Regularized Hierarchical Feature Learning with Non-Negative Sparsity and Selectivity for Image Classification

Bingyuan Liu[1], Jing Liu[1], Chunjie Zhang[2],Xiao Bai[3] and Hanqing Lu[1],
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences.
[2]School of Computer and Control Engineering, University of Chinese Academy of Sciences.
[3] School of Computer Science and Engineering, Beihang University.
[1]{byliu,jliu,luhq}@nlpr.ia.ac.cn, [2]zhangcj@ucas.ac.cn,[3]baixiao.buaa@gmail.com.

***Abstract*—Recently, many deep networks are proposed to learn hierarchical image representation to replace traditional hand-designed features. To enhance the ability of the generative model to tackle discriminative computer vision tasks (e.g. image classification), we propose a hierarchical deconvolutional network with two biologically inspired properties incorporated, i.e., non-negative sparsity and selectivity. First, we propose a single layer deconvolutional model with a raw image as input, attempting to decompose the input as a weighted sum of feature maps convolving with filters. Here, the filters are the model parameters common to all the inputs, while the feature maps and the summing weights are specific to the input. The non-negative sparsity is formulated as the l_1-norm regularizer on the feature map, which is used to generate feature representations for image classification. And the selectivity is forced on the filters to make different filters active different inputs, through requiring the sparsity on the summing weights specifically. The two properties are summarized into an overall cost function, which can be solved with an alternatively iterative algorithm. Then, we build multiple layer deconvolutional network by stacking the single models, where the next-layer inputs are the results of a 3-D max-pooling operation on the inferred feature maps of the front layer, and train the network in a greedy layerwise scheme. Finally, we explore the feature maps of each layer to generate the image representations and input them to a SVM classifier for the classification task. Experiments on two image benchmark datasets of Caltech-101 and Caltech-256 demonstrate the encouraging performance of our model compared with other deep feature learning models as well as some hand-designed features.**

***Keywords*-image classification; deep learning**

I. Introduction

Most computer vision systems critically rely on the quality of image representations. However, how to build a suitable feature extraction model remains challenging. In the past decade, manually-designed image descriptors dominate this area and some well designed features, such as Gabor, SIFT and HOG, have precipitated dramatic success. Although these hand-crafted features may benefit from human ingenuity and prior domain knowledge to some extent, the choice of features is highly dependent on dataset, task and experience. It is also difficult to detect more complex features, including mid and high-level structures in images. Over the recent years, there is a growing interest in building deep models for learning hierarchical image representations to replace traditional engineered descriptors. These include Deep Belief Networks[1], Deep Boltzmann Machines[2] and Deconvolutional Networks[3], which attempt to build hierarchical generative models of the input datas. The deep architecture is motivated by the hierarchical nature of human vision cortex and may be improved with more knowledge from the human cognitive process incorporated.

Usually a hierarchical architecture consists of feature detector units arranged layer by layer with the objective of reconstructing input data. However, two critical properties are ignored in most previous models. One is the non-negative sparsity of the latent features. Some biological evidences[4] indicate the non-negative sparse coding model is relatively similar to the signal process within the brain, and it has been demonstrated that this non-negative sparse constraints is useful to extract features from natural images. The other one is the selectivity of the hierarchical model. That is, a certain neuron unit only senses a particular image structure while keeps silent for other stimulations. It is suggested from the biological evidence that the response possibilities of some neurons in the temporal cortex are highly selective for certain object categories like faces or hands and even specific people[5]. The selective property is also significant to enhance the discrimination of the learned features especially when performing discriminative tasks (*e.g.* image classification). Therefore how to incorporate the non-negative sparsity and selective property into the hierarchical architecture is essential to improve the performance.

To address above issues, we propose a deconvolutional network architecture with non-negative sparsity and selectivity regularization. The deconvolutional network decomposes an input image in a hierarchical fashion with multiple alternating layers of deconvolution and max pooling. First, we present a single deconvolutional layer with raw image as input. To integrate non-negative sparsity, we employ l_1 regularizer on the latent maps and enforce them to be non-negative meanwhile. Though the non-negativity sparsity reduces the flexibility of the model, we demonstrate it does help to learn better representations. To introduce selectivity, it is difficult to directly regularize the feature maps. Instead, we introduce weighting parameters between the input and filters to measure the selectivity and then employ l_1 regularization on these parameters encouraging that seldom filters are activated given a certain input. In this way, the selectivity is enhanced as each filter is activated for some particular input structures and different filters respond to different kinds of input.

