Pattern Recognition 48 (2015) 2645–2655

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Detection guided deconvolutional network for hierarchical feature learning

Jing Liu*, Bingyuan Liu, Hanqing Lu

National Laboratory of Pattern Recognition, CASIA, Beijing 100190, China

ARTICLE INFO

Article history:
Received 2 September 2014
Received in revised form
2 January 2015
Accepted 2 February 2015
Available online 11 February 2015

Keywords:
Image representation
Deep leaning
Object recognition

ABSTRACT

Deep learning models have gained significant interest as a way of building hierarchical image representation. However, current models still perform far behind human vision system because of the lack of selective property, the lack of high-level guidance for learning and the weakness to learn from few examples. To address these problems, we propose a detection-guided hierarchical learning algorithm for image representation. First, we train a multi-layer deconvolutional network in an unsupervised bottom-up scheme. During the training process, we use each raw image as an input, and decompose an image using multiple alternating layers of non-negative convolutional sparse coding and max-pooling. Inspired from the observation that the filters in top layer can be selectively activated by different high-level structures of images, *i.e.*, one or partial filters should correspond to a particular object class, we update the filters in network by minimizing the reconstruction errors of the corresponding feature maps with respect to certain object detection maps obtained by a set of pre-trained detectors. With the fine-tuned network, we can extract the features of given images in a purely unsupervised way with no need of detectors. We evaluate the proposed feature representation on the task of object recognition, for which an SVM classifier with spatial pyramid matching kernel is used. Experiments on the datasets of PASCAL VOC 2007, Caltech-101 and Caltech-256 demonstrate that our approach outperforms some recent hierarchical feature descriptors as well as classical hand-crafted features.

1. Introduction

How to build a suitable image representation is one critical problem for computer vision tasks. Some manually designed image descriptors (*e.g.* Gabor, SIFT and HOG) have precipitated dramatic success in past decades. Although these hand-crafted features may benefit from human ingenuity and prior domain knowledge to some extent, it is task-dependent and difficult to detect more complex features, including mid and high-level structures in images.

Over the recent years, a growing amount of researches focus on deep learning models for hierarchical image representations [24,20,37]. The deep architecture is motivated by the hierarchical nature of human vision cortex, and attempts to learn the hierarchical structures in a data-driven manner. Therefore, it can easily borrow knowledge from the human cognitive process and be applied to various vision tasks.

Usually, deep architectures consist of feature detector units arranged in layers. Lower layers detect simple features and feed into higher layers, which in turn detect more complex features. However, two important properties are neglected in most of previous hierarchical feature learning schemes. One is the response selectivity of nodes in each layer of hierarchy. 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 [6] and even specific people [26]. That is, class-specific neurons exist in the human brain. The other is the lack of high-level concept as a type of top-down guidance. Expert suggestions and personal experiences usually have great influence on the human cognitive process. Thus, how to incorporate the selective property and certain high-level structure assumptions for feature learning is essential to enhance the performance of deep architectures. It is also noted that the recently popular deep convolutional neural network [21] requires a large number of labeled images to train. However, people and animals are able to learn from very few examples. We attempt to train an effective hierarchical model with fewer training samples by incorporating more high-level guidance.

To address above problems, we propose a detection-guided deconvolutional network for hierarchical feature learning, as shown in Fig. 1. In particular, we require the top-layer filters that have selective activations from different object categories and incorporate object detection maps as a kind of high-level guidance for network learning. First, we train a multi-layer deconvolutional network in an unsupervised scheme, which can be easily stacked by alternating non-negative

* Corresponding author. Tel.: +86 10 82544507; fax: +86 10 82544594.
E-mail address: jliu@nlpr.ia.ac.cn (J. Liu).

http://dx.doi.org/10.1016/j.patcog.2015.02.002
0031-3203/© 2015 Elsevier Ltd. All rights reserved.

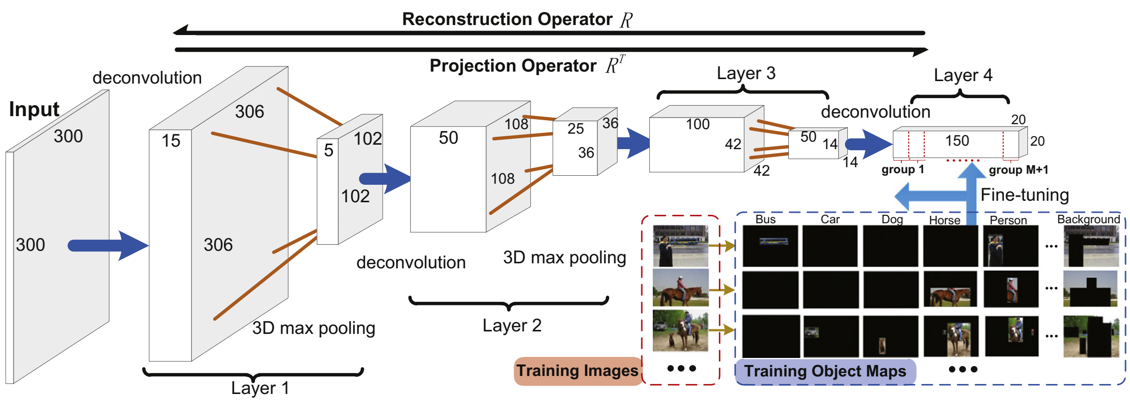

Fig. 1. Deep architecture of our proposed model. We firstly train a deep deconvolutional network in an unsupervised scheme. Then we fine-tune the network with the guidance of training object maps generated by object detectors.

convolutional sparse coding and max-pooling operations. Each layer learning is to decompose an input as a linear sum of 2D feature maps convolving with filters by minimizing the reconstruction error while encouraging sparsity on feature maps. The non-negativity property is enforced on the feature maps to obtain more sensible and explainable features, which is inspired from some biological evidences [17]. Second, we use the detection maps obtained by a set of pre-trained detectors to guide the learning of the top layer and fine-tune the networks below. We divide the filters of the top layer into several groups with each corresponding to a particular object category, and the cost function for fine-tuning is to minimize the reconstruction error of each group of feature maps with respect to the corresponding object detection maps. We also use the top-down influence of the detection information to update the filters in the lower layers. With the trained network, each layer can capture diverse image information: low-level edges, mid-level edge junctions, high-level object parts and complete objects (as discussed in Section 4). Then we can extract an over-complete image representation with only raw image pixels as an input and utilize it as the input of SVM classifier with a spatial pyramid matching kernel to perform the task of object recognition. Extensive experiments on the benchmark datasets of PASCAL VOC 2007, Caltech-101 and Caltech-256 demonstrate the effectiveness of the proposed method compared with other related works.

The rest of the paper is organized as follows. Section 2 reviews the related works of hierarchical feature learning models. In Section 3, we elaborate our proposed architecture, present how to pre-train the network in an unsupervised scheme and fine-tune with the object detection guidance. The experimental evaluation is given in Section 4, and we conclude in Section 5.

2. Related work

Many researches have focused on building a hierarchical representation model as an alternative to manually designed feature. One class of the hierarchical feature extracting framework is a family of biologically-inspired heuristic models [27,29]. These models are mainly motivated by the simple-complex cell model of mammal vision system [19] and constructed by alternating between convolutional filtering and max pooling, extracting features without any training procedure.

