Efficient Feature Coding Based on Auto-encoder Network for Image Classification

Guo-Sen Xie(✉), Xu-Yao Zhang, and Cheng-Lin Liu

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
{guosen.xie,xyz,liucl}@nlpr.ia.ac.cn

Abstract. Local descriptor coding is one crucial step in traditional Bag of Words (BoW) framework for image categorization. However, the slow coding speed of previous methods is one limitation for applications in large scale problems. Recently, neural network based models have been widely applied in various classification tasks. Using neural network models for descriptor coding is straightforward and efficient due to their fast forward propagation. In this paper, we propose to use the Auto-Encoder (AE) network as a local descriptor coding block, and further embed AE network in the BoW framework for the purpose of image classification. To make the hidden activities of AE network to be both selective and sparse, we add an efficient and effective regularization term into the learning process of AE network, which can promote sparsity of the hidden layer for each input descriptor as well as the selectivity for each hidden node. By incorporating the AE network coding with the BoW framework, we can achieve better results and faster speeds than other state-of-the-art feature coding methods on Caltech101, Scene15 and UIUC 8-Sports databases.

1 Introduction

Image classification is one basic and challenging vision task which aims at classifying images into correct categories. The original Bag of Words (BoW) model is derived from the document retrieval field, and has been successfully applied to image classification [1] with state-of-the-art performance [2–5]. An unified framework of BoW model consists of several steps including (a) local feature (e.g., SIFT, HOG, SURF, etc.) extraction [6–8], (b) dictionary learning and feature coding, (c) pooling (Max, Average, and Sum) of the coded features [2–4], and (d) classifier learning (e.g., one-vs-all SVMs). To incorporate spatial information into the BoW framework, Lazebnik et al. [9] proposed the spatial pyramid matching (SPM) model to improve the performance which is usually denoted as BoW+SPM.

Local descriptor coding is an essential step in the BoW framework, which has drawn great attention in recent years. Given the codebook (dictionary), descriptor coding can be seen as the process of activating a small number of codewords (based on the coding process of the descriptor), which can then generate a coding

D. Cremers et al. (Eds.): ACCV 2014, Part I, LNCS 9003, pp. 628–642, 2015.
DOI: 10.1007/978-3-319-16865-4_41

vector with the same size (dimension) of the codebook [10]. One class of local descriptor coding methods is the reconstruction based model, which is usually designed to minimize the norm (distance) of the descriptors and a linear combination of codewords (defined as reconstruction error), along with constraints on the coding vectors. For example, besides the reconstruction term, sparse coding (SC) [2] adds sparsity constraint on the coding vectors, and locality-constrained linear coding (LLC) [3] incorporates local neighborhood based constraints. Moreover, voting based coding [1,4,11] can also be viewed as reconstruction based models with some special constraints on the coding vector.

The optimization formulations of the above models are all based on minimization of the reconstruction error and subjection to various kinds of constraints on the final coding vectors. However, they deal with different descriptors independently. To further improve the performance, Gao et al. [12] proposed the Laplacian sparse coding (LaSC) which incorporates Laplacian constraints of the descriptors into the sparse coding model. Another class of models considering the relationships of descriptors are based on the context information of descriptors [13–15].

Recently, the development of deep neural networks (DNNs) has triggered many interests of using neural network (NN) models for automatical feature learning in image classification, e.g., the convolutional neural network (CNN) [16] and the restricted Boltzmann machines (RBM) [17] models. The learned weights of the neural networks can also be viewed as the codebook when compared with traditional coding methods. One obvious advantage of NN based feature learning is the fast speed of descriptor coding after learning the weights. Goh et al. [18] proposed a feature learning model based on regularized RBM. The regularizer used there can guarantee the sparsity of hidden activates for each input descriptor, and meanwhile the selectivity of one hidden node for a batch of input descriptors. Sohn et al. [19] also advocated complex sparse convolutional RBM for feature learning.

Considering the slow learning process of CNN and complex contrastive divergence (CD) [20] algorithm used in the RBM training process, in this paper, we explore to use another neural network for local descriptor learning, namely the Auto-Encoder (AE) network [21]. The training of AE network is based on an efficient accessible back propagation (BP) algorithm [22], and we also use an improved version of regularizer proposed by [18] to constrain the learning process of AE network. It is obvious that the regularized AE network is also in the class of reconstruction based coding method. Compared with traditional BoW models as mentioned above, the selectivity of hidden node in the AE network can be explained as one kind of context constraint derived from one batch descriptors, which can bring superior performance for image classification.

The contributions of this paper are: (1) Integrating AE network based feature learning into BoW+SPM framework (Fig. 1) which is both efficient and effective. To the best of our knowledge, this is the first trial to integrate AE network into BoW framework. (2) Incorporating the improved regularizer proposed by [18] into the AE network learning process. (3) Extensive experiments on Caltech101, Scene15, and UIUC 8-Sports demonstrate the effectiveness of the proposed method.

