Locality-constrained Linear Coding for Image Classification

Jinjun Wang[†], Jianchao Yang[‡], Kai Yu[§], Fengjun Lv[§], Thomas Huang[‡], and Yihong Gong[†]
[†]Akiira Media System, Palo Alto, California
[‡]Beckman Institute, University of Illinois at Urbana-Champaign
[§]NEC Laboratories America, Inc., Cupertino, California
[†]{jjwang, ygong}@akiira.com, [‡]{jyang29, huang}@ifp.uiuc.edu, [§]{kyu, flv}@sv.nec-labs.com

Abstract

The traditional SPM approach based on bag-of-features *(BoF) requires nonlinear classifiers to achieve good image classification performance. This paper presents a simple but effective coding scheme called Locality-constrained Linear Coding (LLC) in place of the VQ coding in traditional SPM. LLC utilizes the locality constraints to project each descriptor into its* local-coordinate system*, and the projected coordinates are integrated by* max pooling *to generate the final representation. With linear classifier, the proposed approach performs remarkably better than the traditional nonlinear SPM, achieving* state-of-the-art *performance on several benchmarks.*

Compared with the sparse coding strategy [22], the objective function used by LLC has an analytical solution. In addition, the paper proposes a fast approximated LLC method by first performing a K-nearest-neighbor search and then solving a constrained least square fitting problem, bearing computational complexity of $\mathcal{O}(M + K^2)$. Hence even with very large codebooks, our system can still process multiple frames per second. This efficiency significantly adds to the practical values of LLC for real applications.

1. Introduction

The recent *state-of-the-art* image classification systems consist of two major parts: *bag-of-features* (BoF) [19, 4] and *spatial pyramid matching* (SPM) [15]. The BoF method represents an image as a histogram of its local features. It is especially robust against spatial translations of features, and demonstrates decent performance in whole-image categorization tasks. However, the BoF method disregards the information about the spatial layout of features, hence it is incapable of capturing shapes or locating an object. Of the many extensions of the BoF method, including the generative part models [7, 3, 2], geometric correspondence search [1, 14] and discriminative codebook learning [13, 17, 23], the most successful results were reported by using SPM [15]. Motivated by [10], the SPM method partitions the image into increasingly finer spatial sub-regions and computes histograms of local features from each sub-region. Typically, $2^l \times 2^l$ subregions, $l = 0, 1, 2$ are used. Other partitions such as 3×1 has also been attempted to incorporate domain knowledge for images with "sky" on top and/or "ground" on bottom. The resulted "spatial pyramid" is a computationally efficient extension of the orderless BoF representation, and has shown very promising performance on many image classification tasks.

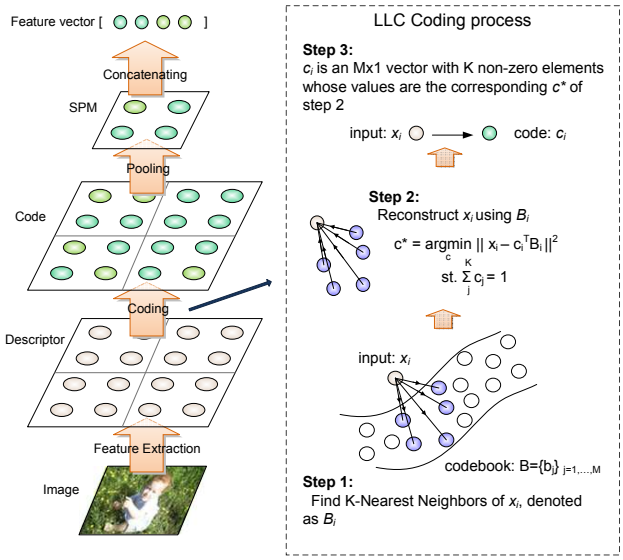


Figure 1. Left: flowchart of the spatial pyramid structure for pooling features for image classification. Right: the proposed LLC coding process.

A typical flowchart of the SPM approach based on BoF is illustrated on the left of Figure 1. First, feature points are detected or densely located on the input image, and descriptors such as "SIFT" or "color moment" are extracted from each feature point (highlighted in blue circle in Figure 1). This obtains the "Descriptor" layer. Then, a codebook with M entries is applied to quantize each descriptor and gen-

erate the "Code" layer, where each descriptor is converted into an $\mathbb{R}^M$ code (highlighted in green circle). If hard vector quantization (VQ) is used, each code has only one non-zero element, while for soft-VQ, a small group of elements can be non-zero. Next in the "SPM" layer, multiple codes from inside each sub-region are pooled together by averaging and normalizing into a histogram. Finally, the histograms from all sub-regions are concatenated together to generate the final representation of the image for classification.