Convolutional Neural Network (CNN) is firstly proposed by LeCun et al. [23], which is trained purely supervised by a back propagation algorithm. Some recent extension works further improve the performance of CNN. Hinton et al. [16] proposed “dropout” by randomly omitting half of the feature detectors on each training case to prevent overfitting and Wan et al. [32] generalized this idea by setting a randomly selected subset of weights within the network to zero for regularizing large fully-connected layers. Zeiler and Fergus [35] proposed to replace the conventional deterministic pooling operations with a stochastic procedure by randomly picking the activation with each pooling region according to a multinomial distribution. Recently, there has been a surge of interest in CNN as it achieves impressive results in many real and difficult situations, *e.g.*, image classification [21,36], object detection [11] and face recognition [31]. This is due to the available large-scaled tagged training samples and scalable computation resources such as thousands of CPU cores and GPU. However, CNN requires large training examples to achieve excellent performance and may fail with small datasets [1], while human brains have the ability to learn from very few examples. Our proposed method is able to train an effective deep model with small datasets (*e.g.* PASCAL VOC and Caltech-101).

Another class of deep feature learning algorithms is based on the encoder–decoder architecture [18]. The input is fed to the encoder which produces a feature vector and the decoder module then reconstructs the input from the feature vector with the reconstruction error measured. It is trained to minimize the average reconstruction error with certain constraints for good property, *e.g.* sparsity.

Our proposed model is developed with the deep learning framework of stacking shallow generative model by greedy layer-wise unsupervised learning scheme. Deep Belief Networks (DBN) [15] builds multiple layers of directed sigmoid belief nets with the top layer as a Restricted Boltzmann Machines (RBM). It is trained layer by layer to maximize the likelihood of the input. Lee et al. [24] extended DBN with convolution operation for the purpose of extracting latent features from raw image pixels. Yu et al. [34] proposed a hierarchical sparse coding model to learn image representations from local patches. The closest models to ours are those based on convolutional sparse coding. Kavukcuoglu et al. [20] proposed to extract features through learned convolutional filter banks and construct a multi-stage convolutional network. In [38], a deconvolutional network is developed by alternately stacking convolutional sparse coding and max-pooling layers. Different from [38], we further constrain the network with non-negative property motivated by the biological evidence [17]. Most of these models neglect the selective property and trained in an unsupervised scheme without any top-down guidance of high-level information, while our model integrates the object detectors as a high-level guidance to fine-tune the network. Some previous works have shown that the performance of object recognition can be obviously improved by incorporating object detection or localization [13,30], while our method explores this information to learn better image representations.

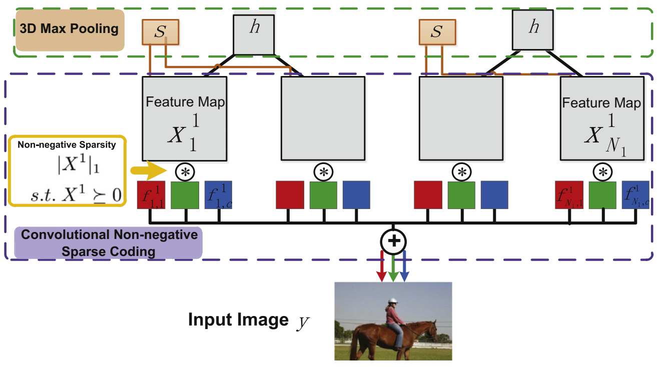

Fig. 2. Single Deconvolutional Network Layer, consisting of convolutional non-negative sparse coding and 3D max pooling.

3. The proposed deconvolutional network

3.1. Single layer

We firstly introduce a single deconvolutional network layer, consisting of two operations: convolutional non-negative sparse coding (deconvolution) and max-pooling, as depicted in Fig. 2. Taking an image y of N_0 color channels $y_1, y_2, \ldots, y_{N_0}$ as an input, the model decomposes each channel as a linear sum of N_1 latent 2-D feature maps $x^1_i (i = 1, 2 \ldots, N_1)$ convolving with filters $f^1_{i,j} (i = 1, 2, \ldots, N_1, j = 1, 2, \ldots, N_0)$. To yield a stable solution, the objective of the deconvolution is to minimize the reconstruction error while encouraging sparsity on the learned feature maps. Thus the cost function of the single layer is

$$\min_{x^1, F^1} C_1(y) = \frac{\lambda_1}{2} \sum_{c=1}^{N_0} \| \sum_{i=1}^{N_1} x^1_i * f^1_{i,c} - y_c \|_2^2 + \sum_{i=1}^{N_i} | x^1_i |_1$$
$$\text{s.t.} \quad x^1_i \geqslant 0 \qquad (1)$$

Different from [38], we enforce non-negative sparsity to further constrain the solutions to be both sparse and non-negative. For notational brevity, we combine the convolution and summing operations into a single operation matrix F^1, convert x^1_i into a vector x^1, and denote the reconstruction as $\hat{y}^1 = F^1 x^1$.

After the deconvolution, a 3-D max-pooling operation is performed on the feature maps x^1, to introduce some local invariance property and allow the next layer to learn higher level structures at a larger scale. The max-pooling is 3-D because it occurs both within each feature map and between adjacent maps. Non-overlapping 3-D cubes are employed to partition the feature maps with both values and location switches of the maximum recorded. This operation is denoted as $[h, s] = P(x)$, where h and s indicate the pooled maps and location switches respectively. The 3-D max pooling operation is nonlinear. However, if s is fixed it becomes a linear operation as $h = P_s x$, where P_s plays the role of a binary selection matrix. On the contrary, a max unpooling operator U_s should be defined for reconstruction, which takes the elements of h back into x at the coordinates recorded in s with the remaining elements set to zero. Note that this is also a linear operation when s is fixed and denoted as $\hat{x} = U_s h$.

3.2. Multiple layers

With the single layer module described above, we can easily build deep networks by stacking the single layer in a hierarchical fashion and treating the pooled maps h^l of layer l as an input for layer $l+1$, as shown in Fig. 1. The number of feature maps N_l in each layer may vary. For high-level layer we calculate the reconstruction $\hat{y}^l$ through the filters and location switches of the current layer and below. The higher layer is also trained with respect to the original input image, rather than the input maps of this layer. In other words, the objective is to keep the reconstruction $\hat{y}^l$ close to the original input image y. Therefore, the cost function of layer l is formulated as

$$\min_{x^l, F^l} C_l(y) = \frac{\lambda_l}{2} \| \hat{y}^l - y \|_2^2 + \sum_{i=1}^{N_l} |x^l_i|_1$$
$$\text{s.t.} \quad x^l_i \geqslant 0 \qquad (2)$$

For notational brevity, a reconstruction operator R_l is defined, taking the feature maps x^l of layer l backward down to the original input space with alternative convolution and unpooling operations: $\hat{y}^l = F^1 U_{s_1} F^2 U_{s_2} \ldots F^l x^l = R^l x^l$. For simplicity of the gradient computation, a projection operator $(R^l)^T$ is also defined to project the input of the network forward up to the feature maps of the layer l: $(R^l)^T = (F^l)^T P_{s_{l-1}} (F^{l-1})^T P_{s_{l-2}} \ldots P_{s_1} (F^1)^T$, where $(F^l)^T$ is the flipped version of the filters F^l. Note that given the location switches s, both the reconstruction and projection operators are linear.

