J. Vis. Commun. Image R. 59 (2019) 563–573

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

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

Principal characteristic networks for few-shot learning ☆

Yan Zheng, Ronggui Wang, Juan Yang *, Lixia Xue, Min Hu

School of Computer and Information, Hefei University of Technology, 230009 Hefei, China

ARTICLE INFO

Article history:
Received 24 May 2018
Revised 19 September 2018
Accepted 3 February 2019
Available online 10 February 2019

Keywords:
Few-shot learning
Principal characteristic
Mixture loss function
Embedding network
Fine-tuning

ABSTRACT

Few-shot learning aims to build a classifier that recognizes unseen new classes given only a few samples of them. Previous studies like prototypical networks utilized the mean of embedded support vectors to represent the prototype that is the representation of class and yield satisfactory results. However, the importance of these different embedded support vectors is not studied yet, which are valuable factors that could be used to push the limit of the few-shot learning. We propose a principal characteristic network that exploits the principal characteristic to better express prototype, computed by distributing weights based on embedded vectors' different importance. The high-level abstract embedded vectors are extracted from our eResNet embedding network. In addition, we proposed a mixture loss function, which enlarges the inter-class distance in the embedding space for accurate classification. Extensive experimental results demonstrate that our network achieves state-of-the-art results on the Omniglot, miniImageNet and Cifar100 datasets.

1. Introduction

Recently, deep learning has achieved significant progress in various tasks with large datasets in image classification [32,31,17,44], object detection [3,35,36,41], machine translation [2,33,34] and so on. In some areas, classification, recognition and detection abilities have exceeded humanity. However, these achievements rely on the deep models that are trained with a huge number of labeled samples. When it comes to small training sets, the deep networks suffers from the overfitting problem. Facing this problem, few-shot learning [4,26,18] is flourishing in recent years. Few-shot learning indicates that a classifier needs to correctly recognize new classes that are not available in training, when only few labeled samples in these new classes are given. This task may be difficult for existing deep models, but it is quite easy for humans. Even children can outline the concept of "tiger" by giving few pictures of a tiger that has never been seen before. Then they can generalize this concept, and recognize tigers correctly in other pictures. The motivation for accomplishing this task is not only this, but also many applications, such as recognition and classification of the rare cases in medical imaging for auxiliary diagnosis, search and recognition of suspects from the massive surveillance video for assistant reconnaissance and so on. The major advantage is that it only needs few labeled samples to achieve reasonable results, rather than millions of labeled samples. Thus sample annotation cost is greatly alleviated. For this type of few-shot learning task, what first comes to our mind is transfer learning [5,37,38], which can train on large datasets in advance, then fine-tune on the target small dataset to get a better result on target categories. However, studies have shown that when target categories diverge from training categories, the performance of pre-trained networks is greatly decreased [6]. In this case, abstracting the concept of the class hierarchy is needed rather than sample hierarchy. In addition, since the target small dataset only has few or even one labeled sample per class, direct fine-tuning cannot learn the class concept of the target small dataset well.

Solutions for few-shot learning tasks include the following methods: data augmentation, meta-learning and metric learning. For data-starved classes, data augmentation can be used to alleviate overfitting. The corresponding solution is to augment data at the feature level, such as hallucinating features [27,28]. Their approaches have a certain accuracy improvement in few-shot classification, but cause of the extremely small data regime, the transformation mode is very limited and cannot solve the overfitting problem. Meta-learning methods [10,14,15,23,12] are widely favored for few-shot learning because they are built on the basis of tasks and learn high-level strategies between similar tasks. By learning good initial conditions [10,14], task-level update strategies [10] or constructing external memory storages [15,23] via RNNs to remember large amounts of information for comparison during testing, these approaches achieved good results. However,

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author at: School of Computer and Information, Hefei University of Technology, Hefei, Anhui 230009, China.
E-mail address: yangjuan6985@163.com (J. Yang).

https://doi.org/10.1016/j.jvcir.2019.02.006
1047-3203/

due to the use of RNNs, network architectures are complex and less efficient, while the metric learning methods are more simple and efficient. This kind of methods first learns the embedded vector of a sample from an embedding network, then directly computes nearest neighbor in the embedding space to accomplish prediction and classification [4,8,9,16,29,30]. By using episodes [8] training methods, improved embedding space [9,16] and learnable distance metrics [29,30], the few-shot classification performance is further improved.

