Sparse-Coded Features for Image Retrieval

Tiezheng Ge
getzh@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Qifa Ke
qke@microsoft.com
Microsoft Bing
Sunnyvale, CA, US

Jian Sun
jiansun@microsoft.com
Microsoft Research Asia
Beijing, China

Abstract

State-of-the-art image retrieval systems typically represent an image with a bag of low-level features. Since different images often exhibit different kinds of low-level characteristics, it is desirable to represent an image with multiple types of complementary features. The systems scalability is, however, significantly lowered when increasing the number of feature types, as the amount of data is also increased rapidly both in index and in query representation.

In this paper, we apply sparse coding to derive a compact yet discriminative image representation from multiple types of features for large-scale image retrieval. We first convert each feature descriptor into a sparse code, and aggregate each type of sparse-coded features into a single vector by max-pooling. Multiple vectors from different types of features are then concatenated and compressed to obtain the final representation. Our approach allows us to add more types of features to improve discriminability without sacrificing scalability. In particular, we design a new *micro feature* which is complementary to existing local invariant features. By combining our micro feature with various local invariant features using the sparse-coding framework, our final compact representation outperforms the state of the art both in retrieval performance and in scalability.

1 Introduction

The bag-of-features(BOF) image representation is popular in large-scale image retrieval [10, 18, 23, 25], where local invariant features [15, 17] are vector-quantized into visual words such that scalable text retrieval approaches, in particular inverted file indexing, can be applied. With BOF, the memory to store the inverted index file and the search complexity are both approximately linearly increased with the number of images.

Various approaches (c.f. [1, 6, 13, 27, 28]) have been proposed to address the retrieval efficiency and/or the memory constraint problem. However, these approaches often trade accuracy for efficiency, or require exhausted scan of the whole image database. An alternative approach is to aggregate local descriptors in one image into a single vector using Fisher Vector [21] or Vector of Local Aggregated Descriptor (VLAD) [9]. It has been shown in [9] that with as few as 16 bytes to represent an image, the retrieval performance is still comparable to that of the BOF representation.

In this paper, we propose to use sparse coding with max-pooling to aggregate local descriptors for image retrieval. Sparse coding [19] learns an over-complete set of bases where an image can be represented by a high-dimensional but sparse vector. In statistics community sparse coding is also known as Lasso regression (see [7], page 72). It has been used in image processing and analysis (c.f. [5, 16]). For image recognition/categorization applications, it has been shown that such sparse representation is more linearly separable—our approach is motivated by the successful application of sparse coding in image classification [2, 24, 29, 33, 34]. In particular, we propose using sparse coding to aggregate local features for image retrieval, where local features extracted from an image are first projected onto the learned bases (codebook) using sparse coding, and the resulted sparse codes are then max-pooled together into a single vector.

Existing aggregation approaches typically use a single type of low-level features. A major advantage of our approach is the capability to aggregate multiple types of local features into a compact yet discriminative representation. Each type of features from one image is independently aggregated into a single vector by sparse coding and max-pooling. We concatenate all vectors from different types, and compress them. The final representation is a low-dimension vector for each image.

Since the sparse-coding allows us to use multiple types of features to improve the retrieval performance without sacrificing search efficiency and scalability, we are motivated to not only aggregate existing popular features, but also seek for new features that are complementary to existing ones. In particular, we design a new feature called *micro feature*. *Micro feature* exploits self-similarity existed within an image [35], is simple to compute, yet exhibits surprisingly good performance by itself. By combining *micro features* with other local features using the sparse-coding, we have derived a compact image representation that outperforms the state-of-the-art in terms of retrieval performance, efficiency, and scalability.

2 Background: aggregating local descriptors

The goal of aggregating local descriptors is to convert a bag of variable number of local features in an image into a global, fixed-size vector representation. In this section, We briefly review two leading aggregation approaches: Fisher Vector [20, 21, 22] and VLAD [9], and reformulate BOF [25] in the aggregation form.

Fisher Vector

Fisher kernel [8] was originally proposed to extract discriminative features. Perronnin et.al. [21] applied Fisher kernel to aggregate local descriptors. They use Gaussian Mixture Model (GMM) as the generative model and aggregated normalized gradient vectors into a fixed-size *Fisher* vector.