The two characteristics are summarized into an overall cost function, and a 3-D max pooling operation is applied on the inferred feature maps. Then, we build multiple layers of deconvolutional network by stacking the single models and train in the greedy layerwise scheme. Finally, with the trained model we can extract an over-complete image representation in a purely unsupervised way and explore it as the input of SVM classifier with a spatial pyramid matching kernel to perform image classification. Extensive experiments on the benchmarks of Caltech-101 and Caltech-256 demonstrate the effectiveness of the proposed method compared to baselines and related works.

II. Related Work

Recently, many researches have focused on building hierarchical feature learning model as an alternative to classical hand-crafted descriptors. One class of hierarchical architecture is a family of biologically-inspired heuristic models[6], motivated by the simple-complex cell model of mammal vision system. Convolutional Neural Network(CNN)[7] is one of the most successful hierarchical models. It is constructed by alternating between convolutional filtering and max-pooling, and trained in a purely supervised scheme by back propagation algorithm. Recently reported performances of CNN in some databases have outperformed classical bag-of-feature model[8].

Our proposed model is developed with another deep learning framework of stacking shallow generative model by greedy layerwise unsupervised learning scheme. Deep Belief Networks(DBN)[9], a pioneer work proposed by Hinton *et al.*, builds multiple layers of directed sigmoid belief nets with the top layer as a Restricted Boltzmann Machines(RBM). Lee *et al.*[10] extended DBN by incorporating convolution operation for extracting latent features from raw image pixels. The closest models to ours are those based on convolutional sparse coding. Kavukcuoglu *et al.*[11] proposed to extract features through learned convolutional filter banks and construct a multi-stage convolutional network, and Zeiler *et al.*[3] built a deconvolutional network by alternately stacking convolutional sparse coding and max-pooling layers.

How to regularize the deep architectures for effectively learning well representations is critical, as deep models may easily suffer from over-fitting problem. Motivated by the successful application of sparse model[12], some works introduce sparse regularizer into feature learning model to improve the discriminative performance. In [10], the sparse constraints are achieved by specifying a desired probability of being active and then adding an additional penalty term to encourage the actual distribution close to the desired. Hinton et al.[13] proposed dropout by randomly omitting half of the feature detectors on each training case to prevent overfitting, while Wan et al.[14] generalized this idea by setting a randomly selected subset of weights within the network to zero for regularizing large fully-connected layers. Different from them, we jointly explore the non-negative sparsity and selectivity as the regularizer in our model, to enhance the discriminative ability of the inferred hierarchical feature representation. Some other works employ the regularization of manifold property to learn proper features[15][16].

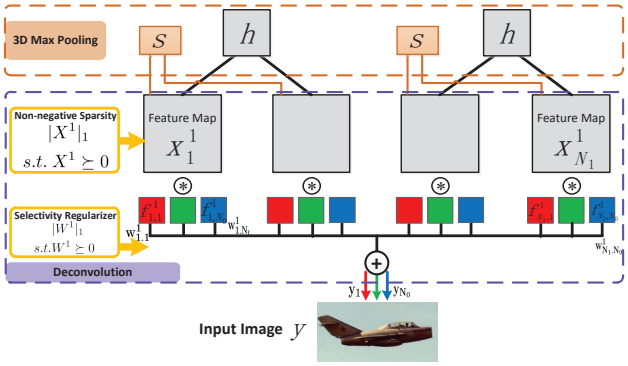

Figure 1. Single deconvolutional network layer consists of convolutional non-negative sparse coding and 3-D max pooling operations.

III. The Proposed Deconvolutional Network

A. Single Layer

Firstly, we consider a single deconvolutional layer applied to an image, as shown in Figure 1. With an image y composed of N_0 channels $y_1, \ldots, y_{N_0}$ (*e.g.* RBG channels) as input, the layer decomposes it via two operations: deconvolution and max-pooling.