One of representative achievements of metric learning methods is the matching networks proposed by Vinyals et al. [8], in which an attention mechanism is used to predict the categories of query images based on the embedding network that learns from support sets. It uses sampled mini-batches called episodes during training to simulate test tasks. The approach makes the training environment more similar to the test environment, which improves generalization performance during the testing phase. Another major contribution is the proposed miniImageNet dataset for few-shot learning tasks, which is also widely used as a benchmark. Snell et al. [9] further explore the relationship among class embedded vectors in the embedding space. There exists a representation called "prototype" for each category. The corresponding class embedded vectors cluster around the class prototype, and the prototype is the mean value of embedded support vectors. Based on this, the prototypical network is proposed. After the class prototypes are obtained, the classification problem becomes the nearest neighbor of an embedded query vector versus various class prototypes in the embedding space. This work achieves good performance.

Our paper is based on the model training method of matching networks [8], and inspired by the idea of prototype [9] to propose principal characteristic networks. Aiming at the problem that it [9] simply uses the mean of embeddings to express class prototype, and cannot distinguish the different contributions of class embedded support vectors to the class prototype well, we propose the concept of principal characteristic. Exploiting different embedded support vectors on the prototype based on the target which is obvious or not in the image, different weights are distributed, and then weighted summation is used to obtain the principal characteristic vector to better express the prototype, which achieves better results. Specifically, we distribute weights based on the sum of absolute difference (SAD) of each embedded support vector versus the rest in a class. In addition, we put forward the mixture loss function in consideration of the situation that existing similar classes in the embedding space cause misclassification. Basic loss combined with relative error loss function is used to increase inter-class distances among embeddings in the embedding space, which reduces misclassification of similar classes while maintaining correct classification of normal classes. Moreover, we propose the eResNet structure based on residual networks [17] for the embedding network. Compared with common CNNs used by prior approaches, it can effectively increase the network depth to improve the general feature extraction degree without degradation. Compared to the smallest ResNet-18, we reduced the 88.86% of the parameters and achieved better results. In experiments after fine-tuning with support sets during testing, we achieve currently known state-of-the-art results on the Omniglot, miniImageNet and Cifar100 datasets.

The rest structure of this paper is organized as follows: In Section 2 we describe related work in the few-shot learning area. In Section 3 we define and explain the methods of our paper. First, we introduce the general architecture of our principal characteristic networks how to handle few-shot classification task, and then introduce each optimization method in detail. In Section 4 we show our experimental settings and experiment performances, which demonstrate the excellent performance of our paper on various datasets. In Section 5 we present our conclusions and prospect the next work.

2. Related work

After Yip et al. [1] started exploration of one or few-shot learning area using machine learning, the related works are roughly divided into the following types of methods. Beginning with data spaces, the related method is generative models and data augmentation. The generative model based on stroke [25] or part [24] has good performance in specific fields such as hand-written characters [25]. The generative model is a joint probability distribution of sub-parts, parts, and relationships among them. It is computed by Bayesian program learning [25]. The probability distribution can be used to generate new types and calculate the probability of generating a type. Such generative models can also have good performance on datasets with little difference in intra-class examples, such as Caltech 101 [26]. Dixit et al. [27] used an attribute labeled dataset to generate additional sample features by changing attribute values. This augmentation is on feature vectors instead of on images in common sense. Hariharan et al. [28] proposed an approach for generating additional sample features by "hallucinating" new sample feature vectors of new classes from the transformation patterns among sample feature vectors of training classes. The above approaches have attained certain results in few-shot learning tasks. However, according to complex datasets such as miniImageNet, their different parts of targets are not easily identified, and the interrelationships among the parts are too complex and diverse. These problems result in a huge combination space of the generative model, and lead to poor performance in face of such datasets. In addition, although feature augmentation can alleviate overfitting phenomenon, the few-shot data space leads to limited transformation patterns, which makes feature augmentation unable to solve the overfitting problem. As a result, we do not optimize our architecture using data augmentation, but our experiments still prove to have excellent results.