Let $\mathtt{X} = \{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_T\}$ be a set of local descriptors (e.g. SIFT descriptors [15]) in a given image, and $\{(\omega_i, \mu_i, \Sigma_i), i = 1...N\}$ is a pre-learned GMM model with N components, where $(\omega_i, \mu_i, \Sigma_i)$ are the weight, mean, and covariance of the i-th component. If we simplify the covariance to a diagonal matrix ($\Sigma_i = \sigma_i$) and only compute the derivatives of the Gaussian means, then the normalized, concatenated gradient vector for the descriptor $\mathbf{x}$ is:

$$g(\mathbf{x}) = [\zeta_1^x, ..., \zeta_i^x, ..., \zeta_N^x], \quad \zeta_i^x = \frac{1}{\sqrt{\omega_i}} \gamma_x(i)[(\mathbf{x} - \mu_i)./\sigma_i], \tag{1}$$

where $\gamma_x(i) = \omega_i p_i(\mathbf{x}) / \sum_j w_j p_j(\mathbf{x})$ is the probability that the feature $\mathbf{x}$ belongs to the i-th Gaussian component, and $./$ is element-wise division. Thus, a D-dimensional descriptor $\mathbf{x}$ is

encoded into a $N \times D$-dimensional Fisher vector $g(\mathbf{x})$.

To form the vector representation $G(\mathtt{X})$ for the whole image, all encoded Fisher vectors are aggregated together:

$$G(\mathtt{X}) = \frac{1}{T} \sum_{t=1}^{T} g(\mathbf{x}). \tag{2}$$

The dimension of $G(\mathtt{X})$ is also $N \times D$. The typical value of N in the Fisher Vector framework is 64. So, the size of the final image representation is $64 \times 128 = 8192$ for 128-dimensional SIFT-like descriptors.

VLAD: vector of locally aggregated descriptors

VLAD [9] can be considered as a simplification of the Fisher vector. Instead of using GMM, it uses k-means clustering to learn a codebook $\mathtt{C} = \{\mathbf{c}_1, ..., \mathbf{c}_N\}$ of N visual words. For a given image, each local descriptor $\mathbf{x}$ is first assigned to its nearest visual word $\mathbf{c}_i = NN(\mathbf{x})$, and the differential (residual) vector $\mathbf{x} - \mathbf{c}_i$ is computed. The descriptor $\mathbf{x}$ is then mapped to a vector by:

$$g(\mathbf{x}) = [0, ..., \mathbf{x} - \mathbf{c}_i, ..., 0]. \tag{3}$$

The final VLAD is the aggregation of all mapped vectors:

$$G(\mathtt{X}) = \sum_{t=1}^{T} g(\mathbf{x}_t). \tag{4}$$

Typically the codebook size $N = 64$, and the resulting dimension of VLAD is also 8192 for 128-dimensional SIFT-like descriptors.

BOF: bag of features

BOF [25] can be reformulated in a way similar to VLAD. Using a codebook $\mathtt{C} = \{\mathbf{c}_1, ..., \mathbf{c}_N\}$, each descriptor is mapped to a high dimensional binary vector by:

$$g(\mathbf{x}) = [\delta(\mathbf{c}_1 = NN(\mathbf{x})), ..., \delta(\mathbf{c}_i = NN(\mathbf{x})), ..., \delta(\mathbf{c}_N = NN(\mathbf{x}))], \tag{5}$$

where $\delta(true) = 1$ and $\delta(false) = 0$. Then BOF is the sum of all mapped vectors:

$$G(\mathtt{X}) = \sum_{t=1}^{T} g(\mathbf{x}_t). \tag{6}$$

Note that in BOF, the size of codebook N could be as large as a million for image retrieval.

Discussion

As we can see, Fisher Vector, VLAD, and BOF can be derived in two steps: encoding and pooling. The encoding step encodes each D-dimensional local descriptor into another high-dimensional (usually much higher than D) vector. Equation (1)(3)(5) are three different encoding methods. The pooling step aggregates all encoded vectors into a single vector. Fisher Vector, VLAD and BOF use simple average/sum pooling, as shown in Equation (2)(4)(6).

The success of these two-step approaches naturally leads to two questions—are there better encoding/pooling alternatives? Can we find better local descriptors for the same encoding/pooling framework? We answer these questions in following sections.

