IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received August 26, 2018, accepted September 18, 2018, date of publication September 20, 2018, date of current version October 17, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2871621

Fine-Grained Image Classification With Gaussian Mixture Layer

JINGYUN LIANG[1], JINLIN GUO[1], (Member, IEEE), XIN LIU[2], AND SONGYANG LAO[1]
[1]College of System Engineering, National University of Defense Technology, Changsha 410072, China
[2]Computer Science and Engineering Department, University of Oulu, 90570 Oulu, Finland

Corresponding author: Xin Liu (linuxsino@gmail.com)

This work was supported by the National Natural Science Foundation of China under Project 61571453.

ABSTRACT Fine-grained image classification aims at recognizing different subordinates in one basic-level category, for example, distinguishing species of birds. Compared with basic-level classification, it has both low inter-class and high intra-class variances. Therefore, utilization of discriminative parts is crucial for fine-grained classification. In this paper, we propose a Gaussian mixture model, which fuses part features by Gaussian mixture layer. More specifically, it first generates a set of part proposals by selective search. Then, we extract image feature maps from mid-layers of convolutional neural networks. Feature maps and part proposals are used for calculating part features via spatial pyramid pooling. Next, Gaussian mixture layer treats part features as data points and uses several Gaussian components to model their distribution. It finds clusters for input and generates output features based on combination of cluster center. Finally, the output feature can represent the whole image and is used for classification. Training process of the model consists of two loops. The outer loop is the optimization of the whole network, and the inner loop is about the EM algorithm used in Gaussian mixture layer. Experiments demonstrate higher or similar performance on four fine-grained data sets compared with the state-of-the-arts. More discussions on Gaussian mixture layer are also provided.

INDEX TERMS Fine-grained image categorization, Gaussian mixture model, convolutional neural network.

I. INTRODUCTION

Fine-grained image classification focuses on recognizing similar subordinates in the same basic-level category. It has promising applications in species identification, vehicle monitoring and online shopping. People can easily cope with traditional image classification, such as classifying dogs, cats and bicycles in the ImageNet challenge dataset [1]. However, in fine-grained tasks like bird species classification, different subcategories have almost the same global appearances, and their differences mainly exist in local and subtle areas. Such low inter-class variances make it difficult to distinguish hundreds of subordinates even for human beings. At the same time, intra-class variances are large and diversified. Different poses, views and illumination conditions in one class make the task further challenging.

Many fine-grained datasets, such as CUB-200-2011 [2], Stanford Cars [3], FGVC-Aircraft [4] and Stanford Dogs [5] (shown in Figure 1), have been collected for research. With the rapid development of deep learning [6], fine-grained classification have made great progress [7]–[11] in recent years. As other computer vision problems, they achieve significant performance boosts compared with hand-crafted features. Convolutional neural networks [12] can mine inherent patterns of data and learn more effective deep convolutional features for fine-grained classification. Though CNN is powerful in feature representation, it usually lacks modelling of semantic object parts in mid-layers [13]. Therefore, many models add effective modifications on convolutional networks, including localization modules and feature encoding methods.

With local and subtle variances, different subcategories are often distinguished by parts, such as color of eyes, texture of feathers and shape of beaks for birds. As a result, using these discriminative parts, instead of similar parts, is crucial for fine-grained classification. The localization of discriminative parts becomes a core problem. Some methods [14]–[16] directly use pre-annotated part locations, which achieves mentionable performance. However, it is hard to collect part annotations for every dataset, and human-defined locations are not necessarily suitable for computers.

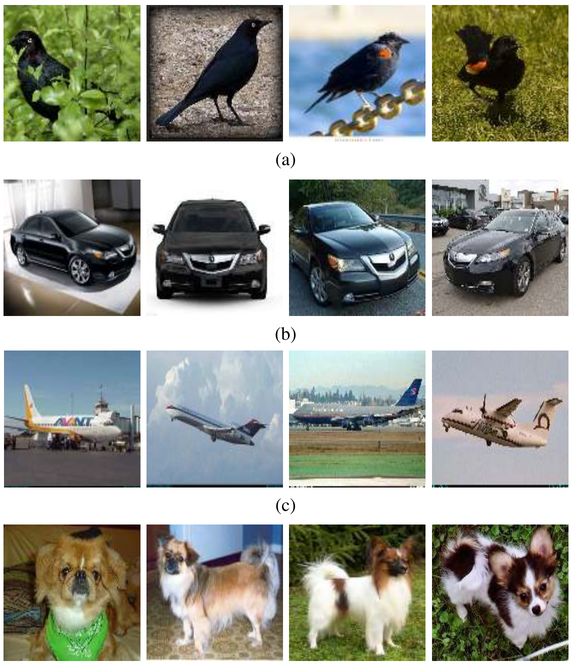


FIGURE 1. Illustration of four fine-grained image datasets, including CUB-200-2011 [2], Stanford Cars [3], FGVC-Aircraft [4] and Stanford Dogs [5]. (a) CUB-200-2011 (bird species. (b) Stanford Cars (car models). (c) FGVC-Aircraft (aircraft variants). (d) Stanford Dogs (dog breeds).

Therefore, we introduce an unsupervised mechanism for discriminative part discovery in fine-grained image classification. Using image-level labels only, we design a Gaussian mixture network (GMNet) equipped with a proposed Gaussian mixture layer. In GMNet, image parts are selected iteratively in the Gaussian mixture layer. Then, they are projected back to convolutional feature maps and jointly optimized with other model parameters. In detail, we first generate part proposals for input images. These proposals might contain discriminative and non-discriminative parts as well. To compare part features and select informative ones, fixed-length features are extracted for image parts. We use convolutional neural networks (e.g. VGG16, VGG19 [17]) as feature extractors, and operate on their high layers (e.g. layer *relu*5_3 of VGG16). It is because high-level features are thought to be more representative and class-specific [18]. Then, part features are viewed as a set of data points and feed into the Gaussian mixture layer.

Gaussian mixture layer is inspired by the widely used Gaussian mixture model [19], [20], which can fit theoretically any distribution using a number of Gaussian distributions. Different Gaussian components learn different views of the input data. The mean value of a Gaussian component is regarded as a clustering center. Therefore, we use mean values of Gaussian components as representatives of part features, and these representatives can be weighted and summed to be one feature. In training, the clustering finding problem can be solved by Expectation Maximization algorithm [21]. With Gaussian mixture layer, part features are fused, in a discriminative way, to form a final feature, which can represent the whole image. As for following classification, we further add a softmax layer to give prediction for different subordinates.

It is interesting to compare our GMNet with other discriminative part finding methods. The two-level attention model [22] uses spectral clustering to find groups in mid-level CNN filters according to their interested parts. In stead of using filters as part detectors, our model directly cluster part features, which can be incorporated into the forward-backward training process and fit data-dependent distributions accordingly. Another model [23] also selects useful parts by clustering, and part proposals assigned to important clusters are chosen. However, it is based on raw images, and thus susceptible to illumination, view and distortion. In comparison, GMNet employs high-level convolutional features of parts, which keeps essential characteristics and remits effects of other factors.

