A Bilinear Multi-Scale Convolutional Neural Network for Fine-grained Object Classification

Qinghe Zheng, Mingqiang Yang, Qingrui Zhang and Jiajie Yang

***Abstract*—Coarse-grained object classification with simple backgrounds has become quite mature, but the fine-grained object classification task against complex backgrounds is still challenging because there are only subtle differences in the local areas between different classes of fine-grained objects. In this paper, we propose a novel multi-scale convolutional neural network (msCNN) architecture that used for fine-grained object classification, which can extract discriminate local features at different scales in pyramid scale space. And a simplified bilinear model is used to carry out the end-to-end training for object classification. In the training phase, we design a distributed learning method with sample penalty term based on the sample distribution to optimize the network, which can improve the generalization ability of the network. In addition, we avoid using costly manual annotations like bounding box throughout the training and classification process. Finally, we present extensive experiments and visualizations on *CUB-200-2011* dataset and *ILSVRC2012_Dog* dataset that analyze the effects of the bilinear msCNN model on the fine-grained object classification task. The classification accuracy shows that the significant improvement of our msCNN model on the fine-grained object classification.**

***Index Terms*—fine-grained object classification, multi-scale CNN, weakly-supervised, sample penalty**

I. Introduction

Fine-grained object classification is a challenging research topic in the field of computer vision, which aims to classify subordinate-level categories under basic-level category, such as different bird types [1], dog breeds [2], flower species [3], aircraft models [4] etc. As shown in Fig. 1, the visual differences between different categories are very small, and are easily affected by the following conditions: the position and the posture of fine-grained object, the viewpoint of shot and the illumination conditions. In the box on the left, Indigo Bunting and Lazuli Bunting are two different breeds of birds that may be difficult to distinguish for people who are not experts. However, coarse-grained object classification is relatively simple, e.g., most people can easily recognize that the box on the left contains birds while the box on the right contains dogs. In particular, the difficulty of fine-grained object classification comes from the fact that discriminative features are focused not just on foreground object, but more importantly on the parts of object (like the body and wing of a bird) [5]. How to find and make use of the useful local area information is the key to the fine-grained object classification. Therefore, most of the fine-grained object classification algorithms follow this principle: locating the key area of foreground object ("where") to extract effective features ("what").

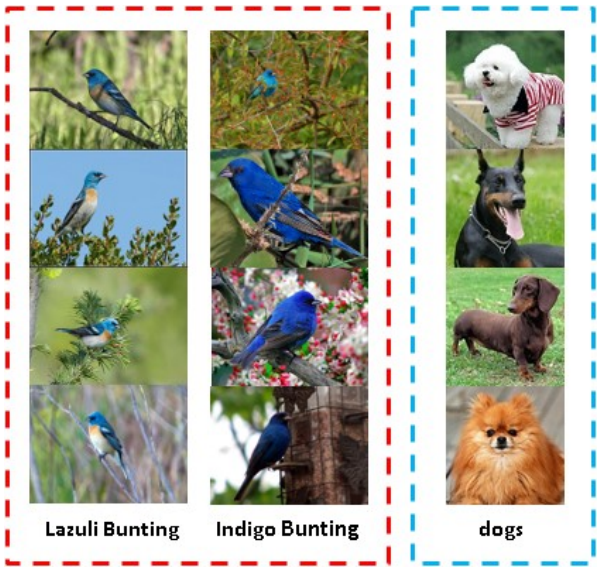


Fig. 1. Fine-grained object categories vs. Coarse-grained object categories. Fine-grained object categories (in the box on the left) include visually similar objects, e.g., to recognize Indigo Bunting and Lazuli Bunting. Coarse-grained object categories (in the box on the right) typically include visually distinct objects such as birds and dogs.

In the fine-grained object classification task, finding foreground object and key parts can be seen as a two-level attention process (object-level and part-level). And they heavily rely on costly manual labels like bounding box for object-level process and part landmarks for part-level process. With the help of these information, the background noise can be eliminated and the foreground object can be effectively detected. Some CNN-based models have been shown great improvements on the earlier work by using hand-crafted features [6, 7, 30, 33]. The disadvantages of this method include not only the difficulty of obtaining valuable label samples, but also the optimal choice of marking information

Manuscript received August 13, 2017; revised October 19, 2017. This work was supported by National Natural Science Foundation of China (Grant 61571275) and Shandong Provincial Natural Science Foundation (Grant ZR2014FM030, ZR2014FM010).

Qinghe Zheng is with the School of Information Science and Engineering, Shandong University, Jinan 250100, China (e-mail: 15005414319@163.com).

Mingqiang Yang is with the School of Information Science and Engineering, Shandong University, Jinan 250100, China (corresponding author, e-mail: imageinstitute@outlook.com).

Qingrui Zhang is with the School of Information Science and Engineering, Shandong University, Jinan 250100, China (e-mail: zhangqingrui1993@126.com).

Jiajie Yang is with Department of Science, University of British Columbia, Vancouver V6T1Z4, British Columbia, Canada (e-mail: 1040051920a@gmail.com).

for fine-grained object classification.

On the other hand, feature extraction ability is the key factor to determine the accuracy of image classification. Finding a more distinguished feature has always been the goal pursued by researchers. Some of the outstanding traditional feature descriptors are artificially designed like VLAD [8] or Fisher vector [9] with SIFT features [10]. By replacing SIFT with features extracted from convolution layers of a deep CNN pre-trained on ImageNet [11], some of the past classification models achieve state-of-the-art results on many object classification tasks [12, 27, 46]. These algorithms are easily applied to a variety of datasets because they do not rely on manual annotation information, such as bounding boxes. But their performance is lower than the best part-based architectures, especially when the background of object is complex. In addition, the effect of end-to-end training of these models has not been adequately studied.

In this paper, our main contribution consists of two parts: 1. we build a bilinear msCNN architecture that improves feature extraction ability on the basis of the traditional CNN; 2. we design a distributed learning method with sample penalty term to improve the generalization ability of the network based on the sample distribution. By simulating the automatic zooming function of biological vision and the construction of pyramid scale space in the SIFT feature extraction process, we add the scale space to the whole CNN structure and adjust the scale automatically, which in order to detect the key part areas and improve the feature extraction ability. Then we use a simplified bilinear model [13] to couple the two neural networks for feature extraction and object classification. It consists of two feature extractors based on msCNN whose outputs are connected by fully connected layers. Although we do not explore this connection further, this architecture is related to the two stream hypothesis of visual processing in the human brain [14] where contains two main pathways or streams. The dorsal stream ("where" pathway) is related to process the object's spatial location. The ventral stream ("what" pathway) is related to object identification and recognition. In the training phase, we build an anomalous sample penalty mechanism according to the sample distribution to establish an optimized feature boundary, which can improve the generalization ability of the network.

Our experimental results demonstrate the effectiveness of the bilinear msCNN architecture (in Section VI). With the only use of category labels of image, we reduce the gesture classification error rate of *CIFAR-100* dataset from 18.5% to 13.2% under single msCNN. On the fine-grained dataset of *CUB200-2011* [15] and *ILSVRC2012_Dog* [16], we reach the classification accuracy of 85.3% and 74.9% in the bilinear msCNN model, respectively, better than other methods that even use stronger supervisions. The rest of the paper is organized as follows. We first introduce some related works of CNN for feature extraction and fine-grained object classification in Section II. Details of the msCNN architecture are described in Section III. The distributed learning method with sample penalty term is introduced in Section IV and the training process of bilinear msCNN model is covered in Section V. Finally, we discuss what we learned, our conclusions and future works in Section VII.

II. Related Works

Fine-grained object classification has been paid more and more attention by researchers recently [1, 2]. Previous works mainly focus on the following three aspects to improve the classification accuracy: 1. foreground object localization; 2. discriminative features extraction for fine-grained objects; 3. human in the loop. Since our purpose is automatic fine-grained object classification, we focus on the related works of the first two aspects. In the first part, we introduce some of the improvement work of traditional CNN in image classification. In the second part, we introduce the CNN based application on the fine-grained object classification with weak supervision.

