Learning Hierarchical Image Representation with Sparsity, Saliency and Locality

Jimei Yang
jyang44@ucmerced.edu
Ming-Hsuan Yang
mhyang@ucmerced.edu

University of California, Merced
California, USA

Abstract

This paper presents a deep learning model of building up hierarchical image representation. Each layer of hierarchy consists of three components: sparse coding, saliency pooling and local grouping. With sparse coding we identify distinctive coefficients for representing raw features of each lower layer; saliency pooling helps suppress noise and enhance translation invariance of sparse representation; we group locally pooled sparse codes to form more complex representations. Instead of using hand-crafted descriptors, our model learns an effective image representation directly from images in a unsupervised data-driven manner. We evaluate our algorithm with several benchmark databases of object recognition and analyze the contributions of different components. Experimental results show that our algorithm performs favorably against the state-of-the-art methods.

1 Introduction

In this paper, we present a deep learning model for hierarchical image representation in which we build the hierarchy by stacking up the base models layer by layer. In each layer, the base model receives the features of the lower layer as input and produces a more invariant and complex representation. The bottom layer receives raw images as input and the top layer produces an image representation that can be used for high-level vision tasks. In this paper, we focus on category-level object recognition and our base model consists of three components: sparse coding, saliency pooling and local grouping.

Sparse coding It is well known that natural images can be sparsely represented by a set of localized, oriented filters [16]. By imposing ℓ_1 norm regularization on representation coefficients, sparse coding can be solved efficiently [10]. Recent progress in computer vision has demonstrated that sparse coding is an effective tool for representing visual data on different levels, *e.g.* image denoising [3] and image classification [25]. We use sparse coding in our base model to learn a set of atom or basis signals from the lower layer so that raw features fed to the current layer can be well quantified. At each layer, we need to encode a large set of raw features in the image domain, and thus the sparse coding is the main computational bottleneck of our model. We develop a parallel implementation of ℓ_1 norm sparse coding by a coordinate descent algorithm. This implementation allows us to encode raw features of the entire image domain simultaneously and significantly improve the computational performance of our model.

Saliency pooling The sparse codes (i.e., encoded raw features) are generally computed with an over-complete dictionary of atom signals. As a result of using the over-complete dictionary, independent sparse codes are favored and inevitably the sparse representation is more sensitive to variation (e.g., slight translation or rotation) and noise. To alleviate this, pooling functions are often used to characterize the statistics of sparse codes within certain local image region. Among several pooling functions (e.g., average, energy and max) in the literature, max pooling has been shown to perform well in several tasks. For image representation, we observe that irrelevant parts of the image (background, non-target objects) may have large sparse coefficients. Consequently, such sparse representations may encode more non-essential visual information. We propose a saliency-weighted max pooling function to address this problem. Visual saliency [7] models the attentional mechanism of biological vision. The most distinctive features are identified within certain the image region by measuring the saliency based on the principles of center-surround contrast and information maximization. Therefore, visual saliency will bias the pooling function toward the image regions where the targets are likely to appear. By using the bottom-up saliency to guide pooling, in general better sparse representations focusing on the foreground objects can be obtained.

Local grouping Grouping has been studied extensively in computer vision for extracting mid-level visual information from pixels or features [13] [22]. Local grouping is a key component for our base model. By grouping the pooled sparse codes in local neighborhood, the base model can produce increasingly complex representation for the use of the upper layer in the sense of bridging the semantic gap between successive layers. In this paper, we use the sub-window based grouping for simplicity. We first concatenate the pooled sparse codes in the square grids (*e.g.* $3 \times 3, 4 \times 4$) to form a vector. As the dimension of this vector is high, we use PCA (Principle Component Analysis) to reduce the dimension while preserving the representation ability for local grouping.

In this paper, we exploit the basic properties of image data: sparsity, saliency and locality in the base model and organize them in hierarchical fashion to form a deep representation architecture. We refer the proposed model as HSSL (Hierarchical model with Sparsity, Saliency and Locality) model and Figure 1 illustrates the architecture and processes.