The contributions of this paper include: 1) we design a straightforward and effective framework named Gaussian mixture network. It extracts part features and fuses them for fine-grained image classification. The model does not need bounding box or part annotation, and it can be trained in a jointly optimized way. 2) we propose a Gaussian mixture layer, which aggregates key information from a set of part features. It can be inserted as a module of feature fusion, bringing a data-driven method compared with commonly used concatenation and summation. 3) we conduct numerous experiments on fine-grained datasets, including CUB-200-2011, Stanford Cars, FGVC-Aircraft and Stanford Dogs. The performance is improved by a significant margin compared with the state-of-the-art. We also give comprehensive analyses of the proposed GMNet and the Gaussian mixture layer.

The rest of the paper is organized as follows: Section II gives brief reviews of fine-grained image classification and Gaussian mixture model. The main model is presented in Section III in detail. Section IV presents related experiments and gives analyses. At last, we conclude the paper in Section V.

II. RELATED WORK

Our work is mainly related to two lines of research: fine-grained image classification and clustering methods. We review recent works in the literature in this section.

A. FINE-GRAINED IMAGE CLASSIFICATION

Fine-grained image classification is closely related to generic image classification. Traditionally, image classification is dominated by delicate hand-crafted features [24]–[28], e.g., SIFT, LBP and HOG. Early works [29]–[32] in fine-grained classification also use these traditional feature descriptors. However, design of descriptors are time-consuming and performance of them are unsatisfactory. In recent years, convolutional neural networks (CNN) [12], [17], [33] have revolutionized the field of computer vision. It performs remarkably good in classification and is supposed to be able to extract much more powerful representations than traditional features. Most recent works are based on CNN. Therefore, fine-grained image classification also moves to the era of deep learning.

Reference [15] is one of the early works based on deep learning. It follows the routine of R-CNN [34] to generate semantic part prediction with geometric constraints and classify images using pose-normalized representation. It shows that better localization of parts does lead to further improvement of classification. Reference [14] trains different networks for different kinds of parts, for example, heads and bodies of birds. Features of them are concatenated to form one long feature, followed by a classifier. Besides, the location information comes from official annotation of CUB-200-2011, so it is not applicable for other datasets without part annotation. Reference [16] also concatenates the image feature and its part features, but it presents an extra localization network for generation of object parts. Reference [13] treats fine-grained classification as object detection. It detects semantic object parts and combines them for recognition. Nevertheless, it is also dependent on extra annotation.

The intuition of [22] is to find foreground object and parts for discriminative feature extraction. It integrates bottom-up attention and object-level attention and part-level attention at the same time. Object-level FilterNet is used to filter out background patches, while part-level DomainNet is employed as semantic part selector. Spectral clustering is used to find specific detectors from filters. Reference [23] also chooses a weakly-supervised way. It generates multi-scale part proposals at first. Then, clustering algorithm is used to find important clusters and select important parts. Final image representation is computed via Fisher Vector [35].

There are also many other models based on attention mechanism or higher order representation in fine-grained classification. Reference [36] progressively learns coarse to fine region attention by its attention proposal sub-network. It trains a feature learning network for each scale, and features generated by different scales of images are concatenated for final prediction. Reference [37] generates multiple part attentions from feature channels. Different part features are also concatenated into a fully-connected fusion layer. LSTM [38] can also used to optimize positions of discriminative candidate parts in [10] and [39]. Besides attention models, higher order representation learning is also explored. Reference [40] produces second-order bilinear features through pairwise feature interactions and pooling. Reference [41] generalizes average pooling and bilinear pooling to the α-pooling, which allows for modification of pooling strategy. Reference [7] and [42] further expand second-order pooling to higher-order pooling by kernel fusion.

B. CLUSTERING METHODS

Clustering is a kind of unsupervised learning method, which groups a set of data points to ensure that data points in one cluster are more similar than those in other clusters. K-means [43] is an iterative clustering algorithm that tends to find local maxima in each iteration. It is often used in the computer vision literature, such as image classification [44], object detection [45] and image segmentation [46]. In fine-grained classification, [47] proposes a hierarchical part matching model and uses K-means to training the codebook. Reference [48] employs K-means to cluster similar classes and learns different networks for each subsets. Reference [49] uses K-means to cluster local descriptors and generate visual dictionary. Encoded feature vectors are classified by a SVM. Though K-means is very fast and effective, it suffers from naive use of mean value for the cluster center. For example, it cannot handle the situation where mean values of clusters are close to each other.

Gaussian mixture model [19] assumes that data points are Gaussian distributed, which is more flexible than K-means' assumption of being circular. Besides, it makes soft assignments instead of hard clustering that assigns each data point to a cluster center. It is also widely used for clustering in computer vision. Reference [50] uses a Gaussian mixture model to model distribution of features, while [11] uses it for Fisher Vectors. Based on Gaussian mixture model, [51] and [52] design a foreground color model for co-segmentation, which follows a graph-cut approach.

III. METHODOLOGY

In this section, we first introduce the pipeline of the whole model named GMNet. Then, we focus on the proposed Gaussian mixture layer. The details of model training are clarified in the end.

A. THE PIPELINE

In fine-grained image classification, parts that are beneficial for classification exist in local and subtle areas. Small visual differences are easily overwhelmed by other factors like poses, views and illumination. Therefore, to fully utilize discriminative object parts for recognition, we propose the Gaussian mixture model (GMNet). At the beginning, we generate possible discriminative regions by selective search [53] to alleviate the negative impacts of backgrounds of images. After that, we treat a convolutional neural network as a feature extractor to extract deep features. For the benefit of efficiency, we extract image feature for a whole image, and then use spatial pyramid pooling [54], [55] to calculate features of parts. In the end, part features of the same image are input into the Gaussian mixture layer, which models the distribution of features and generates a representative feature of them. The architecture of the model is shown in Figure 2, and the details are as follows.

For the generation of possible discriminative regions, some methods [14]–[16] directly use provided part annotations. However, we tend to use image labels only. Selective search is an unsupervised and bottom-up candidate part generation algorithm. It groups pixels by segmentation and generates regions that might contain objects. Let X denote a input

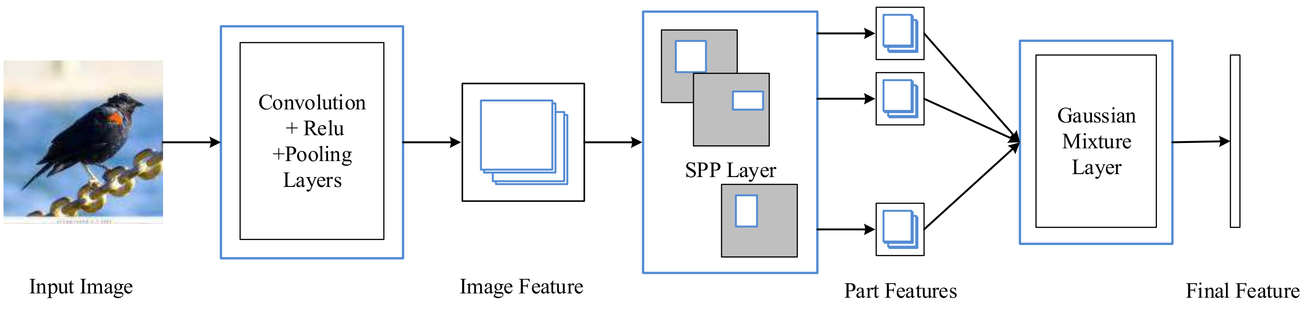


FIGURE 2. The architecture of Gaussian mixture network (GMNet). It aims to make better use of discriminative parts for fine-grained image classification. First, a backbone convolutional neural network, including several convolution, Relu and pooling layers, is used to extract image features. Then, we generate part proposals by Selective Window, and use SPP layer to extract part features from the image feature. At last, the Gaussian mixture layer fuses part features to form a representative final feature for classification.

image, SSW outputs a set of part proposals $\mathcal{P} = \{P_i\}_{i=1}^{N}$, where N is the number of image parts. In another view, it is also a kind of data augmentation, which provides multiple scales and views of the original image. Expansion of training data is beneficial for the optimization of convolutional neural networks.

