SINGLE-LAYER UNSUPERVISED FEATURE LEARNING WITH L2 REGULARIZED SPARSE FILTERING

Zhao Yang, Lianwen Jin, Dapeng Tao, Shuye Zhang, Xin Zhang

College of Electronic and Information, South China University of Technology, China
yangdxng100@126.com, lianwen.jin@gmail.com, dapeng.tao@gmail.com

ABSTRACT

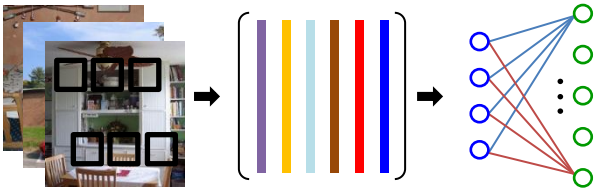
Patch-based Single-layer Unsupervised Feature Learning (SUFL) has been successfully applied in several tasks of computer vision. In the feature learning process, the key ingredient is how to learn a good feature mapping that connects patches to feature vectors. Among various feature mapping methods, the sparse filtering is easy to be implemented and hyper-parameter free. However, the standard sparse filtering method only considers the sparsity distribution of the learned features, ignoring the feature mapping matrix itself. This will lead to a random magnitude for mapping matrix and further weaken the generation performance. In this paper we proposed L2 regularized sparse filtering for the feature mapping in SULF. Classification experiments on three different datasets, i.e., CIFAR-10, small Norb, and subsets of CISIA-HWDB1.0 handwritten characters, show that our method has better performance comparing with the standard sparse filtering.

Index Terms— unsupervised feature learning, single-layer network, sparse filtering, L2 regularization

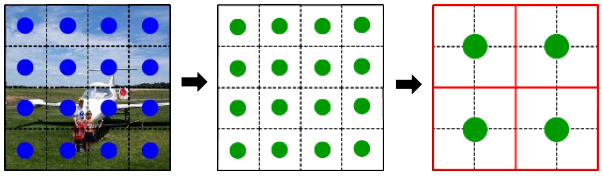
1. INTRODUCTION

Unsupervised feature learning [1][2][3][4] has been extensively used in a wide spectrum of computer vision applications along with the development of deep learning[5][6]. It becomes a promising tool to learning effective feature presentations from unlabeled data in recent years.

In general, unsupervised feature learning is based on two types of framework: deep architecture [1][2][3] and shallow architecture with a single-layer network [4][7]. Though recent researches on deep learning have shown that deep architecture could learn multi-level hierarchies of features and obtain constantly improvements on several benchmark datasets [3][8], the single-layer network is still a favored and commonly used method and draws more and more attentions. There are two reasons for this. First, compared to deep architecture, single-layer network is simple and easy to use [4]. The single-layer network does not rely on complex selection of the hype-parameters and time-consuming fine-tune of the network to get a satisfactory result. Sometimes the single-layer network could obtain comparable performance with the deep architectures [9]. Second, since deep learning [5] was brought out in 2006, it becomes increasing popular to use a single-layer network to build blocks for a deep architecture, resulting in various deep learning methods [10][11]. The performance of the single-layer network influences the whole performance of the deep architecture.



(a) Training stage: patches extraction, pre-processing, and feature mapping learning.



(b) Testing stage: extract features by convolution and pooling operations for a new input image.

Fig. 1. The framework of the single-layer unsupervised feature learning.

Commonly, there are typical pipelines for unsupervised feature learning using single-layer network, as seen in Figure 1. It involves training and testing stages: (a) The random patches extraction, pre-processing for training and feature mapping learning; (b) Convolutional feature extraction and pooling for new images. In above steps, the feature mapping learning is the most important step that generates a mapping matrix from the patches to feature vectors. It determines the performance of the feature. There are many algorithms available for the feature mapping learning, such as sparse restricted Boltzmann machines (RBM) [12], sparse autoencoders [13][14], K-means [9], ICA [15], sparse filtering [16] and so on. In these methods, sparse RBM and sparse auto-encoder have a lot of hyper-parameters to tune; K-means could obtain satisfying results only when the number of the feature is very large; and ICA relies on whitening operation and is difficult to learn over-

complete feature representation. In contrast, sparse filtering is a hyper-parameter free algorithm given various numbers of features with less pre-preprocessing operation.