3.3. Inference phase

The efficiency of inference and learning algorithms is crux for a hierarchical model. In the inference phase, the objective is to find the optimized feature maps x given an input image y, with the filters F of all the layers fixed. In this paper, we employ the efficient ISTA [2] scheme layer by layer, which is an iteration algorithm including gradient and shrinkage steps.

Gradient step: In ISTA, the gradient of the reconstruction error term in Eq. (2) with respect to the feature maps x is needed:

$$\nabla x^l = \lambda_l (R^l)^T (R^l x^l - y) \qquad (3)$$

To compute this gradient, we firstly reconstruct the feature maps to the input space to calculate the reconstruction error, and then propagate it forward up to the current layer by the projection operator $(R^l)^T$. The gradients are actually computed by alternately perform the convolution and pooling operations layer by layer, which is similar to the back propagation algorithm. Taking the gradient ∇x^l, we update the feature maps x^l:

$$x^l = x^l - \beta_l \nabla x^l \qquad (4)$$

where β_l denotes the size of the gradient step.

Shrinkage step: Following the gradient step, a per-element shrinkage operation is performed to clamp small elements in x^l to zero, increasing its sparsity:

$$x^l = \max(|x^l| - \beta_l, 0) \qquad (5)$$

Note that the non-negative constraint is satisfied by projecting the solution into the non-negative set.

A single ISTA iteration consists of the two steps above, where all the operations are amenable to parallelization. Usually several iterations are needed to obtain a stable and satisfied solution.

3.4. Learning phase

The learning objective is to estimate the parameters of the networks, *i.e.*, filters of all layers. When training the networks, both x^l and F^l are unknown in the cost function. Thus we alternately minimize the cost function $C_l(y)$ over the feature maps with filters fixed, and then minimize with respect to filters while keeping the feature maps fixed. To update the filters, the gradient of the cost function with respect to filters in current layer is needed:

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

where the gradient is a convolution between two terms. The left term is the reconstruction error propagating up to layer $l-1$, while the right term is the feature map of layer l. Thus this can be easily computed for all the filters of the current layer simultaneously. In the implementation, the linear conjugate gradients (CG) algorithm is utilized to update the filters as the model is linear in F^l with the feature maps and pooling switches fixed.

3.5. Object detection guided fine-tuning

After greedily learning the filters in every layer, we fine-tune the model with the object detection maps obtained by a set of pre-trained detectors as top-down guidance. Firstly, we generate a set of training object maps set prepared for fine-tuning by decomposing the training images with the object detectors trained by the DPM [9] model. Suppose the number of object categories is M, all the M detectors are applied to obtain the estimated bounding boxes of the objects within the image. Then we separately use the object bounding boxes to dig out the objects in the image by remaining the intensities within the bounding box while setting the intensities outside to zero, as shown in Fig. 1. Thus M object maps with only the object regions for the particular classes remained are obtained for each image. We also add a background map indicating the rest area of the image without any objects, since the context information is also important to understand an image. The object maps set is denoted as $O = \{o_1, o_2, \ldots, o_{M+1}\}$, where $o_1, o_2, \ldots, o_M$ denote the M classes of object maps and o_{M+1} represents the background map. For a particular image, only a small subset of O is positive because of the limited number of objects within the image.

Algorithm 1. Object detection guided fine-tuning.

```
Input:
        Training set $Y$, Detection maps set $O$, Number of layers $L$,
        Epochs $E$, ISTA steps $T$, Regularization coefficients $\lambda_l$
Output:
        Filters $F$, Feature maps $X$, Location switches $S$
1:  Init. filters $F^l$ by the unsupervised pre-training
2:  for $epoch = 1 : E$ (Epoch iteration) do
3:    for $i = 1 : N$ (Loop over images) do
4:      Inference phase
5:      for $l = 1 : L$ (Loop over layers) do
6:        Solve Eq. (2) by T steps of ISTA and obtain $x^l$
7:        Pooling operation: $[h^l, s^l] = P(x^l)$
8:      end for
9:      Solve Eq. (7) by ISTA and update $x^L$
10:     Learning phase
11:     Compute gradient $\nabla f^L$ by Eq. (8) and update $f^L$ by CG
12:     for $l = L-1 : 1$ (Loop from top to bottom) do
13:       Compute gradient $\nabla f^l$ by Eq. (9) and update $f^l$ by CG
14:     end for
15:   end for
16: end for
```

In the fine-tuning, to improve the selective property we hope each top-layer filter only responds to a particular class of object while keeping silent for all the other categories. In the context of our model, this means that the reconstruction of a certain feature map is only close to a particular category of objects within the original images. We divide the filters of the top layer L into $M+1$ groups $(F^L)_{g_m}$ $(m = 1, 2, \ldots, M+1)$ and let each group correspond to a particular object category. With the mth group of feature maps denoted as $(x^L)_{g_m}$ the corresponding reconstruction is $\hat{o}_m^L = R^{L-1} U_{s_{L-1}} (F^L)_{g_m} (x^L)_{g_m}$. Thus we define the cost function of the fine-tuning phase as

$$\min_{x^L, F^L} C_L(O) = \frac{\lambda_L}{2} \sum_{m=1}^{M+1} \|\hat{o}_m^L - o_m\|_2^2 + \sum_{i=1}^{N_L} |x_i^L|_1$$
$$\text{s.t.} \quad x_i^L \geqslant 0 \tag{7}$$

This is to minimize the sum of errors between the reconstruction of each feature map group $\hat{o}_m^L$ and the corresponding object maps o_m. In this way, we lead each filter in the top layer only focusing on a particular class of object in the image. Since the cost function is in the same form as Eq. (2), we also adopt ISTA and CG algorithms to solve it.

Beyond updating the top-layer filters guided by the object detection maps, we fine-tune all the lower layers through the top-down propagated influence of this information, to further enhance the selectivity and discrimination property of the whole network. It can be performed by the CG algorithm by utilizing the gradient of the cost function with respect to each layer's filters:

$$\nabla f_{i,j}^L = \lambda_L [P_{s_{L-1}} (R^{L-1})^T (\hat{o}_m^L - o_m)]_i * x_j^L, \quad j \in g_m \tag{8}$$

$$\nabla f_{i,j}^l = \lambda_l \sum_{m=1}^{M+1} [P_{s_{l-1}} (R^{l-1})^T (\hat{o}_m^L - o_m)]_i$$
$$* [U_{s_l} F^l \ldots (F^L)_{g_m} (x^L)_{g_m}]_j, \quad 1 \le l < L \tag{9}$$

Eq. (8) is the gradient with respect to the filters in the top layer, which is similar to Eq. (6) except that only the reconstruction error of the mth group of feature maps is computed. Eq. (9) is the gradient with respect to the filters of the lower layer l, where the left term is the sum of errors with respect to each object map propagated to layer $l-1$ and the right term is the top down reconstruction to the current layer l. This indicates the property that the filters are affected by both the bottom-up generative and top-down high-level information. While the pre-training of the networks is performed from bottom layer to top, the fine-tuning is opposite from top to bottom. The overall training algorithm of the fine-tuning is summarized in Algorithm 1.

3.6. Object recognition pipeline

With the fine-tuned network, we extract the features of a given image with no need of detectors by decomposing the input image into multiple layers of feature maps. To perform object recognition, our model must be combined with a supervised classifier. Here we turn to the Spatial Pyramid Matching (SPM) [22] model, which is demonstrated to be very effective in image classification task [33]. In this section, we will describe the pipeline to combine our feature learning model with the SPM classifier.