Except the above approaches of learning directly on datasets, meta-learning [14] performs better on few-shot learning tasks. Unlike traditional learning models that directly learn the inductive bias of data points, meta-learners learn a high-level strategy of tasks based on data points to guide learners, which can be significantly better than traditional learning methods. Representative approaches include: memory-augmented neural networks proposed by Santoro et al. [15] based on the idea of Neural Turing Machines [7] (NTMs). NTMs [7] can learn strategies for storing information in memory and use the information for prediction. As a result, it can accurately predict those data which have only appeared once. Ravi and Larochelle [10] used LSTM [22] to simulate gradient descent for learning parameter updating algorithm based on meta-learning methods, where LSTM [22] is a meta-learner who uses its cell state to update the parameters of the classifier network (learner), and eventually learns how to initialize and update the parameters of the classifier network in new classification tasks. Finn et al. [14] proposed model-agnostic adaptive meta-learning networks, which make it possible to quickly perform gradient descent without overfitting on few samples, and make the model easily fine-tune. Their core idea is that the learning model needs to find some parameters that are sensitive to various tasks, so that small parameter changes will produce a larger loss when the gradient direction changes. Then the latent embedding [42,43] was proposed to learn a low-dimension latent representation of model parameters to perform well in the limited data space. The prior approaches have achieved good results in few-shot learning tasks. However, due to the use of RNN-related memory structure, the architecture is relatively complex, or the

use of external memory to memorize all historical information requires more storage. Our principal characteristic networks use a simple and efficient feed-forward CNNs combined with no-parameter distance metric to complete fast learning.

Another method that focuses on feature matching in the feature space is metric learning [11], and it has a long history. The basic solving strategy is to learn an embedding space from inputs, and then use a distance measurement method to classify query vectors in the embedding space. The mainstream approaches are: the siamese neural networks proposed by Koch et al. [4], which use weight tying, two symmetrical networks to learn the embedding space of support and query sets, and use regular cross-entropy loss function via the L1 distance to learn. The matching networks proposed by Vinyals et al. [8] also use distance metrics to learn in the embedding space. Its main innovation is in the other two aspects. In the aspect of modeling, the matching networks [8] use memory and attention to attain fast learning. In the aspect of training, they proposed an episode training method that makes training and testing phases maintain consistency, which effectively improves the classification performance during testing. Snell et al. [9] proposed the prototypical networks. They believe that each class has a prototype surrounded by its embedded support vectors, and this prototype is the mean of the embedded vectors. When classifying, the distances from the embedded query vector to various class prototypes are measured, instead of embedded support vectors, and the effect is significant. Fort [16] proposed the Gaussian prototypical networks built on prototypical networks. The Gaussian covariance matrix of samples is generated together with the embedding space, and then the orientation and class dependent distance metric is constructed in the embedding space. The improving point is similar to ours. However, the Gaussian covariance matrix costs a large amount of computation and it is inefficient. Otherwise, they only used the simple dataset Omniglot to verify effect in the experiments, which is not convincing. As a contrast, our SAD is more efficient and we conduct extensive experiments on Omniglot, miniImageNet and Cifar100. In addition, we make further improvements in the loss function, which furthers the distance among various classes in the embedding space. In these metric learning methods, in addition to the above fixed distance metric methods: L1 distance [4], cosine distance [8], Euclidean distance [9,16], etc., there is another kind of method that uses a learnable distance metric. For example, Mehrotra et al. [29] proposed skip residual pairwise networks. Its structure combines with the residual network and the siamese network structure. It accepts the inputs of a pair of pictures and outputs the distance measure directly for classification tasks. The relation network proposed by Sung et al. [30] uses the concatenation of embedded support and query feature maps to input a relation module and output the relation score to get the predicted query image category. Although these approaches work well, they are more dependent on network design. Beyond few-shot learning, Cheng et al. [39,40] proposed that a metric learning regularization term was added to the classification loss function to solve the intra-class diversity and inter-class similarity problems, and obtained good results.