Convolution neural network (CNN) is employed to extract deep features from images. It often consists of multiple stacked layers, for example, convolution layers, Relu layers and pooling layers. Commonly used networks include AlexNet [12], VGG [17] and ResNet [56]. Generally, fine-grained datasets often have several hundred subordinates, but each subordinate only has dozens of training images. Due to the large parameters of CNN, direct training might lead to over-fitting. As a result, we use networks that are pre-trained on the ImageNet challenge dataset. Based on trained parameters, we go on fine-tuning networks to adapt to the special fields of fine-grained data. To gain features with necessary spatial information, we remove rear fully-connected layer and extract features from mid-level layers, e.g., *relu*5_3 of VGG16. Let f denote the process of feature extraction, we can use $f(X)$ to represent the image feature of X.

Given part proposals and the original image feature, we can directly obtain part feature by spatial pyramid pooling (SPP). SPP divides a part's corresponding area on image feature map into several spatial bins and pools them to form a fixed-length representation of the part. It saves numerous repeated calculation compared with feeding image parts to CNN separately, because many parts are overlapped. The part features $f(X;\mathcal{P})$ is represented as follows:

$$f(X;\mathcal{P}) = SPP(f(X), \mathcal{P}) \tag{1}$$

where $f(X)$ is the original image feature, and $\mathcal{P}$ is the part proposals. Part features are feed into the Gaussian mixture layer, which is detailed in the next subsection.

B. GAUSSIAN MIXTURE LAYER

We propose a Gaussian mixture layer for feature fusion. In this layer, the input is the part features while the output is the fused feature. The kernel of the layer is to train a Gaussian mixture model for input data points. It can fit the distribution of input and extract key information. The process is illustrated in Figure 3.

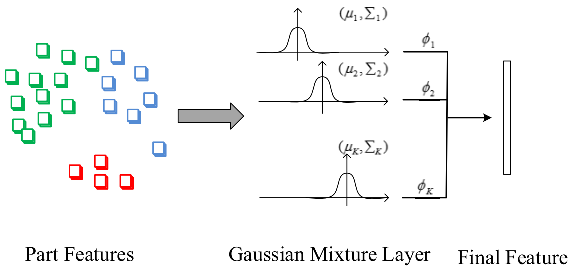


FIGURE 3. The illustration of the proposed Gaussian mixture layer. The input is a set of part features extracted from one image, and the output is the final feature for the whole image. The Gaussian mixture model is used to model part features' distribution and fuse them as well.

Gaussian mixture model is a probability-based clustering method, which uses linear combination of multiple Gaussian distribution functions to fit multimodal distributions. Given multi-dimensional random variable $\boldsymbol{x}$, Gaussian mixture model is as follows:

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{3}$$

where K is the number of components and ϕ_i is the mixture component weight for the i-th component. There is a constraint that $\sum_{i=1}^{K} \phi_i = 1$, so that the total probability distribution is normalized. The multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is the model's i-th component, which is defined as follows:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{\exp\left(-\frac{(\boldsymbol{x}-\boldsymbol{\mu}_i)^{\mathrm{T}}\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)}{2}\right)}{\sqrt{(2\pi)^K |\boldsymbol{\Sigma}_i|}} \tag{4}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vector and covariance matrix respectively. Treating generated part features as a set of

Algorithm 1 The Training of the Gaussian Mixture Network Using Back Propagation

```
Require: Training image: $X$; part proposals: $\mathcal{P} = \{P_i\}_{i=1}^N$;
    image label: $y$
Ensure: Network parameters: $W$; final image feature: $z$
1:  Initialize $W$ with ImageNet pre-trained parameters
2:  repeat
3:      procedure Forward
4:          Calculate image feature $f(X)$
5:                  ▷ $f$ denotes the process of feature extraction
6:          Calculate part feature $f(X; \mathcal{P}) \leftarrow SPP(f(X), \mathcal{P})$
7:                  ▷ $SPP$ denotes spatial pyramid pooling
8:          Generate final feature $z \leftarrow GML(f(X; \mathcal{P}))$
9:                  ▷ $GML$ denotes Gaussian mixture layer
10:         Compute loss $L \leftarrow CE(softmax(z), y)$
11:                 ▷ $CE$ denotes the cross-entropy loss
12:     end procedure
13:     procedure Backward
14:         Update $W \leftarrow W - \eta \frac{\partial L}{\partial W}$
15:                 ▷ $\eta$ is the learning rate
16:     end procedure
17: until Learning converges
```

data points, we use the Gaussian mixture model to model their distribution. With the training of convolutional neural network, part features are supposed to reflect substantive characteristics of their original image parts. Though different parts come from different local regions, parts from neighbouring regions exhibit semantic relevance and tend to be more similar. Therefore, we cluster similar part features and use Gaussian mixture model to find different clusters' center, which can represent the surrounding features.

Further, to fuse different clusters, we can refer to the mixture component weight ϕ_i, which reflects the importance of the corresponding Gaussian distribution in the whole model. The final feature is defined as the linear combination of the mean vector of each cluster center. Let z denote the final output feature, it is calculated as follows:

$$z = \sum_{i=1}^{K} \phi_i \boldsymbol{\mu}_i \tag{5}$$

z is an overall representation of the image. Therefore, we add a softmax layer and a cross-entropy loss after the Gaussian mixture layer. The image labels are used to train the whole model.

C. MODEL TRAINING

For the GMNet equipped with Gaussian mixture layer, the training process includes two loops. The outer loop is the forward-backward propagation of the GMNet, and the inner loop is inside the Gaussian mixture layer. The optimization of the Gaussian mixture layer is conducted in every forward step of the whole network's training. For a set of part features

Algorithm 2 Training of the Gaussian Mixture Layer. $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^N$ (for Clarity, We Use $\boldsymbol{x}$ to Represent Part Features $f(X; \mathcal{P})$) Is a Set of Input Data Points, and z Is the Generated Final Image Feature. N Is the Number of Part Proposals, While K Is the Number of Gaussian Components. ϕ_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ Are the k-th Mixture Component's Weight, Mean Vector and Covariance Matrix Respectively

```
1: procedure EM algorithm($\boldsymbol{x}$)
2:     Initialize $\gamma_{i,k} \leftarrow \frac{1}{K}$
3:     for $T$ iterations do
4:         $\phi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \leftarrow M-STEP(\gamma_{i,k}, \boldsymbol{x})$
5:         $\gamma_{i,k} \leftarrow E-STEP(\phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{x})$
6:     end for
7:     $z = \sum_{i=1}^K \phi_i \boldsymbol{\mu}_i$
8:     return $z$;
9: end procedure

1: procedure $M-STEP(\gamma_{i,k}, \boldsymbol{x})$
2:     $\forall k : \phi_k \leftarrow \frac{1}{N} \sum_{i=1}^N \gamma_{i,k}$
3:     $\forall k : \boldsymbol{\mu}_k \leftarrow \frac{\sum_{i=1}^N \gamma_{ik} \boldsymbol{x}_i}{\sum_{i=1}^N \gamma_{ik}}$
4:     $\forall k : \boldsymbol{\Sigma}_k \leftarrow \frac{\sum_{i=1}^N \gamma_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^2}{\sum_{i=1}^N \gamma_{ik}}$
5:     return $\phi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$
6: end procedure

1: procedure $E-STEP(\phi, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{x})$
2:     $\forall i, k : \gamma_{i,k} \leftarrow \frac{\phi_k \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \phi_j \mathcal{N}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$
3:     return $\gamma_{i,k}$
4: end procedure
```

from one image, we train the Gaussian mixture layer to fuse them and output a representative feature for the image.