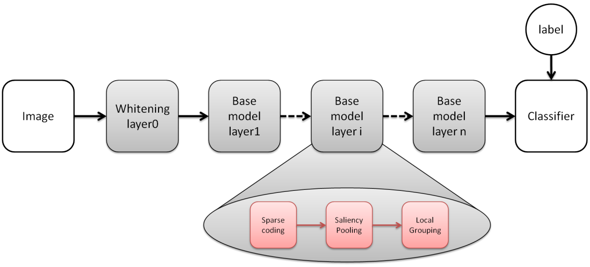

Figure 1: Illustration of the HSSL model. The shadowed components denote the learning process. At layer 0, a standard preprocessing is applied to reduce noise (e.g., whitening and normalization). From layer 1 and onward, the base model at layer i is repeatedly built upon layer $i-1$. The output of the layer n is fed into classifiers for vision tasks.

Instead of using hand-crafted descriptors (SIFT, HoG, Gabor, LBP), the proposed HSSL model learns effective representation directly from images in a unsupervised data-driven manner. The proposed model requires minimum expert knowledge for specific tasks or laborious labeling process. Therefore, it can be easily applied to vision tasks with different sensor data such as infrared and depth images where descriptive features cannot be easily crafted.

2 Related Work

The proposed HSSL model is developed with the deep learning framework [5] in which feature hierarchy is constructed directly from data by layer-wise progressive training. Each layer defines a base model consisting of a set of functions mapping from input to output. The crux of this architecture is to learn the base model that can be applied repeatedly across multiple layers. Each layer generates latent representation that is not directly related to the final tasks, so the mapping functions need to be well constrained. One class of deep learning algorithms [19][8] define a pair of encoder and decoder functions as the base model in which the input is mapped to the output and vice versa. By using this encoder and decoder pairs, variations of the generated representation can be constrained by reconstruction error. Another line of deep learning uses restricted Boltzmann machines (RBM) [5][11] as the base model. Each RBM learns a generative model so that encoder and decoder functions can be derived by marginalization. However, training generative models is still a challenging task. Our model employs a feedforward mapping of sparse decomposition. In this sense, our work bears some resemblance to the deconvolutional network algorithm [26]. The base model of the deconvolutional networks decomposes input raw features of entire image domain into a linear combination of sparse code maps convolved with learned filters. The deconvolutional networks produce smooth sparse code maps at the expense of complex deconvolution computation. Our model performs sparse decomposition of input raw features independently using learned atom signals. Furthermore, sparse coding is followed by saliency pooling and local grouping, which equips our base model with nonlinear mapping from input raw features to latent representations.

In addition, our HSSL model shares a similar framework with a family of biologically-inspired hierarchical models [20] [21] [14] [17] for object recognition. These models are constructed by alternating between convolutional filtering and max pooling, which mimic the simple-complex cell model of Hubel and Wiesel [6]. Instead of validating the proposed model with neural evidence, we aim to integrate learning techniques in order to develop effective image representations for various vision tasks.

We note that some techniques used in our base model are similar to those of [25] but at its core the approach and goals are different. In [25], Yang *et al*. extract well established SIFT [12] descriptors as image representation. Sparse coding and max pooling are utilized in a spatial pyramid matching structure to generalize the bag-of-words model for object categorization. However, our HSSL model learns deep representations by layer-wise training instead of using task-specific descriptors, *e.g*. SIFT for object categorization, HoG for human detection and Gabor for face recognition, to name a few. Thus, it can be easily applied to different vision tasks and data without designing features first.