In consideration that data augmentation cannot solve the overfitting problem, and meta-learning methods are high in complexity and low in execution efficiency, we mainly focus on simple and efficient metric learning methods. Based on matching networks training method [8] and inspired by the prototype [9] idea we propose the principal characteristic networks. By using episodes training method, the images are mapped into the embedding space, and then weighted clusters of embedded support vectors are used as principal characteristics to classify query images. The novelty of our method is to give different weight to embedded support vectors based on their different contribution to the prototype, rather than equal treatment as in the prototypical networks [9]. We use SAD values of each support vector versus the rest as the evaluation criterion; softmax is then used to compute the weight of each support vector contributing to the principal characteristic. Each support vector in a class is then multiplied by the corresponding weight and the summation of them is computed to obtain the class principal characteristic. This method makes certain samples that deviate from the principal characteristic have smaller weights, and have lower impact on the principal characteristic, so that the principal characteristic expresses the prototype better than the mean and improves classification accuracy. Next we put forward the mixture loss function, which adds relative error loss to the basic loss to increase inter-class distances in the embedding space, and to reduce misclassification of similar classes. We also make further improvements to the embedding network. Based on the residual network [17] we propose the eResNet embedding network, by adding depth to further abstract general representative features of complex datasets such as miniImageNet, and using the shortcut connection structure to avoid degradation after increasing the depth of the network.

3. Methods

The basic procedure of processing few-shot classification tasks in our paper is: we first input support and query images of an episode into our eResNet embedding network, and obtain the corresponding embedded vectors. In the embedding space, we use the proposed principal characteristic method to obtain the principal characteristic per class. Then we calculate the cosine distances from embedded query vectors to each class principal characteristic, and then use the proposed mixture loss function to compute loss based on the distances, finally apply back-propagation to obtain gradients to update network, as shown in Fig. 1. Episodes training method is used to train our network. The following details our various optimization methods.

3.1. eResNet embedding network

In the existed metric learning approaches [8,9], most of embedding networks are simple CNNs, using stacked four modules, each module is like this:

$$3 \times 3 \ \text{conv} \rightarrow \text{batch normalization} \rightarrow \text{ReLU} \rightarrow 2 \times 2 \ \text{max pool}$$

the numbers of convolutional kernels per module are [64, 64, 64, 64].

In consideration that this simple CNN structure cannot learn the nonlinearity of complex datasets such as miniImageNet well, we improved the embedding network. Inspired by the structure of the residual network proposed by He et al. [17], we propose the eResNet embedding network. The residual block used therein contains three 3×3 convolutional layers, then uses a shortcut connection structure followed by ReLU activation and a 2×2 maximum pooling layer. The structure of the residual block and the entire network is presented in Fig. 2.

The main advantage of residual networks [17] is shortcut connections, which add an identity mapping to the standard structure:

$$y = \mathcal{R}(x) + x \tag{1}$$

where x denotes the input vector of the network module, y denotes the output vector, and $\mathcal{R}(x) = y - x$ is the residual mapping to be learned, whereas in the basic network the output is obtained by direct mapping of the input via the network module: $y = \mathcal{F}(x)$. Learning the residual mapping $\mathcal{R}(x)$ is much easier than learning $\mathcal{F}(x)$ directly, and this shortcut connections can effectively solve the degradation problem as the network depth increases. Theoretical analysis shows that if the network has reached an optimal level