A. Traditional CNN

In recent years, the deep learning technique, represented by CNN, transforms the original data into a high-level expression to enhance the most discriminant features and to weaken the non-correlation features through the combination of nonlinear model [17]. Therefore, it has obvious advantages in dealing with multi-dimensional signals such as images. In the past time, CNN made the best results in the application of small-scale image problems like handwriting recognition, but the application of large-scale images cannot achieve the desired results. In order to solve this problem, the depth of the neural network [18] and the selection of hyperparameters [19] are gradually being discussed. In theory, it is easier to learn more abstract and complex features with a deeper structure of the convolution neural network. As the representative of CNN, VGGNet [20] and GoogLeNet [21] have also made impressive achievements in a variety of image identification areas. But these structures are difficult to get effective features and are easy to overfit in small sample datasets, which require ultra large datasets to get effective training. In order to solve the overfitting problem, the dropout [22] is used to prevent co-adaptation of feature detectors by omitting some perceptions from the hidden layer on each training instance. This method is effective, but the theoretical guidance and mathematical proof is not yet perfect. Another technique, batch normalization [23], reduces the overfitting and increases the training speed of the network. The strategy of locating discriminant regions and excluding background disturbances provides an excellent breakthrough for solving the problem [24]. However, the image recognition network will lose the speed advantage of end-to-end network structure after adding the pre-segmentation step.

B. CNN-based Fine-grained Object Classification

Two level attention algorithm [25] is the first attempt to complete fine-grained object classification task without using additional label information that solve the problem of how to detect the local area in the case of only class labels. However, the accuracy of local region obtained by clustering algorithm in this method is very limited. Zhang [26] proposed an algorithm that can select the discriminative local region feature from CNN that reduces the computational overhead based on the selective search method [47]. But these features contain a large number of irrelevant information, which need to remove noise. Simon et al. [28] designed a novel local area detection and extraction program. By visualizing the features

extracted by convolution layers, Simon found that regions with relatively strong responses often correspond to some of the potential local regions in the original image. The convolutional activation features are regarded as detection scores, and the regions with high response value represent the local areas detected in the original image. Branson [29] proposed pose normalized CNN in order to solve the interference of pose information. The algorithm makes use of the prototype to align the image and extracts the features in different layers for different regions, which in order to construct a more discriminative feature representation. Another important work in multi-scale feature extraction task is MOP-CNN [36], which extracts unordered multi-scale features of image by using CNN. Then, through the PCA dimension reduction and VLDA coding, the three level features are cascaded into 3*4096 dimensional features. Finally, some better classification results are achieved by using the linear one-vs-all support vector machine (SVM). Lin [13] designed a novel bilinear CNN model that achieves the classification accuracy of 84.1% in *CUB200-2011* dataset. The two networks coordinate with each other to accomplish the most important tasks in the process of fine-grained object classification: region detection and feature extraction. Our approach employs a similar simplified bilinear model, and the feature extraction and object classification is completed by an end-to-end training structure without using SVM.

III. The Details of msCNN

In this section, we introduce the construction of bilinear msCNN in detail and explain the rationality of the structure from many aspects. The overall framework, as well as the two training strategies used in the training process are presented in each part.

A. Structure of msCNN

In the whole structure, we add Gaussian smoothing layers compared to the traditional CNN architecture (e.g., Alexnet and VGGNet). The first hidden layer of the structure is set to the Gaussian smoothing layer (G1), then add the Convolution layer (C1), Down sampling layer (P1), the Gaussian smoothing layer (G2), and repeat until the three Fully connected layers (Full), and finally we get the results through the Softmax layer (see Fig. 2). In each Gaussian smoothing layer, we use Gaussian convolution kernels in different scales to generate the scale maps and add the feature maps of the upper layer to the current layer as the original scale map. The feature of the generated scale map is extracted by the convolution layer. Finally, multi-scale feature extraction is performed on the scale space constructed by down sampling layer and Gaussian smoothing layer. The reason for using the Gaussian convolution kernels is that it is the only correct kernel function to approximate scale space.

In the first Gaussian smoothing layer, a number of Gaussian convolution kernels are set, and the number of Gaussian convolution kernels corresponding to each feature map is increased as the number of layer increases. It can be found in some studies on visualization of neural network [31, 45], the bottom layer extracts some low-level features of the image, such as colors and edges. The middle layer extracts more

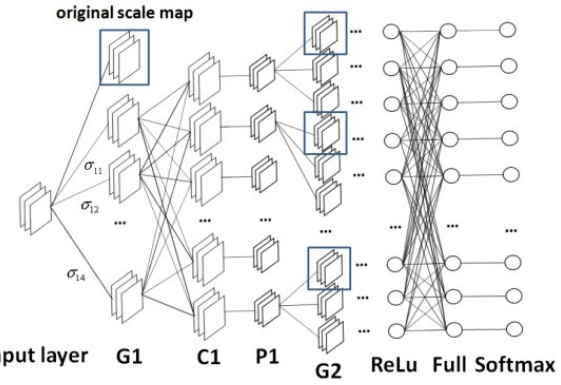


Fig. 2. The msCNN architecture with Gaussian smoothing layers. The scale map in the blue box is the original feature map without Gaussian smoothing.

complex features such as texture. Based on different objects, high layers extract more abstract and more complex features. We argue that the more complex features require more scales to understand. The specific structure of each layer is shown in Fig. 2.

Gaussian smoothing layer. A radial basis function (RBF) is a real-valued function whose value only depends on the distance from the origin. As the most commonly used radial basis function in image processing, Gaussian kernel function has the following excellent properties: 1. Rotational symmetry, that is, the smooth ness of the filter in all directions is the same; 2. Single valued function. It is shown that the weighted mean of the neighborhood pixel of the Gaussian filter is used to replace the pixel value, and the weight of each neighborhood pixel is monotonically increasing with the distance between the point and the center point; 3. The width of the Gaussian filter, which determines the degree of smoothness, is characterized by the parameter σ, and the relationship between σ and smoothness is very simple. The larger the σ, the wider the band of the Gaussian filter, then the better the degree of smoothness. These are also the reasons why we choose the Gaussian kernel function to establish the scale space. The Gaussian function g controlled by variance σ^2 is defined by:

$$g(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \tag{1}$$

where x, y represents the position of the pixel in the image. Using the formula can produce the corresponding Gaussian averaging operator. As shown in Fig. 3, the size of a typical Gaussian filter is 5×5 and the variance σ is 1.

Convolution layer. The main function of convolution layers in CNN architecture is to extract features by convolution operating of the feature map in the upper layer. In

0.002	0.013	0.220	0.013	0.002
0.013	0.060	0.098	0.060	0.013
0.220	0.098	0.162	0.098	0.220
0.013	0.060	0.098	0.060	0.013
0.002	0.013	0.220	0.013	0.002

Fig. 3. A typical Gaussian filter with 5×5 size (σ=1.0).

order to avoid overfitting and increase the generalization ability of the network, it is not necessary for each neuron to perceive the global image, whereas the neuron only needs to be aware of the local area. Then the local information is synthesized at the higher level and the global information is obtained. In addition, the number of parameters is reduced by sharing the weights and bias of convolution kernel. Filter parameters are randomly initialized, and will be changed to become colour, texture or other specific feature extractors. The forward propagation of convolution layer is given by:

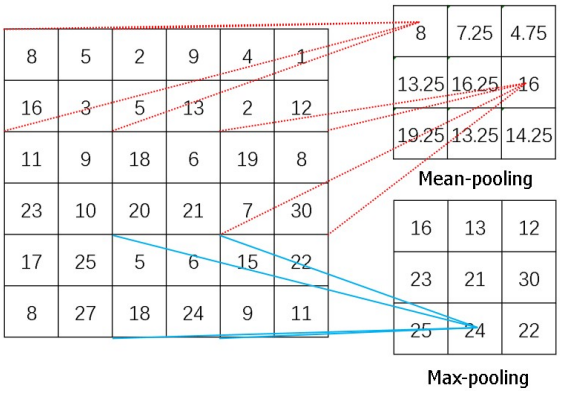


Fig. 4. Mean-pooling function and max-pooling function.

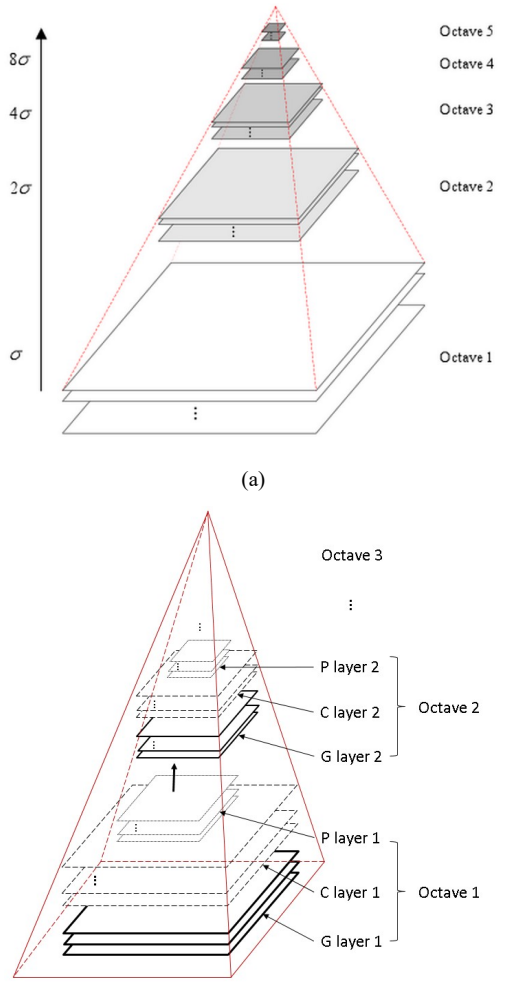


Fig. 5. Comparison of pyramid scale space: (a) pyramid scale space in SIFT algorithm and (b) pyramid scale space in the msCNN architecture.

$$y^j = f(b^j + \sum_i \omega^{ij} * x^i) \tag{2}$$

where x^i is the i^{th} input feature map, and y^j is the j^{th} output feature map. ω^{ij} is the weight coefficients of the convolution kernel. b^j is the bias of the j^{th} output feature map. $f(\cdot)$ represents activation function. * denotes the convolution operation. The convolution layers in our architecture are almost indistinguishable from the traditional network, except for the number and size of convolution kernels.