Given a new image y as an input, we build spatial pyramid representation for each layer by the obtained feature maps and switch configurations. For the 1st layer, we directly use the feature maps $x_i^1 (i = 1, 2, \ldots, N_1)$ as activations of this layer to extract the SIFT-like feature representation. Similar to dense SIFT, each feature map in this layer is densely split into overlapping 16×16 patches. Then the absolute value of activations in each patch is pooled in 4×4 regions and the pooled values of all the adjacent feature maps are concatenated leading to a $16 \times N_1$ dimensional descriptor. Replacing SIFT by this descriptor, we can compute a spatial pyramid representation for the first layer. With respect to higher layers, although the filters are shared between images, the location switches are not. Thus the feature maps of two images are not directly comparable as they use different bases R^l. In the top layer of our model, we lead the filters of group m responding more strongly to the mth object category. To take advantage of this discriminative property, we propose a novel approach to apply the learned features. We only remain the largest activations within each feature map in layer l and separately reconstruct each one down to the input space. Then N_l different reconstructed images $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{N_l})$ are obtained with each one containing certain image parts. After that, we respectively use the corresponding feature maps in the 1st layer of these N_l reconstructions $(\hat{x}^{1,1}, \hat{x}^{1,2}, \ldots, \hat{x}^{1,N_l})$ in the same way that we adopt to treat the actual 1st layer feature maps, leading to N_l spatial pyramid representation. By averaging the N_l representation, we get the representation corresponding to layer l. The obtained spatial pyramid representation in each layer may be concatenated as the final feature to train SVM classifiers to further improve the performance.

4. Experiments

Our experiments are conducted by the Deconvolutional Networks toolbox developed by Zeiler [38], the GPU library GPUmat[1] and the C++ GPU convolutional library.[2] The entire model is trained on a 24-core CPU and a Nvidia Tesla M2075 GPU for parallelization and fast computing. We evaluate our model on three image benchmarks: PASCAL VOC 2007, Caltech-101 and Caltech-256.

We implement a 4-layer model with sizes of filters set to 7×7 for all the layers. The number of feature maps in each layer is $15, 50, 100, 150$ and the pooling sizes are $3 \times 3 \times 3, 3 \times 3 \times 2, 3 \times 3 \times 2, 3 \times 3 \times 2$ respectively, as elaborated in Fig. 1. Before inputting into the model, each image is converted to gray-scale and resized to 300×300 (zeros padding to preserve the aspect ratio). A local subtraction with Gaussian filter is also performed and the intensities of the image are finally normalized to $[0, 1]$. In the fine-tuning phase, we directly employ the ground truth of the object bounding boxes to create the training object maps set since the performance of the state-of-the-art object detector is still unsatisfied. When performing recognition combined with the SPM classifier, we densely extract descriptors from the feature maps over a grid with 3 pixels spacing in multiple scales and the size of codebook is fixed as 2000. The spatial regions are obtained by dividing the feature maps into non-overlapping regions with $1 \times 1, 2 \times 2, 3 \times 1$ grids on PASCAL VOC 2007 and $1 \times 1, 2 \times 2, 4 \times 4$ on Caltech-101 and Caltech-256. χ^2 kernels are adopted to train SVM classifiers.

In the following subsections, we firstly in-depth analyze our model on PASCAL VOC 2007 by visualizing the learned filters and feature maps. Then we report the recognition performance on the three datasets. We mainly compare our method with some famous hand-designed descriptors (*e.g.* HOG [5], SIFT and its color-variant Opponent-SIFT [28], Object-Bank [25]), and some hierarchical feature learning models (*e.g.* CDBN [24], ConvSC [20], HSC [34], DN [38]).

4.1. Results on PASCAL VOC 2007

The PASCAL VOC 2007 dataset [7] is widely used as test bed for evaluating algorithms for image understanding tasks. This database is considered to be an extremely challenging one because all the images are natural photos obtained from Flicker where the size, viewing angle, illumination, appearances of objects and their poses vary significantly, with frequent occlusions. The dataset consists of 9963 images from 20 classes, where 5011 images are used for training and validation, and 4952 images for testing. The classification performance is evaluated by the Average Precision (AP) measure, a standard metric used by PASCAL challenge. When performing fine-tuning, we divide the first 120 feature maps of the 4th layer into 20 groups, *i.e.*, every 6 adjacent feature maps corresponding to a particular category and the rest 30 representing the background context. Our model is trained by the training and validation images and evaluated on the testing images. It takes about 6 h to entirely train a 4-layer model.

4.1.1. Model visualization

Fig. 3 shows the learned filters in all the four layers before and after fine-tuning. In the first layer, it is convenient to show what we have learned by directly visualize the filter matrixes. However, for higher layers it is meaningless by directly visualizing the filter matrixes, since the higher layer works by assembling the filters from the current layer to the bottom. To show the high-level filters, we respectively take each feature map in current layer and then pick the single largest activation over the entire training set to reconstruct it down to the input space. In other words, we show the filters by evaluating what kind of structures mostly activate them. In layer 1, we see a range of edge structures of different orientations and scales, which is similar to edge-based SIFT descriptors. Thus the recognition performance of layer 1 is very similar to SIFT. In layer 2, the filters capture some mid-level image structures, including some edge junctions, curves, parallel lines and other basic geometric elements, while the filters in layer 3 also represent some mid-level features with more complex structures. Some filters in layer 3 already capture some high-level object parts, such as the plane-like and bird-like filters. In layer 4, our model is able to capture very high-level image structures, *i.e.*, key object parts and fairly complete objects. The comparison of filters before and after fine-tuning demonstrates the effectiveness of our fine-tuning scheme, as obviously better filters are obtained after fine-tuning especially on the 3rd and 4th layer with improved selective and discriminative property and different filters in layer 4 respond to particular classes of objects (*e.g.* aeroplane, bicycle, bird, and boat). We think the unsupervised learning is effective at extracting low-level information in the lower layers, while high-level information guided fine-tuning is significant to integrate the low-level features and obtain high-level representations.

With the trained model, multiple layers of feature maps are inferred given a new image as an input. In Fig. 4, we visualize some examples of the obtained feature maps in layer 4 by picking and reconstructing the largest 3 activations respectively. It is shown that different filters focus on different classes of objects in the image. For example, the left image in the 4th row contains two categories of objects, *i.e.* person and motorbike, which respectively activate different filters. It is also shown that our model decomposes the left image in the 3rd row into bus map and person map. These evidences demonstrate that the learned filters have semantic meanings and selectively respond to different categories of objects in the input image. In the last two rows of Fig. 4, we display some hard examples that our model does not handle well. This is mainly because of the extremely large variability of PASCAL VOC datasets. For example, in the left image of the last row, the person is in such a small upper left corner of the image that our model fails to represent. For the right image in the last row, the feature maps are not able to capture the sofa and tv well, because only a small part of the sofa exists in the bottom and the tv is heavily occluded by the dog.