Although the traditional SPM approach works well for image classification, people empirically found that, to achieve good performance, traditional SPM has to use classifiers with nonlinear Mercer kernels, e.g., Chi-square kernel. Accordingly, the nonlinear classifier has to afford additional computational complexity, bearing $\mathcal{O}(n^3)$ in training and $\mathcal{O}(n)$ for testing in SVM, where n is the number of support vectors. This implies a poor scalability of the SPM approach for real applications.

To improve the scalability, researchers aim at obtaining nonlinear feature representations that work better with linear classifiers, e.g. [26, 22]. In particular, Yang *et al*. [22] proposed the ScSPM method where sparse coding (SC) was used instead of VQ to obtain nonlinear codes. The method achieved state-of-the-art performances on several benchmarks. Yu *et al*. [24] empirically observed that SC results tend to be *local* – nonzero coefficients are often assigned to bases nearby to the encoded data. They suggested a modification to SC, called Local Coordinate Coding (LCC), which explicitly encourages the coding to be local, and theoretically pointed out that under certain assumptions locality is more essential than sparsity, for successful nonlinear function learning using the obtained codes. Similar to SC, LCC requires to solve L1-norm optimization problem, which is however computationally expensive.

In this paper, we present a novel and practical coding scheme called Locality-constrained Linear Coding (LLC), which can be seem as a fast implementation of LCC that utilizes the locality constraint to project each descriptor into its *local-coordinate system*. Experimental results show that, the final representation (Figure 1) generated by using LLC code can achieve an impressive image classification accuracy even with a linear SVM classifier. In addition, the optimization problem used by LLC has an analytical solution, where the computational complexity is only $\mathcal{O}(M + M)$ for each descriptor. We further propose an approximated LLC method by performing a K-nearest-neighbor (K-NN) search and then solving a constrained least square fitting problem. This further reduces the computational complexity to $\mathcal{O}(M + K)$, where K is the number of nearest neighbors, and usually $K < 0.1 \times M$. As observed from our experiment, using a codebook with 2048 entries, a 300×300 image requires only 0.24 second on average for processing (including dense local descriptors extraction, LLC coding and SPM pooling to get the final representation). This efficiency significantly adds to the practical values of LLC for many real applications.

The remainder of the paper is organized as follows: Section 2 introduces the basic idea of the LLC coding and lists its attractive properties for image classification; Section 3 presents an approximation method to allow fast LLC computation; In Section 4, an incremental training algorithms is proposed to construct the codebook used by LLC; In Section 5, image classification results on three widely used datasets are reported; and Finally in Section 6, conclusions are made, and some future research issues are discussed.

2. Locality-constrained Linear Coding

Let $\mathbf{X}$ be a set of D-dimensional local descriptors extracted from an image, *i.e*. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N] \in \mathbb{R}^{D\times N}$. Given a codebook with M entries, $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_M] \in \mathbb{R}^{D\times M}$, different coding schemes convert each descriptor into a M-dimensional code to generate the final image representation. This section reviews two existing coding schemes and introduces our proposed LLC method.

2.1. Coding descriptors in VQ

Traditional SPM uses VQ coding which solves the following constrained least square fitting problem:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{B}\mathbf{c}_i||^2 \qquad (1)$$
$$s.t. \|\mathbf{c}_i\|_{\ell^0} = 1, \|\mathbf{c}_i\|_{\ell^1} = 1, \mathbf{c}_i \succeq 0, \forall i$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N]$ is the set of codes for $\mathbf{X}$. The cardinality constraint $\|\mathbf{c}_i\|_{\ell^0} = 1$ means that there will be only one non-zero element in each code $\mathbf{c}_i$, corresponding to the quantization id of $\mathbf{x}_i$. The non-negative, ℓ^1 constraint $\|\mathbf{c}_i\|_{\ell^1} = 1, \mathbf{c}_i \geq 0$ means that the coding weight for $\mathbf{x}$ is 1. In practice, the single non-zero element is found by searching the nearest neighbor.

2.2. Coding descriptors in ScSPM [22]

To ameliorate the quantization loss of VQ, the restrictive cardinality constraint $\|\mathbf{c}_i\|_{\ell^0} = 1$ in Eq.(1) can be relaxed by using a sparsity regularization term. In ScSPM [22], such a sparsity regularization term is selected to be the ℓ^1 norm of $\mathbf{c}_i$, and coding each local descriptor $\mathbf{x}_i$ thus becomes a standard sparse coding (SC) [16] problem[1]:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{B}\mathbf{c}_i\|^2 + \lambda\|\mathbf{c}_i\|_{\ell^1} \qquad (2)$$

[1]The non-negative constraint in Eq.(1) is also dropped out, because a negative $\mathbf{c}_i$ can be absorbed by flipping the corresponding basis.

The sparsity regularization term plays several important roles: First, the codebook $\mathbf{B}$ is usually over-complete, *i.e.*, $M > D$, and hence ℓ^1 regularization is necessary to ensure that the under-determined system has a unique solution; Second, the sparsity prior allows the learned representation to capture salient patterns of local descriptors; Third, the sparse coding can achieve much less quantization error than VQ. Accordingly, even with linear SVM classifier, ScSPM can outperform the nonlinear SPM approach by a large margin on benchmarks like Caltech-101 [22].