In the forward propagation of GMNet, we follow the pipeline described in subsection III-A. The network maps the input (training image X and part proposals $\mathcal{P}$) into a final image feature z through layer-by-layer propagation. We use the cross-entropy loss function to decide the optimization direction. In backward propagation, network parameters, including all weights and biases in the backbone network, are optimized using stochastic gradient descent with momentum. The whole procedure is described in Algorithm 1.

As for Gaussian mixture layer, it is usually analytically impossible to find maximum likelihood solution through differentiating the log likelihood. so we choose the expectation-maximization algorithm (EM algorithm) to estimate the parameters of the Gaussian mixture model. It enables that the maximum likelihood strictly increases during iterations. Therefore, the algorithm can approach a saddle point or a local minimum during optimization. The EM algorithm consists of two repeatedly iterative steps: E-step and M-step. Given the parameters mixture component weight ϕ_k, mean vector μ_k and covariance matrix σ_k, the E-step calculates the expectation of component assignments $\gamma_{i,k}$ of data points $\boldsymbol{x}$. The M-step updates the parameters ϕ_k, μ_k and σ_k by maximizing the expectations given $\gamma_{i,k}$. We run EM algorithm for several iterations until parameters have been finalized in Gaussian mixture layer. The detailed algorithm is shown in Algorithm 2.

Particularly, the Gaussian mixture layer's optimization is independent on the whole model's, but it still can propagate gradient from its next layer to its previous layer. The whole network is trained in a jointly optimized way. When training is done, we compute the final values of ϕ_k, μ_k, σ_k and $\gamma_{i,k}$. In the forward propagation of the Gaussian mixture layer, we have:

$$\begin{aligned} z &= \sum_{k=1}^{K} \phi_k \boldsymbol{\mu}_k \\ &= \sum_{k=1}^{K} \left(\frac{\phi_k}{\sum_{i=1}^{N} \gamma_{ik}} \sum_{i=1}^{N} \gamma_{ik} \boldsymbol{x}_i \right) \end{aligned} \tag{6}$$

In the back propagation of the layer, the gradient is calculated as follows:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{x}_i} &= \frac{\partial L}{\partial z} \frac{\partial z}{\partial \boldsymbol{x}_i} \\ &= \frac{\partial L}{\partial z} \sum_{k=1}^{K} \left(\frac{\phi_k}{\sum_{i=1}^{N} \gamma_{ik}} \gamma_{ik} \right) \end{aligned} \tag{7}$$

IV. EXPERIMENT

To prove the effectiveness of our GMNet, as well as gain more insights from the model, we conduct comprehensive experiments in this section. The results and analyses are given as follows.

A. DATA PREPARATION AND EXPERIMENTAL SETTING

We adopt four widely-used fine-grained image dataset for experiments, including CUB-200-2011 [2], Stanford Cars [3], FGVC-Aircraft [4] and Stanford Dogs [5].

- **CUB-200-2011** It is one of the most popular dataset for fine-grained image classification. It has 11,788 images from 200 bird subordinates. 5,994 images are selected for training, while the rest 5,794 images for testing. Approximately, 30 images are used in training for each subordinate, and 11∼30 for testing. In addition, it provides the most detailed annotations among datasets, including a subordinate label, a bounding box, 15 part locations and 312 binary attributes for each image.
- **Stanford Cars** It is a collection of car models, which contains 16,185 images of 196 subordinates. 8,144 and 8,041 images are selected as training set and testing set respectively. Each subordinates has 24∼68 training images and 24∼68 testing images. Labels at the level of Make, Model and Year, along with a bounding box, are provided.
- **FGVC-Aircraft** It contains 6,667 images for training and 3,333 images for testing. In total 100 aircraft model variants are collected, and each variant has 100 images (about 67 images for training and 33 for testing). Images are annotated with Model, Variant, Family and Manufacturer. A tight bounding box is also provided.
- **Stanford Dogs** It has 20,580 images of 120 dog breeds. It is divided as follows: 12,000 images for training and 8,580 images for testing. Each breed has 100 training images and 48∼152 testing images. Class labels and bounding boxes are annotated.

In the experiments, to avoid over-fitting, we adopt two data augmentation techniques for training set: five-scale image resizing (resize the short sides of images to be 250, 350, 450, 550 and 650 randomly while keeping aspect ratios) and horizontal image flip. We use default parameters in selective search method for part proposal generation, and 500 parts are collected. For fair comparison, we choose VGG16 and VGG19 as GMNet's backbone networks, whose parameters have been pre-trained on the ImageNet challenge dataset. Image features are extracted from *relu*5_3 (or *relu*5_4 for VGG19), and layers after that are removed. Next, a 7×7 max pooling SPP layer is added. It extracts $7 \times 7 \times 512$ features for each parts. Two fully-connected layers (4096 channels and 200 channels respectively) are used to reduce part features' dimensions. After that, we input these features into the Gaussian mixture layer, which performs well with 10 Gaussian components across different datasets. Last, a fully-connected layer and a softmax layer are attached for training and prediction.

In training, learning rate is initialized to be 1e-4 and divided by 10 every 10 epochs. Weight decay and momentum are 5e-4 and 0.9 respectively. For the convenience of implementation, batch size is set to 1. The stochastic gradient descent with momentum is used to optimize the loss function. 20 epochs are enough for a good convergence. As for the Gaussian mixture layer, it reaches a stable state in abut 10 iterations. All the experiments are implemented by Pytorch [57] and tested on a computer with Intel i7-3930K CPU, 64G main memory, and a Nvidia Titan X GPU. In testing, the short sides of images are scaled to be 650. The accuracy (the ratio between the number of correctly classified images and the number of testing images) is used to evaluate the classification performance.

B. COMPARISONS WITH STATE-OF-THE-ART METHODS

This subsection demonstrates the classification results on four fine-grained datasets. Overall, GMNet achieves better or comparable results across these datasets. Table 1 shows the comparison between our GMNet and previous methods on CUB-200-2011 [2]. The performance of different methods are mainly evaluate by the accuracy, and the dependences on bounding box or part annotation are also given in the table. Pose-normalized CNN [14] designs pose normalization schemes for deep convolutional features, which outperforms traditional features by a large margin. Part R-CNN [15] and Part-alignment CNN [51] are based on image parts. They generates pose-normalized representations by geometric constraints or co-segmentation. Part-stacked CNN [16] stacks image features and part features for classification, while Multi-granularity CNN [58] stacks image features from multi-granularity descriptors. Our GMNet is also based

TABLE 1. Comparison of results on CUB-200-2011 [2].