Down-pooling layer. The down-pooling layer usually uses two functions, as shown in Fig. 4. Our architecture uses max-pooling for a reason that it preserves more texture information. There are two sources of errors in the feature extraction process in CNN: one is that the increased estimated variance due to the limited size of the neighborhood, another one is the deviation of the estimation error caused by the precision errors of convolution layer parameters. We reduce first types of error by using max-pooling filter. The mean-pooling layer reduces the second types of error while preserving more background information, which is what we do not want to see.

Analysis of structure. In the msCNN structure, we add three fully connected layers in front of the output layer, which have been abandoned in many current CNN structures. Specific reasons will be given in Section IV. Each kind of layer is introduced in the architecture, and the local response normalization layer has been abandoned. Next, we analyze the significance of this structure. In some traditional network architectures [19], the weights of convolution layer are initialized as Gaussian convolution kernels, but which do not have the advantages of Gaussian smoothing layer. Because in the training process, parameters adjustment in each layer leads to the collapse of pyramid scale space immediately. And the Gaussian smoothing layer only adjusts the size of the scale by training, so that the entire learning process is carried out in a stable scale space. On the other hand, the SIFT algorithm is not clearly defined and contains several free sets of parameters, which needs further improvements during testing phase. So we build the optimal scale space by training neural network to solve the problem. In the training process, the network adjusts the scale of feature maps in a bad convergence by observing the accuracy of validation set and finally establish the most appropriate scale space for feature extraction (see Fig. 5). Specifically, there are many separable features that are easy to distinguish for coarse-grained image classification tasks, such as cars and aircrafts. But for the fine-grained image classification such as different types of aircraft in a chaotic background, multi-scale analysis has a significant effect in this case that the feature configuration is almost the same.

B. Selection of Loss Function and Activation Function

The network is finally output using softmax layer and log-likelihood function is used as the loss function. Suppose that neurons in the last layer of a multi-layer neural network are linear neurons, which means the outputs are simply $a_j^L = z_j^L$. For a training example x, if we use the quadratic loss function, the output error δ^L is given by:

$$\delta^L = a^L - y \tag{3}$$

In the output layer, the partial derivatives of loss function on the weights and bias are given by:

$$\frac{\partial C}{\partial \omega_{jk}^L} = \frac{1}{n}\sum_x a_k^{L-1}(a_j^L - y_j) \tag{4}$$

$$\frac{\partial C}{\partial b_j^L} = \frac{1}{n}\sum_x (a_j^L - y_j) \tag{5}$$

where C is the quadratic loss function, the entries of the weight matrix ω^L are just the weights connecting to the l^{th} layer of neurons, that is, the entry in the j^{th} row and k^{th} column is ω^L_{jk}. B_j^L is the j^{th} components of the bias vector b^L. And finally, we define an activation vector a^L whose components are the activations a_j^L. y is the corresponding desired output. n is the total number of training samples. It is shown that if the output neuron is linear, the quadratic loss function will no longer cause the learning rate to decline. In this case, the quadratic loss function is an appropriate choice. And the cross-entropy function and log-likelihood function have the same property. In fact, the cross-entropy function [32] is a better loss function when the output neuron activation function is a sigmoid function. Due to the random initialization of the network weights and biases may produce considerable errors for some training samples, the quadratic loss function will lead to a decline in learning speed. Softmax function is a natural way to ensure that the output activations form a probability distribution, which has monotonicity and non-locality. The combination of softmax and log-likelyhood function is more suitable for scenarios where the output activation value needs to be interpreted as a probability. It is more effective in the classification of fine-grain objects with overlapping features. The final experiment also proves the effectiveness of the combination.

Parametric Rectified Linear unit [43] (PReLu) is set to be activation function as in (6), which corrects the data distribution and avoids the neuronal necrosis caused by an inappropriate initialization, that is, some neurons cannot be trained after being set zero.

$$\text{PReLu}(x) = \begin{cases} x & \text{for} \quad x > 0 \\ \alpha x & \text{for} \quad x \le 0 \end{cases} \tag{6}$$

where α is a small fixed value (e.g. 0.25).

C. Bilinear msCNN Model for Fine-grained Object Classification

Bilinear CNN model [13] is a classification model that solves several problems of both part-based and texture models. Here we employ a simplified bilinear model for image classification. It consists of two feature extractors based on msCNN and is classified by fully connected layer and softmax layer (Fig. 6). The two msCNN are linear in parallel, and one of them is used to locate the critical region of the object in the ideal state, and the other is used to extract the feature. First of all, a quadruple $\mathbf{B} = (f_A, f_B, P, C)$ is defined as

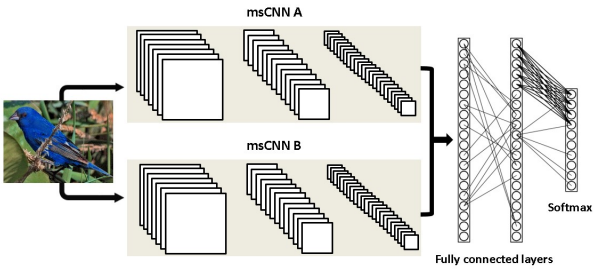


Fig. 6. Bilinear msCNN model.

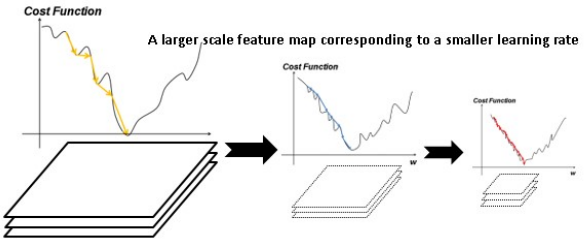


Fig. 7. Influence of learning rate under different scales.

bilinear model for fine-grained object classification. Here f_A and f_B are feature extraction functions (msCNN). P represent the last pooling function for connecting two networks and C is the softmax function used to classification. A feature function is a feature mapping f: $I \rightarrow \mathrm{R}^{c \times D}$ that takes an image I, which maps the input image I to a $c \times D$ feature. The bilinear combination feature of f_A and f_B is given by bilinear $(f_A, f_B) = f_A^T f_B$. Then the resulting bilinear feature vector $\mathbf{x}$ in fully connected layer is passed through softmax layer.

Since the structure of the entire bilinear msCNN model is a directed acyclic graph, the parameters can be trained by back-propagating the gradients of the loss function (log-likelihood). Suppose that the outputs of two msCNN are matrices A and B of size $L \times M$ and $L \times N$ respectively, then the bilinear combination feature is $\mathbf{x} = A^T B$ of size $M \times N$. Let $dl / d\mathbf{x}$ be the gradient of the loss function l, then according to the chain derived rule we can get:

$$\frac{dl}{dA} = B\left(\frac{dl}{d\mathbf{x}}\right)^T, \frac{dl}{dB} = A\left(\frac{dl}{d\mathbf{x}}\right) \tag{7}$$

The gradient of classification layer is simple and clear, and the gradient of the last pooling layer before fully connected layers can be computed according to the chain derived rule, as shown in (7).

IV. Distributed Learning Method with Sample Penalty Term

After the network structure has been set up, we give a detailed introduction to the improved training methods in this section.