Deconvolution: The deconvolution operation decomposes each channel of the input as a sum of N_1 latent 2-D feature maps $x_i^1 (i = 1, 2..., N_1)$ convolved with filters $f_{i,j}^1 (i = 1, 2, ..., N_1, j = 1, 2, ..., N_0)$. To measure the selectivity of the filter, we introduce weighting parameters $w_{i,j}^1 (i = 1, 2, ..., N_1, j = 1, 2, ..., N_0)$ between the input channel i and the feature map j. Thus the reconstruction of the input channel is formulated as a weighted linear sum of convolutions:

$$\hat{y}_c^1 = \sum_{i=1}^{N_1} w_{i,c}^1 (x_i^1 * f_{i,c}^1) \tag{1}$$

where we denote the reconstruction of channel c as $\hat{y}_c^1$. For notational brevity, we combine the convolution and summing operations into a single operation matrix F^1 and denote the weighting operation as W^1, thus the reconstruction is denoted as $\hat{y}^1 = F^1 W^1 x^1$.

To guarantee a stable solution, the objective of the deconvolution is to minimize the reconstruction error while impose certain regularization on latent feature maps. As non-negative sparsity is helpful to learn reasonable and discriminative features, we employ a l_1 norm regularizer on x^1 and enforce it to be non-negative with a strict constraint. Note that the introduced weighting parameters $w_{i,j}$ may be regarded as a selectivity measurement of the filters because the distribution of the non-zero $w_{i,j}$ decides which filters are activated. A good selective property means that seldom filters are activated for a particular kind of input and different filters respond to different inputs. To enhance it, a l_1 norm regularizer is also applied on

the weighting matrix. These two regularizations may be combined into an overall cost function for the single layer:

$$C_1(y) = \frac{1}{2}\|\hat{y}^1 - y\|_2^2 + \lambda_1 \sum_{i=1}^{N_i} |x_i^1|_1 + \beta_1 |W^1|_1 \quad s.t.\ x_i^1 \succeq 0, W^1 \succeq 0 \tag{2}$$

where λ_1 and β_1 are the tradeoff parameters to balance the relative contributions of the reconstruction error and the regularization terms. A non-negative constraint is also imposed on W^1 to obtain reasonable solutions. The filters are the network parameters common to all inputs, while features maps x^1 and weights W^1 are specific to input.

Max-pooling: After the above deconvolution, a 3-D max-pooling is performed on the learned features maps, introducing some local invariance and allowing higher layer to capture structure at a larger scale. Beyond regular max-pooling, 3-D max-pooling occurs both within the feature map and between adjacent maps. 3-D cubes are firstly applied to partition the feature maps and then both values and location switches of the maximum are recorded as shown in Figure 1. We denote h and s as the pooled maps and location switches respectively, and formulate the 3-D max pooling as $[h, s] = P(x)$, which is a nonlinear operation. However, given s fixed, it becomes a linear operation denoted as $h = P_s x$, with P_s being a binary selection matrix set by switches s. Oppositely, a 3-D max-unpooling operator U_s is utilized for reconstruction, which takes the elements of h back into x at the locations recorded in s with the others set to zero. It is also a linear operation when s is fixed, denoted as $\hat{x} = U_s h$.

B. Multiple Layers

It is easy to build a deep deconvolutional network by stacking the single layer module, where the outputs of layer l are taken as the inputs to layer $l+1$, as shown in Figure 2. The architecture remains the same for each layer but the number of feature maps N_l may vary. The higher layer of our model is also trained with respect to the original input image rather than the input of current layer. With the reconstruction of layer l denoted as $\hat{y}^l$, the objective is to minimize the reconstruction error to the original input image y with the regularization terms:

$$C_l(y) = \frac{1}{2}\|\ \hat{y}^l - y\|_2^2 + \lambda_l \sum_{i=1}^{N_l} |x_i^l|_1 + \beta_l |W^l|_1 \quad s.t.\ x_i^l \succeq 0, W^l \succeq 0 \tag{3}$$

We denote R^l as the reconstruction operator taking the feature maps x^l backward down to the original input space with alternative convolution, weighting and unpooling operations: $\hat{y}^l = F^1 W^1 U_{s_1} F^2 W^2 U_{s_2} ... F^l W^l x^l = R^l x^l$. A reverse projection operator $(R^l)^T$ is also needed for calculating gradients, which takes the input of the network forward up to the feature maps of layer l: $(R^l)^T = (W^l)^T (F^l)^T P_{s_{l-1}} (W^{l-1})^T (F^{l-1})^T \dots P_{s_1} (W^1)^T (F^1)^T$, where $(F^l)^T$ is the flipped version of filters F^l.