Approach	Bounding box	Part annotation	Accuracy
Pose-normalized CNN [14]	√	√	75.7
Part R-CNN [15]	√	√	76.4
Part-stacked CNN [16]	√	√	76.6
Part-alignment CNN [51]	√		82.8
Multi-granularity CNN [58]	√		83.0
FCAN [62]	√	√	84.3
SPDA-CNN [13]	√	√	85.1
Mask-CNN [59]		√	85.7
MSML [63]			67.9
VGG16 (fine-tuning) [17]			72.1
Two-level Attention CNN [22]			69.7
VGG19 (fine-tuning) [17]			77.8
DVAN [39]			79.0
NAC [64]			81.0
Multi-granularity CNN [58]			81.7
PDFS [11]			81.7
RACNN(scale2) [36]			82.4
Bilinear CNN [40]			84.1
Saliency-guided model [60]			85.1
RACNN(scale1+2+3) [36]			85.3
MACNN(2parts+object) [37]			85.4
OPAM [61]			85.8
MACNN(4parts+object) [37]			**86.5**
GMNet (VGG16)			84.5
GMNet (VGG19)			86.3

TABLE 2. Comparison of results on Stanford Cars [3].

Approach	Bounding box	Accuracy
FCAN [62]	√	91.5
MDTP [65]	√	92.5
Part-alignment CNN [51]	√	92.8
VGG16(fine-tuning)		83.8
VGG19(fine-tuning)		84.9
DVAN [39]		87.1
RACNN(scale2) [36]		90.0
Bilinear CNN [40]		91.3
MACNN(2parts+object) [37]		91.7
OPAM [61]		92.2
RACNN(scale1+2+3) [36]		92.5
MACNN(4parts+object) [37]		92.8
GMNet (VGG16)		91.7
GMNet (VGG19)		**93.5**

on image parts, but we find patterns from part features instead of concatenating them by rote. SPDA-CNN [13] and Mask-CNN [59] focus on semantic object parts like head and tail of birds. They resort to extra annotations like bounding box and part annotation in the training process. In contrast, GMNet only need subordinate labels and performs even better than those requiring extra information. As baselines, we fine-tuning VGG16 and VGG19 [17] on this dataset, with the last fully-connected layer modified only. GMNet surpasses these two baselines by large margins (12.4% and 8.5% respectively), which prove the effectiveness of our framework. Two-level Attention CNN [22] fuses object-level and part-level attention for classification. It also has an unsupervised module named part detector, which uses spectral clustering to find groups in parts. Compared with the multi-stage training process of the Two-level Attention CNN, we benefit from the end-to-end framework of GMNet, which can be optimized in a jointly way. Also, GMNet performs much better (more than 16%) than the network. PDFS [11] generates candidate patches like GMNet, but it uses patches to train deep filters. In comparison, GMNet directly uses these patches for feature extraction and fusion, which demonstrates better accuracies than PDFS. The Saliency-guided model [60] adds a saliency extraction network on the basis of faster R-CNN, it achieves 85.1% with the testing speed of 10.1 frames per second. With similar speed, GMNet outperforms the Saliency-guided model by a margin of 1.2%. Similar to the Two-level Attention CNN [22], OPAM [61] also combines two level attentions. Spatial constraints, saliency extraction and part alignment are designed to achieve better results. Two recent models are RACNN [36] and MACNN [37]. Both of them are attention models, which take advantage of feature map for discriminative part localization. Though they achieve comparable results with GMNet, they suffer from complex training strategies and combination of multiple models.

As illustrated in Table 2, GMNet achieves great results on Stanford Cars [3]. FCAN [62] designs attention networks for part attention, which finds important regions and crops them for later classification. The mechanism is a kind of 'hard' discriminative feature selection, which discards the rest regions. In contrast, generated parts in GMNet nearly cover all regions of images, and the Gaussian mixture layer can fuse different part features in a 'soft' way. MDTP [65] chooses triplet mining to train mid-level representations, while Part-alignment CNN [51] generate parts with co-segmentation and alignment. Nevertheless, they are all inferior to GMNet in classification performance. Compared with baselines VGG16 and VGG19 [17], GMNet still boosts the accuracies by 7.9% and 8.6%, achieving 91.7% and 93.5% respectively. The GMNet surpasses OPAM [61] with a gain of 1.3%, and it also performs better than both RACNN [36] and MACNN [37].

The results on FGVC-Aircraft [4] are shown in Table 3. The GMNet (VGG19) outperforms those methods with bounding box, including Multi-granularity CNN [58] and MDTP [65], by at least 2.1%. Bilinear CNN [40] is one of the pioneering works in higher order representation. It achieves 84.1% on this dataset. Compared with baselines VGG16 and VGG19 [17], GMNet improves the performance to 88.1% and 90.5% respectively. Besides, RACNN [36] and MACNN [37] report 88.2% and 89.9% as their best results, which are slightly lower than ours.

Table 4 demonstrates the classification results on Stanford Dogs [5]. AlignmentModel [66] is one of the early

TABLE 3. Comparison of results on FGVC-Aircraft [4].

Approach	Bounding box	Accuracy
Multi-granularity CNN [58]	√	86.6
MDTP [65]	√	88.4
VGG16(fine-tuning) [17]		80.1
VGG19(fine-tuning) [17]		83.3
Bilinear CNN [40]		84.1
RACNN(scale1+2+3) [36]		88.2
MACNN(2parts+object) [37]		88.4
MACNN(4parts+object) [37]		89.9
GMNet (VGG16)		88.1
GMNet (VGG19)		**90.5**

TABLE 4. Comparison of results on Stanford Dogs [5].

Approach	Bounding box	Accuracy
AlignmentModel [66]	√	50.1
FCAN [62]	√	84.2
NAC [64]		68.6
PDFS [11]		71.9
VGG16(fine-tuning) [17]		76.7
VGG19(fine-tuning) [17]		80.3
DVAN [39]		81.5
RACNN(scale2) [36]		85.9
RACNN(scale1+2+3) [36]		87.3
GMNet (VGG16)		86.0
GMNet (VGG19)		**88.1**

works in this field. It extracts traditional features for different parts of image and introduces both supervised and unsupervised alignments to the object. It only gets a 50.1% accuracy on this dataset, which demonstrates that the feature extraction process is crucial for fine-grained classification. NAC [64] designs an unsupervised part model discovery method by finding constellations of neural activations. By selecting a few parts for classification, it obtains 68.6% accuracy. DVAN [39] uses attentional LSTM to predict attention regions from feature maps, improving accuracy from the baseline 76.7% to 81.5%. On this dataset, GMNet also outperforms all the methods, reaching 88.1% accuracy.

GMNet is also competitive in the aspect of speed and memory usage. For training speed, it runs at about 5.3 images per second on with a Nvidia Titan X GPU. Generally, the training time of different datasets range from 8 hours to 17 hours. In testing, it achieves 10.0 frames per second, which is much faster than RACNN and MACNN (1 and 5.2 frames per second respectively). For memory usage, we take input images with random scaling. Therefore, GPU usage fluctuates between 4.2 GB and 11.8 GB, which is acceptable for most GPU devices.