3 Sparse coding for image retrieval

In this section, we revisit the sparse coding(SC) framework proposed in [33]. By analyzing the relationship between SC and local feature aggregation [9, 21], we propose to apply SC to local feature aggregation for image retrieval, and it also allows combining different types of feature descriptors in a compact representation

3.1 Data sets

Two common datasets are used in describing our approach and conducting the experiments.
INRIA Holidays [10]. The original INRIA Holidays dataset contains 1491 holiday images, 500 of them being used as queries. The accuracy is measured by mean Average Precision (mAP). For large-scale experiments, we mix the Holiday dataset with 4 million distracter images downloaded from Flickr. We call this extended dataset *Holidays 4 million*.
University of Kentucky Benchmark (UKB) [18]. This set consists of images of 2550 objects, each of which is represented by 4 images. The most commonly used performance metric for this dataset is the average number of relevant images in the top 4 retrieved images (one of which is the query itself). The score thus ranges from 0 to 4.

For any codebook training, we randomly sample local features from 100K images in an independent dateset, the **ImageNet** [3].

3.2 Sparse coding for local descriptor aggregation

The sparse coding framework encodes the bag of features from an image into an N-dimensional sparse vector. We review sparse coding in this subsection using the same matrix formulation in [33]. It consists of two steps—Step 1 encodes each local descriptor into a sparse code, and Step 2 pools all sparse codes from Step 1 into a single vector.
Step 1: Encoding. Each local descriptor $\mathbf{x}$ from an image is encoded into an N-dimensional vector $\mathbf{u} = [u^1, u^2, ..., u^N]$ by fitting a linear model with sparsity (L_1) constraint:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \|\mathbf{x} - \mathbf{u}\mathtt{V}\|_2^2 + \lambda \|\mathbf{u}\|_1, \\ \text{subject to} \quad & \mathbf{u} \succeq 0 \end{aligned} \tag{7}$$

Here $\mathtt{V} = [\mathbf{v}_1, ..., \mathbf{v}_N]^\top$ is the codebook, i.e., a set of over-complete bases.It's learned beforehand by alternative optimization [33]. The second term is the L_1 penalty term to enforce sparsity on the vector $\mathbf{u}$, and λ is the parameter to control the sparsity.
Step 2: Pooling. For a given image with T descriptors $\mathtt{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$, after obtaining their corresponding sparse codes $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_T\}$ in Step 1, we now pool them into a single N-dimensional vector $\mathbf{y} = [y^1, ..., y^i, ...y^N]$. In image classification, there are two often-used pooling methods—average pooling and max pooling:

$$\text{average pooling:} \qquad y^i = \sum\nolimits_{t=1}^{T} u_t^i \tag{8}$$

$$\text{max pooling:} \qquad y^i = \max\{u_t^i \,|\, t = 1, ..., T\} \tag{9}$$

The pooled vector $\mathbf{y}$ is subsequently normalized by $\mathbf{y} := \mathbf{y}/\|\mathbf{y}\|_2$ to generate the final single-vector representation. We call the normalized vector $\mathbf{y}$ the sparse-coded vector.

The local feature aggregation approaches for image retrieval we presented in Section 2, including Fisher Vector, VLAD, and BOF, share a common two-step framework as the sparse coding framework in encoding a bag of local descriptors $\mathtt{X} = \{\mathbf{x}_t \mid t = 1, ..., T\}$:

$$\text{encoding:} \quad g(\mathbf{x}) = \mathbf{u}, \tag{10}$$

$$\begin{aligned} \text{pooling:} \quad & G(\mathtt{X}) = \sum\nolimits_{t=1}^{T} g(\mathbf{x}_t) \quad \text{or} \\ & G(\mathtt{X}) = \max\{g(\mathbf{x}_t) \,|\, t = 1, ..., T\}. \end{aligned} \tag{11}$$

The only difference among them is in how the encoding is obtained. Such a common framework, together with the success of local feature aggregation in image retrieval and the success

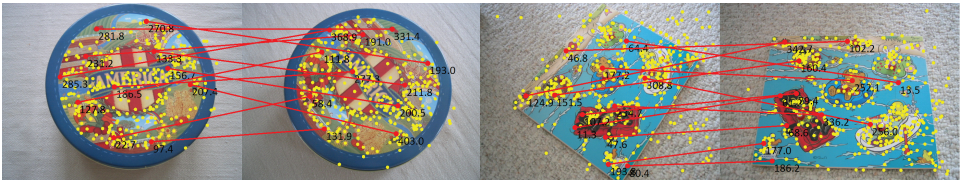

Figure 1: Two examples of matching image pairs. Active features are shown by yellow dots. Some co-active and matching feature pairs are linked by red lines. The number at each point is the code value at the active dimension.

of sparse coding in image classification, motivates us to apply the sparse coding framework to the image retrieval task.