Algorithm 1 Training of the proposed model

Input:
Training set Y, Number of layers L, Epoch E, ISTA step T, Regularization coefficients λ_l, β_l, Shrinkage parameter θ_l
Output:
Filters F, Feature maps X, Location switches S, Weight W

for $l = 1 : L$ (*Loop over layers*) **do**
  Init. feature maps $x^l \sim 0$,filters $F^l \sim N(0, \epsilon)$
  for $epoch = 1 : E$ (*Epoch iteration*) **do**
    for $i = 1 : N$ (*Loop over images*) **do**
      # Inference phase
      for $t = 1 : T$ (*ISTA iteration*) **do**
        Compute gradient G_{x^l}, G_{W^l} using Eq.4 and Eq.5
        Gradient step: $x^l = x^l - \frac{\theta_l}{\lambda_l} G_{x^l}$, $W^l = W^l - \frac{\theta_l}{\beta_l} G_{W^l}$
        Shrink step:$x^l = max(|x^l| - \theta_l, 0) \cdot sign(x^l)$, $W^l = max(|W^l| - \theta_l, 0) \cdot sign(W^l)$
        Project step: $x^l = max(x^l, 0)$, $W^l = max(W^l, 0)$
        Pooling operation: $[h^l, s^l] = P(x^l)$
      end for
      # Learning phase
      Compute gradient G_{f^l} using Eq.6
      Update f^l by conjugate gradient algorithm
    end for
  end for
end for

C. Model Learning

Our proposed model can be efficiently learned in a layer-by-layer scheme starting from the bottom layer to the top. In our deconvolutional network, the only parameters to be estimated is the filters of each layer. However, when training the network, the x^l, W^l and F^l are all unknown. Therefore, we alternately minimize $C_l(y)$ over the feature maps and weights with filters fixed, and update the filters while keeping the other variables.

With the filters fixed, the objective is to solve Eq.3 by finding the optimal feature maps x and weights W given an input image y. Here, we adopt ISTA[17], which is an iterative framework consisting of gradient, shrinkage and projection steps. To apply ISTA, we firstly compute the gradients of the reconstruction error term with respect to x^l and W^l via the following equations:

$$G_{x^l} = (R^l)^T (R^l x^l - y) \tag{4}$$

$$G_{W^l} = (F^l)^T (R^{l-1})^T (x^l)^T (R^l x^l - y) \tag{5}$$

Then we may update x^l and W^l by gradient descent. Following the gradient step, a per-element shrinkage operation is performed to clamp small elements in x^l and W^l to zero, increasing the sparsity. To satisfy the non-negative constraints, the solutions are finally projected onto non-negative set. Some detailed notes are shown in Algorithm.1. As all the above operations are amenable to parallelization, the iterations are executed very efficiently.

To update filters, we utilize linear conjugate gradient algorithm as the model is linear in F^l when fixing other variables. The gradient of the cost function with respect to filters is obtained by the following function:

$$G_{f_{i,j}^l} = \frac{\partial C_l}{\partial f_{i,j}^l} = [P_{s_{l-1}} (R^{l-1})^T (R^l x_l - y)]_i * W^l x_j^l \tag{6}$$

This is a convolution between two terms, where the left term is the reconstruction error propagating up to layer

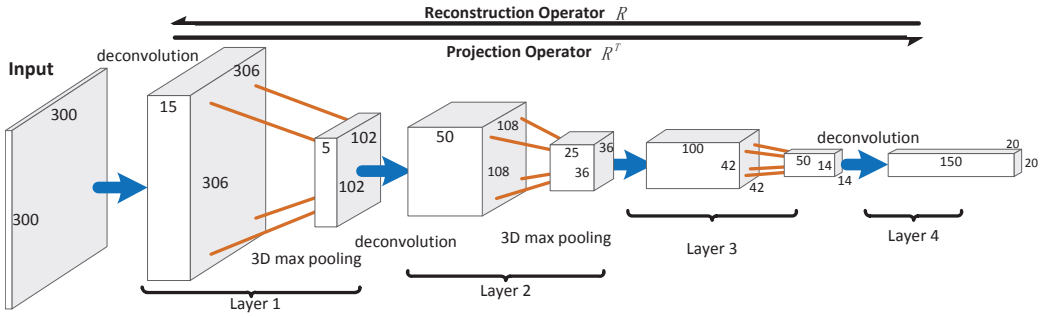

Figure 2. Deep architecture of our proposed model. It is built by stacking the single layer modules.