The sparse filtering works by imposing constraints on sparse distribution of features. However, it does not consider the property of the mapping learning matrix itself. In this paper, we propose a L2 regularized sparse filtering method for unsupervised feature learning. For concreteness, a L2 weight decay constraint item is added to feature learning object function, which tends to decrease the magnitude of the weights in the mapping function and improve the generalization ability. Hence, better weights are learnt for the feature learning. We evaluate our method on three different datasets, i.e., CIFAR-10, small Norb, and subsets of CISIA-HWDB1.0 handwritten characters. Classification experiments show that our proposed method has better performance than the standard sparse filtering.

In the following, we first describe the single-layer unsupervised feature learning in Section 2; Section 3 presents the details of the proposed method, followed by classification experiments in Section 4. Section 5 outlines the conclusion.

2. SINGLE-LAYER UNSUPERVISED FEATURE LEARNING

We first describe the single-layer unsupervised feature learning process whose general pipeline is as follow [4].

1. Collect a set of small patches from the training set randomly and conduct pre-processing.

2. Learn a feature mapping matrix to build a mapping from input patches to feature vectors using certain single-layer network.

3. After the learning process, we can extract features for a new image using the learned mapping matrix by convolution and pooling operations.

2.1. Patch extraction and pre-processing

Let $\mathbf{D}=\{I^{(1)},I^{(2)},...,I^{(m)}\}$, $I^{(i)}\in\mathbb{R}^{n_W\times n_H\times d}$ be the unlabeled images used for training (d is the channel. If the input image is an RGB image, $d=3$, otherwise $d=1$ for a gray image). Given a $w\times w$ receptive field, N patches of size $w\times w\times d$ are extracted at random location from all the images, followed by represented into column vectors. Let $\mathbf{P}=\{p^{(1)},p^{(2)},...,p^{(N)}\}$, $p^{(i)}\in\mathbb{R}^{w\cdot w\cdot d\times 1}$ denote the patches. Generally, in order to get good performance, there are some pre-processing operations, such as the mean value subtraction, contrast normalization, whitening and so on. Here, we use $X=\{x^{(1)},x^{(2)},...,x^{(N)}\}$, $x^{(i)}\in\mathbb{R}^{w\cdot w\cdot d\times 1}$ to denote the pre-processed patches.

2.2. Feature mapping learning

Once we get the patches, a single layer network (only one hidden layer) is trained to learning a function $f:\mathbb{R}^{w\cdot w\cdot d}\to\mathbb{R}^{K}$ that maps a patch $x^{(i)}$ to a new feature vector $f^{(i)}$ of K dimensions using various criterions or algorithms. Many types of methods can be applied for this purpose. For example, RBM is defined by restricting the interactions of input layer and hidden layer in the Boltzmann energy function; Auto-encoder restricts the hidden layer to be a compressed or sparse represent of the input; Sparse filtering works by optimizing the sparsity distribution of the hidden layer. By this, the hidden layer will be a meaningful representation of the original patches. After training, we call the learned weights matrix W of the network as filters, base or feature extractors.

2.3. Convolution and pooling

By the learned weights W, we could compute the feature representation for a new input image. Specifically, the W is applied to conduct convolution operation with every $w\times w$ patch of the input image to yield a feature vector. Formally, we use $f^{i}\in\mathbb{R}^{K}$ to denote the feature of an input patch. We can get a feature mapping image with dimension of $(n_H-w+1)\times(n_W-w+1)\times d$. (We can also extract features convolutionally over the whole image with a larger steps, the situation described above is when the step is equal 1 pixel.)

Typically we need to reduce the dimensionality of the feature represent image by pooling operation while obtaining some invariance. The pooling works by splitting a feature mapping image into four equal-sized quadrants, and summing up or getting the maximum of each quadrant into a vector. This yields a $2\times 2\times d$ feature image. Finally all the vectors are concatenated into a feature vector of dimension $4K$ to represent the image.

3. L2 REGULARIZED SPARSE FILTERING

Sparse filtering is a feature mapping method proposed by Ngiam [16]. It works by optimizing the sparsity distribution of the hidden layer (features) in three principles: population sparsity, lifetime sparsity, high dispersal. For example, given a finite input $\mathbf{X}_{D\times N}=\{x^{(1)},x^{(2)},...,x^{(N)}\}$, with a mapping matrix $\mathbf{W}_{K\times D}$ the output of the network is:

$$\mathbf{F}_{K\times N}=\mathbf{W}_{K\times D}\times\mathbf{X}_{D\times N}=\begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1N} \\ f_{21} & f_{22} & \cdots & f_{2N} \\ \cdots & \cdots & f_{ij} & \cdots \\ f_{K1} & f_{K1} & \cdots & f_{KN} \end{bmatrix} \quad (1)$$