4.1.2. Object recognition

The recognition performances of our model on PASCAL VOC 2007 are shown in Table 1. Our best result is achieved by combining the features in all the 4 layers denoted as ‘Our($l1+l2+l3+l4$)’ and leads

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

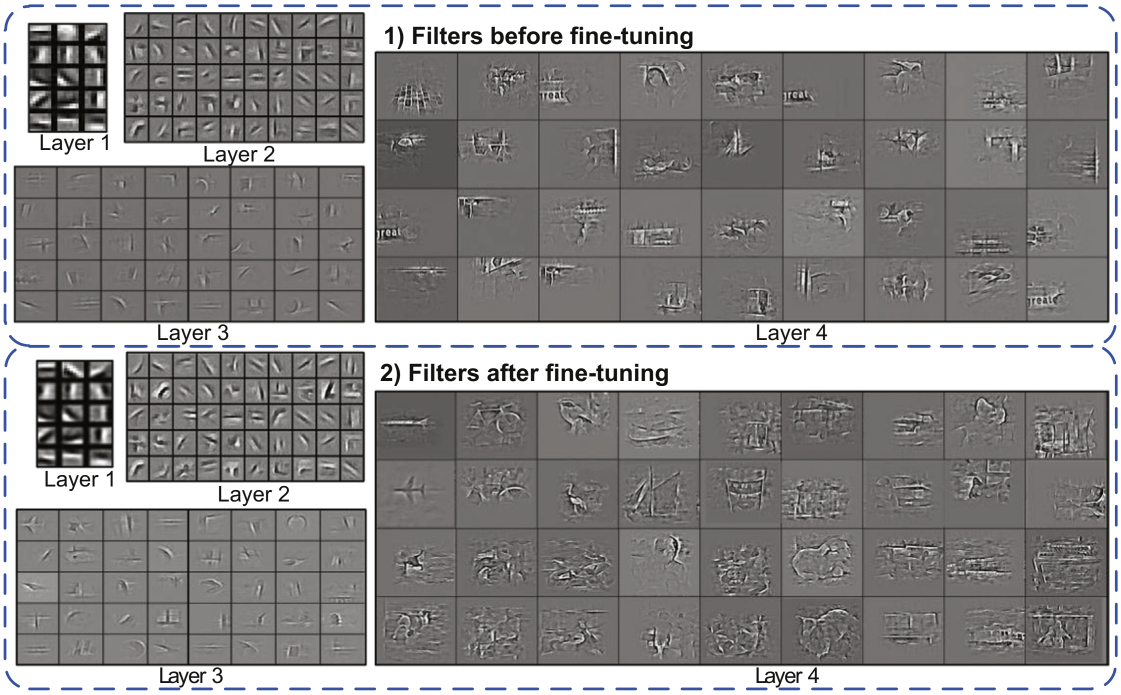

Fig. 3. The visualization of filters learned in each layer of our model. (1) Filters of all the 4 layers before fine-tuning. (2) Filters of all the 4 layers after fine-tuning.

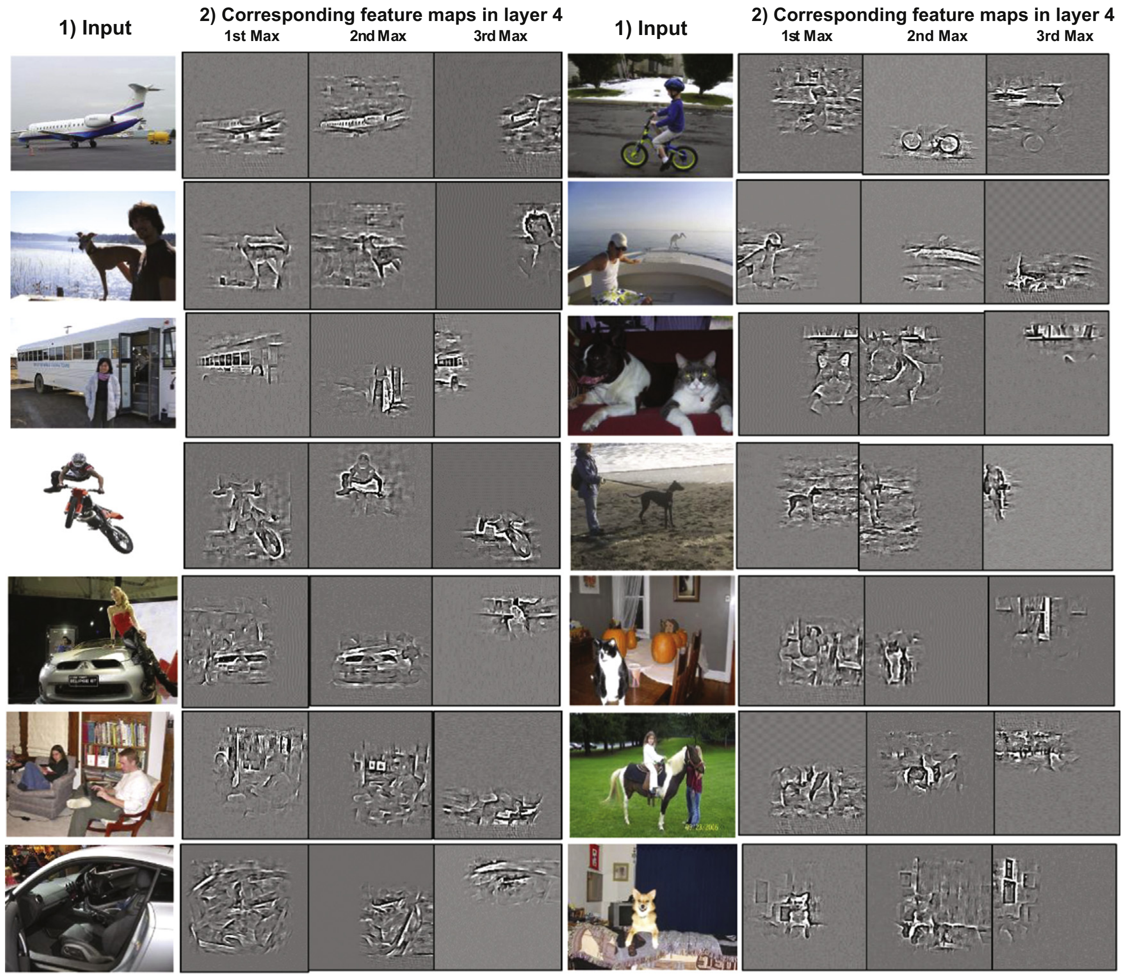
1) Input
2) Corresponding feature maps in layer 4
1st Max
2nd Max
3rd Max
1) Input
2) Corresponding feature maps in layer 4
1st Max
2nd Max
3rd Max

Fig. 4. Visualization of the feature maps in layer 4 when inputting a raw image. (1) Examples of input images. (2) Corresponding largest 3 feature maps in layer 4, visualized by reconstructing the feature maps in the 4th layer into the input space.

Table 1
Recognition performance (AP in %) comparison on VOC 2007. (The table is divided into two parts due to the limitation of space.)