2.3. Coding descriptors in LLC

In this paper, we present a new coding algorithm called Locality-constrained Linear Coding (LLC). As suggested by LCC [24], locality is more essential than sparsity, as locality must lead to sparsity but not necessary vice versa. LLC incorporates locality constraint instead of the sparsity constraint in Eq.(2), which leads to several favorable properties as explained in Subsection 2.4. Specifically, the LLC code uses the following criteria:

$$\min_{\mathbf{C}} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{B}\mathbf{c}_i\|^2 + \lambda \|\mathbf{d}_i \odot \mathbf{c}_i\|^2 \qquad (3)$$
$$s.t.\ \mathbf{1}^\top \mathbf{c}_i = 1,\ \forall i$$

where $\odot$ denotes the element-wise multiplication, and $\mathbf{d}_i \in \mathbb{R}^M$ is the locality adaptor that gives different freedom for each basis vector proportional to its similarity to the input descriptor $\mathbf{x}_i$. Specifically,

$$\mathbf{d}_i = \exp\left(\frac{\text{dist}(\mathbf{x}_i, \mathbf{B})}{\sigma}\right). \qquad (4)$$

where $\text{dist}(\mathbf{x}_i, \mathbf{B}) = [\text{dist}(\mathbf{x}_i, \mathbf{b}_1), ..., \text{dist}(\mathbf{x}_i, \mathbf{b}_M)]^T$, and $\text{dist}(\mathbf{x}_i, \mathbf{b}_j)$ is the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{b}_j$. σ is used for adjusting the weight decay speed for the locality adaptor. Usually we further normalize $\mathbf{d}_i$ to be between $(0, 1]$ by subtracting $\max\big(\text{dist}(\mathbf{x}_i, \mathbf{B})\big)$ from $\text{dist}(\mathbf{x}_i, \mathbf{B})$. The constraint $\mathbf{1}^T\mathbf{c}_i = 1$ follows the shift-invariant requirements of the LLC code. Note that the LLC code in Eqn. 3 is not sparse in the sense of ℓ^0 norm, but is sparse in the sense that the solution only has few significant values. In practice, we simply threshold those small coefficients to be zero.

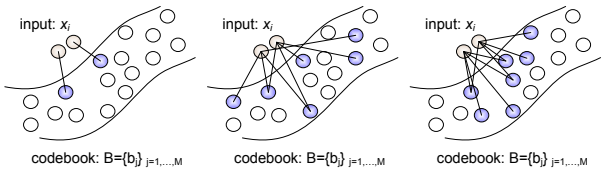
2.4. Properties of LLC

To achieve good classification performance, the coding scheme should generate similar codes for similar descriptors. Following this requirement, the locality regularization term $\|\mathbf{d}_i \odot \mathbf{c}_i\|^2$ in Eq.(3) presents several attractive properties:

1. **Better reconstruction.** In VQ, each descriptor is represented by a *single* basis in the codebook, as illustrated in Fig. 2.a. Due to the large quantization errors, the VQ code for similar descriptors might be very different. Besides, the VQ process ignores the relationships between different bases. Hence non-linear kernel projection is required to make up such information loss. On the other side, as shown in Fig. 2.c in LLC, each descriptor is more accurately represented by multiple bases, and LLC code captures the correlations between similar descriptors by sharing bases.
2. **Local smooth sparsity.** Similar to LLC, SC also achieves less reconstruction error by using multiple bases. Nevertheless, the regularization term of ℓ^1 norm in SC is not smooth. As shown in Fig. 2.b, due to the over-completeness of the codebook, the SC process might select quite different bases for similar patches to favor sparsity, thus losing correlations between codes. On the other side, the explicit locality adaptor in LLC ensures that similar patches will have similar codes.
3. **Analytical solution.** Solving SC usually requires computationally demanding optimization procedures. For instance, the Feature Sign algorithm utilized by Yang *et al.* [22] has a computation complexity of $\mathcal{O}(M \times K)$ in the optimal case [16], where K denotes the number of non-zero elements. Unlike SC, the solution of LLC can be derived analytically by:

$$\tilde{\mathbf{c}}_i = \big(\mathbf{C}_i + \lambda \text{diag}(\mathbf{d})\big) \setminus \mathbf{1} \qquad (5)$$
$$\mathbf{c}_i = \tilde{\mathbf{c}}_i / \mathbf{1}^\top \tilde{\mathbf{c}}_i, \qquad (6)$$

where $\mathbf{C}_i = (\mathbf{B} - \mathbf{1}\mathbf{x}_i^\top)(\mathbf{B} - \mathbf{1}\mathbf{x}_i^\top)^\top$ denotes the data covariance matrix. As seen in Section 3, the LLC can be performed very fast in practice.