where each column corresponds to the feature of an input sample. First each row is normalized by its L2 norm across the samples $\tilde{\mathbf{f}}_{i\cdot}=\mathbf{f}_{i\cdot}/\left\|\mathbf{f}_{i\cdot}\right\|_2$, then each column is normalized

by its L2 norm $\hat{\mathbf{f}}_{\cdot j} = \tilde{\mathbf{f}}_{\cdot j} / \left\| \tilde{\mathbf{f}}_{\cdot j} \right\|_2$ .The object function of sparse filtering is the sparseness constraints for each example, namely *minimize* $\sum_{i=1}^{N} \left\| \hat{\mathbf{f}}_{\cdot j} \right\|_1$. A more detailed study can be referred in [16].

The above method only considers the constraints of the features' sparsity distribution, ignoring the mapping matrix itself. Thus, we proposed L2 regularized sparse filtering by adding a weight decay item using L2 regularization. By this, the network could learn a better weight matrix and have improved generalization performance for the new data. The optimization problem is defined as followed:

$$\begin{aligned} \textit{minimize } \mathrm{J}(\mathbf{W}) &= \sum_{j=1}^{N} \left\| \hat{\mathbf{f}}_{\cdot j} \right\|_1 + \frac{\lambda}{2} \sum_{i=1}^{D} \sum_{j=1}^{N} w_{ij}^2 \\ &= \sum_{j=1}^{N} \left\| \frac{\tilde{\mathbf{f}}_{\cdot j}}{\left\| \tilde{\mathbf{f}}_{\cdot j} \right\|_2} \right\|_1 + \frac{\lambda}{2} \sum_{i=1}^{D} \sum_{j=1}^{N} w_{ij}^2 \end{aligned} \tag{2}$$

where λ is the regularization parameters, controlling the relative importance of sparsity distribution and weight decay. The objection function can be easily implemented with the off-the-shell minimization method L-BFGS [17].

4. EXPERIMENTS

We evaluated the L2 regularized sparse filtering for classification experiments on three different dataset: CIFAR-10 [18], small Norb [19], subsets of CASIA-HWDB1.0 [20]. Comparative experiments were provided to illustrate better performance of proposed method over the standard sparse filtering. We used the following protocol for all the experiments.

1. We obtained a collection of 100000 patches with certain receive field randomly from the training set, followed by pre-processing using brightness and local contrast normalization to alleviate the variety of colors and brightness:

$$x^{(i)} = \frac{p^{(i)} - mean(p^{(i)})}{\sqrt{\mathrm{var}(p^{(i)}) + \varepsilon}} \tag{3}$$

where ε was used to avoid division by zeros and get some purpose of noise suppression, and it was set as 10 during the whole experiments.

2. The regularization parameter λ was fixed at 0.01, and the objection function was optimized using the L-BFGS package [21] until convergence.

3. For training and test sets, features were extracted convolutionally with every patch and pooled into quadrants as the final feature representation of an image. A Linear SVM was trained used for classification experiments, and different hidden features were evaluated for proper comparison.

4.1. CIFAR-10 classification

The CIFAR-10 dataset [18] is a collection of 32×32 color natural object images, consisting of ten object categories: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. There are 5000 training images and 1000 test images per class. The dataset is challenging due to its low resolution and vast variability, and has been extensive employed to evaluate the performance of feature learning methods. Some example images can be seen in Figure 2(a).

Table 1. Comparative classification accuracy on CIFAR-10.

#Features	100	200	400	800
Sparse filtering	**57.26**	59.97	61.78	63.46
L2 Sparse filtering	57.13	**60.39**	**62.05**	**63.89**

Considering the resolution of the image, we select receptive field size $w = 6$ for feature learning. The learned filters are shown in Figure 3(d), in which each filter corresponds to a row of the weight matrix W . As we can see, they are oriented, localized edge filters. Table 1 shows the classification results with different features. As we can see, the L2 regularized sparse filtering yields higher performance.

4.2. Small Norb classification

The small Norb dataset [19] is a collection of 96×96 gray images from 3D toys. It contains 50 toys belonging 5 generic classes: four legged animals, human figures, airplanes, trucks and cars. Figure 2(b) shows some examples of the dataset. Following the partitioning scheme in [19], there are 48600 images for training and 48600 images for testing in total. The learned filters are shown in Figure 2(e) (receptive field size $w = 12$), and Table 2 lists the comparative experimental results. Again, the L2 sparse filtering achieves higher accuracy.