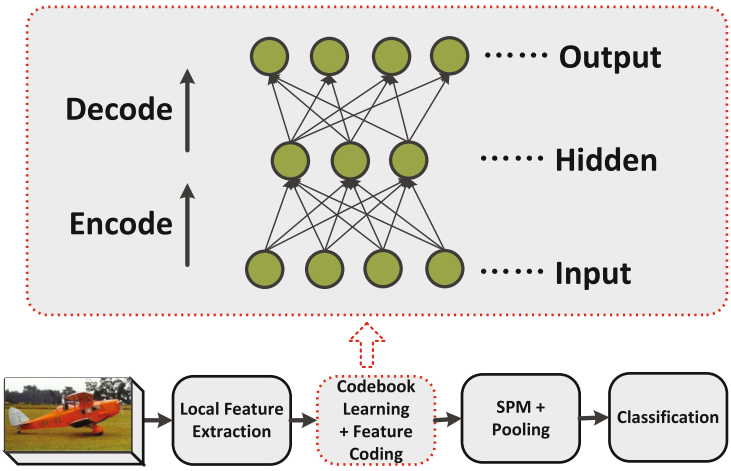

Fig. 1. The hybrid BoW+SPM framework, based on regularized AE network learning for local descriptor coding. After learning the weights and biases of the AE network, the hidden activates are coding vectors of descriptors.

The rest of this paper is organized as follows: The related work about descriptor coding are presented in Sect. 2. The proposed regularized AE network is detailed in Sect. 3. Experimental results, influence of parameter settings and comparison of coding speed are presented in Sect. 4. Section 5 concludes this paper and discusses the future works.

2 Related Work

Descriptor coding methods can be roughly classified into five categories [10]: (i) Voting based methods, e.g., hard assignment coding (HAC) [1], soft assignment coding (SAC) [11] and local soft assignment coding (LSAC) [4]. (ii) Reconstruction based methods, e.g., sparse coding (SC) [2], local coordinate coding (LCC) [23], locality-constrained linear coding (LLC) [3], and so on. (iii) Salient coding (SaC) [5] and group salient coding (GSC) [24]. (iv) Fisher coding [25,26]. (v) Local tangent coding, e.g., super vector coding [27].

Let $X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{d \times N}$ be N $d-$dimensional local features extracted from an image. $B = [b_1, b_2, \cdots, b_M] \in \mathbb{R}^{d \times M}$ denotes the codebook which is usually obtained by k-means algorithm. And $V = [v_1, v_2, \cdots, v_M] \in \mathbb{R}^{M \times N}$ is the coding matrix of these N features, here v_i is the coding vector of x_i, with $i = 1, 2, \cdots, N$. Notations denoted above are used in the whole paper.

The initial BoW model was proposed by [1], where hard assignment was taken to quantize the local descriptors. HAC records the number of local descriptors assigned to each codeword in the codebook B, thus obtaining one M dimensional coding vector for these descriptors. The assignment criteria for descriptor x_i is:

$$v_{i,j} = \begin{cases} 1, & \text{if } j = \arg\min_k \|x_i - b_k\|_2^2, \\ 0, & \text{else.} \end{cases} \tag{1}$$

Different to HAC, SAC [11] assigns weighted value to all codewords of the codebook for each descriptor. An improved version of SAC is localized SAC (LSAC) [4] which only assigns value to neighbor codewords of the descriptor. When the size M of the codebook is large, the searching of neighbor codewords is time consuming. The performances of HAC, SAC and LSAC are all relative poor when compared with other more complicated models.

Besides the reconstruction error, SC [2] was proposed to add another sparse constraint onto the coding vector v. SC can be formulated as an optimization problem:

$$v^* = \arg\min_v \|x - Bv\|_2^2 + \lambda\|v\|_1, \tag{2}$$

where λ controls the tradeoff of the reconstruction and sparsity. We can get the best coding v^* of descriptor x through optimizing Eq. (2). Then LCC [28] was proposed to add locality constraint instead of sparse constraint. Though, the performances of SC and LCC are better than voting based coding, the optimization of them are both time consuming. To speed up, LLC was proposed by [3], which is formulated as follow:

$$v^* = \arg\min_v \|x - Bv\|_2^2 + \lambda\|dist \odot v\|_2, \tag{3}$$

where $\odot$ denotes element-wise product and $dist = [\exp(\|x - b_1\|_2/\delta), \exp(\|x - b_2\|_2/\delta), \cdots, \exp(\|x - b_M\|_2/\delta)]^T \in \mathbb{R}^M$. In Eq. (3), LLC incorporates locality constraint by L2-norm of element-wise product.

To consider the saliency of codewords, salient coding was proposed by [5], wherein relative proximity is used to represent the salient response of the codewords. Given K nearest neighbor codewords of one descriptor, salient coding utilizes the difference between the nearest codeword and the rest $K-1$ neighbor codewords to represent saliency. Recently, group saliency coding (GSC) [24] was also proposed to utilize all the saliency responses of a group of codewords, where the final coding vector of a descriptor on one codeword is the maximum value of all responses under different group sizes [10].