VQ SC LLC

Figure 2. Comparison between VQ, SC and LLC. *The selected basses for representation are highlighted in red*

3. Approximated LLC for Fast Encoding

The LLC solution only has a few significant values, or equivalently, solving Eq.(3) actually performs feature selection: it selects the *local bases* for each descriptor to form a *local coordinate system*. This suggests that we can develop an even faster approximation of LLC to speedup the encoding process. Instead of solving Eq.(3), we can simply use the K ($K < D < M$) nearest neighbors of $\mathbf{x}_i$ as the local bases $\mathbf{B}_i$, and solve a much smaller linear system to get the

codes:

$$\min_{\tilde{\mathbf{C}}} \sum_{i=1}^{N} ||\mathbf{x}_i - \tilde{\mathbf{c}}_i \mathbf{B}_\mathbf{i}||^2 \quad (7)$$
$$st.\ \mathbf{1}^\top \tilde{\mathbf{c}}_i = 1,\ \forall i.$$

This reduces the computation complexity from $\mathcal{O}(M^2)$ to $\mathcal{O}(M + K^2)$, where $K \ll M$. The final implementation of such approximated LLC process is illustrated in Figure.1 right. Though this approximated encoding appears to be similar to Local Linear Embeding [18], the whole procedure of LLC itself differs from LLE clearly, because LLC incorporates an additional codebook learning step where the inference is derived from Eq.(3). The codebook learning step will be further explained in Section 4.

As K is usually very small, solving Eq.(7) is very fast. For searching K-nearest neighbors, we applied a simple but efficient hierarchical K-NN search strategy, where each descriptor is first quantized into one of L subspaces, and then in each subspace an $\mathbb{R}^{M\times D}$ codebook was applied. The effective size of the codebook becomes $L \times M$. In this way, a much larger codebook can be used to improve the modeling capacity, while the computation in LLC remains almost the same as that in using a single $\mathbb{R}^{M\times D}$ codebook.

4. Codebook Optimization

In all the above discussions, we have assumed that the codebook is given. A simple way to generate the codebook is to use clustering based method such as K-Means algorithm [15]. According to our experimental results in Subsection 5.4, the codebook generated by K-Means can produce satisfactory accuracy. In this work, we use the LLC coding criteria to train the codebook, which further improves the performance. In this section, we present an effective on-line learning method for this purpose.

Revisiting Eq.(3), now we seek to factorize each training descriptor into the product of an LLC code and a codebook. Hence an optimal codebook B^* can be obtained by

$$\arg\min_{\mathbf{C},\mathbf{B}} \sum_{i=1}^{N} ||\mathbf{x}_i - \mathbf{B}\mathbf{c}_i||^2 + \lambda ||\mathbf{d}_i \odot \mathbf{c}_i||^2 \quad (8)$$
$$st.\ \mathbf{1}^\top \mathbf{c}_i = 1,\ \forall i \quad (9)$$
$$||\mathbf{b}_j||^2 \leq 1,\ \forall j$$

Eq.(8) can be solved using Coordinate Descent method to iteratively optimizing $\mathbf{C}(\mathbf{B})$ based on existing $\mathbf{B}(\mathbf{C})$. However, in practice, the number of training descriptors N is usually large (*e.g.*, 2,000,000+ in our experiment), such that holding all the LLC codes together in each iteration is too memory consuming. Hence we apply an on-line method that reads a small batch of descriptors $\mathbf{x}$ at a time and incrementally update the codebook $\mathbf{B}$.

To elaborate, we first use a codebook trained by K-Means clustering to initialize $\mathbf{B}$. Then we loop through all the training descriptors to update $\mathbf{B}$ incrementally. In each iteration, we take in a single examples $\mathbf{x}_i$ (or a small batch of them), and solve Eq.(3) to obtain the corresponding LLC codes using current $\mathbf{B}$. Then, as explained in Section 3, we regards this step as a feature selector, *i.e*. we only keep the set of basis $\mathbf{B}_i$ whose corresponding weights are larger than a predefined constant, and refit $\mathbf{x}_i$ without the locality constraint. The obtained code is then used to update the basis in a gradient descent fashion. Finally, we project those basis outside the unit circle onto the unit circle. The above process is illustrated in Alg.4.1.