C. THE EFFECT OF SUPER-PARAMETERS AND DATA AUGMENTATION

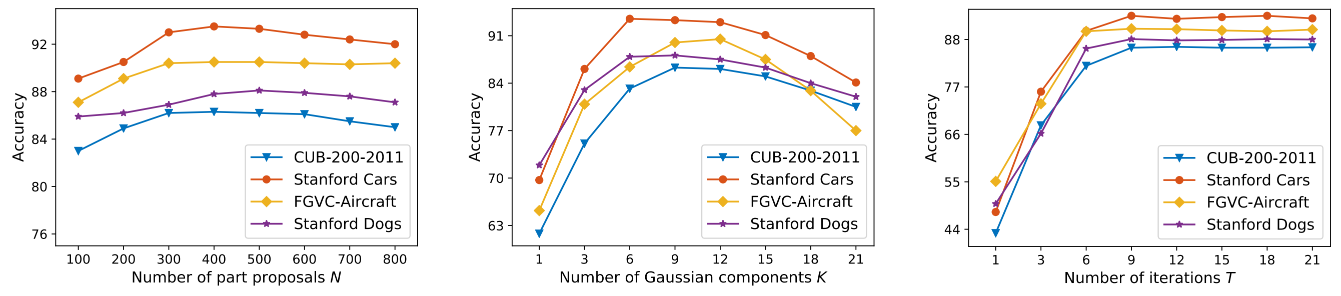
In the proposed GMNet, there are some parameters need to be chosen ahead of time. To measure the impacts of these super-parameters on performance, we conduct various experiments on different datasets using VGG19 as the backbone. In the investigation of one super-parameter, other settings are kept the same in experiments.

In part generation, we use selective search algorithm to generate abundant parts with high object confidence. As the authors recommend, most settings are set to default. In GMNet, we take part features as input, and the number of part proposals is exactly the number of input data points for the Gaussian mixture layer. Figure 4 (a) shows the effect of the number of part proposals N. When N is small, part proposals do not contain enough information. Therefore, the performance is not satisfactory. When N rises from 100 to 400, the accuracy increases as well. For large N ($N > 400$), the increase of N does not lead to further performance. It is because enough parts have nearly covered all discriminative regions, and more parts will not bring more useful information. When N continues to rise, there is a slight drop because there are many data points for the Gaussian mixture layer to model. Besides, more part proposals lead to more computation time and memory usage. We choose $N = 500$ for all datasets.

In the Gaussian mixture layer, we use combination of several Gaussian components to model the distribution of part features. As a kind of clustering method, we have to determine the number of cluster center, that is, the number of components K. Theoretically, too few components cannot describe the distribution, which is hard for the model to converge to a good state. However, increasing K will always reduce the error of clustering. Too many components cannot grasp the main characteristics of the distribution and give a good compression of the input. As can be seen from the Figure 4 (b), when $K < 9$, the accuracies keep rising with the increase of K. When $K > 12$, increase of K instead makes the accuracies drop. K between 9 and 12 performs best across different datasets, so K is set to be 10 in rest experiments.

The training of the Gaussian mixture layer is a relatively independent inner loop. The EM algorithm consists of two repeatedly iterative processes: E-step and M-step. Generally, after initialization, we set a fixed number of iterations T in training. Figure 4 (c) indicates the impact of number of iterations T on final results. Clearly, when T is smaller than 9, the accuracies keep rising, which corresponds to the convergence of the model. When T is larger than 9, the accuracies remain stable basically. It demonstrates that the algorithm has reached a good convergence. As a result, we set $T = 10$ for other experiments.

Data augmentation is widely used in the deep learning literature. We randomly rescale and flip input images in the training process. It provides multi-scale and multi-view training images for the network, which can relieve the problem of overfitting and make the model more robust. Table 5 shows the results on different datasets with or without data augmentation. The use of data augmentation can boost the accuracies by at least 0.8%. Also, such data augmentation



(a) (b) (c)

FIGURE 4. Effects of different super-parameters in GMNet. (a) shows the effect of number of part proposals N on accuracy, (b) demonstrates the impact of the number of Gaussian components K, and (c) presents the effect of number of iterations T on performance.

TABLE 5. Performance of GMNet with or without data augmentation.

With/without data augmentation	CUB-200-2011 [2]	Stanford Cars [3]	FGVC-Aircraft [4]	Stanford Dogs [5]
w	**86.3**	**93.5**	**90.5**	**88.1**
w/o	85.4	92.7	89.4	86.7

TABLE 6. Performance of different feature fusion methods.

Dataset	Gaussian mixture layer	K-means	Summation	Concatenation
CUB-200-2011 [2]	**86.3**	81.1	82.1	-
Stanford Cars [3]	**93.5**	88.4	89.1	-
FGVC-Aircraft [4]	**90.5**	84.1	86.4	-
Stanford Dogs [5]	**88.1**	80.7	82.4	-
Training speed(frames/sec)	5.3	7.1	10.4	0.4
Parameter	10*200*200	10*200	0	500*200*200

is pretty efficient in implementation. In default, we use data augmentation for other experiments.

D. HOW EFFECTIVE IS THE GAUSSIAN MIXTURE LAYER

To some extent, the Gaussian mixture layer is a kind of feature fusion methods. It models the distribution of different part features from an image and fuses them to form a final feature for classification. To evaluate the effectiveness of the Gaussian mixture layer, we compare it with other popular methods, including the widely-used K-means clustering algorithm, summation and concatenation. For K-means, we follow the routine of Gaussian mixture layer, we conduct a inner loop to train the algorithm. The number of the clustering center and the number of iterations are set to be 10 and 10 respectively. After training, the cluster with most data points is treated as the main cluster, and the center of it is used for further classification. For summation, we directly add all the part features and calculate their average feature. For concatenation, we concatenate these part features and use an extra fully-connected layer for dimension reduction.

The performance, including accuracies, training speed and parameters, on different datasets are shown in Table 6. It is clear that the Gaussian mixture layer performs well among these methods across different fine-grained datasets. K-means achieves at least 6.1% lower accuracy compared with the Gaussian mixture layer. It is because K-means cannot model the distribution well. In fact, it only makes use of a part of part features for final classification. Summation demonstrates surprising performance in experiments. It might due to their usage of all part features. However, concatenation does converge well and give reasonable results, because the added fully-connected layer contains too many parameters and might lead to over-fitting. For training speed, Gaussian consumes more time than K-means and summation. As for parameters in feature fusion, Gaussian mixture model has three kind of parameters: mixture component weight ϕ_i, mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. The amount of parameters is about $10 * 200 * 200$. K-means only needs to store the position of cluster center, so the parameter usage is $10 * 200$. Summation is the operation between part features and does not contain any parameter. On the whole, though Gaussian mixture layer contains more parameters and consumes more time, it achieves much better performance across datasets. It is still acceptable and practicable.