3 Auto-Encoder (AE) Network Based Descriptor Coding

In this paper, the descriptor coding process is based on Auto-Encoder (AE) network [21]. The AE network consists of an encoder process of descriptors and an decoder process of hidden outputs (Fig. 1). The objective is to minimize the reconstruction error and the activate functions of the hidden layer are usually nonlinear, (e.g., sigmoid and tangent), therefore, AE network is a reconstruction based and nonlinear coding process. To further improve the descriptor coding performance, we also add a regularization term in the learning process to guarantee both sparsity and selectivity of the AE network.

3.1 Auto-Encoder Network

We use the notations denoted in Sect. 2 for convenience. The number of nodes for both the input and output layer are d, which is the same as the descriptor dimension. In the BoW framework, we have M codewords in the codebook B, therefore, we also set the number of hidden nodes in the AE network as M, which means the coded vectors of AE network and other BoW models have the same discrete dimensionality (M can be set as 1024, 2048, etc.).

Let $W^{(1)} \in \mathbb{R}^{d\times M}, b^{(1)} \in \mathbb{R}^M$ be the weight and bias of the input-hidden layer, $W^{(2)} \in \mathbb{R}^{M\times d}, b^{(2)} \in \mathbb{R}^d$ be the weight and bias of the hidden-output layer. Figure 1 shows the architecture of the AE network. Suppose we have extracted N descriptors: $X = [x_1, x_2, \cdots, x_N] \in \mathbb{R}^{d\times N}$ from the training images around N interesting points (by dense sampling or interesting points detection). Then the formulation of AE network is:

$$\arg\min_{\Theta} \mathcal{L} = \frac{1}{2N}\sum_{t=1}^{N} \|x_t - h(x_t)\|_2^2 + \frac{\lambda_1}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2), \tag{4}$$

where $\Theta = \{W^{(i)}, b^{(i)} | i = 1, 2\}$ and λ_1 controls the elements of the weights to be small, which can avoid over-fitting. The $h(x_t)$ is the reconstruction of x_t which is defined as

$$h(x_t) = \sigma({W^{(2)}}^\top \sigma({W^{(1)}}^\top x_t + b^{(1)}) + b^{(2)}), \tag{5}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activate function. Then Eq. (4) is optimized by batch gradient decent (BGD) based on BP algorithm [22].

3.2 Sparsity and Selectivity Regularizer with Random Distortion

In the striate complex cells, the selectivity is the response distribution of a neuron across a set of stimuli, and sparsity is the response distribution of several neurons to one single stimulus [29]. In the AE network, each hidden node can be viewed as a neuron and each descriptor can be viewed as a stimuli. To make the AE network to be both sparse and selective, i.e., the output hidden activate vector of one input descriptor is sparse, and the activate vector of one hidden node for the batch descriptors is selective, we should regularize our AE network learning process with sparsity and selectivity just the same as the striate complex cells [29].

To facilitate the next steps of pooling and classification in the hybrid BoW framework (Fig. 1), Goh et al. [18] have proposed one regularizer into the learning process of RBM. In this paper, we also incorporate this regularizer into the AE network. To further improve the performance, we also add random distortion into the constructing of the regularization term.

Let $X_{batch} = [x_1, x_2, \cdots, x_K] \in \mathbb{R}^{d\times K}$ be one batch descriptors during the AE network training, $H = [\sigma({W^{(1)}}^\top x_1 + b^{(1)}), \sigma({W^{(1)}}^\top x_2 + b^{(1)}), \cdots, \sigma({W^{(1)}}^\top x_K + b^{(1)})] \in \mathbb{R}^{M\times K}$ be the forward activates matrix of X_{batch}. The sparsity and selectivity of the hidden activates can be introduced by forcing $\gamma(W^{(1)}, b^{(1)}) = \|H - P\|_F^2 \to 0$ during the AE network learning process, where $P \in \mathbb{R}^{M\times K}$ is