3 Base Model

The base model defines a nonlinear mapping from the lower layer to the next latent layer within a restricted domain. We denote Ω^ℓ as the working domain of layer ℓ. Let $\mathbf{x}_i^{\ell-1}$ be a feature vector of layer $\ell-1$ and $\mathbf{X}^{\ell-1} = \{\mathbf{x}_i^{\ell-1}\}_{i\in\Omega^\ell}$ be the feature set in the working domain

Ω^ℓ of layer ℓ. The nonlinear mapping function is defined as

$$\mathbf{x}^\ell = f(\mathbf{X}^{\ell-1}), \|\mathbf{x}^\ell\| \leq 1. \tag{1}$$

The function f maps a set of feature vectors from a lower layer to a single feature vector in the current layer. As a result, the first layer extracts features from small local patches and the last layer extracts a single feature of the entire image domain. Formally, we have $\Omega^1 \subset \Omega^2 \subset \ldots \subset \Omega^n = E$, where E denotes the entire image domain. For example, the $\mathbf{x}_i^0$ denotes a patch extracted from the image at the site $i \in \Omega^1$ and $\mathbf{x}^1$ is the feature of first layer extracted from a set of image patches defined in Ω^1. The base model consists of three component functions: sparse coding s, salient pooling p and local grouping g. Thus, the nonlinear function f is composed of a chain: $f = g \circ p \circ s$.

3.1 Sparse Coding

Given a set of d-dimensional atom signals $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_k], \in \mathbb{R}^{d \times k}$, the sparse coding of an input signal $\mathbf{x} \in \mathbb{R}^d$ can be found by solving an ℓ_1-norm minimization problem,

$$\mathbf{s}(\mathbf{x}, \mathbf{B}) = \arg\min_{\mathbf{s}} \frac{1}{2}\|\mathbf{x} - \mathbf{B}\mathbf{s}\|^2 + \gamma\|\mathbf{s}\|_1, \tag{2}$$

where $\|\cdot\|_1$ denotes the ℓ_1-norm. In this paper, we instead solve the elastic net problem [27] to improve the stability by integrating the additional ℓ_2-norm regularization,

$$\mathbf{s}(\mathbf{x}, \mathbf{B}) = \arg\min_{\mathbf{s}} \frac{1}{2}\|\mathbf{x} - \mathbf{B}\mathbf{s}\|^2 + \gamma\|\mathbf{s}\|_1 + \frac{\lambda}{2}\|\mathbf{s}\|_2^2. \tag{3}$$

We reformulate the problem in Eqn. 3 in a quadratic form,

$$\mathbf{s}(\mathbf{x}, \mathbf{B}) = \arg\min_{\mathbf{s}} \frac{1}{2}\mathbf{s}^\top \mathbf{A}\mathbf{s} - \mathbf{s}^\top \mathbf{h} + \gamma\|\mathbf{s}\|_1, \tag{4}$$

where $\mathbf{A} = \mathbf{B}^\top\mathbf{B} + \lambda\mathbf{I}$ ($\mathbf{I}$ is identity matrix) and $\mathbf{h} = \mathbf{B}^\top\mathbf{x}$. The objective function in Eqn. 4 is non-differential. Classic constrained quadratic programming algorithms such as interior point method do not scale well to high dimensionality [24]. Coordinate descent algorithms decompose the original problem into subproblems and the solution can be found by sequentially updating each coordinate by fixing the others:

$$s_j^* = \begin{cases} 0 & \text{if} \quad |\mu_j - a_j s_j| < \gamma \\ (-\mu_j + a_j s_j + \gamma)/a_j & \text{if} \quad \mu_j - a_j s_j > \gamma \\ (-\mu_j + a_j s_j - \gamma)/a_j & \text{if} \quad \mu_j - a_j s_j < -\gamma \end{cases}, \tag{5}$$

where $\mu = \mathbf{A}\mathbf{s} - \mathbf{h}$ and $a_j = \mathbf{A}_{jj}$. The computational complexity turns out to be $O(kt)$, where t is the number of iterations. We need compute sparse coding across the whole image domain in each layer. Assume that there are n^l feature vectors in the l^{th} layer so that the total computational complexity is $O(ktn^l)$. To alleviate this main computational burden, we adapt the algorithm to parallel computing so that it can run on GPUs. we develop a coordinate descent method in a way similar to [18]. We compute s_j^* in parallel with fixed other coefficients in the last iteration so that the coordinate direction is $\mathbf{d} = \mathbf{s}^* - \mathbf{s}$. The objective is reduced in the current iteration by doing the line search in the direction $\mathbf{d}$,