Implementation differences from image classification. We note two important differences between our image retrieval and image classification in using sparse coding: 1) the input local invariant descriptors are computed from sparse interest/key points instead of densely sampling grids; 2) while spatial pyramid matching (SPM) [14, 33] is a standard practice in image classification, we do not use any form of SPM as the spatial location is sensitive to translation and rotation, two typical scenarios to handle in image retrieval.

Max-pooling is essential. In image classification max-pooling is reported to perform better than average pooling [14, 24, 33]. We also empirically found maximum pooling performed significantly better than average pooling when applying sparse coding to image retrieval. Table 1 shows the retrieval results on datasets described in 3.1, with a single type feature representation. The reason is that max-pooling tends to pick up the distinctive features that are more likely to be repeatable, an important factor in image retrieval.

	Holiday(mAP)	UKB(score/4)
Average pooling	0.553	3.05
Max pooling	0.599	3.40

Table 1: Retrieval results using max pooling vs. average pooling. Features are detected by the Harris detector and represented by DAISY descriptors.

3.3 Discussion:Why does sparse coding work for image retrieval?

To understand the rationale behind using sparse coding for image retrieval, we perform empirical analysis below.

A descriptor $\mathbf{x}$ is *active* in an image (represented by T descriptors), if there exists at least one dimension i such that its sparse code $\mathbf{u}$ satisfies $u^i = \max\{u_t^i | t = 1, ..., T\}$. In other words, the i-th dimension of the final max-aggregated vector is solely determined by the active descriptor $\mathbf{x}$. We say a pair of descriptors from two images are *co-active*, if they are active in one or more common dimensions of their sparse codes, respectively.

Descriptors from visually similar patches are closer to each other and are more likely to be co-active, and will generate similar values in the co-active dimensions of their sparse codes. Relevant images have more visually similar corresponding features than non-relevant images, and thus their aggregated vectors should be more similar. On the other hand, if two descriptors (from two images respectively) are co-active at one or more dimensions in $\mathbf{u}$, due to *sparsity* they have high chances to be visually similar. Thus two similar aggregated vectors are more likely to come from relevant images.

scale \ rotation	0°	30°	60°
1.33	64%	60%	61%
1.00	-	61%	62%
0.75	62%	52%	53%

Table 2: Statistics of co-active and matching feature pairs. We pick an image from UKB as query, and manually transform it with various scales and rotations. The table shows the percentage of co-active pairs among 500 matching feature pairs.

Figure 1 shows two pairs of relevant images in which yellow dots are active descriptors. We show some of the many co-active feature pairs, which are linked by red lines. As we expect, there exist many such correct corresponding pairs (for clarity, not all pairs are drawn) which are implicitly established by the sparse coding method.

For a more quantitative measure, we use an image from UKB as query and create its several rotated and scaled versions.Then, we calculate the percentage of co-active feature pairs among true matching ones. As listed in Table 2, under large image variations, there is still a large percentage of matching features that are also co-active.

3.4 Multiple feature pooling and compression

In image retrieval, it has been shown that increasing the number of descriptors (by using several key point detectors to detect more feature points) considerably improves retrieval performance. But it is typical that only one single type of local invariant descriptor is used in existing image retrieval systems. We observed that different images exhibit different appearance characteristics. While some images are well represented by some local invariant descriptor for retrieval purpose, some images may be well better described by different descriptors such as color histograms.

The sparse coding framework is well suited for aggregating multiple types of descriptors without increasing the size of the final image representation. In particular, we propose the following *multiple feature pooling and compression*:

1. Extract multiple local descriptor sets from a given image (one set for each type of descriptors);
2. Apply sparse coding to each descriptor set to derive a single aggregated vector;
3. Jointly compress all sparse-coded aggregated vectors to derive the final representation in a compact vector.