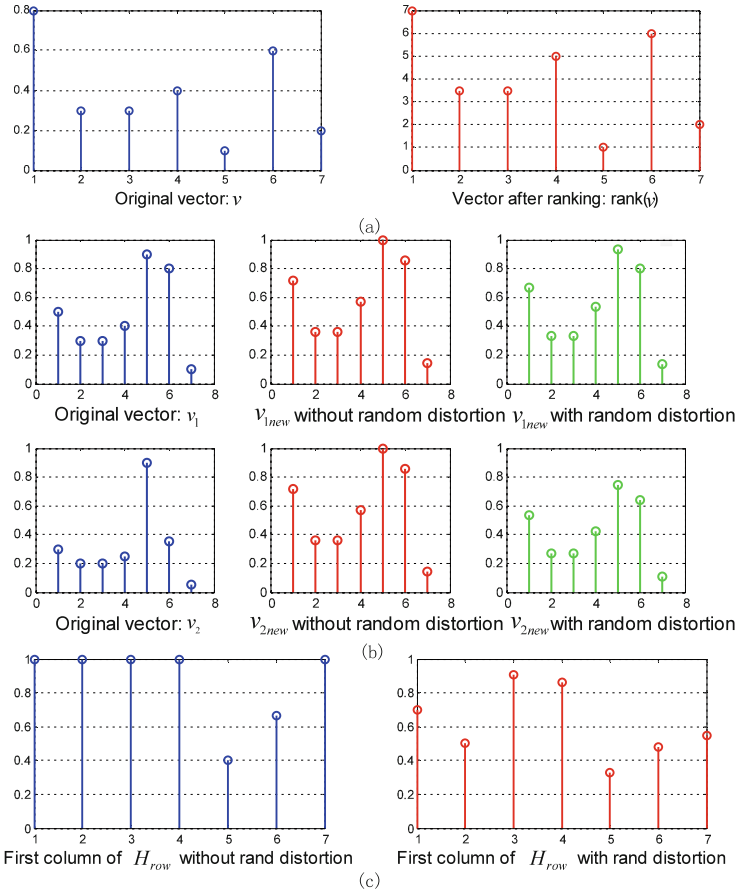

Fig. 2. (a) v and its corresponding rank(v), here v =[0.8 0.3 0.3 0.4 0.1 0.6 0.2], rank(v)=[7 3.5 3.5 5 1 6 2]. (b) $v_i, (i = 1, 2)$ and their normalized ranking vectors, without and with random distortion. It can be seen that $v_{1new} = v_{2new}$ when no distortion added, $v_{1new} \neq v_{2new}$ when distortion added. (c) H_{row} is the matrix obtained by row ranking of H, without or with random distortion. Left: first column of H_{row} without distortion. Right: first column of H_{row} with distortion. It's obvious we get better H_{row} with distortion added.

a matrix constructed based on H forcing both sparsity and selectivity. Then the gradient updating of $W^{(1)}, b^{(1)}$ is modified by adding the increments of $\gamma(W^{(1)}, b^{(1)})$ w.r.t. its parameters during the regularized AE network learning process.

In practice, many choices of penalty satisfying $H \rightarrow P$ can be taken. In order to embed the derivative calculation into the residual error during the back propagation [30], we use another penalty, i.e., KL divergence of H and P instead of the F-norm of them. The KL divergence of H and P is:

Algorithm 1. Construction of P

Input:
Hidden activates: $H \in \mathbb{R}^{M\times K}$, μ_1, μ_2
Output: $P \in \mathbb{R}^{M\times K}$
Process column begins:
$randDistort = rand(1, K)$.
for t $= 1 \rightarrow$ K **do**
  $P(:, t) = rank(H(:, t))$;
  $Max_P = max(P(:, t))$;
  $P(:, t) = P(:, t)/(Max_P + randDistort(t))$;
  $P(:, t) = P(:, t)^{1-1/\mu_1}$;
end for
Process row begins:
$randDistort = rand(1, M)$.
for t $= 1 \rightarrow$ M **do**
  $P(t, :) = rank(P(t, :))$;
  $Max_P = max(P(t, :))$;
  $P(t, :) = P(t, :)/(Max_P + randDistort(t))$;
  $P(t, :) = P(t, :)^{1-1/\mu_2}$;
end for

$$\mathcal{T}(W^{(1)}, b^{(1)}) = \sum_{i=1}^{M}\sum_{j=1}^{K} p_{ij} \log \frac{p_{ij}}{h_{ij}} + (1 - p_{ij}) \log \frac{1 - p_{ij}}{1 - h_{ij}}, \tag{6}$$

where $H = [h_{ij}]^{M\times K}$ and $P = [p_{ij}]^{M\times K}$. It is easy to proof that the derivative of Eq. (6) w.r.t. $W^{(1)}, b^{(1)}$ is the same as the derivative obtained by cross-entropy of H, P, used in [18].

Based on the batch data X_{batch}, the overall optimizing objective function is

$$\mathcal{L} = \frac{1}{2K}\sum_{t=1}^{K} \|x_t - h(x_t)\|_2^2 + \frac{\lambda_1}{2}(\|W^{(1)}\|_F^2 + \|W^{(2)}\|_F^2) + \lambda_2 \mathcal{T}(W^{(1)}, b^{(1)}), \tag{7}$$

where λ_2 controls the tradeoff between sparsity, selectivity and the reconstruction.

Construction of Sparse and Selective Matrix P, based on H. In this Subsection, we present the construction of P.