Aiming at the problem that the training of neural networks is slow and difficult to converge in the appropriate position, we design a distributed multi learning rate training method. Inspired by light propagation properties, we consider that each octave is different medium in the error back propagation

process. Just like light travels in different mediums at different speeds, we use different learning rates in each octave in the training progress and adjust them according to the change of the scales. In the training process of the traditional CNNs, the weights in the front hidden layers are more difficult to train than the that in the hidden layers behind it. So we used a larger learning rate on the small scale front hidden layer, and a smaller learning rate corresponding to the large scale hidden layer. For example, as a function of one parameter w, the scale of the different layer may lead to different densities of the valley. At this time a different learning rate will bring different results, as shown in Fig. 7. The specific learning rate needs to be set according to the specific dataset, and we adjust the learning rate according to (8).

$$\alpha_{new_i} = \alpha_{old_i} \frac{\sum_{k=1}^{m} \sigma_{old_k}}{1.2 \sum_{k=1}^{m} \sigma_{new_k}} \tag{8}$$

where α_{old_i} and α_{new_i} is the learning rate before and after the update of i^{th} layer, σ_{old_k} and σ_{new_k} is the corresponding scales of k^{th} scale map before and after the update of i^{th} layer. With the iteration of training, the scale becomes larger and the image becomes blurred, the learning rate will gradually decrease according to (8), which is consistent with our intuitive understanding.

We apply this learning rate adjustment criterion based on the well-known back-propagation (BP) algorithm [35]. The current CNN training method consists of two parts: 1. the training samples are propagated forward to the final output layer of the network, and finally the loss function is calculated; 2. The error is transmitted back layer by layer from top to bottom, and weights are updated in respective layers based on the back-propagated errors. Finally, the log-likelihood loss function in the output layer is given by

$$C = -\ln a_y^L \tag{9}$$

When training the network in general, there are multiple images per batch. Then the loss function becomes

$$C = -\sum_{j=0}^{m} \ln a_y^L = \ln\left(\sum_{i=0}^{n} e^{z_i} - \sum_{j=0}^{m} z_j\right) \tag{10}$$

where m is batch size. When back propagation occurs, the first step is to calculate the loss gradient by the partial derivative in accordance with (11).

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} = \begin{cases} f(z_j^L) - 1, & z_i^L = z_j^L \\ f(z_j^L), & z_i^L \neq z_j^L \end{cases} \tag{11}$$

where

$$f(z_j^L) = \frac{e^{z_j^L}}{\sum_{i=1}^{n} z_j^L} \tag{12}$$

where z^L represent the weighted input to the neurons in layer L. Then update the weights and biases of each layer according to

$$W_{(iter+1)_t} = W_{(iter)_t} - \Delta history_{(iter+1)_t} \tag{13}$$

$$\Delta history_{(iter+1)_t} = mt * \Delta history_{(iter+1)_t} + \alpha_t * \frac{\partial C}{\partial W} \tag{14}$$

where mt is momentum, α_t is the learning rate in octave t.

A. Optimized Feature Boundary

In this part, we explain why we still use the fully connected layers at the end of the network, and how to use it to seek the optimal feature boundary.

At present, lots of CNNs have abandoned the use of fully connected layer, like Deep Residual Network [34]. Fully connected layers have a large number of parameters, easily lead to overfitting. And the removal of fully connected layer makes the whole network similar to a multi-feature voting mechanism, which has a better generalization ability for large and complex image samples. The network with fully connected layer is similar to the single feature classification mechanism, which is better for fine-grained images with common features. So we still keep it in the structure. Suppose that a fully trained, overfitted CNN model has a feature distribution of samples as shown in Fig. 8. The circle and triangle represent the eigenvectors of two kinds of samples, respectively. In this case, wrong and invalid features may be extracted from these anomalous samples due to noise and the specificity of the sample itself. In order to make these samples correctly classified, the training method based on the minimization of loss function can only forcibly fit them. Thus weakening the feature extraction and classification ability of CNN, results in the overfitting problem. In the case of insufficient samples, we believe that these anomalous samples will have a negative impact on training and produce misclassified areas as shown in the shaded section of Fig. 8. Therefore, the loss function based on the anomalous degree is used to train the network. Based on this assumption, we believe that these features extracted from anomalous samples

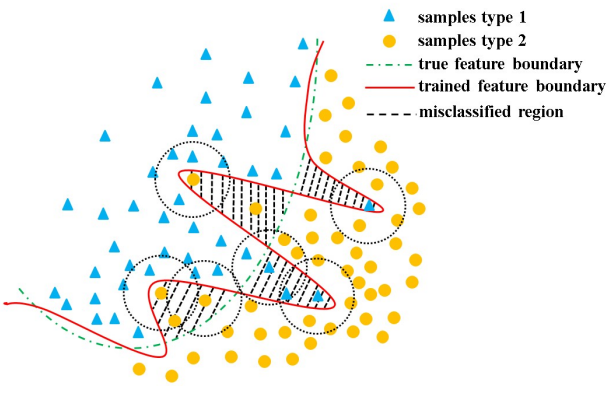


Fig. 8. Feature distribution of samples under a preliminary model.

centered together with features extracted from samples of other labels, as shown in the black dashed circle in Fig. 8.

Therefore, K-nearest neighbor algorithm is used to determine the anomalous degree μ_i of sample x_i. As shown in (15), the anomalous degree μ_i of sample x_i is determined by the proportion of different classes of samples in the surrounding k samples.

$$\mu_i = \frac{k + 0.01 - c_i}{k + 0.01} \tag{15}$$

where c_i is the number of samples of the same category in the nearest k samples, and the distance measure is Euclidean distance. 0.01 is added to numerator and denominator to avoid the case where anomalous degree is equal to 0. Then we can obtain the judgement method of anomalous samples from:

$$x_i = \begin{cases} \textit{Anomalous samples}, & \text{if } \mu_i > U \\ \textit{Normal samples}, & \text{if } \mu_i \le U \end{cases} \tag{16}$$

where U is the threshold of anomalous degree and ranges from 0 to 1, with closed intervals. Then we design the penalty factor η_i that used to update the loss function according to the threshold U and the anomalous degree μ_i of sample x_i, as in (17).

$$\eta_i = \left[U\left(1 - \frac{\log \mu_i}{5 \log U}\right) \right] * \varepsilon(\mu_i - U) \tag{17}$$

where ε represent the step function. Then we can get the new loss function $f_{newloss\_i}$ for each sample x_i according to (18).

$$f_{newloss\_i} = (1 - \eta_i) f_{loss\_i} \tag{18}$$

where f_{loss_i} is the original loss function, such as the cross-entropy function and log-likelihood function.

B. Drop-path and Freeze-path Technique

In this section, we introduce a strategy for optimizing the msCNN structure. In the training phase, we use dropout technique in convolution layer that is adjacent to the Gaussian smoothing layer, which is often only applied to the fully connected layers in the traditional structure. This means that a portion of feature map with smaller dropout weight will be removed at each convolution layer after several training epochs. The specific application strategy is that we use dropout when the classification error rate of the validation set is no longer decreasing. In this case, we think that the feature extraction ability of network is caught in the bottleneck, and some indistinguishable features interfere with the classification of networks. In other words, the network no longer continues to learn features, but begins to overfit the training samples.

There are two ways to implement dropout technique: one way called drop-path is to directly remove the feature map in each convolution layer, which means the weights of the feature map are set to zero and no longer being trained at the same time. Another way called freeze-path is to stop adjusting the weights of feature maps, that is, to maintain it unchanged. In the final experiment, the convergence rate and the classification accuracy of the two methods are compared. In order to determine the importance of a feature map in the msCNN structure, dropout-weight DW is defined as a comprehensive consideration for each feature map in (19).

$$\begin{aligned} DW_i^t = {} & \beta_1 \cdot \sum_k \omega_{i,j} + \beta_2 \cdot \left(\sum_k \omega_{i-1,j} + \sum_k \omega_{i-2,j}\right) \\ & + \beta_3 \cdot \left(\sum_k \omega_{i+1,j} + \sum_k \omega_{i+2,j}\right) \end{aligned} \tag{19}$$

where DW_i^t is the Dropout-Weight of t^{th} feature map in i^{th} convolution layer of the architecture in octave t, and $w_{i,j}$ is the j^{th} weight coefficient of feature map in the i^{th} layer, k is the number of weight parameters in each layer. β_i is the weight coefficient of each convolution layer (β_1=1.2, β_2=0.7, β_3=0.9). We consider the weights of adjacent two layers as the basis for judging the importance of the scale map. The parameters in current layer have the highest weights, and the latter two layers have higher weights than the previous two layers.

Instead of extracting the effective features and then classify, the training of small datasets easily leads to a phenomenon that the top-level neuron and the input data have a point-to-point memory. If the feature cannot be extracted, mandatory error monitoring training will make the model directly fit the input data. Dropout technique used in the convolution layer can dropout the unstable features through the training again and again. A good foundation can bring beneficial effects to the abstract features extraction of deeper layers. And it forces a neuron to work together with selected neuron to achieve good results, eliminates the joint adaptability between the nodes and enhances the generalization ability. So we can prevent co-adaptation in learning process and improve the overfitting problem by this strategy.