E. VISUALIZATION

We use the Gaussian mixture model to model the distribution of part features, under the hypothesis that a set of part features extracted from one image has its special distribution. However, it is hard for human to understand the high-dimensional distribution. Therefore, we turn to the t-distributed stochastic neighbour embedding (t-SNE) [67], which is a dimension

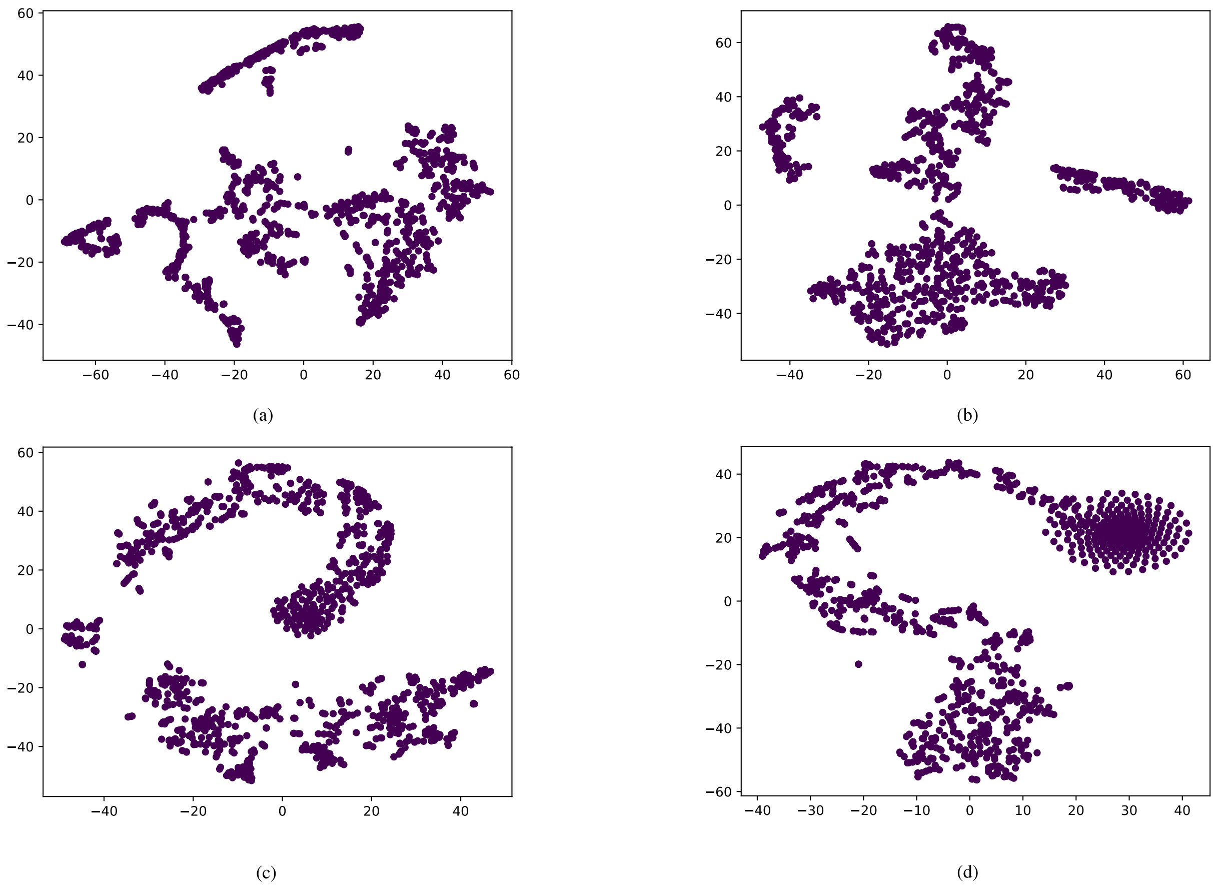


FIGURE 5. The t-SNE visualizations of part feature distributions. (a)(b)(c)(d) are from CUB-200-2011 [2], Stanford Cars [3], FGVC-Aircraft [4] and Stanford Dogs [5] respectively. One part feature correspond to one data point in this two-dimensional space. The Gaussian mixture layer takes these part features (data points) as input.

reduction technique widely used for high-dimensional data visualization. We use trained model to extract part features of one image from the GMNet, more specifically, just before the Gaussian mixture layer. The 200-dimensional part features are input into the t-SNE algorithm, and a 2-dimensional distribution map is the output. The visualizations of part feature distributions from different datasets, including CUB-200-2011 [2], Stanford Cars [3], FGVC-Aircraft [4] and Stanford Dogs [5], are illustrated in Figure 5. The distances in the map demonstrate the similarity of different features. Some features are similar and adjacent, while some features are far away from each other. The distribution of part features accords with our hypothesis. It is reasonable to use a clustering method (Gaussian mixture model) to cluster data points and model the distribution.

V. CONCLUSION

In this paper, we propose a Gaussian mixture model for fine-grained image classification. It first uses selective search for part proposal generation. Then, image feature maps are extracted from the backbone convolutional neural network. Spatial pyramid pooling is used to extract part features directly from the image feature map. Next, part features are input into the Gaussian mixture layer. The Gaussian mixture layer uses several Gaussian components to model the distribution of part features and extract key information. It uses EM algorithm for training and outputs a final feature for classification. Experiments on four fine-grained classification datasets demonstrate the effectiveness of the Gaussian mixture model. We achieve competitive performance compared with the state-of-the-arts, while enjoying acceptable computation time and memory usage. In addition, we conduct experiments to investigate the effects of three super-parameters and compare the Gaussian mixture layer with other feature fusion methods.

The future works are of two aspects: First, we will try to take advantage of higher order representations, such as the bilinear feature. It will promote the description ability of convolutional features. Second, we will attempt to avoid generate part proposals and use regional features from the image feature map directly. Both of them will be explored for better classification performance.

REFERENCES

[1] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[2] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-UCSD birds-200-2011 dataset," Univ. California, California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2011-001, 2011.

[3] J. Krause, M. Stark, D. Jia, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Jun. 2014, pp. 554–561.

[4] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. (2013). "Fine-grained visual classification of aircraft." [Online]. Available: https://arxiv.org/abs/1306.5151

[5] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *Proc. 1st Workshop Fine-Grained Vis. Categorization, IEEE Conf. Comput. Vis. Pattern Recognit.*, Colorado Springs, CO, USA, Jun. 2011, pp. 1–2.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[7] S. Cai, W. Zuo, and L. Zhang, "Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 511–520.

[8] S. Kong and C. Fowlkes, "Low-rank bilinear pooling for fine-grained classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7025–7034.

[9] X. He and Y. Peng. (2017). "Visual-textual attention driven fine-grained representation learning." [Online]. Available: https://arxiv.org/abs/1709.00340

[10] M. Lam, B. Mahasseni, and S. Todorovic, "Fine-grained recognition as HSnet search for informative image parts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6497–6506.

[11] X. Zhang, H. Xiong, W. Zhou, W. Lin, and Q. Tian, "Picking deep filter responses for fine-grained image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1134–1142.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, no. 2, 2012, pp. 1097–1105.

[13] H. Zhang *et al.*, "SPDA-CNN: Unifying semantic part detection and abstraction for fine-grained recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1143–1152.

[14] S. Branson, G. van Horn, S. Belongie, and P. Perona. (2014). "Bird species categorization using pose normalized deep convolutional nets." [Online]. Available: https://arxiv.org/abs/1406.2952

[15] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.

[16] S. Huang, Z. Xu, D. Tao, and Y. Zhang, "Part-stacked CNN for fine-grained visual categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1173–1182.