$$\mathbf{s} = \mathbf{s} + \alpha\mathbf{d}, \tag{6}$$

where α is the step length. With Eqn. 6, we can update the sparse coefficients of feature vectors simultaneously on GPUs. Ideally, the computational complexity reduces to $O(kt)$.

The dictionary $\mathbf{B}$ is crucial to the base model. The dictionary learning is formulated into the optimization problem,

$$\arg\min_{\mathbf{B},\mathbf{s}_i} \sum_{i=1}^{n} (\|\mathbf{x}_i - \mathbf{B}\mathbf{s}_i\|_2^2 + \gamma\|\mathbf{s}_i\|_1 + \frac{\lambda}{2}\|\mathbf{s}_i\|^2). \tag{7}$$

The training data $\{\mathbf{x}_i\}_{i=1}^n$ are randomly sampled from the lower layer. The problem in Eqn. 7 is not convex for the dictionary $\mathbf{B}$ and the sparse codes $\mathbf{S}^\ell = \{\mathbf{s}_i\}_{i=1}^n$ at the same time. We update the dictionary and sparse codes iteratively,

1. solve the elastic net (Eqn. 3) for each training data by fixing the dictionary $\mathbf{B}$;

2. update the dictionary $\mathbf{B}$ by the Lagrange dual method [10] given the sparse codes $\mathbf{S}^\ell$.

Figure 2 shows a typical dictionary learned in the first layer where the atom signals are similar to Gabor filters. After sparse coding, the set of raw features $\mathbf{X}^{\ell-1}$ are mapped into a matrix $\mathbf{S}^\ell$ where each column is a sparse code $\mathbf{s}_i$.

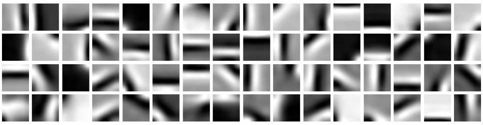

Figure 2: The first layer dictionary learned from the Caltech101 image database.

3.2 Saliency Pooling

Saliency pooling extracts translation invariant distinctive features from the sparse codes $\mathbf{S}^\ell$. We first partition the working domain into a set of M disjoint pooling sub-windows based on the grouping grid, *e.g.* $\Omega^\ell = \bigcup_{m=1}^M \Delta_m$. Accordingly, the sparse codes are partitioned into M subsets $\{\mathbf{S}^\ell_{\Delta_m}\}_{m=1,2,\ldots,M}$. We note that the partition can be operated in multiple scales. The pooling function works on the subsets of sparse codes in the sub-window Δ_m independently. Typical pooling functions include average, max, and energy. Here we use max pooling due to its robust statistical characteristics, manifested in recent literature [8][25]. Note that the pooling function operates row-wise on the matrix of sparse codes $\mathbf{S}^\ell$ corresponding to the same atom signal in $\mathbf{B}$. The max pooling function p is defined as

$$p(\mathbf{S}^\ell_\Delta) = \max(|\mathbf{S}^\ell_\Delta|), \tag{8}$$

where $|\cdot|$ denotes the point-wise absolute value. Real-world object images almost always contain some background clutters. As pooling is a process to select features spatially without notion of foreground objects, features extracted from backgrounds may have stronger responses to the learned dictionary. Thus this operation alone may inevitably introduce noise into the representation. To cope with this problem, we propose a pooling function weighted by saliency map to help pooling operation focus on distinctive features and improve the representability. Figure 3 illustrates the saliency pooling in the working domain. We compute the saliency map $\mathbf{w}^\ell$ within the working domain Ω^ℓ, and the saliency weighted max pooling function is as follows

$$p(\mathbf{S}^\ell_\Delta) = \max(|\mathbf{S}^\ell_\Delta| \cdot \mathbf{w}^\ell). \tag{9}$$