More formally, if we extract K sets of local descriptors and apply sparse coding and max pooling on each set independently to obtain aggregated vectors $\{\mathbf{y}_1, ..., \mathbf{y}_K\}$. Then, the final image representation is a concatenated vector:

$$\mathbf{y}^* = [\mathbf{y}_1^\top, ..., \mathbf{y}_K^\top]^\top. \tag{12}$$

Since there exist plenty of redundancies between sparse-coded vectors, the concatenated vector $\mathbf{y}^*$ can be effectively compressed.In this paper, we found PCA performs well in reducing the dimension while preserving the discriminative power of $\mathbf{y}^*$. We achieve further compression (*e.g.*, to 20-40 bytes) with product quantization [12].

For a given image, we can apply different local feature detectors [17], such as LOG, Harris, MSER, to detect multiple features. For each detected local feature, we can then compute different types of descriptors, such as SIFT [15] or DAISY [26, 32]. We experiment with various combinations of multiple detectors and descriptors in the context of image retrieval,

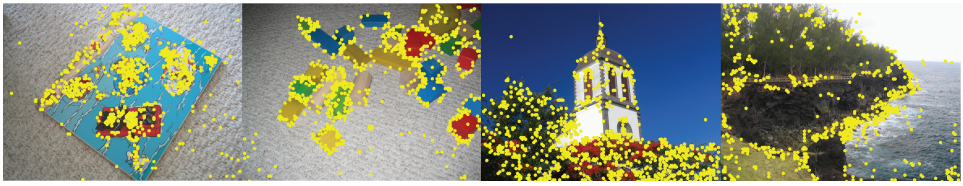

Figure 2: Active micro features. Note that the "fired" features are generally distinctive and a few representative features are detected from the smooth regions.

	LOG/SIFT	Harris/DAISY	Harris/SIFT
LOG/DAISY	0.647	0.637	0.641
LOG/SIFT		**0.663**	0.634
Harris/DAISY			0.623

Table 3: Combining two different detectors (LOG and Harris) and descriptors (SIFT and DAISY). The best configuration is "Harris/DAISY + LOG/SIFT", which leverages all information.

using the Holiday dataset described in Section 3.1. Table 3 shows the *m*AP results from different combinations. Interestingly, we found that the best two-type combination is Harris-DAISY and LOG-SIFT, which covers two different detectors and two different descriptors. It indicates that the multi-feature sparse coding framework is indeed effectively exploiting the *complementary power among different features*.

3.5 Sparse-coded micro feature

Since our sparse coding framework allows aggregating multiple types of features in a compact way, we are motivated to seek new features, especially those complementary to existing local invariant features. Inspired by the work of bag-of-colors(BOC) [30], where an image is represented by its color histogram, we propose a novel local feature called *micro feature*.

First, our micro features are simply detected by densely sampling the image on a grid with some predefined step size s. Then, at each sampled point, we extract a vector consisting of color values of a small $k \times k$ color patch in the CIE-Lab color space, resulting in a $3k^2$ dimensional vector, the *micro feature*. Finally, we feed all extracted vectors into the same sparse coding framework to produce our *sparse-coded micro features*.

We experiment with various values of s and k using the Holiday and the UKB datasets. Surprisingly, we found that $s = 2$ and very small patch size $k = 4$ work best, and even comparable to strong local invariant features(c.f. Section 4)! We conjecture that the excellent performance is resulted from the proper combination of the micro feature itself and the sparse coding framework, as we explain below.

First, by using small patches, our micro feature is robust to small affine deformations.

Second, the sparse coding and maximum pooling framework effectively picks *meaningful* patches on edge and corner locations. Figure 2 shows the *active* micro features from some example images. Patches from smooth regions are mostly excluded except for a few representatives, which are still useful for retrieval.

Third, our micro feature effectively exploits image self-similarity [35]. Suppose $\mathbf{x}$ and $\mathbf{x}'$ are corresponding points from a pair of matching images I and I', respectively. Then with self similarity, $\mathbf{x}'$ has a set of similar features $\mathtt{X}'$ in I'. When $\mathbf{x}$ is active in the aggregated vector of I, with a high probability a feature from I', either $\mathbf{x}'$ itself or one in $\mathtt{X}'$, is also active. Thus, the self-similarity contributes to the similarity of the final image vectors aggregated from the sparse coded micro features.