V. Training Process

In this section, we introduce the parameters setting and the whole training process of the linear msCNN model.

Fig. 9 shows the flow chart of the training process of network. In the first stage, we build the whole structure consists of several octaves and fully connected layers; we finally get the results through the softmax layer. Each octave in the pyramid scale space contains a Gaussian smoothing layer, two convolution layers and a max-pooling layer. The network structure is shown in Table I. The structure consists of two basic neural networks: msCNN A and msCNN B. msCNN A has a total of 9 convolution layers. In the first three layers (Octave 1), each layer consists of 128 convolution kernels of size 5×5. In the middle three layers (Octave 2), each layer consists of 256 convolution kernels of size 3×3. The last three layers contain 512 convolution kernels of size 3×3 per layer. In order to break the symmetry of the two networks, msCNN B uses a slightly different structure that includes only 6 convolution layers, and sets different parameter initialization. The last two fully connected layers

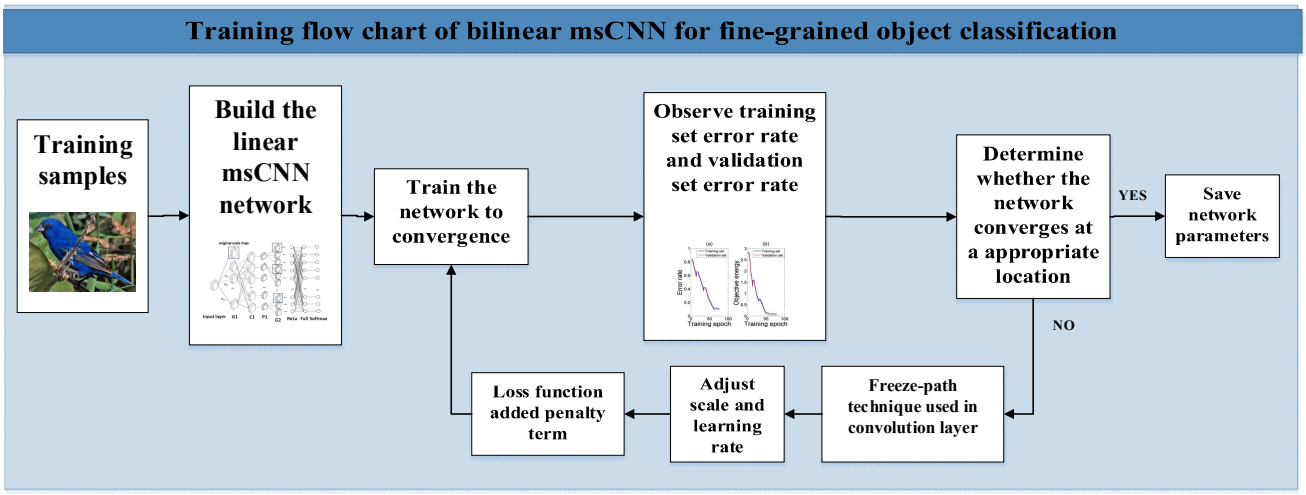


Fig. 9. Training flow chart of bilinear msCNN.

TABLE I
THE BILINEAR MSCNN NETWORK STRUCTURE

Input (224 × 224 RGB image)	
msCNN A	msCNN B
Gaussian smoothing layer (Octave 1)	
Conv5-128 Conv5-128 Conv5-128	Conv3-128 Conv3-128
Max-pooling layer	
Gaussian smoothing layer (Octave 2)	
Conv3-256 Conv3-256 Conv3-256	Conv3-128 Conv3-128
Max-pooling layer	
Gaussian smoothing layer (Octave 3)	
Conv3-512 Conv3-512 Conv3-512	Conv3-256 Conv3-256
Max-pooling layer	
Fully connected layer-4096	
Fully connected layer-4096	
Fully connected layer	
Softmax layer	

TABLE II
GAUSSIAN SMOOTHING SCALES OF EACH OCTAVE IN MSCNNS

Architecture	Octave1	Octave2	Octave3
msCNN A (for *CIFAR-100*)	σ_1=0.5(5*5)	σ_1=0.7(3*3)	σ_1=0.9(3*3)
	σ_2=0.7(5*5)	σ_2=0.9(3*3)	σ_2=1.2(3*3)
	σ_3=1.2(5*5)	σ_3=1.2(3*3)	σ_3=1.4(3*3)
	no σ_4	σ_4=1.4(3*3)	σ_4=1.5(3*3)
	no σ_5	no σ_5	σ_5=1.6(3*3)
msCNN B	σ_1=0.6(5*5)	σ_1=0.8(3*3)	σ_1=1.0(3*3)
	σ_2=0.8(5*5)	σ_2=1.0(3*3)	σ_2=1.3(3*3)
	σ_3=1.1(5*5)	σ_3=1.3(3*3)	σ_3=1.5(3*3)
	no σ_4	σ_4=1.5(3*3)	σ_4=1.6(3*3)
	no σ_5	no σ_5	σ_5=1.7(3*3)

are connected to two networks, each with 4096 neurons. The number of neurons in the last layer needs to be determined according to the class of the dataset.

Then we set the initial learning rate at each octave to the same value $\alpha_1=\alpha_2=\alpha_3=0.005$, batch size $m=128$, momentum $mt=0.6$, and weight decay $wd=0.001$. Gaussian smoothing scales in each octave are shown in Table II. The number of kernel functions of Gaussian smoothing layer increases gradually, and the size becomes smaller. With all these settings completed, we start training the neural network to converge and start observing the validation set error rate. If the validation set error rate no longer falls, we use the drop-path technique or freeze-path technique to solve the overfitting problem. At this point, the loss function is replaced by a function with a sample penalty term. Then the entire network continues to be trained to converge to its best position. Throughout the process, validation set is not used to train the network. It is only used to observe the convergence state of the network as well as a criterion for selecting the optimal model.

VI. EXPERIMENTS

The experiments are divided into the following three parts. In the first part, we prove the effectiveness of a single msCNN on *CIFAR-100* dataset, and evaluate the classification ability of the network under various strategies. In the second part, we show the classification accuracy of bilinear msCNN on *CUB200-2011* dataset and *ILSVRC2012_Dog* dataset. In the last part, we illustrate the working mechanism by visually analyzing the convolution kernels and samples with the highest classification error rate. And all experiments are conducted under weak supervision, which means that only category labels are available.

A. Verification of different techniques on CIFAR-100 Dataset

In this section, we use the *CIFAR-100* dataset to validate the effectiveness of the network structure and training strategy, which includes 100 categories of 60000 images. 40000 images of the dataset are used as training sets, 10000 as validation sets, and 10000 as test sets. By observing the classification error rate and convergence state of the network, we can judge the feature extraction ability and object classification ability of CNN.

As shown in Table III, we use different network structure and various training strategies to observe the image classification accuracy. From the comparison between network 1 and network 2, we can see that the distributed multi-learning method with the penalty term achieves higher classification accuracy. The distributed learning method improves the classification accuracy of the network over the test set by about 8%. At the same time, it can be found that the

TABLE III
CLASSIFICATION ERROR RATE OF MSCNN WITH DIFFERENT TECHNIQUE ON CIFAR-100

Network number	Model	Drop-path	Freeze-path	Single learning rate	Distributed multi learning rate	Penalty term	Test set error rate(%)
1	msCNN A			√			45.5
2					√	√	37.3
3		√			√		18.5
4			√		√		20.7
5			√		√	√	**13.2**
6	Alexnet					×/√	46/36.3
7	VGG-16					×/√	37.9/29.2
8	GoogLeNet					×/√	26.5/17.4

TABLE IV
TOP-1 ERROR RATE ON *CUB-200-2011* AND *ILSVRC2012_DOG* TEST SET

Method	*CUB-200-2011*(%)	*ILSVRC2012_Dog*(%)
Domain Net [25]	30.3	27.2
Two-level Attention (Alexnet) [25]	30.3	38.1
Two-level Attention (VGGnet) [25]	22.1	29.7
Zhang et al. [26]	20.7	25.4
Constellations [28]	19.0	26.7
Spatial Transformer Net [37]	**15.9**	22.4
J. Krause et al. [38]	18.0	24.1
B-CNN [D, M] [13]	**15.9**	19.8
Chen et al. [39]	17.2	18.3
Jacobsen et al. [40]	24.5	27.3
Szegedy et al. [41]	17.4	**16.5**
msCNN A	**19.6**	**22.4**
Bilinear msCNN	**14.7**	**15.1**

distributed multi learning rate method converges faster than the single learning rate method by observing the convergence curves of two networks in Fig. 10 (a) and Fig. 10 (b). The process of network training to convergence is reduced from 23 epochs to 17 epochs. From the network 3 and 4, we can see that the addition of drop-path and freeze-path technique is helpful to improve the performance of network classification ability. But as shown in Fig. 10 (c) and Fig. 10 (d), the addition of two technologies increases the number of epochs and training time. Drop-path has a better effect than freeze-path, but the epoch number of convergence is also higher than freeze-path. Due to the deletion of the feature map, network with drop-path technique takes a long time to converge. In the same initial parameter setting, the network using freeze-path can reduce the convergence of five epochs. Comparing network 4 and 5, we can see that the loss function with penalty term can improve the generalization ability of the network effectively. Finally, we fuse a variety of strategies in the msCNN training process and achieve very good results in Network 5. We also compare the effect of training methods on several classical structures (Network 6, 7, 8). The classification performance of each network has been improved by about ten percent. But the classification accuracy of bilinear msCNN model is higher than the classic three CNN models, even though these models are enhanced by improved training methods.