$l-1$ and the right one is the weighted feature maps of current layer. The overall training algorithm of our proposed model is summarized in Algorithm 1.

IV. Image Classification Pipeline

With the trained model, hierarchical feature maps are obtained for a given input image by applying ISTA algorithm layer-by-layer. As our model is conducted in a totally unsupervised manner, a supervised classifier must be combined to perform image classification. We utilize the Spatial Pyramid Matching(SPM[18]) to construct final image representation by replacing the standard SIFT descriptors with the features learned by our model and then turn to train a SVM to classify the images.

Although the filters are shared between images, the location switches and weighting parameters are not. Thus the feature maps of two images are not directly comparable. Taking advantage of the introduced selective property, we separately reconstruct feature maps of the top layer to extract features as input to SPM. For an input image, our network decomposes it into multiple layers of feature maps. First, we take the largest activations of each feature map in the top layer L and separately reconstruct each one down to the input space, obtaining N_L reconstructed images $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{N_L})$ with each one including different image parts. Because features in the 1st layer are roughly similar to SIFT descriptors, we extract local features from corresponding feature maps of the N_L reconstructions in layer 1 and obtain N_L distinctive spatial pyramid representations. These features are finally combined as the image representation for SVM classifier. We also directly employ the actual feature maps of layer 1 to construct image representations, and the two levels of representations may be combined to further enhance the performance.

V. Experiments and Results

We implement our method with the toolbox developed by Matthew Zeilier[3], the GPU library GPUmat[1] and the efficient GPU convolutional library[2]. Our model is trained on a 24-core CPU and an Nvidia Tesla M2075 GPU for parallelization and fast computing. We train 4-layer model with sizes of filters in each layer set to 7×7 on two image benchmarks Caltech-101[19] and Caltech-256[20]. The parameters setup of our model is presented in Figure 2, where the number of feature maps in each layer is $15, 50, 100, 150$ and the sizes of pooling are $3\times3\times3, 3\times3\times2, 3\times3\times2, 3\times3\times2$ respectively. The tradeoff coefficients λ and β of the 4 layers are both set as $[1, 10, 50, 100]$ according to our implementation. A little preprocess is performed before inputting images into the network by converting each image to gray-scale and resizing to 300×300 with zeros padding to preserve the aspect ratio. A local subtraction with Gaussian filter is also conducted and the intensities are normalized to $[0, 1]$.

With the trained model, we perform image classification by integrating with SPM classifier. Compare to the standard SPM model, the only difference is that we replace the SIFT descriptors with our learned features. We densely extract local descriptors by the method described in Section IV over a grid with 4 pixels spacing on patches of 16×16 and the size of codebook is fixed as 2000. The spatial pyramid representations are obtained by dividing the maps into non-overlapping regions with $1\times1, 2\times2, 4\times4$ grids, and χ^2 kernels are utilized to train SVM classifiers.

In the following subsections, we report the classification performance on Caltech-101 and Caltech-256. We mainly compare our method to standard SPM model with SIFT descriptors and some hierarchical feature learning methods. Some learned filters are also visualized to investigate what we learn and demonstrate the effectiveness of our model to capture hierarchical image structure information.

A. Results on Caltech-101

The Caltech-101 dataset[19] contains $9,144$ images from 102 different categories, including 101 object categories and 1 additional background class, with high shape variability. The number of images per category varies from 31 to 800. We train a 4-layer model on 30 images per category and the overall training time is about 2 hours. When performing classification task, we follow the common experiment setup, training the SPM classifier on 30 images per category and testing on the rest.