Approaches	Dim	Holiday (mAP)				UKB (score/4)			
		D	→128	→64	→32	D	→128	→64	→32
Fisher (HD)	6656	0.566	0.530	0.499	0.458	3.15	3.09	3.02	2.84
VLAD (HD)	6656	0.559	0.527	0.496	0.454	3.14	3.05	3.00	2.84
SC (HD)	5000	0.599	0.525	0.505	0.479	3.40	3.29	3.21	3.06
SC (Micro)	1024	0.673	0.661	0.637	0.599	3.41	3.38	3.33	3.21
SC (LS)	5000	0.591	0.530	0.506	0.466	3.04	3.04	2.97	2.81
SC (HD+LS)	10000	0.664	0.599	0.562	0.528	3.50	3.45	3.37	3.21
SC (HD+LS+Micro)	11024	0.767	0.727	0.700	0.664	3.76	3.67	3.62	3.52

Table 4: Performance comparison of different aggregation methods (including Fisher Kernel, VLAD, and Sparse Coding) and dimension reductions. For Fisher and VLAD, HD performs better than LS, and we only report their HD results. Column Dim is the original dimension of aggregated features. Notation: HD for Harris Detector + DAISY descriptor; LS for LOG detector + SIFT descriptor.

	Holiday(mAP)		UKB(score/4)	
	L1	L2	L1	L2
BOC (512-D) [30]	0.617	0.573	3.40	3.12
Micro (1024-D)	0.664	0.673	3.44	3.41
Micro (512-D by PCA)	-	0.678	-	3.41

Table 5: Comparing our micro feature with bag of colors (BOC) [30].

	HD	LS	HD+LS	micro	HD+LS+micro
Holiday	0.522	0.522	0.587	0.658	0.725
UKB	3.26	3.00	3.42	3.36	3.65

Table 6: Performances with product quantization at 40Bytes/Image

4 Experiments

We use two standard datasets (Holiday and UKB) to evaluate the performance of sparse-coded local invariant features, micro features, and their combinations. In our method, we follow the *sparse coding + pooling + L2 normalization + feature combination* pipeline and use L2 distance as default metric.

Sparse coded local-invariant feature

We compare the performance of local invariant features aggregated by different methods. The features are detected by the Harris detector and an 104-D DASIY descriptor [31] is computed for each feature. The descriptors are then aggregated by Fisher Kernel, VLAD, and sparse coding (SC). The codebook size is $k = 64$ for Fisher and VLAD, resulting in a $104 \times 64 = 6656$ dimensional aggregated vector. The results are shown in the first three rows of Table 4, at original dimension, and various PCA-reduced dimensions. As we can see, sparse-coding performs significantly better than Fisher Kernel and VLAD on the UKB dataset, and considerably better on the Holiday dataset except for the 128-D case.

Sparse coded micro feature

We first compare our micro feature with BOC [30], both of which explore the color information and use dense sampling. Table 5 shows that our micro feature outperforms BOC on both datasets, using L1 or L2 image distance metric. Note that the performance of BOC decreases by about 8% when using L2 metric, while micro feature performs similar for either L1 or L2 metric—a desired property as efficient dimension reduction algorithms, such as PCA, require L2 metric.

	Method	Holiday	UKB
BOF-based representation	BOW k=20K [11]	0.572	2.95
	HE [10]	0.745	3.30
	LBOC [30]	0.789	3.50
Aggregation	VLAD 8192-D [9]	0.526	3.17
	FK 4096-D [21]	<0.60	<3.30
	FK+Attr. 6755-D [4]	0.699	
	FK+Attr. 128-D [4]	0.633	
Proposed method	SC (HD+LS) 10000-D	0.664	3.50
	SC (Micro) 1024-D	0.673	3.41
	SC (HD+LS+Micro) 11024-D	0.767	3.76
	SC (HD+LS+Micro) PCA128-D	0.727	3.67

Table 7: Comparison with the state of the art. For VLAD and Fisher Kernel (FK), the codebook size k is 64.

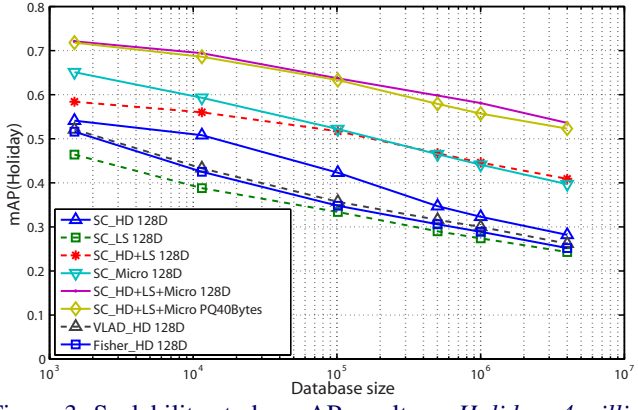

Figure 3: Scalability study: mAP results on *Holidays 4 million*.