Table 2. Comparative classification accuracy on small Norb.

#Features	100	200	400	800
Sparse filtering	84.19	85.76	87.21	88.19
L2 Sparse filtering	**84.58**	**85.91**	**87.63**	**88.61**

4.3. Offline handwritten Chinese character recognition

Offline handwritten Chinese character recognition is a difficult task due to large variability of stroke and writing style. Many types of classification framework are proposed to improve the recognition rate based on gradient feature, which has been granted as the best feature for character recognition [22]. Here, we try to use L2 regularized sparse filtering to learning feature for recognition.

The dataset we used is CASIA-HWDB1.0 [20], containing 3866 classes' characters with 420 samples per class. Each character is a 64×64 gray image. For simplify, we randomly selected three groups of data from the large

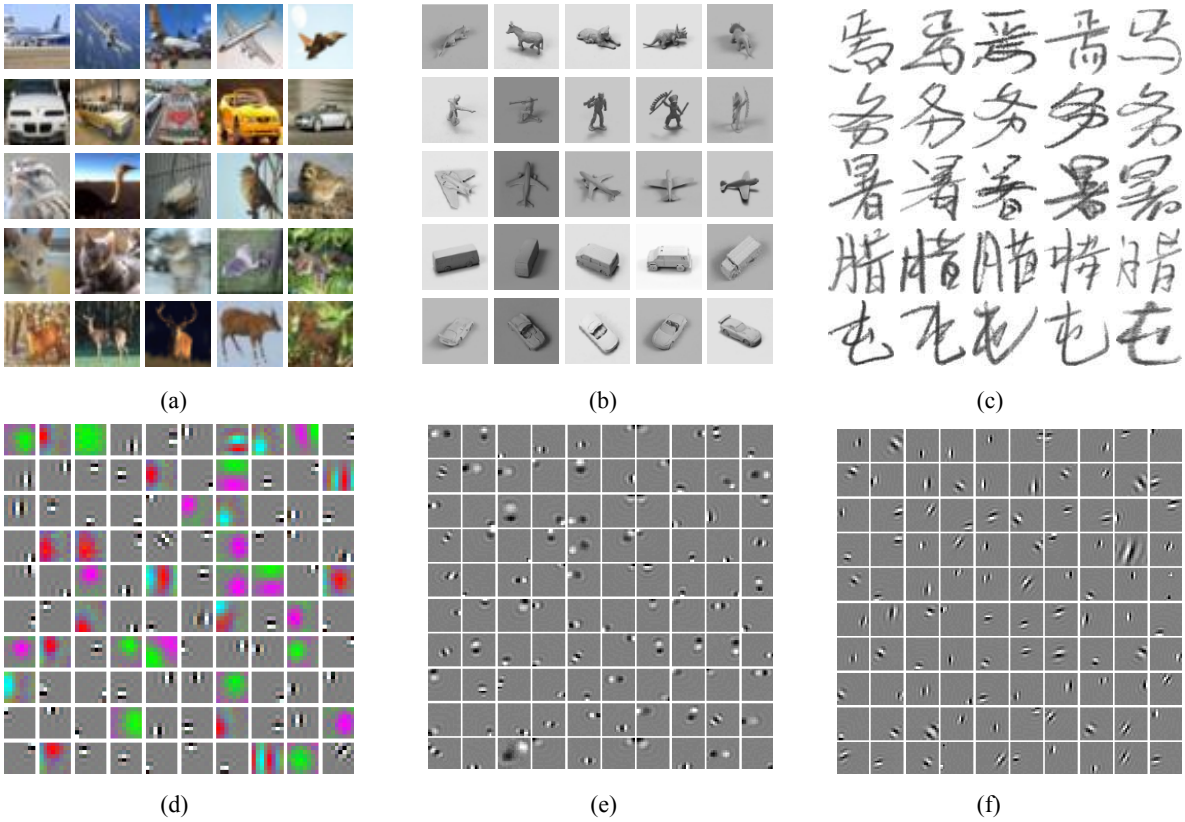


Fig. 2. (a), (b), (c): Sample images from three datasets, CIFAR-10, small Norb, subsets of CIASIA-HWDB1.0, and corresponding learned filers in (d), (e), (f).

dataset, each group had 10 classes characters. Random 320 images were used for training and the remains were used for test per class.