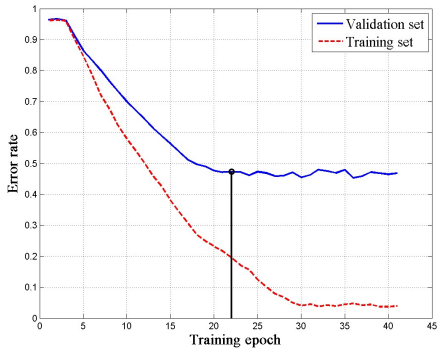
From the training process of msCNN, it can be seen that the classification accuracy of the training set is approaching about 100%, but the classification accuracy of test set is just over 85%. Overfitting problems still exist, but according to the test set results we can see that the generalization ability of the network has been improved.

B. CUB200-2011 Dataset and LSVRC2012_Dog Dataset

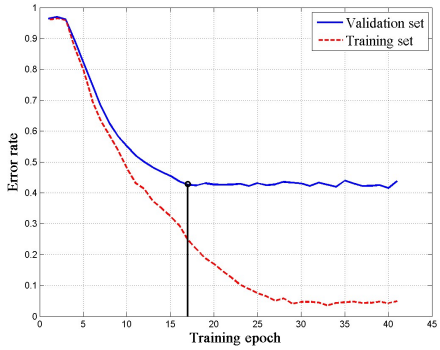
In this section, we verify the effectiveness of optimized structure and training methods on two fine grained image datasets. The *CUB-200-2011* dataset contains 11,788 images of 200 bird species and *ILSVRC2012_Dog* dataset contains 153,773 images of 118 breeds of dog; 1/5 of the whole are used as validation set and test set. During the training phase, the input to our model is a fixed-size 224×224 RGB image. The only preprocessing that we do before training is subtracting the mean RGB value and computing over the training set from each pixel. Our bilinear msCNN model in all experiments is invariant, except the number of neurons of the output layer is set as number of categories when required. Table IV summarizes the top-1 error rates of two datasets under different algorithms.

From experimental results, our bilinear msCNN model achieves the lowest classification error rate on the *CUB-200-2011* and *ILSVRC2012_Dog* dataset. Even a single msCNN structure has achieved very competitive results. The two level attention model [25] can solve the problem of how to detect the local region in the case of only class labels. However, the accuracy of the local region obtained by the clustering algorithm is very limited. And a more effective integration of the object-level and part-level attention needs to be explored. The Constellations [28] algorithm has achieved good results by generating candidate regions, but it faces huge computational costs and waste of resources. By extracting the convolutional activation features in the CNN and fisher coding and then input into the SVM, Chen [39] achieved good results, but this method is not an end-to-end way to complete the training and classification. The end-to-end structure used for image classification has not been fully studied yet, but it has obvious advantages from the experimental results. B-CNN model combining M-Net [42] and the "very deep" network [20] has achieved the start-of-the-art accuracy of 74.1% at that time on *CUB-200-2011* dataset. Compared to this algorithm our network has achieved better results with the improved training method based on the simplified bilinear model. Szegedy [41] made the best results of 83.5% at that time on *ILSVRC2012_Dog* dataset. However, the network structure of this method is complex and huge, and the training stage consumes a great deal of time. Compared with the past state-of-the-art classification methods, we improve the error rates of fine-grained object classification on the *CUB-200-2011* and *ILSVRC2012_Dog* dataset from 15.9%

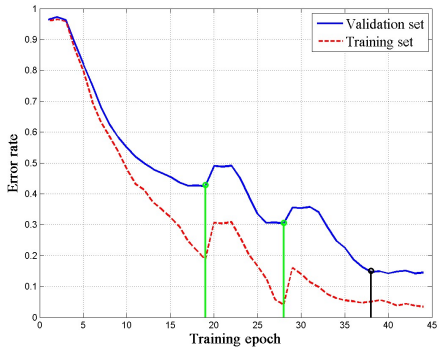
to 14.7% and 16.5% to 15.1%, respectively. The experimental results show that the classification ability of network and the generalization ability on test set are improved compared with the past CNN structures.



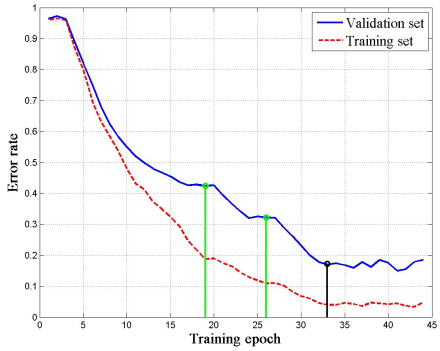
(a)



(b)



(c)



(d)

Fig. 10. Training curves for each network and corresponding strategies. The black and green vertical lines (the first two vertical lines in (c) and (d)) represent the locations of network convergence and where two training strategies are used, respectively. (a) Network training with single learning rate (network number 1 in Table III). (b) Network training with distributed learning method (network number 2). (c) Network with drop-path technique (network number 3). (d) Network with freeze-path technique (network number 4).

C. *Visual Analysis*

Confusion matrix [44] is a kind of visualization method of classification results commonly used in supervised learning (Fig. 11), which can intuitively express the precision rate and recall rate of classification model. Element $M_{i,j}$ in confusion matrix represents the number of samples in class i that are assigned to class j. The value of the matrix can be normalized between 0 and 1. Based on this ratio, we give each element of the matrix a hue from blue to red. The elements on the main diagonal represent the proportion of samples that are correctly classified. The closer the color of the main diagonal of confusion matrix is to black, the higher the classification rate. We can observe the precision and recall rate according to the confusion matrix. Fig. 12 shows the confusion matrix of the

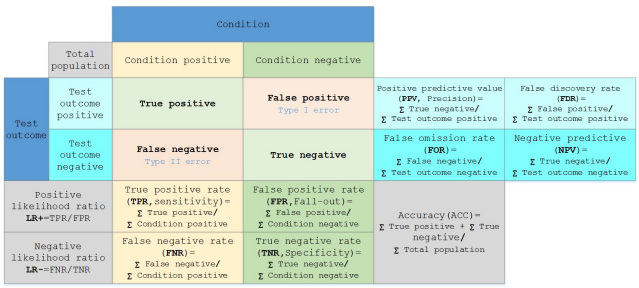


Fig. 11. The composition and calculation method of confusion matrix.

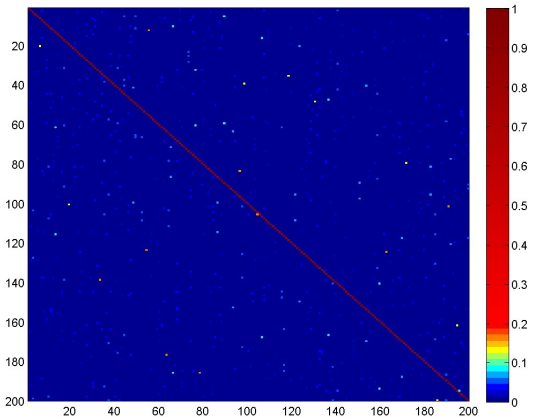


Fig. 12. Classification confusion matrix on *CUB-200-2011* dataset.

Fig. 13. Top four pairs of categories that are most misclassified. In each row we show the objects in the left column that are most confidently classified as the category in right column.