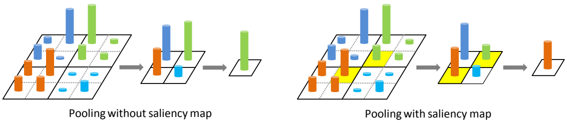

Figure 3: Illustration of saliency pooling. The yellow marked regions represent the salient parts. Although the sparse codes in the salient regions are not the maximum, the saliency pooling helps identify those sparse codes.

For each pooling regions of working domain, we compute the pooled sparse codes $\mathbf{z}_m^\ell = p(\mathbf{S}_{\Delta_m}^\ell)$.

3.3 Local Grouping

After saliency pooling, local invariant features $\mathbf{z}_m^\ell$ are grouped within working domain for more structured representation

$$g : \mathbf{y}^\ell = [(\mathbf{z}^\ell)_1^\top, (\mathbf{z}^\ell)_2^\top, \ldots, (\mathbf{z}^\ell)_M^\top]^\top. \tag{10}$$

Fig. 4 illustrates the local grouping in successive layers. As the size of working domain is increased $\Omega^{\ell-1} \subset \Omega^\ell$, the grouped features can describe more complex image content. These grouped features $\mathbf{y}^\ell$ are usually high dimensional, and thus we apply a dimensionality reduction technique (e.g., principal component analysis) to obtain a low-dimensional compact representation.

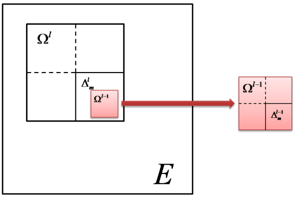

Figure 4: Illustration of local grouping in successive layers. The current working domain Ω^ℓ is partitioned into 2×2 pooling sub-windows $\Delta_1^\ell, 2, 3, 4$. The working domain of the lower layer $\Omega^{\ell-1}$ is restricted within pooling sub-windows.

$$\mathbf{x}^\ell = (\mathbf{P}^\ell)^\top \mathbf{y}^\ell, \tag{11}$$

where $\mathbf{P}^\ell$ is the PCA projection matrix. This simple yet effective spatial grouping process is a key component in our base model for constructing low-dimensional structured representations. By applying sparse coding, saliency pooling and local grouping sequentially, we obtain the current layer representation $\mathbf{x}^\ell$ that can be fed into the next layer.

4 Learning the Hierarchy

To construct a hierarchical representation, we need to learn the base model layer by layer. By removing the domain notations for presentation clarity, learning hierarchical image representation is defined by a recursive function

$$\mathbf{x}^n = \underbrace{f \circ f \circ \cdots \circ f}_{n}(\mathbf{x}^0). \tag{12}$$

In each layer, the base model function f is bounded so as the recursive function. Thus, the variations of hierarchical representations can be well controlled. Similar to other hierarchical models, it is not clear how the optimal number of layers can be learned easily. In this paper, we only use two layers in our HSSL model for object recognition.

5 Experiments

In this section, we present implementation details of our HSSL model and experimental results on object recognition using several benchmark datasets, with comparisons to reported findings in the literature. More analysis and results can be found in the supplementary material. All the experiments are carried out on a 2.8 GHz desktop computer with MATLAB and GPU implementation.

5.1 Caltech 101

We first validate our HSSL model with object categorization experiments using the Caltech 101 image database. All the 8,677 images of 101 object categories are used in our experiments.

Whitening. The images from the Caltech101 dataset are normalized similarly to [8]. First each input image is converted to grayscale and resized by bicubic interpolation so that the largest dimension is 151 pixels while preserving its aspect ratio. Each image is then locally normalized over 9 × 9 neighborhoods. That is, each pixel is subtracted by the mean pixel value in a 9 × 9 window centered at that pixel and divided by the standard deviation of this window, if it is greater than the standard deviation of the whole image. Finally, each image is zero-padded to have 143 × 143 pixels. Hereinafter, we will use the same whitening method to pre-process the images.