Algorithm 4.1 Incremental codebook optimization

input: $\mathbf{B}_{init} \in \mathbb{R}^{D\times M}$, $\mathbf{X} \in \mathbb{R}^{D\times N}$, λ, σ
output: $\mathbf{B}$

$\mathbf{B} \leftarrow \mathbf{B}_{init}$.
for i = 1 to N **do**
  $\mathbf{d} \leftarrow 1 \times M$ zero vector,
  {locality constraint parameter}
  for j = 1 to M **do**
    $d_j \leftarrow \exp^{-1}\big(-\|\mathbf{x}_i - \mathbf{b}_j\|^2/\sigma\big)$.
  end for
  $\mathbf{d} \leftarrow normalize_{(0,1]}(\mathbf{d})$
  {coding}
  $\mathbf{c}_i \leftarrow \text{argmin}_\mathbf{c} ||\mathbf{x}_i - \mathbf{B}\mathbf{c}||^2 + \lambda||\mathbf{d} \odot \mathbf{c}||^2$
    s.t. $\mathbf{1}^\top\mathbf{c} = 1$.
  {remove bias}
  $id \leftarrow \{j|abs\big(\mathbf{c}_i(j)\big) > 0.01\}$, $\mathbf{B}_i \leftarrow \mathbf{B}(:, id)$,
  $\tilde{\mathbf{c}}_i \leftarrow \text{argmin}_\mathbf{c} ||\mathbf{x}_i - \mathbf{B}_i\mathbf{c}||^2$
    s.t. $\sum_j \mathbf{c}(j) = 1$.
  {update basis}
  $\Delta\mathbf{B}_i \leftarrow -2\tilde{\mathbf{c}}_i(\mathbf{x}_i - \mathbf{B}_i\tilde{\mathbf{c}}_i)$, $\mu \leftarrow \sqrt{1/i}$,
  $\mathbf{B}_i \leftarrow \mathbf{B}_i - \mu\Delta\mathbf{B}_i/|\tilde{\mathbf{c}}_i|_2$,
  $\mathbf{B}(:, id) \leftarrow proj(\mathbf{B}_i)$.
end for

5. Experimental Results

In this section, we report results based on three widely used datasets: Caltech-101 [7], Caltech-256 [11], and Pascal VOC 2007 [6]. We used only a single descriptor, the Histogram of Oriented Gradient (HOG) [5], throughout the experiment. In our setup, the HOG features were extracted from patches densely located by every 8 pixels on the image, under three scales, 16×16, 25×25 and 31×31 respectively. The dimension of each HOG descriptor is 128. During LLC processing, only the approximated LLC was used, and the number of neighbors was set to 5 (Section 3) with the shift-invariant constraint.

In the "SPM" layer, for each spatial sub-region, the codes

of the descriptors (e.g., VQ codes, SC codes or LLC codes) are pooled together to get the corresponding pooled feature. These pooled features from each sub-region are concatenated and normalized as the final image feature representation. Specifically two pooling methods have been used

- sum pooling [15]:: $\mathbf{c}_{out} = \mathbf{c}_{in1}+, ..., +\mathbf{c}_{in2}$
- max pooling [22]: $\mathbf{c}_{out} = \max(\mathbf{c}_{in1}, ..., \mathbf{c}_{in2})$

where "max" functions in a row-wise manner, returning a vector the same size as $\mathbf{c}_{in1}$. These pooled features can then normalized by

- sum normalization: $\mathbf{c}_{out} = \mathbf{c}_{in}/\sum_j \mathbf{c}_{in}(j)$
- ℓ^2 normalization: $\mathbf{c}_{out} = \mathbf{c}_{in}/\|\mathbf{c}_{in}\|_2$

Different combinations can be explored, e.g., "sum pooling" followed by "sum normalization" with VQ codes produces the histogram. In our LLC framework, we used "max pooling" combined with "ℓ^2 normalization" as in [22].

All the experiments were conducted on a Dell PowerEdge 1950 server with 16G memory and 2.5Ghz Quad Core CPU.

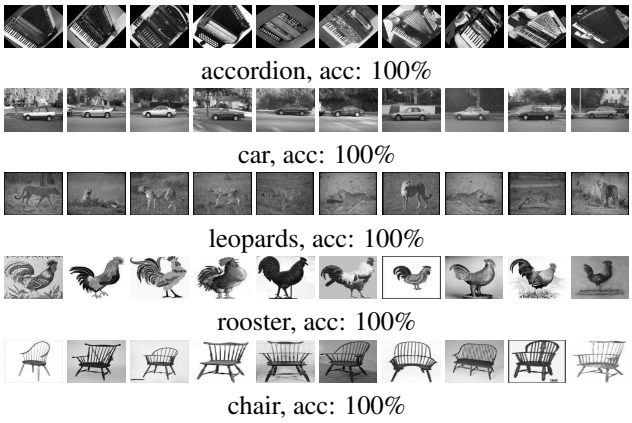
5.1. Caltech-101

The Caltech-101 dataset [7] contains 9144 images in 101 classes including animals, vehicles, flowers, etc, with significant variance in shape. The number of images per category varies from 31 to 800. As suggested by the original dataset [7] and also by many other researchers [25, 11], we partitioned the whole dataset into 5, 10, ..., 30 training images per class and no more than 50 testing images per class, and measured the performance using average accuracy over 102 classes (*i.e*. 101 classes and a "background" class). We trained a codebook with 2048 bases, and used 4×4, 2×2 and 1×1 sub-regions for SPM. In the experiment, the images were resized to be no larger than 300×300 pixels with preserved aspect ratio.