Based on the initial algorithm in [18], we propose to add random distortion after getting the rank score of the column or row of H. Denote ν as one column or row of H, the distorted and normalized new vector ν_{new} is:

$$\nu_{new} = rank(\nu)/(max(\nu) + \tau * rand), \tag{8}$$

where τ controls the extent of random distortion, $\tau = 1$ in all our experiments. And rank(ν) returns the ranks of the values in ν. If any values in ν are equal, rank(ν) returns their average rank (Fig. 2(a)).

Adding random number into Eq. (8) can enhance the robustness of ν_{new}. On the one hand, when two different vectors ν_1, ν_2 have the same ranking results: $\nu_{1new} = \nu_{2new}$, adding random number can make them also different (Fig. 2(b)), which is more reasonable for afterward processing. On the other hand, after obtaining the ranked matrix H_{row} by ranking all rows of H by Eq. (8), equal value elements will occur in some columns, that is confused for afterward max-pooling operation (Fig. 2(c)). Adding random number in generating H_{row} can avoid this problem.

After ν_{new} is obtained, another step, transforming it into a long-tailed vector is carried out as follow:

$$\nu_{last} = \nu_{new}^{(1-1/\mu)}, \tag{9}$$

where μ controls the shape of the long-tailed vector. Inspired by [18,23], we first carry out column ranking of H, then row ranking. Complete constructing procedure of P is listed in Algorithm 1.

3.3 The Hybrid Algorithm of AE Network with Sparse and Selective Regularization

In this Subsection, we summary the modified BP algorithm used to train the regularized AE network. BP algorithm [22] is usually run on batch data. Based on notations in the above Sections, the batch mode gradient descending procedure on $X_{batch} = [x_1, x_2, \cdots, x_K] \in \mathbb{R}^{d \times K}$ is listed in Algorithm 2. The $\odot$ and $\oslash$ denote element-wise product and division of matrices respectively.

Based on Algorithm 2, we can train the regularized AE network to cover all training data for several iterative epoches. After $(W^{(i)}, b^{(i)}), i = 1, 2$ have been learned, we can use them to do forward propagation to code all local descriptors, e.g., given descriptor $x_i \in \mathbb{R}^d$, its coding vector is

$$v_i = \text{sigmoid}(W^{(1)^T} * x_i + b^{(1)}) \in \mathbb{R}^M.$$

After coding the local descriptors by our trained AE network, $1 \times 1, 2 \times 2, 4 \times 4$ spatial partitions (SPM) [9] are adopted to incorporate spatial information. Then, max pooling on each sub-regions are adopted. By concatenating all the pooled sub-region vectors, we get the final image representations, which are fed into the linear SVM [31] (one-vs-rest) to train classifier.[1]

4 Experiments

In this Section, we first present the experimental settings in Subsect. 4.1. Then Subsect. 4.2 gives classification accuracy and illustrates the impact of different hidden node size on three datasets. Subsection 4.3 shows the impact of μ_1, μ_2 and λ_2. Subsection 4.4 compares the speed of the proposed methods with several other coding methods.

[1] We utilized lib-linear toolkit [32] in this paper for SVM training.

Algorithm 2. One batch Learning of Regularized AE Network

Input:

Batch data: $X_{batch} \in \mathbb{R}^{d \times K}$

$\lambda_1 = 0.0001, \lambda_2, \mu_1, \mu_2, hiddenNodes = M$

$momentum = 0.05, learnRate = 1, layer = 3$

Output: $(W^{(i)}, b^{(i)}), i = 1, 2$

Forward propagation:

$H^{(1)} = X_{batch}$;

for t = 2 → layer **do**

  $H^{(t)} = sigmoid(repmat(b^{(t-1)}, 1, K) + {W^{(t-1)}}^T * H^{(t-1)})$;

end for

Calculate P based on $H^{(2)} \in \mathbb{R}^{M \times K}$

Back propagation:

$D^{(layer)} = -(X_{batch} - H^{(3)}) \odot H^{(3)} \odot (1 - H^{(3)})$;

for t = $layer - 1 \rightarrow 2$ **do**

  $SparsitySelectivity = \lambda_2 * ((1 - P) \oslash (1 - H^{(t)}) - P \oslash H^{(t)})$;

  $D^{(t)} = (W^{(t)} * D^{(t+1)} + SparsitySelectivity) \odot H^{(t)} \odot (1 - H^{(t)})$;

end for

for t = 1 → layer-1 **do**

  $DW^{(t)} = H^{(t)} * {D^{(t+1)}}^T / K$;

  $Db^{(t)} = sum(D^{(t+1)}, 2)/K$;

end for

Gradient updating :

for t = 1 → layer-1 **do**

  $vW^{(t)} = momentum * vW^{(t)} + learnRate * (DW^{(t)} + \lambda_1 * W^{(t)})$;

  $vb^{(t)} = momentum * vb^{(t)} + learnRate * Db^{(t)}$;

  $W^{(t)} = W^{(t)} - vW^{(t)}$;

  $b^{(t)} = b^{(t)} - vb^{(t)}$;

end for

4.1 Experimental Settings

In this paper, three datasets, i.e., Caltech101 [33], Scene15 [9], and UIUC 8-sports [34] are used to validate the proposed model. Images in Caltech101 and Scene15 datasets are resized to be no larger than 300 in height or width, and images in UIUC 8-sports no larger than 400. In all our experiments, single scale (16×16) 128-dim SIFT [6] are densely extracted from all images. The step sizes of extracting SIFT for Caltech101, Scene15 and UIUC 8-sports datasets are 6 pixels, 8 pixels and 4 pixels respectively. 20 k descriptors are used for regularized AE network learning in three datasets.