First layer. We randomly extract 100,000 8 × 8 patches from the pre-processed images and train a dictionary with 64 atom signals. For each input image, we compute the local sparse codes of 8 × 8 patches with a sift of one pixel over the entire image. This results in 136 × 136 64-dimensional feature maps. We pool the sparse features within each 4 × 4 non-overlapping window with max operator and generate 34 × 34 64-dimensional feature maps. Note that saliency is not used to weight pooling in this layer since the working domain is small the saliency does not influence the pooling significantly. We group the pooled features within each 4 × 4 window to form the 1024-dimensional feature vectors. These features are computed at each pixel and thus we have 31 × 31 1024-dimensional feature vectors. The feature vectors are then projected down to a 96-dimensional subspace spanned by the largest principal components.

Second layer. In this layer, we train a dictionary of 2048 bases using a set of 100,000 features randomly selected from the first layer. As a result, we obtain 31 × 31 2048-dimensional feature vectors. Next, we use a variant of [1] to compute saliency map of 31 × 31 dimensions. The max pooling operator is performed on the saliency map within 1 × 1, 2 × 2 and 4 × 4 sub-windows, and they are grouped into a single feature vector of $43,008$ dimensions.

Classification. We follow the same protocol commonly adopted in experiments with the Caltech101 dataset. In each experiment, we use a training set of either 15 and 30 images randomly selected from each category and test all the other images. We train linear support vector machines (SVMs) with one-versus-all paradigm for multi-class classification using LIBSVM [2]. Experimental results (average classification accuracy per category) are reported by repeating 10 trials with different random selection of training samples.

We first compare our method with biologically-inspired [21] [14] [17] and deep learning methods [8] [11] [26]. Using 30 training samples per class, our HSSL model outperforms the state-of-the-art biologically inspired method by 9%, and outperforms the best deep learning methods by 9.2%. We also compare HSSL with SIFT-based methods [9] [25]. The proposed method outperforms the leading mehod [25] by almost 3%. The results are summarized in Table 1. For the sensitivity analysis of algorithm parameters, please refer to the supplementary materials.

Table 1: Experimental results with Caltech 101 dataset.

	Method	15 samples	30 samples
Bio-inspired	Serre [21]	51.0%	42%
	Mutch [14]		56%
	Pinto [17]		67%
Deep Learning	Jarrett [8]		65.5%
	Lee [11]	57.7%	65.4%
	Zeiler [26]	58.6 ± 0.7%	66.9 ± 1.1%
	HSSL	**68.7± 0.4%**	**76.1± 1.3%**
SIFT-based	Lazebnik [9]	54.0%	64.6 ± 0.8%
	Yang [25]	67.0 ± 0.45%	73.2 ± 0.54%

5.2 Caltech 256

The Caltech 256 dataset contains 30,607 images of 256 object categories and one clutter background. The minimum number of images in any category is 80. Compared with the Caltech 101 dataset, it contains much more intra-class variability in terms of object size, appearance and location. In the first layer, We train a dictionary of 64 atom signals and use the same parameters as in the experiment of Caltech101 to perform saliency pooling and local grouping. In the second layer, we train a large dictionary of 6144 atom signals to cope with large variations between and within categories. Following the common setup, we test our method with 15, 30, 45 and 60 training samples per class, respectively. As shown in Table 2, our HSSL model consistently outperforms the SIFT-based methods.

Table 2: Experimental results with Caltech 256 dataset.