In our evaluation, totally 13 classes achieve 100% classification accuracy with 30 training image per class. Figure 5.1 illustrates five out of these 13 classes that have more than 10 testing images. We compared our result with several existing approaches. Detailed results are shown in Table 1, and it can be seen that in most cases, our proposed LLC method leads the performance. In addition, the average processing time for our method in generating the final representation from a raw image input is only 0.24 second.

Table 1. Image classification results on Caltech-101 dataset

training images	5	10	15	20	25	30
Zhang [25]	46.6	55.8	59.1	62.0	-	66.20
Lazebnik [15]	-	-	56.40	-	-	64.60
Griffin [11]	44.2	54.5	59.0	63.3	65.8	67.60
Boiman [2]	-	-	65.00	-	-	70.40
Jain [12]	-	-	61.00	-	-	69.10
Gemert [8]	-	-	-	-	-	64.16
Yang [22]	-	-	**67.00**	-	-	73.20
Ours	**51.15**	**59.77**	65.43	**67.74**	**70.16**	**73.44**



5.2. Caltech-256

The Caltech-256 dataset holds 30,607 images in 256 categories. It presents much higher variability in object size, location, pose, etc than in Caltech-101. Each class contains at least 80 images. Similar to Subsection 5.1, we resized the images to be less than 300×300 pixels with aspect ratio kept. We followed the common setup during experiment, *i.e*., we tried our algorithm on 15, 30, 45, and 60 training images per class respectively. We trained a codebooks with 4096 bases, and used 4×4, 2×2 and 1×1 sub-regions for SPM. As can be seen from Table 2, the results are very impressive: under all the cases, our LLC method outperforms the best of the existing techniques by more than 6 percent. Besides, the average time in processing each image is 0.3 second. Figure 5.2 lists 20 classes that are easiest to be classified using 60 training images per class.

Table 2. Image classification results using Caltech-256 dataset

training images	15	30	45	60
Griffin [11]	28.30	34.10	-	-
Gemert [8]	-	27.17	-	-
Yang [22]	27.73	34.02	37.46	40.14
Ours	**34.36**	**41.19**	**45.31**	**47.68**

5.3. Pascal VOC 2007

The PASCAL 2007 dataset [6] consists of 9,963 images from 20 classes. These images range between indoor and outdoor scenes, close-ups and landscapes, and strange viewpoints. The dataset is an extremely challenging one because all the images are daily photos obtained from Flicker where the size, viewing angle, illumination, etc appearances of objects and their poses vary significantly, with frequent occlusions (Figure 5.3).

The classification performance is evaluated using the Average Precision (AP) measure, a standard metric used by PASCAL challenge. It computes the area under the Precision/Recall curve, and the higher the score, the better the performance. In Table 3, we list our scores for all 20 classes

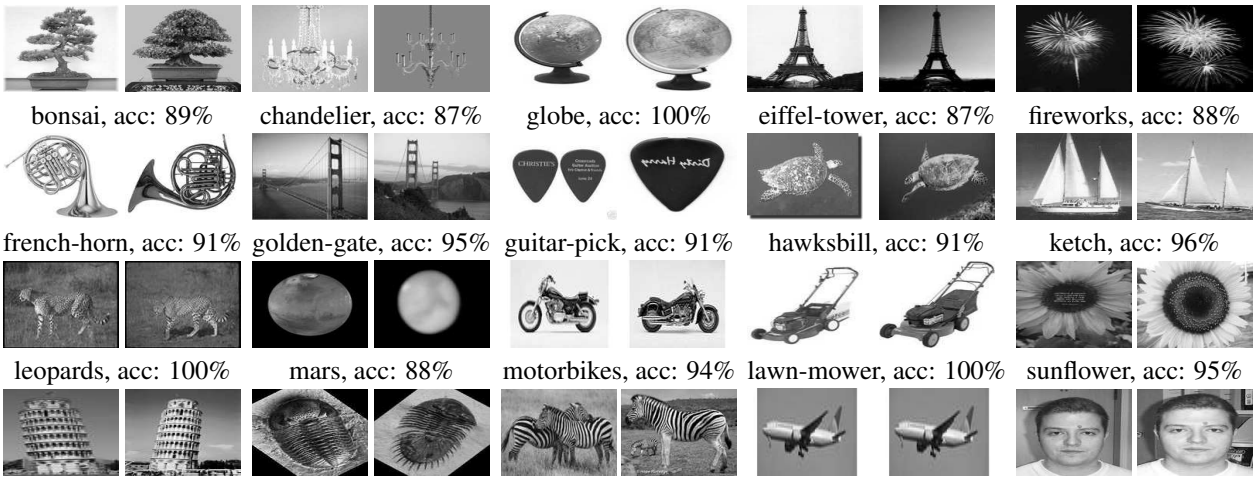


Figure 3. Example images from classes with highest classification accuracy from the Caltech-256 dataset

in comparison with the best performance of the 2007 challenge [6], as well as another recent results in [20]. To make fair comparison, two noteworthy comments need to be made:

1. In [6], multiple descriptors were used with nonlinear classifier in the winner system.
2. In [20], multiple decision systems were combined to get the final results.
3. For our LLC algorithm, we only used single descriptor (HoG) and simple linear SVM as the classifier.