As for training-test set partition, we randomly select 30 training images per category for training, the rest for testing for Caltech101 dataset. For Scene15 dataset, we follow the partition manner in [9], i.e., randomly select 100 images per category for training, the rest for testing. For UIUC 8-sports dataset [34], 70 training images and 60 test images are randomly chosen from each category. The final result is the mean accuracy and standard deviations, which is based on 5 times experiments with different random partition of the training and test set.

Table 1. Classification rate (%) comparison on Caltech101, Scene15 and 8-Sports datasets.

Algorithms	Caltech101(30train)	Scene15	8-Sports
Hard Assignment [9]	64.60±0.80	81.40±0.50	–
Soft Assignment [4]	72.56±0.65	81.09±0.43	82.04±2.37
Local Soft Assignment [4]	74.21±0.81	82.70±0.39	82.29±1.84
ScSPM [2,12]	73.20±0.50	80.30±0.90	82.74±1.46
LLC [3,4]	73.40	81.53±0.65	81.41±1.84
SaC [5]	–	82.55±0.41	–
GSC [24]	73.4±1.20	83.2±0.4	–
LC-KSVD [35]	73.6	–	–
Ours	**74.24±0.96**	**83.27±0.83**	**85.29±1.49**

During our regularized AE network training, we fix the following parameters, i.e., learning rate = 1, batchsize = 100 or 200, momentum = 0.05, $\lambda_1 = 0.0001$, max iterative epoches ≤ 6. Now, we only have three free parameters: λ_2, μ_1, μ_2.

4.2 Classification Accuracy

Table 1 lists the average accuracies and standard deviations of different models. We can find that our model is better than other state-of-the-art models. Because we did not know the exact parameter settings and the implementation details of the compared models, the results of the compared models are borrowed from the cited references. It can be concluded that our proposed method is superior to other coding methods (reconstruction based coding, voting based coding, saliency coding, etc.) consistently. It is worthy to mention that our model is only based on single scale SIFT feature, while other methods are usually based on multi-scale SIFT features. Moreover, the maximal hidden node size (dictionary size) of our AE network is 3072, while some other methods use much larger dictionary than ours, e.g., GSC [24] uses a dictionary with 8192 codewords to obtain their results in Table 1.

We also explore accuracies under different hidden node size, i.e., the dimensionality of the coding vectors. We list the results on Table 2. It can be seen that different hidden node size has different performance. Best results are obtained with hidden node size 2048 for Caltech101 and 8-Sports datasets. While the best hidden node size is 3072 for scene15 dataset.

4.3 Impact of Hyper-Parameters

In this Subsection, we discuss the impact of the hyper-parameters, i.e., λ_2 in Eq. (7), μ_1, μ_2 in Algorithm 1. To save processing time, we only discuss these parameters on Caltech101 and Scene15 datasets. Figures 3 and 4 present the classification rate change tendency along different λ_2 and μ_1, μ_2.

Table 2. Classification rate (%) comparison on Caltech101, Scene15 and 8-Sports datasets under different size of hidden node.

Hidden node number	Caltech101(30train)	Scene15	8-Sports
1024	72.26±1.53	81.57±0.34	85.13±1.35
2048	**74.24±0.96**	82.61±0.18	**85.29±1.49**
3072	73.62±0.90	**83.27±0.83**	84.92±1.25

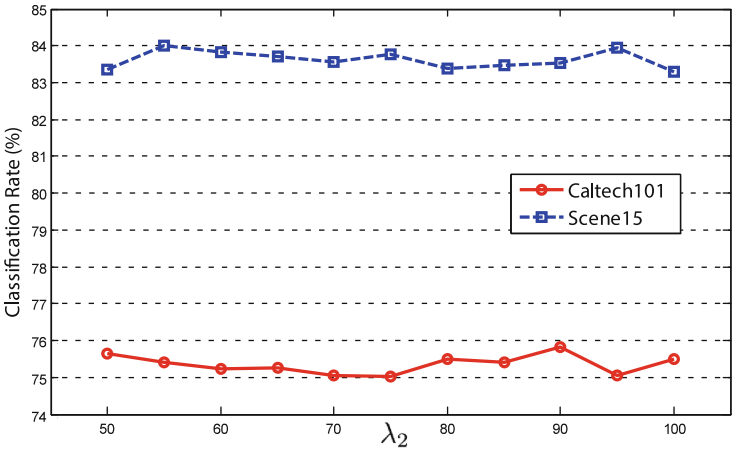

Fig. 3. Changing tendency of the classification rate along λ_2 on Caltech101 and Scene15 datasets (color figure online).