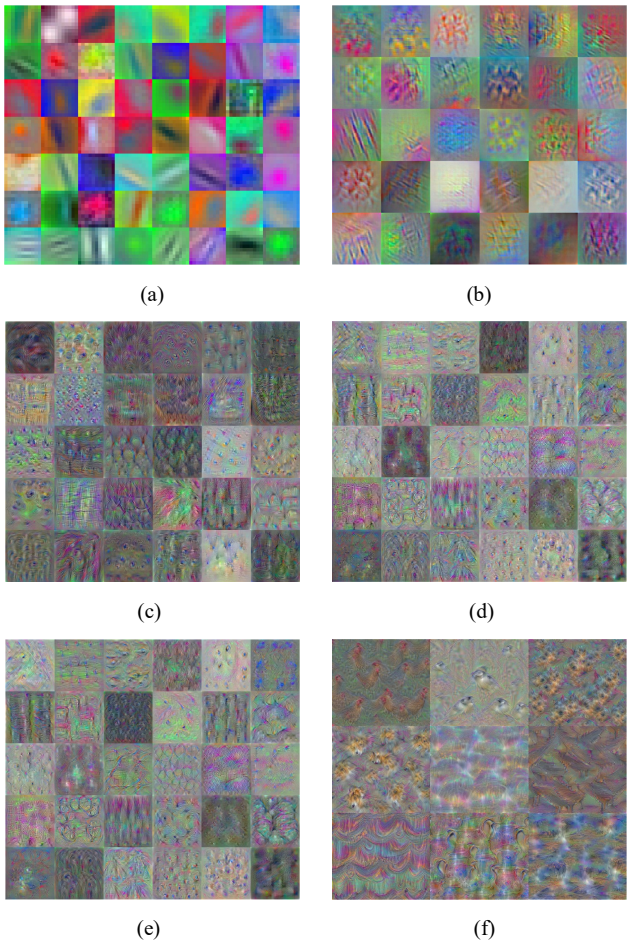


Fig. 14. Convolution kernel visualization results in six convolution layers of msCNN.

classification results of *CUB-200-2011* dataset. In order to better represent the samples of error classification, we extend the partial color gamut (0-0.2) to make the samples more obvious. The sample in the white box is the category with the highest misclassification error rate. According to the result of confusion matrix, we selected the top four pairs of categories that are misclassified by bilinear msCNN model, as shown in Fig. 13. The most confused pair of categories is "American crow" and "Common raven", which look remarkably similar. The main difference between the two sub categories is concentrated in the complex background, and the foreground target is difficult to distinguish effectively because of the extremely subtle differences. Without the help of some strong supervised information, it is difficult to distinguish them right now. Investigate features by observing which areas in the convolutional layers activate on an image and comparing with the corresponding areas in the original images.

Then, we visualize some of the convolution kernels in the first three convolution layers by deconvolution operation, and judge the features of the network by observing convolution kernels. The calculation method of deconvolution operation is given by

$$y_l = \sum_{k=1}^{K} z_{k,l} * f_{k,l}^{T} \tag{19}$$

where y_l represent the deconvolution output, $z_{k,l}$ is the convolution kernel parameter and $f_{k,l}$ is the weights of the convolution layer. So we get the visualization results of the convolution kernels in the msCNN A on *CUB200-2011*

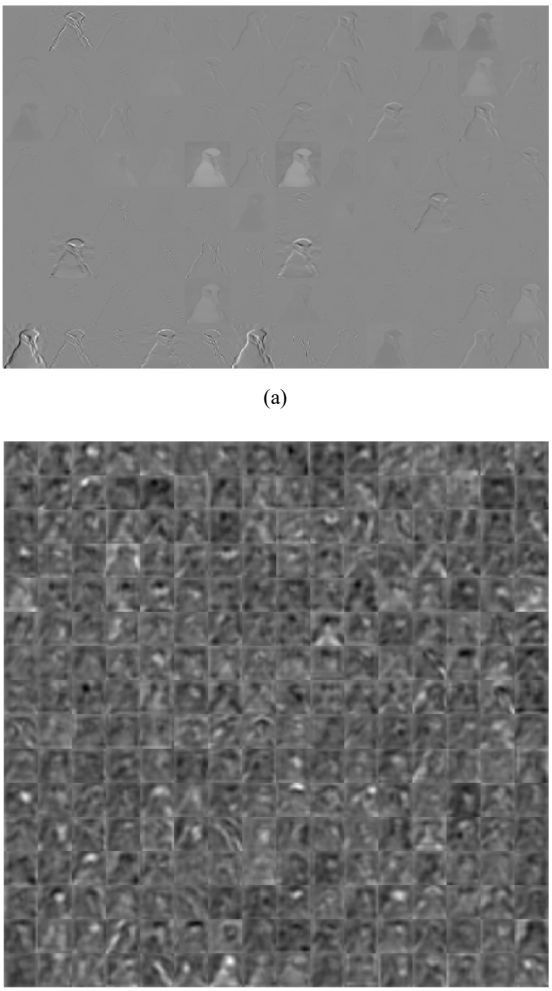


Fig. 15. Feature maps of bottom convolution layers and top level convolution layers in CNN.

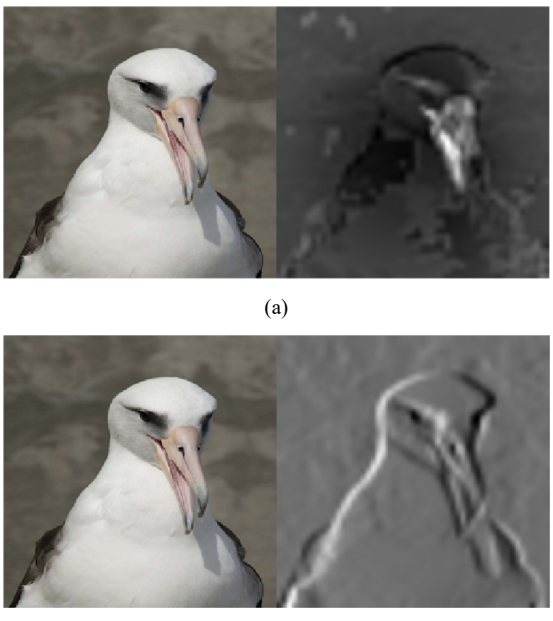


Fig. 16. Two pairs of feature maps with the highest activation degree.

dataset by deconvolution operation. As shown in Fig. 14, (a), (b), (c), (d), (e)and (f) represent the visual image of the convolution kernels in the first layer, the second layer, the forth layer, the fifth layer, the seventh layer and the eighth layer of msCNN A respectively. It can be seen that the convolution kernels in the shallow layer learns more about the bottom features of color, texture, edge and so on. With the increase of the number of convolution layers, the content of convolution kernels is more abstract. We believe that the features extracted by CNN begin to cross the semantic gap and turn into high-level semantic features. In the end, these features are fused and the classification results are output by the softmax layer.

In order to observe the influence of convolutional kernels of CNN on the input samples, we select a part of the output feature maps of the bottom convolutional layers and output feature maps of the top level convolution layer, respectively, as shown in Fig. 15(a) and 15(b). We can investigate features by observing which areas in the convolutional layers activate on an image and comparing with the corresponding areas in the original images. In the image, white pixels represent strong positive activations and black pixels represent strong negative activations. A channel that is mostly gray does not activate as strongly on the input image. The position of a pixel in the activation of a channel corresponds to the same position in the original image. A white pixel at some location in a channel indicates that the channel is strongly activated at that position. It can be seen from Fig. 15(a) that the contour and edge features of the object are extracted effectively, and the features of the key areas such as the eyes and mouth of the bird are enhanced. Some of the feature maps are complementary, which can be used to express the features in different gray levels and thus can effectively avoid the influence caused by the inconsistency of illumination and color. It can be seen from the Fig. 15(b) that the absolute position of the object is weakened after the maximum filtering of the pooling layer, and the relative position relationship between different regions in the image is enhanced, which makes the network invariant to changes in rotation, translation and scale to some extent. Many of the feature maps contain areas of activation that are both light and dark. These are positive and negative activations, respectively. However, only the positive activations are used because of the rectified linear unit (ReLU), even in PRELU, their role in the testing process is very small, only to prevent neuronal necrosis in the training process. As shown in Fig. 16, there are two pairs of the feature maps that are most strongly activated in the CNN. The left is the input image of the object and the right is the feature map of the convolution layer in CNN. The feature map in the Fig. 16(a) reflects the most distinguished local features (beak) in birds, while Fig. 16(b) reflects the overall contour features of birds. These features are fused and encoded by the fully connected layer to form abstract semantic features for final classification.

Previous machine learning approaches often manually designed features specific to the problem, but these deep convolutional networks can learn useful features for themselves. In summary, because of these different kinds of convolution kernels in CNNs, the network has the ability to extract different features. The processing of the subsequent feature maps in the network is based on the results of the previous layer, the network can gradually extract and combine more clear features. So it has a more comprehensive description of the image features and improve the accuracy of image classification.