Table 3. Comparative classification accuracy on subsets of CASIA-HWDB1.0 (The dimension of the final feature representation is #Featues×4 when using feature learning methods, and the dimension of gradient feature is 512).

Groups	G1		G2		G3	
Features	128	512	128	512	128	512
Sparse filtering	95.50	96.75	95.75	97.38	97.13	98.38
L2 Sparse filtering	95.63	97.83	96.75	97.75	97.50	**98.75**
Gradient feature	**99.37**	-	**98.50**	-	98.63	-

Table 3 shows the performance of various feature extraction methods. It is clearly seen that L2 regularized sparse filtering has better performance than sparse filtering, achieving approximate results with the classic 512 dimensional gradient feature [23] (When using feature learning methods, the dimension of the final feature representation is #Featues×4). In addition, it has improved accuracy with more learned features as expected.

The filters learned by L2 sparse filtering depicted in Figure 3(f). As we can see, the filers are oriented, which are very useful for character recognition intuitively.

5. CONCLUSIONS

In this paper, we have present the single-layer unsupervised feature learning with L2 regularized sparse filtering to learn useful feature representation from unlabeled data. This general L2 regularization allowed the network to learn a better feature mapping matrix. Classification experiments on three different datasets were provided to show its superior performance over the standard sparse filtering.

Acknowledgement. This research is supported in part by NSFC (Grant No.: 61075021, 61201348, 61202292), National science and technology support plan (Grant No.:2013BAH65F01, 2013BAH65F04), GDSTP (Grant No.: 2012A010701001, S2012040008016), Research Fund for the Doctoral Program of Higher Education of China (Grant No.: 20120172110023).

6. REFERENCES

[1] M. D. Zeiler, G. W. Taylor and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *ICCV*, 2011.

[2] A. M. Saxe, P. W. Koh, Z. Chen, et al., "On random weights

and unsupervised feature learning," in *ICML*, 2011.

[3] H. Lee, R. Grosse, R. Ranganath, et al., "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representation," in *ICML*, 2009.

[4] A. Coates, H. Lee, A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011.

[5] G. E. Hinton, R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 313(5786): 504-507, 2006.

[6] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends in Machine Learning*, 2(1): 1-127, 2009.

[7] R. Kiros, C. Szepesvari, "On linear embeddings and unsupervised feature learning," in *ICML*, 2012.

[8] L. Wan, M. Zeiler, S. Zhang, et al., "Regularization of neural networks using dropconnect," in *ICML*, 2013.

[9] A. Coates, A. Y. Ng, "Learning feature representation with K-means," *Neural Networks: Tricks of the Trade*, 7700: 561-580, 2012.

[10] P. Vincent, H. Larochelle, Y. Bengio, "Extracing and composing robust features with denoising autoencoders," in *ICML*, 2008.

[11] A. Coates, A. Karpathy, A. Y. Ng, "Emergence of object-selective features in unsupervised feature learning," in *NIPS*, 2012.

[12] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," in *NIPS*, 2008.

[13] M. Ranzato, C. Poultney, S. Chopra, Y. LeCun, "Efficient learning of sparse representations with an energy-based model," in *NIPS*, 2006.

[14] A. Y. Ng, "Sparse autoencoder," *CS294A Lecture notes*, Stanford University, 2011.

[15] A. Hyvärinen, E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, 13(4):411-430, 2000.

[16] J. Ngiam, P. W. Koh, Z. Chen, et al., "Sparse filtering," in *NIPS*, 2012.

[17] Q. V. Le, J. Ngiam, A. Coates, el at., "On optimization methods for deep learning," in *ICML*, 2011.

[18] A. Krizhevsky. "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science*, University of Toronto, 2009.

[19] Y. LeCun, F.J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting", In *CVPR*, 2004.

[20] C.-L. Liu, F. Yin, D.-H. Wang, et al., "CASIA online and offline Chinese handwriting databases," in *ICDAR*, 2011.

[21] M Schmidt. minFunc. http://www.di.ens.fr/~mschmidt/Software/minFunc.html.

[22] C.-L Liu, "Normalization-cooperated gradient feature extraction for handwritten character recognition," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 29(8): 1465-1469, 2007.

[23] C.-L. Liu, F. Yin, D.-H. Wang, et al., "Online and offline handwritten Chinese character recognition: benchmarking on new databases", *Pattern Recognition*, 46(1): 155-162, 2013.