Classification results of one time experiment for Caltech101 and Scene15 datasets are drawn in Fig. 3 (red curve corresponding to Caltech101, blue curve corresponding to Scene15), with training-test set partition the same as the manner in Subsect. 4.1. Here, λ_2 takes $50, 55, \cdots, 100$, other relevant parameters of AE network are fixed. It can be seen that classification performance of our model w.r.t. λ_2 is stable, i.e., the fluctuation of accuracy is less than one percentage with the changing of λ_2.

To visualize the performance of the accuracy w.r.t. μ_1, μ_2, we show the (μ_1, μ_2,accuracy) maps of one time experiment both on Caltech101 and Scene15 datasets. The training-test partition manner is the same as the ones in Subsect. 4.1. The grids of μ_1, μ_2 used to draw the maps in Fig. 4 are in the range of $\mu_1 \in [0.001, 0.1]$ and $\mu_2 \in [0.001, 0.02]$, other parameters are fixed. It can be concluded from Fig. 4, our model is stable w.r.t. μ_1, and best μ_2 is located at interval: $[0.001, 0.006]$ for all datasets.

4.4 Comparison of Coding Speed

In this Subsection, we compare the coding (forward propagation) speed of our method with HAC [1], SAC [11], LSAC [4], SC [2], Approximate SC [2] (first

Table 3. Average single image processing time (second) on Caltech101 dataset under different dictionary size. The numbers in brackets are knn number of corresponding methods. SC(200) is the approximate SC [2] algorithm.

DictionarySize	HAC	SAC	LSAC(5)	SC	SC(200)	LLC(5)	SaC(5)	Ours
1024	0.196	0.047	0.191	1.355	2.007	0.343	0.179	**0.027**
2048	0.351	0.076	0.307	1.885	2.248	0.481	0.301	**0.045**
3072	0.502	0.112	0.449	3.382	2.437	0.721	0.437	**0.050**

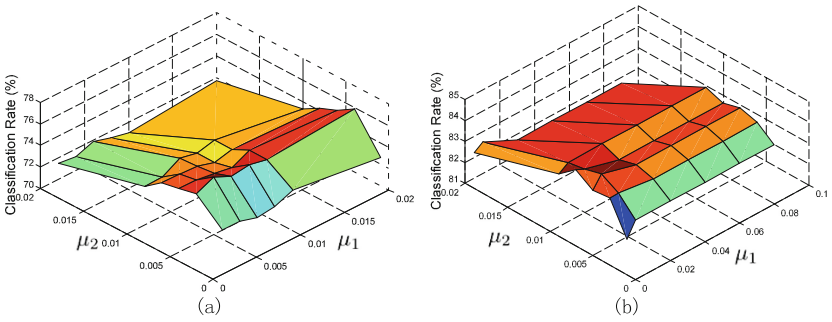

Fig. 4. Changing tendency of the classification rate along μ_1, μ_2 on (a) Caltech101 and (b) Scene15 datasets.

find the neighbor codewords, denoted as $\tilde{B}$, then coding the descriptors based on the sub-codebook $\tilde{B}$ instead of B), LLC [3] and SaC [5]. Average single image processing time is recorded for each method under different codebook (dictionary) size. Here, hidden layer size is the dictionary size for our AE network. We only reported the time on Caltech101 dataset. The average time on other datasets should be similar as Caltech101, because all three datasets have the same image magnitude, i.e., around 300-400 width and height. The experimental platform of comparing coding time is MATLAB8.1.0.604 (R2013a) in a server, with an Intel E5-2670 CPU (2.60 GHz and 16 cores) and 126.1 GB RAM.

To save dictionary learning time of other methods, K-means clustering is used to obtain the needed dictionaries of different size. Meanwhile, the weight $(W^{(1)}, b^{(1)})$ with different hidden node size (the same as dictionary size) should be learned. After that, we randomly select 10 images per class to test the coding speed of all these methods. The compared coding methods are implemented by us based on their public codes.

Results are listed in Table 3. It can be seen that our method is much faster than their counterparts. Moreover, weight learning time of our regularized AE network is also much faster than SC, LLC. In practical AE network learning process, less than six epoches are enough to gain good results, one epoch only takes several minutes. On the contrary, obtaining the dictionary by SC and LLC usually takes several hours or days. The fast inferring speed makes the proposed hybrid model more suitable for real application.