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Table
HOG	69.8	40.2	27.2	51.6	19.6	49.2	67.1	44.7	47.5	29.0	34.5
SIFT [4]	68.7	57.0	39.9	64.6	22.0	58.8	73.9	53.8	52.4	38.6	49.2
OpponentSIFT [28]	69.6	**66.9**	40.3	62.8	17.4	52.3	69.3	39.8	47.9	35.7	45.1
Object bank [25]	68.7	53.4	34.6	61.8	19.8	49.9	**75.0**	42.1	48.7	28.7	50.2
DN [38]	69.0	40.8	42.2	62.3	25.3	50.1	64.1	50.8	50.9	33.7	40.8
Our ($l1$)	70.1	52.1	45.9	66.6	32.8	55.4	68.9	57.6	54.8	42.7	43.9
Our ($l2$)	69.5	53.2	47.5	66.2	32.4	56.3	67.0	56.7	54.8	45.9	43.8
Our ($l3$)	71.1	54.3	48.2	67.8	34.7	56.5	69.3	57.7	56.7	47.7	44.9
Our ($l4$)	75.0	56.3	50.0	68.3	37.5	57.2	71.6	59.6	57.8	49.0	45.8
Our ($l1+l4$)	75.5	57.7	53.9	70.1	42.3	59.7	73.4	64.0	58.9	53.2	49.6
Our ($l1+l2+l3+l4$)	**76.5**	58.5	**55.0**	**71.5**	**43.2**	**60.1**	73.9	**64.8**	**59.5**	**54.6**	**50.3**
Our(**nononnegativity**)	74.9	56.0	53.7	68.8	41.3	58.9	71.9	61.7	57.0	53.1	47.8

Method	Dog	Horse	Motor	Person	Plant	Sheep	Sofa	Train	Tv	Mean
HOG	30.9	69.1	43.7	75.2	18.3	24.8	32.7	65.4	41.0	44.1
SIFT [4]	36.9	75.6	61.6	81.6	20.5	**40.1**	50.9	73.4	**49.2**	53.4
OpponentSIFT [28]	33.3	76.0	59.2	78.2	19.8	41.7	41.5	70.4	42.7	49.9
Object bank [25]	31.8	71.4	53.1	79.6	15.6	29.0	44.3	67.3	49.0	48.7
DN [38]	36.3	77.1	49.6	79.2	20.0	20.7	47.7	66.7	41.4	48.4
Our ($l1$)	37.2	78.7	55.9	80.8	20.9	27.8	51.5	72.8	42.5	52.9
Our ($l2$)	37.6	77.4	53.9	81.9	21.6	30.0	51.3	72.5	43.2	53.1
Our ($l3$)	38.8	79.9	56.9	83.3	23.6	28.7	53.5	72.7	45.6	54.6
Our ($l4$)	40.5	80.8	57.4	84.8	25.2	29.9	55.3	74.5	46.0	56.1
Our ($l1+l4$)	42.0	83.1	61.8	85.5	25.9	32.1	58.2	77.6	47.2	58.6
Our ($l1+l2+l3+l4$)	**42.6**	**83.6**	**62.3**	**86.4**	**27.3**	32.5	**59.5**	**78.7**	48.5	**59.5**
Our (**nononnegativity**)	41.6	80.1	60.1	84.2	26.1	31.4	57.1	75.2	45.9	57.3

the performance compared with the baselines. DN [38] is the most related model, which trains a 4-layer model in an unsupervised scheme. The main differences are that we further constrain the learned feature with non-negativity and fine-tune the model with object detectors, leading to the obvious improvement of more than 10%. As 'Our(no nonnegativity)' denotes the model without non-negative constraint, it is indicated that the non-negative constraints bring about 2% of improvement and the detection-guided fine-tuning contributes more. In comparison with the famous hand-signed features (*i.e.* HOG, SIFT and its color-variant Opponent-SIFT) with the same extracting setup and SPM classifier, our learned features perform much better in most categories and the mean AP. The Object-Bank representations are high-level image representations extracted with a scale-invariant response map of pre-trained generic object detectors. It is shown that our best results also outperform Object-Bank, even though our method dose not need any object detectors when executing object recognition with the trained model.

Table 2
Recognition performance (AP in %) comparison of different object maps setups on VOC 2007.

Class	DN [38]	Our(*l*1((DPM)	Our(*l*4)(DPM)	Our(*l*1)	Our(*l*4)	DPM detection AP
Aero	69.0	69.1	70.3	70.1	75.0	62.2
Bike	40.8	51.6	54.1	52.1	56.3	73.1
Bird	42.2	43.9	43.3	45.9	50.0	56.9
Boat	62.3	64.6	62.3	66.6	68.3	55.6
Bottle	25.3	31.3	31.2	32.8	37.5	52.3
Bus	50.1	55.0	56.8	55.4	57.2	74.3
Car	64.1	68.2	67.7	68.9	71.6	59.5
Cat	50.8	57.2	56.4	57.6	59.6	71.3
Chair	50.9	52.3	50.3	54.8	57.8	49.2
Cow	33.7	42.3	42.7	42.7	49.0	69.3
Table	40.8	43.5	45.2	43.9	45.8	84.0
Dog	36.3	36.9	37.5	37.2	40.5	68.7
Horse	77.1	78.1	78.6	78.7	80.8	74.2
Motor	49.6	54.9	56.2	55.9	57.4	73.7
Person	79.2	78.5	79.4	80.8	84.8	65.6
Plant	20.0	18.3	20.1	20.9	25.2	49.1
Sheep	20.7	26.0	25.1	27.8	29.9	51.6
Sofa	47.7	51.1	54.6	51.5	55.3	79.2
Train	66.7	72.3	73.4	72.8	74.5	74.8
Tv	41.4	41.8	42.4	42.5	46.0	63.5
Mean	48.4	51.8	52.4	52.9	56.1	65.4

In Table 2, we also compare our results with the performance of employing the pre-trained detector provided by [9] to create the training object maps for fine-tuning (denoted as 'Our(*l*1)(DPM)' and 'Our(*l*4)(DPM)' in the table). The average precision of the DPM-based detector on the training and validation subset is 65.4%. In this way, the mean recognition AP of the 1st layer features is 51.8%, while the mean AP of the 4th layer features is 52.4%. Compared to our best results obtained by directly employing the labeled object bounding boxes, the performance of the 4th layer is decreased by about 4 percentages in average, but is still higher than the baseline of 'DN [38]'. This demonstrates the effectiveness of the detection guidance for feature learning, even if the detectors are not strong enough. Obviously, the better the detectors the higher enhancement is achieved for our model. For some categories that the detector performs well, like table (84.0% detection AP), the performance of the recognition accuracy is comparable with the scheme of applying the ground truth of the location (45.2% vs. 45.8%). However, for the categories with weak detection performance, like car (59.5% detection AP), the decrease of the recognition accuracy is more obvious (67.7% vs. 71.6%). Thus it is believed that with stronger detectors the performance may be similar to the way of directly applying the precise location information. Note that the performances of the 1st layer features are similar, as the influence of the fine-tuning phase on the low-level layer is limited.

