43.3
multiple features [28]	1	38.0
spatial HOG [8]	1	26.8
MI	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

4.3. SUN 397

We show accuracy of MI and PQ on the SUN 397 dataset in Table 3. Limited by our main memory size, we do not evaluate accuracy of the uncompressed dataset. Comparing with uncompressed FV [23], MI 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 [23] 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, MI is 0.8% inferior to PQ when compression ratio is 32, but better than

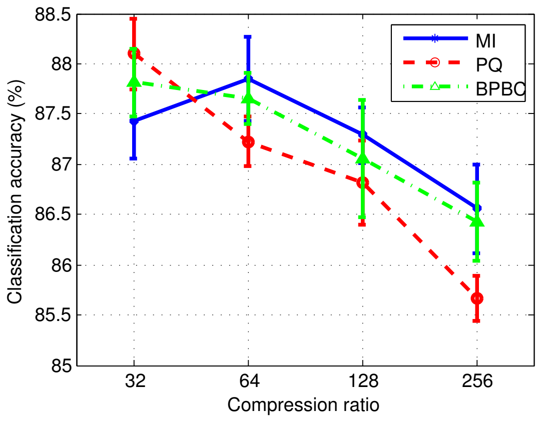

Figure 5. Classification accuracy of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

or comparable with it when compression ratio gets larger. However, as we will show in Sec. 4.4, the required time in feature compression/selection, classifier learning and testing of MI is much less than that of PQ. Thus, overall MI is more efficient and effective than PQ.

One final note for Table 3 is that MI exhibits non-monotone accuracy again: ratio 64 accuracy is slightly higher than ratio 32.

4.4. Scene 15

On the Scene 15 dataset, we not only compare the accuracy of different methods. Because this dataset is small scale, we also perform additional experiments to compare the efficiency of the proposed MI method with PQ and BPBC, including feature compression/selection time, the classifier learning time, and the testing time.

The minimum compression ratio that MI 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. 5. Overall all three methods have similar accuracies. PQ has an edge at ratio 32, while MI 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 methods. The feature selection time in MI 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 time. BPBC costs most time in the feature compression

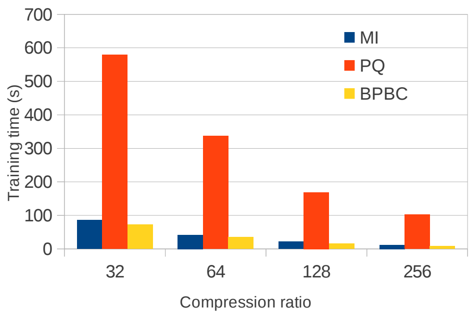

Figure 6. Classifier training time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

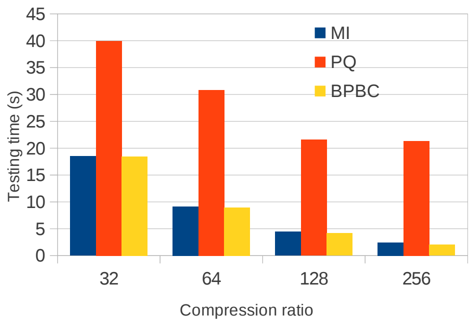

Figure 7. Classifier testing time of FV on Scene 15 using three methods: MI (proposed), PQ, and BPBC.

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 MI or PQ in dimensionality reduction for high dimensional vectors.

Finally, the classifier training and testing time of these three methods are shown in Fig. 6 and Fig. 7, respectively. The training and testing times of MI and BPBC are similar to each other at all compression levels, and both are much less than that of PQ. This is because MI and BPBC only uses 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. 7 for testing time.

5. 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, a fact that also hints on the missing of multicollinearity (*i.e.*, strong linear relationships among more than 2 dimensions). This observation undermines the effectiveness of compression methods. We propose to use mutual information to choose a subset of dimensions for dimensionality reduction, which is a natural choice if we assume both collinearity and multicollinearity do not exist. Feature selection also removes noisy dimensions.

We use 1-BIT quantization to compute mutual information. We show that when quantizing dimensions, the zero value must be a threshold because it leads to bimodal distributions. Later, we also quantize the chosen dimensions into 1-BIT and design a linear SVM classifier to learn directly on the quantized values. Experiments show that the proposed feature selection and quantization method achieves better accuracy than feature compression methods, and is more efficient.

Our future works include the following. First, we will evaluate our method on more high dimensional feature vectors, which can be extracted from multiple descriptors like [22, 23, 16]. Second, more effective feature selection will be studied for higher accuracy. Finally, more efficient classifier learning techniques can be investigated for the compressed or quantized features.

References

[1] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

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

[3] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2010.

[4] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *CVPR*, 2013.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[6] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL VOC 2007 results.

[7] F. Fleuret. Fast binary feature selection with conditional mutual information. *JMLR*, 5:1531–1555, 2004.

[8] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

[9] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2013.

[10] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[11] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[12] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on PAMI*, 33(1):117–128, 2011.

[13] H. Jégou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[14] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on PAMI*, 34(9):1704–1716, 2011.

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

[16] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, and K. Yu. Large-scale image classification: Fast feature extraction and SVM training. In *CVPR*, 2011.

[17] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, 2013.

[18] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on PAMI*, 27(8):1226–1238, 2005.

[19] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid. Towards good practice in large-scale learning for image classification. In *CVPR*, 2012.

[20] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, pages 1–8, 2007.

[21] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010.

[22] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, pages 1665–1672, 2011.

[23] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[24] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *ICCV*, 2009.

[25] M. Tan, L. Wang, and I. W. Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, 2010.

[26] A. Vedaldi and A. Zisserman. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2012.

[27] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

[28] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[29] G.-X. Yuan, C.-H. Ho, and C.-J. Lin. Recent advances of large-scale linear classification. *Proceedings of the IEEE*, 100:2584–2603, 2012.

[30] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.