The classification performances are shown in Table I. DN_N denotes the deconvolutional network with only non-negative sparsity on feature maps, while DN_S represent the model with only selectivity regularizer imposed on the weighting parameters. The method with both non-

[1]http://http://sourceforge.net/projects/gpumat/
[2]http://code.google.com/p/cuda-convnet/

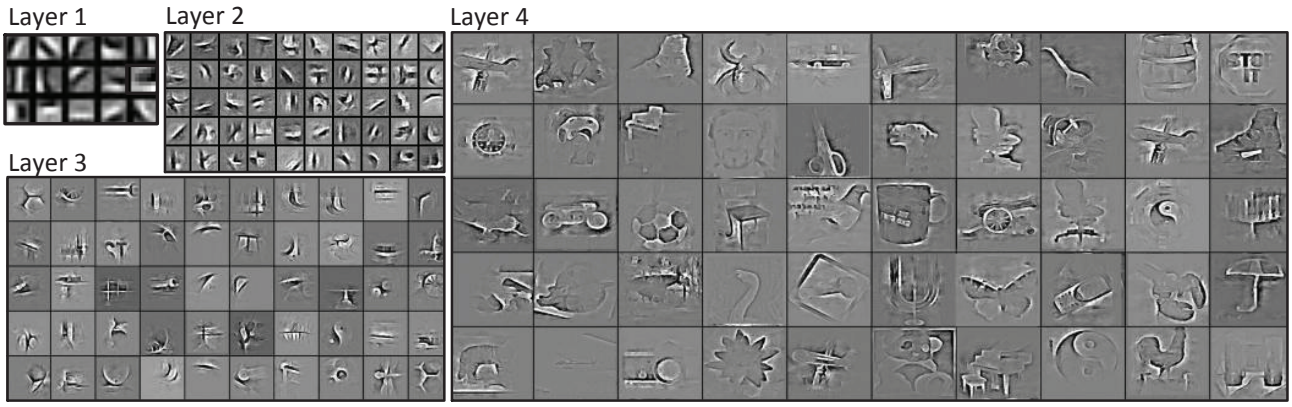

Figure 4. Visualization of the learned filters in each layer of our model. It is shown that our model captures image structures information from low-level edges to mid-level geometric elements and high-level objects.

Table I
CLASSIFICATION RATE(%) COMPARISON ON CALTECH101.

Algorithms	Classification Rate
SIFT[18]	64.60 ± 0.8
Macrofeatures[21]	70.9 ± 1.0
CDBN[10]	65.4 ± 0.5
ConvSC[11]	65.7 ± 0.7
DN(layer 1)[3]	67.8 ± 1.2
DN(layer 4)[3]	69.8 ± 1.2
DN(layer 1+4)[3]	71.0 ± 1.0
DN_N (layer 1)	68.4 ± 0.3
DN_S (layer 1)	68.8 ± 0.2
DN_{NS} (layer 1)	69.0 ± 0.8
DN_N (layer 4)	70.3 ± 0.5
DN_S (layer 4)	71.5 ± 0.4
DN_{NS} (layer 4)	72.5 ± 0.2
DN_{NS} (layer 1+4)	$\mathbf{73.3 \pm 0.4}$

Figure 3. Examples of the Caltech-101 set. Top: the top 3 categories where our model improves most. Bottom: the 3 categories where our method decreases performance. The numbers in the brackets indicate the classification rate (DN(layer 1+4)/DN_{NS}(layer 1+4))

negative sparsity and selectivity is denoted as DN_{NS}. It is shown that the non-negative sparsity and selectivity do help to enhance the discrimination of the learned features, and the best classification rate is achieved by employing the combined features of 1-st and 4-th layer from the model trained with both properties. Compared with the most related model[3] denoted as DN, which is a 4-layer model with only sparsity regularization on feature maps, we obtain an improvement of more than 2%. The enhancement in layer 4 is more obvious, as the integrated properties are more significant to the over-complete feature maps in high-level layer. Our results also outperform some other deep learning model like CDBN[10] and ConvSC[11]. We also compared with the famous SIFT descriptor and its variant Macrofeatures[21], demonstrating our learned features perform better than these hand-designed descriptors. Figure 3 shows some examples of the classes that our model increases the performance most compared to [3] and some bad examples that our model decrease the performance. It is investigated that consistent improvements are obtained for most categories. The complex texture information may influence the performance on some classes.