Methods	15 samples	30 samples	45 samples	60 samples
Griffin [4]	28.3%	34.1%		
Yang [25]	27.8 ± 0.51%	34.0 ± 0.35%	37.5 ± 0.55%	40.1 ± 0.91%
HSSL	**29.8± 0.4%**	**35.4± 0.4%**	**38.7±0.3%**	**41.6± 0.3%**

5.3 Oxford Flowers

The Oxford Flowers dataset [15] includes 17 different categories of flowers with 80 images for each class. These images have large scale, pose and illumination variation, and some flowers are visually very similar. The smallest width or height of any image is 500 pixels. In our experiments, we resize the images so that have the largest dimension is 300 pixels and zero-padding them to have canonical 300 × 300 pixels. The dictionaries for the first layer and the second layer have 64 and 2048 atom signals, respectively. Similarly, the max pooling operator is performed on the saliency map within 1×1, 2×2, 4×4 and 8×8 sub-windows. Table 3 presents the results using our HSSL model and two leading methods [15][23]. In Nilsback's work, they use 40 images per class for training, 20 images per class for validation, and the remaining 20 images for tests. For fair evaluation, we first compare our method with Varma's work [23] without using the validation set. In this setting, we obtain an average classification accuracy of 69.7%, which is better than any single feature results in his work. Next we compare our method with Nilsback's work [15]. Without using a validation set for parameter selection, we add the validation set into the training set (i.e., 60 training samples per class). Our method achieves accuracy of 76.2% which is better than Nilsback's work with single features (73.7%). In the reported experiments, segmentation prior are used to extract flower regions from images [23][15], whereas we do not use this prior. Both Nilsback [15] and Varma [23] exploit feature fusion for better results. When different types of features are effectively combined, the classification accuracy can be further improved.

Table 3: Experimental results with Oxford Flowers dataset.

	Varma [23]			Ours	Nilsback [15]			Ours
Segmentation	Yes			No	Yes			No
Training Samples	40			40	60 (40 training + 20 validation)			60
Test Samples	20			20	20			20
Feature	Color	Shape	Texture	HSSL	Color	Shape	Texture	HSSL
Aver. Classif. Acc.	59.7 ±2.0%	68.9 ±2.0%	59.0 ±2.1%	**69.7 ±2.7%**	73.7%	71.8%	55.5%	**76.2 ±3.8%**

6 Conclusions

In this paper, we present a deep learning model for hierarchical image representation in which we build the hierarchy by stacking up the base models layer by layer. We exploit three basic structures of visual data: sparsity, saliency and locality in our base model. We evaluate the proposed HSSL model in the task of object categorization. The experimental results show that our HSSL model is able to extract effective feature hierarchy from unlabeled images. Plus a linear SVM classifier, the learned features outperform the state-of-the-art deep learning methods. Our model learns representation directly from the image data, without using hand-crafted feature descriptors. In this sense, our model is more adaptive to different vision tasks even when lack of expert knowledge.

References

[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *CVPR*, 2009.

[2] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3336–3745, 2006.

[4] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. Technical Report TR-2007-001, California Institute of Technology, 2007.

[5] Geoffrey E. Hinton and Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:2006, 2006.

[6] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154, 1962.

[7] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene anaysis. *PAMI*, 20(11):1254–1259, 1998.

[8] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, 2009.

[9] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[10] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006.

[11] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009.

[12] David Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[13] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman and Company, 1982.

[14] Jim Mutch and David G. Lowe. Object class recognition and localization using sparse features with limited receptive fields. *IJCV*, 80(1):45–57, 2008.

[15] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, 2006.

[16] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997.

[17] Nicolas Pinto, David D. Cox, and James J. Dicarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4(1):e27+, 2008.

[18] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *ICML*, 2009.

[19] Marc'Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with application to object recognition. In *CVPR*, 2007.

[20] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[21] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 29(3):411–426, 2007.

[22] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 1997.

[23] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007.

[24] Allen Y. Yang, Arvind Ganesh, Zihan Zhou, S. Shankar Sastry, and Yi Ma. *A review of fast l1-minimization algorithms for robust face recognition*, 2010. http://arxiv.org.

[25] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[26] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *CVPR*, 2010.

[27] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67:301–320, 2005.