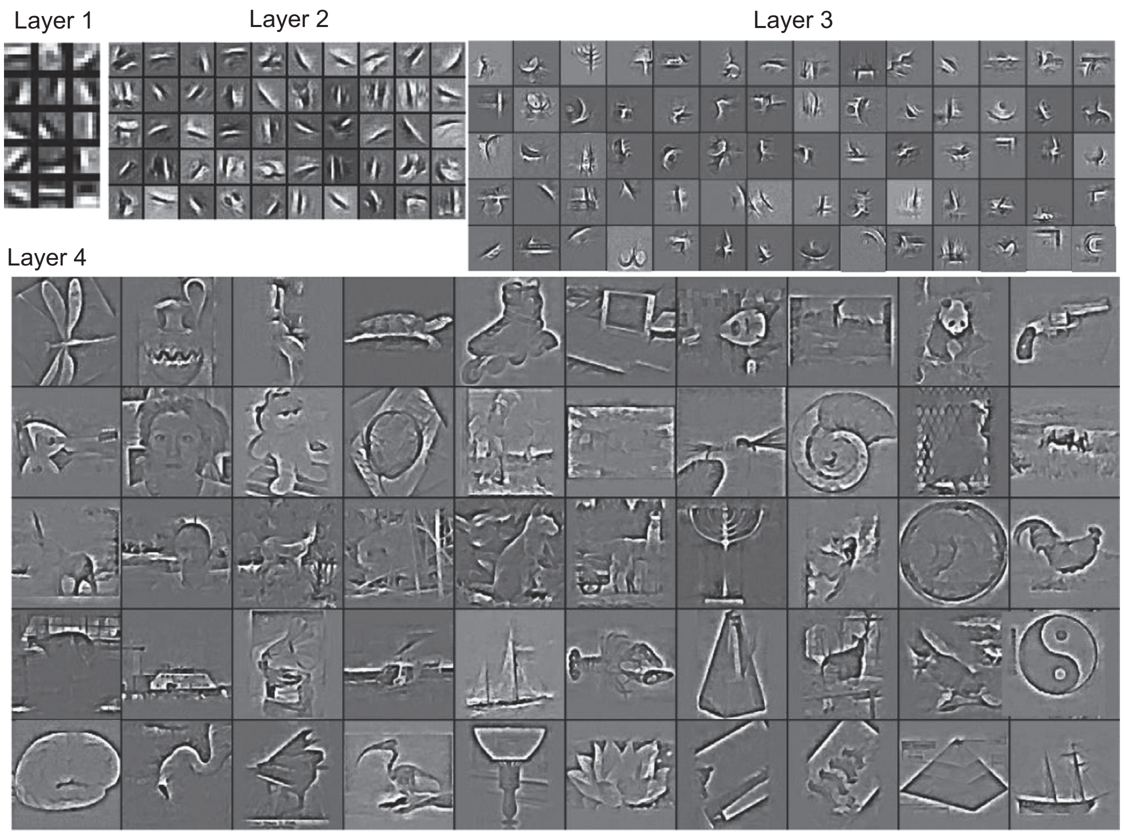

Fig. 5. The visualization of filters from layer 1 to layer 4 learned on Caltech-101.

4.2. Results on Caltech-101

The Caltech-101 dataset [8] contains 9144 images belonging to 101 object categories, with high shape variability. The number of images per category varies from 31 to 800. In the fine-tuning process, we let 101 feature maps of the 4th layer correspond to the 101 classes respectively and the remaining 49 correspond to the background. We follow the common experiment setup, training on 30 images per category and testing on the rest.

In Fig. 5, we show the filters leaned on the Calteh-101 dataset. The 4 layers of filters on this dataset are also able to capture image information in a hierarchical form, *i.e.*, low-level edges, mid-level geometric elements, high-level object parts and complete objects. Different filters of layer 4 focus on particular object classes of Caltech-101 and these filters seem fairly clear and complete as Caltech-101 is a easier dataset with less intro-class and object occlusion variability. It is also noted that the filters of layer 1 and layer 2 are quite similar to those on PASCAL VOC, indicating that our model captures some general image information especially in the low and mid levels.

Table 3 shows the recognition performances comparison in this dataset. Our best result is also obtained by combining the representations of the 4 layers, with an improvement of more than 6% compared with DN. Macrofeature [3] is a kind of hand-designed mid-level descriptors based on SIFT and behaves worse than our learned hierarchial features. Our model also beats some recently developed hierarchial models (*i.e.* CDBN [24], ConvSC [20] and HSC [34]) as these models lack high-level information to guide the model learning. We also evaluate the generalization of our model by employing the network trained on VOC 2007 to evaluate on Caltech-101, denoted as 'Our VOC model' in the table. Even though the performance in this case is less than our best result, it outperforms DN [38] and most published results. This has indicated that the model learned by the challenging VOC images captures some universal image structures and can generalize to entirely new categories. More stat-of-the-art results on this dataset are achieved by the framework of Convolutional Neural Network [36,14], as they employ an auxiliary large-scaled ImageNet dataset to pre-train a very large network. It is noted that CNN [36] performs weak without extra large numbers of training samples, while our proposed model can learn effective features from few examples.

Table 3
Classification rate (%) comparison on Caltech-101.

Algorithms	Classification rate
SIFT [22]	64.6 ± 0.8
Macrofeature [3]	70.9 ± 1.0
CDBN [24]	65.4 ± 0.5
ConvSC [20]	65.7 ± 0.7
HSC [34]	74.0
DN [38]	71.0 ± 1.0
Our VOC model	72.1 ± 0.3
Our ($l1$)	68.4 ± 0.3
Our ($l4$)	75.7 ± 0.2
Our ($l1+l4$)	76.6 ± 0.4
Our ($l1+l2+l3+l4$)	$\mathbf{77.3 \pm 0.2}$
CNN (no extra data) [36]	46.5 ± 1.7
CNN (with ImageNet) [36]	86.5 ± 0.5
CNN-SPP (with ImageNet) [14]	91.4 ± 0.7

Table 4
Classification rate (%) comparison on Caltech-256.

Algorithms	Classification rate
SIFT [33]	29.5 ± 0.5
KCB [10]	27.2 ± 0.5
ScSPM [33]	34.0 ± 0.4
DN [38]	33.2 ± 0.8
Our ($l1$)	34.3 ± 0.5
Our ($l5$)	38.1 ± 0.2
Our ($l1+l2+l3+l4+l5$)	$\mathbf{40.3 \pm 0.3}$
CNN (no extra data) [36]	22.5 ± 0.7
CNN (with ImageNet) [36]	70.6 ± 0.2

4.3. Results on Caltech-256

The Caltech-256 dataset [12] is an extension of Caltech-101, including 29,780 images in 256 object categories where the number of images in each class ranges from 31 to 800. This dataset is very much challenging as it possesses highly intra-class variability and object location variability. Because there are 256 object categories, we train a 5-layer model on this dataset and set the number of filters in the 5th layer to 280, then we let 256 feature maps of the top layer respond to the 256 classes with the rest corresponding to the background. The other parameter setups are transposed from the former experiments. We train on 30 images per class and test on the rest. As shown in Table 4, our model consistently outperforms the related methods on this challenging dataset and a gain of 10.8% is achieved over SIFT descriptors. It is noted that our model also beats ScSPM [33], which performs more complex sparse coding than the hard

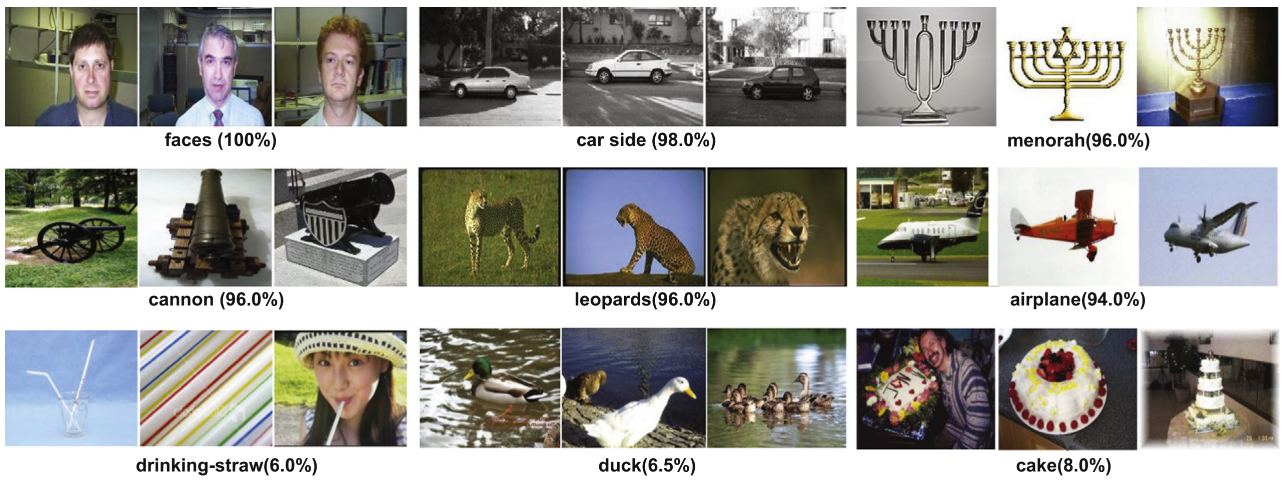

Fig. 6. Examples images of classes with highest and lowest classification accuracy from the Calteche-256 dataset. The top two rows are classes with the highest accuracy and the bottom one displays classes with poor accuracy.