Figure 4 demonstrates what we learn in each layer of the model. How to visualize the trained units in deep model is a critical problem, since it is meaningless by directly showing the filter matrixes in high layer. In this paper, we visualize high-level filters by separately taking the largest activation in each feature map over the entire training set and reconstructing down to the input space. In another view, we inspect the filters by evaluate what kind of input activate them strongest. It is shown that the filers in the 1st layer are the edge structures in different orientations and scales. This is why the features extracted from this layer are similar to SIFT descriptors. In layer 2, the filters capture some edge junctions, curves, parallel lines and other types of edge combinations. The filters in layer 3 represent more complex geometric elements beyond those in layer 2. We can see some filters already capture object parts in this layer, such as wheel and wrench. In the 4-th layer, the learned filters capture object parts and complete objects, such as the shown face, chair and ball, which are very high-level image information. The visualizations demonstrate that our model captures the hierarchical image structures from low-level edges to mid-level geometric elements, and high-level object structures.

B. Results on Caltech-256

The Caltech-256[20] dataset is an extension of Caltech-101, which holds $29,780$ images falling into 256 categories. This is a much more challenging database as it possesses much higher intra-class variability and object location variability. Using the same parameters as the experiments on Caltech-101, we train a 4-layer model on 30 images per class and evaluate on the rest.

The performance comparison results are shown in Table II. In the more challenging dataset, our model also leads

Table II
CLASSIFICATION RATE(%) COMPARISON ON CALTECH256.

Algorithms	Classification Rate
SIFT[20]	29.5 ± 0.5
ScSPM[12]	34.0 ± 0.4
DN(layer 1)[3]	31.2 ± 1.0
DN(layer 4)[3]	30.1 ± 0.9
DN(layer 1+4)[3]	33.2 ± 0.8
DN_{NS} (layer 1)	32.2 ± 0.2
DN_{NS} (layer 4)	33.6 ± 0.2
DN_{NS} (layer 1+4)	$\mathbf{35.0 \pm 0.3}$

the performance and the best result is also obtained by the combined 1-st and 4-th layer features from the proposed deconvolutional network with both non-negative sparsity and selectivity regularization(DN_{NS}). It is shown that our result outperforms ScSPM[12] which is an excellent extension of SPM using sparse coding to encode SIFT features. In the much more challenging condition, consistent improvements are also obtained for most categories, demonstrating the effectiveness of integrating more regularized properties in deep feature learning models.

VI. CONCLUSION

In this paper, we propose an improved deep deconvolutional network by jointly incorporating non-negative sparsity and selectivity property. An overall cost function with the two regularizers is proposed for each layer and then we build a 4-layer network by greedy layerwise unsupervised learning scheme. It is shown that our model effectively learns hierarchical image features from low-level edges to high-level complex objects. Experiments on Caltech-101 and Caltech-256 datasets show outperforming performances compared to related deep learning models as well as classical hand-designed features, demonstrating the effectiveness of our method to enhance the discriminative property of deep feature learning model.

VII. ACKNOWLEDGMENT

This work was supported by 973 Program (2010CB327905) National Natural Science Foundation of China (61272329, 61273034, 61202325,61303154), the President Fund of UCAS and the Open Project Program of the National Laboratory of Pattern Recognition(NLPR).

REFERENCES

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[2] R. R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann machines," in *AISTATS*, vol. 5, 2009, pp. 448–455.

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

[4] P. O. Hoyer, "Modeling receptive fields with non-negative sparse coding," *Neurocomputing*, vol. 52-54, pp. 547–552, 2003.

[5] R. Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried, "Invariant visual representation by single neurons in the human brain," *Nature*, vol. 435, pp. 1102–1107, 2005.

[6] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in *NIPS*, 2012.

[9] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, p. 2006, 2006.

[10] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 609–616.

[11] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, "Learning convolutional feature hierarchies for visual recognition," in *NIPS*, 2010.

[12] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR 2009*, 2009.

[13] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[14] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural network using dropconnect," in *ICML*, 2013.

[15] C. Li, Q. Liu, J. Liu, and H. Lu, "Learning ordinal discriminative features for age estimation," in *CVPR*, 2012.

[16] C. Li, Q. Liu, J. Liu, and H. Lu, "Ordinal regularized manifold feature extraction for image ranking," *Signal Processing*, vol. 93, no. 6, pp. 1651–1661, 2013.

[17] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

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

[19] 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," *CVIU*, vol. 106, no. 1, pp. 59–70, 2007.

[20] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[21] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *CVPR*, 2010.