Even though, as seen from Table 3, our LLC method can still achieve similar performance as the best system in [6] and [20]. In fact, if only dense SIFT (similar to dense HOG in our framework) was used, the best result on the validation set only scored 51.8% in [6], while our method achieved 55.1%, winning a remarkable margin of 3.3%.

5.4. Discussion

To provide more comprehensive analysis of the proposed LLC method, we further evaluated its performance with respect to codebook training, number of neighbors for approximation, various constraints as mentioned in Subsection 2.4, and different types of pooling and normalizing during the "SPM" step. The paper mainly reports the results using the Caltech-101 dataset, and our experiments have shown that the conclusions can be generalized to other dataset as well.

First, we compared the classification accuracies using codebooks trained by K-Means algorithm and by our proposed Alg.4.1. In applying our algorithm, the two related parameters λ and σ in Eq.(4) and Eq.(8) were carefully selected such that the cardinality, *i.e.* the length of id in line 8 of Alg.4.1, could match the number of neighbors used during classification. Finally $\lambda = 500$ and $\sigma = 100$ were used, and the cardinality ranges between $4 \sim 34$ during training. We generated two codebooks with 1024 and 2048 entries for each method respectively. Then the same experimental setup as that reported in Subsection 5.1 were used, and the results are plotted in Figure 5. As shown, under all different numbers of training samples per class, the learned codebook by Alg.4.1 improved the classification accuracy by $0.3 \sim 1.4$ percent over the codebook by K-Means.

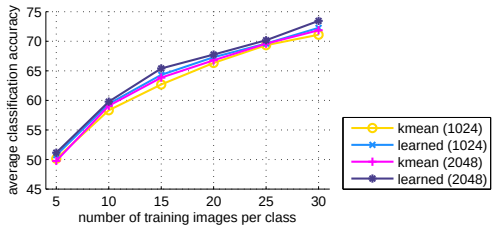


Figure 5. Performance of codebook (Caltech-101)

Second, we studied the effect of different number of neighbors K used for approximated LLC. Figure 6 lists the performance using 2, 5, 10, 20 and 40 neighbors respectively. As can be seen, generally small number of neighbors leads to better classification accuracy. This is desirable, because the smaller the number of neighbors used, the faster the computation goes, and the less the memory consumed. However, when K goes smaller than 5, the performance starts to drop.

Third, we tested the performance under different constraints other than the shift-invariant constraint in Eq. (3) on the LLC code. Other constraints tested were

- Unconstrained.
- Non-negative constraint.
- Non-negative shift-invariant constraint.

Eq.(7). The non-negative constraint is solved using the algorithm in [9]. During our experiments, it is noticed that,

Figure 4. Example images from Pascal VOC 2007 dataset. *A * in the class name means that our method outperforms others*

Table 3. Image classification results using Pascal VOC 2007 dataset

object class	aero	bicyc	bird	boat	bottle	bus	car	cat	chair	cow
Obj.+Contex [20]	**80.2**	61.0	49.8	69.6	21.0	66.8	**80.7**	51.1	51.4	35.9
Best PASCAL'07 [6]	77.5	63.6	**56.1**	**71.9**	**33.1**	60.6	78.0	58.8	53.5	42.6
Ours	74.8	**65.2**	50.7	70.9	28.7	**68.8**	78.5	**61.7**	**54.3**	**48.6**

object class	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	average
Obj.+Contex [20]	**62.0**	38.6	69.0	61.4	84.6	28.7	**53.5**	**61.9**	**81.7**	**59.5**	58.4
Best of PASCAL'07 [6]	54.9	**45.8**	**77.5**	64.0	**85.9**	**36.3**	44.7	50.9	79.2	53.2	59.4
Ours	51.8	44.1	76.6	**66.9**	83.5	30.8	44.6	53.4	78.2	53.5	59.3

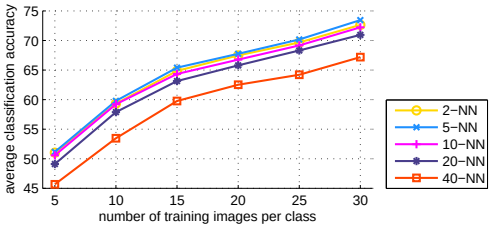


Figure 6. Performance under different neighbors (Caltech-101)

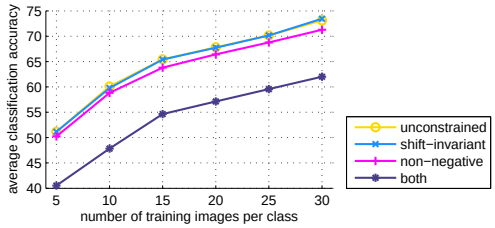


Figure 7. Performance under different constraint (Caltech-101)

since K is small, solving Eq.(7) with different constraints doesn't require significantly different computation. As illustrated in Figure.7, the shift-invariant constraint leads to the best performance. The non-negative constraint, either used alone or together with the shift-invariant constraint, decreases the performance dramatically.