5 Conclusion and Future Work

In this paper, we propose a hybrid framework by combining traditional BoW+ SPM model with the regularized Auto-Encoder (AE) network. We use AE network to learn nonlinear transformation of local descriptors, which can be also seen as codebook learning in the viewpoint of BoW, by viewing the weights $W^{(1)}$ of AE network as the learned dictionary. During the AE network training, a random distortion based sparse and selective regularizer is also incorporated to guarantee the sparsity for each input descriptor and meanwhile the selectivity for each activate node in the hidden layer. After learning the weights $(W^{(1)}, b^{(1)})$ of the AE network, the inference speed (forward propagation) of local descriptors is very fast compared with some other state-of-the-art coding methods (e.g., HAC, SAC, LSAC, ScSPM, LLC and SaC). The classification accuracy of our model also consistently outperforms some recently proposed popular models (ScSPM, LLC, K-SVD, etc.).

In the future, we will consider incorporate discriminative information into regularized AE network and construct deep AE model for better coding of the descriptors. On the other hand, incorporating the relationship information (such as Laplacian constraint) of descriptors into the AE network learning process is also an interesting topic.

Acknowledgement. This work has been supported in part by the National Basic Research Program of China (973 Program) Grant 2012CB316302 and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant XDA06040102).

References

1. Csurka, G., Bray, C., Dance, C., Fan, L.: Visual categorization with bags of keypoints. In: ECCV, Workshop on Statistical Learning in Computer Vision, pp. 1–22 (2004)
2. Yang, J., Yu, K., Gong, Y., Huang, T.S.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR, pp. 1794–1801 (2009)
3. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR, pp. 3360–3367 (2010)
4. Liu, L., Lei, W., Liu, X.: In defense of soft-assignment coding. In: ICCV, pp. 2486–2493 (2011)
5. Huang, Y., Huang, K., Yu, Y., Tan, T.: Salient coding for image classification. In: CVPR, pp. 1753–1760 (2011)
6. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893 (2005)
8. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR, pp. 2169–2178 (2006)

10. Huang, Y., Wu, Z., Wang, L., Tan, T.: Feature coding in image classification: A comprehensive study. IEEE Trans. Pattern Anal. Mach. Intell. **36**, 493–506 (2014)
11. van Gemert, J.C., Geusebroek, J.-M., Veenman, C.J., Smeulders, A.W.M.: Kernel codebooks for scene categorization. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 696–709. Springer, Heidelberg (2008)
12. Gao, S., Tsang, I.W.H., Chia, L.T.: Laplacian sparse coding, hypergraph laplacian sparse coding, and applications. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 92–104 (2013)
13. Galleguillos, C., Rabinovich, A., Belongie, S.: Object categorization using co-occurrence, location and appearance. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
14. Morioka, N., Satoh, S.: Compact correlation coding for visual object categorization. In: ICCV, pp. 1639–1646 (2011)
15. Su, Y., Jurie, F.: Visual word disambiguation by semantic contexts. In: ICCV, pp. 311–318 (2011)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1106–1114 (2012)
17. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**, 504–507 (2006)
18. Goh, H., Thome, N., Cord, M., Lim, J.-H.: Unsupervised and supervised visual codes with restricted boltzmann machines. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part V. LNCS, vol. 7576, pp. 298–311. Springer, Heidelberg (2012)
19. Sohn, K., Jung, D.Y., Lee, H., Hero, A.O.: Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In: ICCV, pp. 2643–2650 (2011)
20. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Comput. **14**, 1771–1800 (2002)
21. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
22. Rumelhart, D., Hintont, G., Williams, R.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
23. Ngiam, J., Koh, P.W., Chen, Z., Bhaskar, S.A., Ng, A.Y.: Sparse filtering. In: NIPS, pp. 1125–1133 (2011)
24. Wu, Z., Huang, Y., Wang, L., Tan, T.: Group encoding of local features in image classification. In: ICPR, pp. 1505–1508 (2012)
25. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
26. Perronnin, F., Dance, C.R.: Fisher kernels on visual vocabularies for image categorization. In: CVPR (2007)
27. Zhou, X., Yu, K., Zhang, T., Huang, T.S.: Image classification using super-vector coding of local image descriptors. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 141–154. Springer, Heidelberg (2010)
28. Yu, K., Zhang, T., Gong, Y.: Nonlinear learning using local coordinate coding. In: NIPS, pp. 2223–2231 (2009)
29. Lehky, S.R., Sejnowski, T.J., Desimone, R.: Selectivity and sparseness in the responses of striate complex cells. Vis. Res. **45**, 57–73 (2005)
30. Ng, A.: Sparse autoencoder. CS294 A Lecture notes (2011)

31. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer-Verlag New York Inc., New York (1995)
32. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
33. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples an incremental bayesian approach tested on 101 object categories. In: Proceedings of the Workshop on Generative-Model Based Vision (2004)
34. Li, L.J., Li, F.F.: What, where and who? classifying events by scene and object recognition. In: ICCV, pp. 1–8 (2007)
35. Jiang, Z., Lin, Z., Davis, L.S.: Label consistent k-svd: Learning a discriminative dictionary for recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**, 2651–2664 (2013)