VII. Conclusion

In this paper, we presented a bilinear msCNN model for fine-grained object classification and proved their effectiveness on various fine-grained object datasets. By constructing the scale space in the CNN and the usage of freeze-path technique in the convolution layers, we can improve generalization ability and obtain stable and effective features in the training process for the final classification. In the training process of network, the new loss function with penalty term based on the sample distribution can be used to establish an optimized feature boundary and improve the overfitting problem. In the experimental part, we analyze the role of CNN in the fine-grained objects classification process through the visualization of convolution kernels and feature maps. Remarkably, the performance is better than methods that rely on manually annotations like part or bounding box for training. Theoretically, generalization ability of this structure makes it applicable to any graph structures.

At the same time, the experience points out a few lessons and future directions, which we summarize as the followings:

- Whether the classification ability of CNN is optimal. If the effective features can be judged and extracted by convolution layers, it needs to be verified if fine-grained object classification results will be improved by using SVM instead of the fully connected layers.
- More appropriate network connection method. Linear parallel may not be the optimal structure. It still needs further research on how to break the symmetry of the network so that they coordinate with each other.
- Over-fitting problem still exists in the training process of the CNN. We plan to further explore how to enhance the generalization ability of CNN from the perspective of data distribution and visualization.

We are actively pursuing the above directions in the future studies.

References

[1] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, “Caltech-UCSD birds 200,” 2010.

[2] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li, “Dataset for fine-grained image categorization. In First Workshop on Fine-Grained Visual Categorization,” In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[3] M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[4] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, “Fine-grained visual classification of aircraft,” *Technical report*, 2013.

[5] Y. Chai, V. Lempitsky, and A. Zisserman, “Symbiotic segmentation and part localization for fine-grained categorization,” In IEEE *International Conference on Compuer Vision*, 2013, pp. 321-328.

[6] L. Bourdev, S. Maji, and J. Malik, "Describing people: A poselet-based approach to attribute classification," In *IEEE International Conference on Computer Vision*, 2011, pp. 1543-1550.
[7] N. Zhang, R. Farrell, and T. Darrell, "Pose pooling kernels for sub-category recognition," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3665-3672.
[8] H. J´egou, M. Douze, C. Schmid, and P. P´ere, "Aggregating local descriptors into a compact image representation," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304-3311.
[9] F. Perronnin, J. S´anchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," In *European Conference on Computer Vision*, 2010, pp. 143-156.
[10] D. G. Lowe, "Object recognition from local scale-invariant features," In *IEEE International Conference on Computer Vision*, 1999, pp. 1150-1157.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Image Net: A Large-Scale Hierarchical Image Database," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248-255.
[12] C. Mircea, M. Subhransu, K. Iasonas, et al, "Deep Filter Banks for Texture Recognition, Description, and Segmentation," International Journal of Computer Vision, 2016, pp. 65-94.
[13] T. Y. Lin, A. Roychowdhury, S. Maji, "Bilinear CNN Models for Fine-Grained Visual Recognition," In *IEEE International Conference on Computer Vision*, 2016, pp. 1449-1457.
[14] M. A. Goodale and A. D. Milner," Separate visual path-ways for perception and action," *Trends in neurosciences*, 15(1), 20–25, 1992.
[15] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," *Technical Report CNS-TR-2011-001*, Cal Tech, 2011.
[16] A. S. Razavian, H. Azizpour, J. Sullivan, et al., "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 512-519.
[17] Lecun. Yann, Y. Bengio, and G. Hinton, "Deep learning," *Nature* 521.7553(2015), 436-444.
[18] K. Jarrett, K. Kavukcuoglu, M. Ranzato, et al., "What is the best multi-stage architecture for object recognition," In *IEEE International Conference on Computer Vision*, 2010, pp. 2146 - 2153.
[19] Bengio. Yoshua, "Practical recommendations for gradient-based training of deep architectures," *Neural Networks: Tricks of the Trade*, Springer Berlin Heidelberg, 2012, pp. 133-144.
[20] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *IEEE Computer Science*, 2015.
[21] C. Szegedy, W. Liu, Y. Jia, et al., "Going deeper with convolutions," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1-9.
[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, et al., "Improving neural networks by preventing co-adaptation of feature detectors," *IEEE Computer Science*, 2012, 3(4), pp. 212-223.
[23] S. Ioffe, C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *IEEE Computer Science*, 2015, pp. 448-456.
[24] Y. Chai, V. Lempitsky, and A. Zisserman, "Symbiotic Segmentation and Part Localization for Fine-Grained Categorization," *IEEE International Conference on Computer Vision*, 2013, pp. 321-328.
[25] T. Xiao, Y. Xu, K. Yang, et al., "The application of two-level attention models in deep convolutional neural network for fine-grained image classification," *IEEE Computer Science*, 2014, 40(1), pp. 842-850.
[26] Y. Zhang, X. S. Wei, J. Wu, et al., "Weakly Supervised Fine-Grained Categorization with Part-Based Image Representation," *IEEE Transactions on Image Processing*, 2016, 25(4), pp. 1713-1725.
[27] J. Yang, Q. Wu, Z. Qu, et al., "An enhanced density clustering algorithm for datasets with complex structures," *IAENG International Journal of Computer Science*, 2017, pp. 44(2):150-156.
[28] M. Simon, E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," In: *Proceedings of the 15th IEEE International Conference on Computer Vision*. Santiago, 2015, pp. 1143−1151.
[29] S. Branson, G. V. Horn, S. Belongie, et al., "Bird Species Categorization Using Pose Normalized Deep Convolutional Nets," *Eprint Arxiv*, 2014.
[30] J. T. Lalis, "A New Multiclass Classification Method for Objects with Geometric Attributes Using Simple Linear Regression," *IAENG International Journal of Computer Science*, 2016, pp. 198-203.
[31] M. D. Zeiler, G. W. Taylor, R. Fergus, "Adaptive Deconvolutional Networks for Mid and High Level Feature Learning," In *IEEE International Conference on Computer Vision,* 2011, pp. 2018-2025.
[32] K.P. Murphy, "Machine learning: a probabilistic perspective, adaptive computation and machine learning," *MIT Press*, 2012.
[33] C. J. Tu, L. Y. Chuang, J. Y. Chang, et al., "Feature Selection using PSO-SVM," *IAENG International Journal of Computer Science*, 2007, pp. 33(1):111-116.
[34] K. He, X. Zhang, S. Ren, et al., "Deep Residual Learning for Image Recognition," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 770-778.
[35] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986, pp. 533-536.
[36] Y. Gong, L. Wang, R. Guo, et al., "Multi-scale Orderless Pooling of Deep Convolutional Activation Features," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 392-407.
[37] M. Jaderber, K. Simonyan, A. Zisserman, et al., "Spatial Transformer Networks," In *Conference and Workshop on Neural Information Processing Systems*, 2016, pp. 2017-2025.
[38] J. Krause, H. Jin, J. Yang, et al., "Fine-grained recognition without part annotations," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5546-5555.
[39] Q. Chen, Z. Song, R. Feris, et al., "Efficient Maximum Appearance Search for Large-Scale Object Detection," In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3190-3197.
[40] J. H. Jacobsen, J. V. Gemert, Z. Lou, et al., "Structured Receptive Fields in CNNs," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2610-2619.
[41] C. Szegedy, S. Ioffe, V. Vanhoucke, et al., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv.org*, 2016.
[42] K. Chatfield, K. Simonyan, A. Vedaldi, et al., "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *IEEE Computer Science*, 2014.
[43] K. He, X. Zhang, S. Ren, et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," In *IEEE International Conference on Computer Vision*, 2015, pp. 1026-1034.
[44] A. D. Forbes, "Classification-algorithm evaluation: Five performance measures based on confusion matrices," *Journal of Clinical Monitoring*, 1995, pp. 11(3):189-206.
[45] M. D. Zeile, R. Fergus, "Visualizing and Understanding Convolutional Networks," In *European Conference on Computer Vision*, 2014, pp. 818-833.
[46] Y. Ghanou, G. Bencheikh, "Architecture optimization and training for the multilayer perceptron using ant system," *IAENG International Journal of Computer Science*, pp. 20-26, 2016.
[47] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013, pp. 154-171.
[48] M. D. Zeiler, G. W. Taylor, R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," In *IEEE International Conference on Computer Vision*, 2011, pp. 2018-2025.

Qinghe Zheng was born in Jining, Shandong, China in 1993. He received his B.S. degree from Xi'an University of Posts and Telecommunications in 2014 and begin work for a M.S. degree in Shandong University in 2015. His research direction is computer vision and machine learning.