Finally, we evaluated the types of pooling and normalization used in the "SPM" layer. As can be observed from Figure.8, the best performance is achieved by the "max pooling" and "l_2 normalization" combination. The good performance of "max pooling" is in consistence with [22]. For the "l_2 normalization", it makes the inner product of any vector with itself to be one, which is desirable for linear kernels.

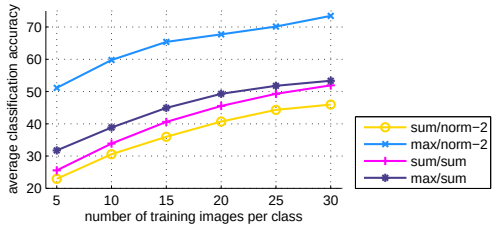


Figure 8. Performance with different SPM setting (Caltech-101)

6. Conclusion and Future Work

This paper presents a promising image representation method called Locality-constrained Linear Coding (LLC). LLC is easy to compute and gives superior image classi-

fication performance than many existing approaches. LLC applies locality constraint to select similar basis of local image descriptors from a codebook, and learns a linear combination weight of these basis to reconstruct each descriptor. The paper also introduces an approximation method to further speed-up the LLC computation, and an optimization method to incrementally learn the LLC codebook using large-scale training descriptors. Experimental results based on several well-known dataset validate the good performance of LLC.

The future work includes the following issues: First, additional codebook training methods, such as supervised training, will be investigated; Second, besides exact-nearest-neighbor search applied in the paper, approximated-nearest-neighbor search algorithms will be evaluated to further improve the computational efficiency of approximated LLC; And third, integration of the LLC technique into practical image/video search/retrieval/indexing/management applications is on-going.

References

[1] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. *Proc. of CVPR'05*, pages 26–33.

[2] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. *Proc. of CVPR'08*, 2008.

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification using a hybrid generative/dicriminative approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2008.

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Proc. of CVPR'05*, pages 886–893, 2005.

[6] M. Everingham, L. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[7] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *In IEEE CVPR Workshop on Generative-Model Based Vision*, 2004.

[8] J. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel codebooks for scene categorization. *Proc. of ECCV'08*, pages 696–709, 2008.

[9] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, pages 1–33, 1983.

[10] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. *Proc. of ICCV'05*, 2005.

[11] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. (7694), 2007.

[12] P. Jain, B. Kullis, and K. Grauman. Fast image search for learned metrics. *Proc. of CVPR'08*, 2008.

[13] F. Jurie and B. Triggs. Creating efficient codebooks for visual recognition. *Proc. of ICCV'05*, 2005.

[14] S. Lazebnik, C. Schmid, and J. Ponce. A maximum entropy framework for part-based texture and object recognition. *Proc. of ICCV'05*, 2005.

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Proc. of CVPR'06*, 2006.

[16] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *Advances in Neural Information Processing Systems, MIT Press*, pages 801–808, 2007.

[17] F. Moosmann, B. Triggs, and F. Jurie. Randomized clustering forests for building fast and discriminative visual vocabularies. *Proc. of NIPS'07*, 2007.

[18] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, pages 2323–2326, 2000.

[19] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. *Proc. of ICCV'03*, pages 1470–1477, 2003.

[20] J. Uijlings, A. Smeulders, and R. Scha. What is the spatial extent of an object? *Proc. of CVPR'09*, 2009.

[21] J. Wang, S. Zhu, and Y. Gong. Resolution enhancement based on learning the sparse association of image patches. *PatternRecognition Lett. (2009) doi:10.1016/j.patrec.2009.09.004*.

[22] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *Proc. of CVPR'09*, 2009.

[23] L. Yang, R. Jin, R. Sukthankar, and F. Jurie. Unifying discriminative visual codebook generation with classifier training for object category recognition. *Proc. of CVPR'08*, 2008.

[24] K. Yu, T. Zhang, and Y. Gong. Nonlinear learning using local coordinate coding. *Proc. of NIPS'09*, 2009.

[25] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest heighbor classification for visual category recognition. *Proc. of CVPR'06*, 2006.

[26] X. Zhou, N. Cui, Z. Li, F. Liang, and T. Huang. Hierarchical gaussianization for image classification. *Proc. of ICCV'09*, 2009.