assignment method in our experiments, further demonstrating the effectiveness of the features learned by our model. Zeiler and Fergus [36] achieve the accuracy of 70.6% on this dataset, since they pre-train a CNN model with large-scaled extra ImageNet dataset, while our model is able to learn good features with much less training samples.

In Fig. 6, we show some example images from classes with the highest and lowest classification accuracies. Our model performs very well on categories with little clutter (*e.g.* faces and car side), but fails in some categories like drinking-straw because of the large intra-class variation and highly diversity as seen in Fig. 6.

5. Conclusion

In this paper, we propose a detection-guided deconvolutional network for hierarchically learning image features and explore it for object recognition. It is shown that our model is effective to learn latent image structures from low-level edges to mid-level geometric elements and high-level complex objects. The selective and discriminative properties of the network are enhanced by our non-negative constraint and detection-guided fine-tuning scheme. Extensive experiments on three famous image benchmark databases (*i.e.* PASCAL VOC 2007, Caltech-101 and Caltech-256) demonstrate that the recognition performances of our model outperform the classical hand-designed features and related hierarchical models. It is also shown that the model trained on PASCAL VOC 2007 can adapt to the new categories in Caltech-101, indicating the well generality of the proposed model.

Conflict of interest

None declared.

Acknowledgment

This work was supported by 973 Program (2012CB316304) and National Natural Science Foundation of China (61272329, 61472422, 61332016).

References

[1] A. Ahmed, K. Yu, W. Xu, Y. Gong, E. Xing, Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks, in: European Conference on Computer Vision (ECCV), 2008, pp. 69–82.

[2] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sci. 2 (2009) 183–202.

[3] Y.L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning mid-level features for recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2559–2566.

[4] K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, in: British Machine Vision Conference (BMVC), 2011, pp. 1–12.

[5] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, vol. 2, pp. 886–893.

[6] R. Desimone, T.D. Albright, C.G. Gross, C. Bruce, Stimulus-selective properties of inferior temporal neurons in the macaque, J. Neurosci. 4 (1984) 2051–2062.

[7] M. Everingham, L. Van Gool, C. Williams, J. Winn, A. Zisserman, The Pascal visual object classes challenge 2007 (voc2007) results (2007), 2007.

[8] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories, Comput. Vis. Image Underst. 106 (2007) 59–70.

[9] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part based models, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 1627–1645.

[10] J. van Gemert, J.M. Geusebroek, C.J. Veenman, A.W.M. Smeulders, Kernel codebooks for scene categorization, in: European Conference on Computer Vision (ECCV), 2008, pp. 696–709.

[11] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.

[12] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset, Technical Report 7694, California Institute of Technology, 2007.

[13] H. Harzallah, F. Jurie, C. Schmid, Combining efficient object localization and image classification, in: International Conference on Computer Vision (ICCV), 2009, pp. 237–244.

[14] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: European Conference on Computer Vision (ECCV), 2014, pp. 346–361.

[15] G.E. Hinton, S. Osindero, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) 1527–1554.

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

[17] P.O. Hoyer, Modeling receptive fields with non-negative sparse coding, Neurocomputing 52–54 (2003) 547–552.

[18] F.J. Huang, Y. lan Boureau, Y. Lecun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.

[19] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex, J. Physiol. 160 (1962) 106–154.

[20] K. Kavukcuoglu, P. Sermanet, Y.L. Boureau, K. Gregor, M. Mathieu, Y. LeCun, Learning convolutional feature hierarchies for visual recognition, in: Advances in Neural Information Processing Systems (NIPS), 2010, pp. 1090–1098.

[21] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[22] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006, vol. 2, pp. 2169–2178.

[23] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.

[24] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: International Conference on Machine Learning (ICML), 2009, pp. 609–616.

[25] L.J. Li, H. Su, E.P. Xing, L. Fei-Fei, Object bank: a high-level image representation for scene classification and semantic feature sparsification, in: Advances in Neural Information Processing Systems (NIPS), 2010, pp. 1378–1386.

[26] R.Q. Quiroga, L. Reddy, G. Kreiman, C. Koch, I. Fried, Invariant visual representation by single neurons in the human brain, Nature 435 (2005) 1102–1107.

[27] M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex, Nat. Neurosci. 2 (1999) 1019–1025.

[28] K.E.A. van de Sande, T. Gevers, C.G.M. Snoek, Evaluating color descriptors for object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 1582–1596.

[29] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, T. Poggio, Robust object recognition with cortex-like mechanisms, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007) 411–426.

[30] Z. Song, Q. Chen, Z. Huang, Y. Hua, S. Yan, Contextualizing object detection and classification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1585–1592.

[31] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: closing the gap to human-level performance in face verification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1701–1708.

[32] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, R. Fergus, Regularization of neural network using dropconnect, in: International Conference on Machine Learning (ICML), 2013, vol. 28, pp. 1058–1066.

[33] J. Yang, K. Yu, Y. Gong, T. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 1794–1801.

[34] K. Yu, Y. Lin, J. Lafferty, Learning image representations from the pixel level via hierarchical sparse coding, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1713–1720.

[35] M. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: International Conference on Learning Representations (ICLR), 2013.

[36] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European Conference on Computer Vision (ECCV), 2014, pp. 818–833.

[37] M.D. Zeiler, D. Krishnan, G.W. Taylor, R. Fergus, Deconvolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 2528–2535.

[38] M.D. Zeiler, G.W. Taylor, R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in: International Conference on Computer Vision (ICCV), 2011, pp. 2018–2025.

Jing Liu received her B.E. degree in 2001 and M.E. degree in 2004 from Shandong University, and her Ph.D. degree from Institute of Automation, Chinese Academy of Sciences in 2008. Currently she is an associate professor in National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. Her research interests include machine learning, image content analysis and classification, multimedia information indexing and retrieval, etc.

Bingyuan Liu received his B.S. degree from Zhejiang University (ZJU), Hangzhou, China, in 2010. He is currently a Ph.D. candidate at National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image content analysis and deep representation learning.

Hanqing Lu received his B.E. degree in 1982 and his M.E. degree in 1985 from Harbin Institute of Technology, and Ph.D. degree in 1992 from Huazhong University of Sciences and Technology. Currently, he is a deputy director of National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include image and video analysis, medical image processing, object recognition, etc. He has published more than 200 papers in these fields.