[17] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." (2014). [Online]. Available: https://arxiv.org/abs/1409.1556

[18] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[19] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE CVPR*, Jun. 1999, p. 2246.

[20] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. Int. Conf. Learn. Represent.*, 2018. [Online]. Available: https://openreview.net/forum?id=HJWLfGWRb

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc. B (Methodol.)*, vol. 39, no. 1, pp. 1–38, 1977.

[22] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang, "The application of two-level attention models in deep convolutional neural network for fine-grained image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 842–850.

[23] Y. Zhang *et al.*, "Weakly supervised fine-grained categorization with part-based image representation," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1713–1725, Apr. 2016.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[25] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[26] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2005, pp. 886–893.

[27] L. Liu and P. W. Fieguth, "Texture classification from random features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 574–586, Mar. 2012.

[28] L. Liu, J. Chen, P. Fieguth, G. Zhao, R. Chellappa, and M. Pietikäinen. (2018). "A survey of recent advances in texture representation." [Online]. Available: https://arxiv.org/abs/1801.10324

[29] R. Farrell, O. Oza, N. Zhang, V. I. Morariu, T. Darrell, and L. S. Davis, "Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 161–168.

[30] S. Yang, J. Wang, J. Wang, and L. G. Shapiro, "Unsupervised template learning for fine-grained object recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 3122–3130.

[31] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur, "Dog breed classification using part localization," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 172–185.

[32] N. Zhang, R. Farrell, and T. Darrell, "Pose pooling kernels for sub-category recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3665–3672.

[33] L. Liu *et al.* (2018). "Deep learning for generic object detection: A survey." [Online]. Available: https://arxiv.org/abs/1809.02165

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[35] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the Fisher vector: Theory and practice," *Int. J. Comput. Vis.*, vol. 105, no. 3, pp. 222–245, 2013.

[36] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1–3.

[37] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1–9.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] B. Zhao, X. Wu, J. Feng, Q. Peng, and S. Yan. (2016). "Diversified visual attention networks for fine-grained object classification." [Online]. Available: https://arxiv.org/abs/1606.08572

[40] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1449–1457.

[41] M. Simon, Y. Gao, T. Darrell, J. Denzler, and E. Rodner, "Generalized orderless pooling performs implicit salient matching," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, May 2017, pp. 1–10.

[42] Y. Cui, F. Zhou, J. Wang, X. Liu, Y. Lin, and S. Belongie, "Kernel pooling for convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1–7.

[43] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[44] S. Ranjan, D. R. Nayak, K. S. Kumar, R. Dash, and B. Majhi, "Hyperspectral image classification: A k-means clustering based approach," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Jan. 2017, pp. 1–7.

[45] R. Buck, "Cluster-based salient object detection using K-means merging and keypoint separation with rectangular centers," Ph.D. dissertation, Dept. Comput. Sci., Utah State Univ., Logan, UT, USA, 2016.

[46] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using K-means clustering algorithm and subtractive clustering algorithm," *Procedia Comput. Sci.*, vol. 54, pp. 764–771, Jan. 2015.

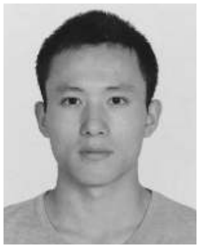
[47] L. Xie, Q. Tian, R. Hong, S. Yan, and B. Zhang, "Hierarchical part matching for fine-grained visual categorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1641–1648.

[48] Z. Ge, C. McCool, C. Sanderson, and P. Corke, "Subset feature learning for fine-grained category classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2015, pp. 46–52.

[49] W. Zhang, J. Yan, W. Shi, T. Feng, and D. Deng, "Refining deep convolutional features for improving fine-grained image recognition," *EURASIP J. Image Video Process.*, vol. 2017, no. 1, p. 27, 2017.

[50] Z. Ge, C. McCool, C. Sanderson, and P. Corke, "Modelling local deep convolutional neural network features to improve fine-grained image classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 4112–4116.

[51] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*. IEEE, Jun. 2015, pp. 5546–5555.
[52] X. He and Y. Peng, "Weakly supervised learning of part selection model with spatial constraints for fine-grained image classification," in *Proc. AAAI*, 2017, pp. 4075–4081.
[53] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.
[55] X. Chen and A. Gupta. (2017). "An implementation of faster RCNN with study for region sampling." [Online]. Available: https://arxiv.org/abs/1702.02138
[56] K. He, X. Zhang, S. Ren, and J. Sun. (2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385
[57] A. Paszke *et al.*, "Automatic differentiation in pytorch," in *Proc. NIPS*, 2017, pp. 1–4.
[58] D. Wang, Z. Shen, J. Shao, W. Zhang, X. Xue, and Z. Zhang, "Multiple granularity descriptors for fine-grained categorization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2399–2406.
[59] X.-S. Wei, C.-W. Xie, and J. Wu. (2016). "Mask-CNN: Localizing parts and selecting descriptors for fine-grained image recognition." [Online]. Available: https://arxiv.org/abs/1605.06878
[60] X. He, Y. Peng, and J. Zhao, "Fine-grained discriminative localization via saliency-guided faster R-CNN," in *Proc. ACM Multimedia Conf.*, 2017, pp. 627–635.
[61] Y. Peng, X. He, and J. Zhao, "Object-part attention model for fine-grained image classification," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1487–1500, Mar. 2018.
[62] X. Liu, T. Xia, J. Wang, Y. Yang, F. Zhou, and Y. Lin. (2016). "Fully convolutional attention networks for fine-grained recognition." [Online]. Available: https://arxiv.org/abs/1603.06765
[63] Q. Qian, R. Jin, S. Zhu, and Y. Lin, "Fine-grained visual categorization via multi-stage metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3716–3724.
[64] M. Simon and E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1143–1151.
[65] Y. Wang, J. Choi, V. Morariu, and L. S. Davis. (2016). "Mining discriminative triplets of patches for fine-grained classification." [Online]. Available: https://arxiv.org/abs/1605.01130
[66] E. Gavves, B. Fernando, C. G. Snoek, A. W. Smeulders, and T. Tuytelaars, "Fine-grained categorization by alignments," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1713–1720.
[67] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.



JINGYUN LIANG received the B.S. degree from the National University of Defense Technology, Changsha, China, in 2016, where he is currently pursuing the M.S. degree in control science and engineering. His research interests include computer vision and deep learning.

JINLIN GUO (M'11) received the B.S. degree from Central South University, and the M.S. and Ph.D. degrees from the National University of Defense Technology, China, in 2008 and 2015, respectively. He is currently a Lecturer with the College of System Engineering, National University of Defense Technology. His research interests focus on computer vision, pattern recognition, and machine learning. He is a member of ACM.

XIN LIU received the Ph.D. degree in information and communication engineering from Xi'an Jiaotong University, Xi'an, China, in 2016. He is currently a Researcher with the Center for Machine Vision and Signal Analysis, University of Oulu, Finland. His research interests include human behavior analysis, image restoration, and object detection.

SONGYANG LAO received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology in 1990, 1993, and 1996 respectively, all in systems engineering. He is currently a Professor with the College of System Engineering, National University of Defense Technology. His research interests include multimedia information systems and human computer interaction.