We also compare our micro feature with local invariant features, as shown by the row of SC(Micro) in Table 4. The sparse-coded micro feature consistently outperforms Harris+DASIY with all three aggregation methods on both datasets.

Multiple feature pooling and compression

We evaluate the performances of different feature combinations and compressions. In particular, we consider the combination of Harris/DAISY+LOG/SIFT (HD+LS, see Section 3.4) for local invariant features, and HD+LS+Micro for combining invariant features and color features. As shown in the last three rows in Table 4, the sparse-coded HD+LS outperforms any single-type features (HD or LS). When combined with micro feature (HD+LS+Micro), we achieve about 14% and 10% improvement over the best individual feature on Holiday and UKB, respectively, and the trend holds for all compressed features.

Product quantization [12] can further compress the aggregated vector. Table 6 shows the results of further compressing each 128-D PCA-compressed aggregated vector into 40 bytes. The performance loss from product quantization is negligible (Compared with Table 4).

Compared to the state of the art

We compare our work to state-of-the-art retrieval systems, including those using BOF-based and aggregation-based representations. The first two rows in Table 7 are the best results from previous works. The results of our system are shown in the third row. On the UKB dataset, our system outperforms all previous works, even when using a compact 128-D image representation. On the Holiday dataset, our PCA-compressed combined-feature representations outperform previous aggregated representations significantly. Although our

128-D feature performs worse than HE and LBOC on the Holiday dataset, our representation is much more compact compared to HE and LBOC which require a set of quantized descriptors (visual words) plus some per-feature signatures to represent each image.

Scalability

We study the scalability of our approach using the *Holidays 4 million* dataset. Figure 3 shows the mAP results vs. database size for different versions of our method. We see that our method scales well. In particular, our final representation(HD+LS+Micro) performs the best, exceeding state-of-the-art feature aggregations systems—Fisher Vector and VLAD—by 28 and 27 absolute mAP points, respectively. Product quantization further compresses (HD+LS+Micro) into a 40-byte representation (HD+LS+Micro PQ), with negligible loss.

5 Conclusion

Based on reformulating the state-of-the-art local feature aggregation methods in a common encoding-pooling framework, we have applied sparse coding to encode feature descriptors for image retrieval. The sparsity of the representation allows us to max-aggregate descriptors from an image into a fixed-size vector, with tolerable loss of information. By integrating multiple types of complementary features, including a novel *micro feature*, using sparse coding framework, we obtain a discriminative yet compact image representation that significantly outperforms the state of the art.

References

[1] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] Ken Chatfield, Victor Lemtexpitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *BMVC*, 2011.

[3] 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.

[4] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *CVPR*, 2011.

[5] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 54(12):3736–3745, 2006.

[6] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001.

[8] Tommi S. Jaakkola and David Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999.

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

[10] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[11] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87(3):316–336, 2010.

[12] Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011.

[13] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

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

[15] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 20:91–110, 2003.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *CVPR*, 2008.

[17] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005.

[18] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[19] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[20] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2006.

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

[22] F. Perronnin, J. Sanchez, and T. Mensink. Fisher kernel for large-scale image classification. In *ECCV*, 2010.

[23] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[24] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML*, 2007.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[26] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *CVPR*, 2008.

[27] Antonio Torralba, Rob Fergus, and Yair Weiss. Small codes and large image databases for recognition. *CVPR*, 2008.

[28] L. Torresani, M. Szummer, and A. Fitzgibbon. Learning query-dependent prefilters for scalable image retrieval. In *CVPR*, 2009.

[29] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, , and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010.

[30] Christian Wengert, Matthijs Douze, and Herve Jégou. Bag of colors for improved image search. In *ACM Multimedia*, 2011.

[31] Simon Winder, Gang Hua, and Matthew Brown. Picking the best daisy. In *CVPR*, 2009.

[32] Simon A. J. Winder, Gang Hua, and Matthew Brown. Picking the best daisy. In *CVPR*, 2009.

[33] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classication. In *CVPR*, 2009.

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

[35] Maria Zontak and Michal Irani. Internal Statistics of a Single Natural Image. In *CVPR*, 2011.