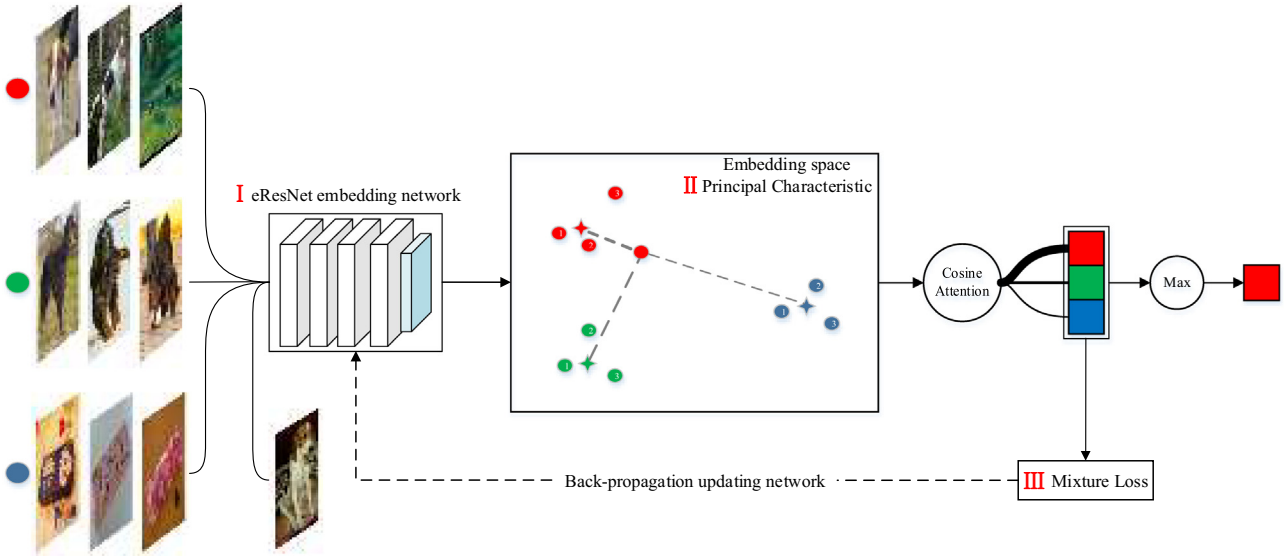

Fig. 1. Overall architecture diagram of the principal characteristic networks. The circle in the embedding space represents the embedded vector of each image, and the classes are represented by red, green and blue respectively. The class embedded support vectors are used to calculate the class principal characteristic. The star in the figure represents each principal characteristic. The dashed line on the bottom of the diagram represents back-propagation course. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

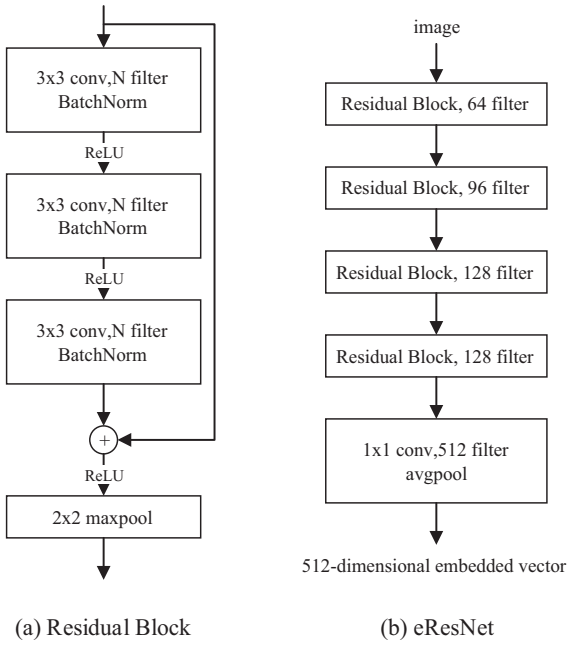

Fig. 2. The left figure shows the structure of the residual block used in our embedded network. The figure on the right is our entire eResNet embedding network using the residual block. An image is gone through four residual blocks, and then a 1×1 convolutional layer is used to extend the depth to 512, finally global average pooling is used to output a 512-dimension embedded vector.

and is deepen continually, the residual mapping $\mathcal{R}(x)$ will be zeroed, which is equivalent to the module performing an identity mapping: $y = x$, so that deepening has no effect on the network and it maintains the optimal status.

Specifically, the input and output dimensions of the module must be equal. When the input and output are of the same dimensions, the element-wise addition operation can be performed directly:

$$y = \mathcal{R}(x, w_i) + x \tag{2}$$

when the dimensions are different, the input vector needs to be performed with a linear projection W by the shortcut connections to match the dimensions, using 1×1 convolution operation to achieve:

$$y = \mathcal{R}(x, w_i) + Wx \tag{3}$$

where w_i denotes i-th convolutional layer parameters in a module.

In contrast to ResNet structure, our eResNet model is more suitable for few-shot learning task. The goal of our design network is to use shortcut connection to deepen network and extract features better, but minimize the number of parameters to prevent overfitting. First, our network uses max-pooling to halve feature map size instead of the convolutional layer with a larger stride in ResNet. Benefited from max-pooling layer which does not need any parameters, we reduce about 460k parameters totally. Second, our network ends with a 1×1 convolution layer to extend depth rather than a fully-connected layer. We believe that fully convolutional network can extract features better. Third, eResNet has 13 parameter layers, and chooses an appropriate number of kernels per block. All these methods achieve the goal of using as few parameters as possible while ensuring depth and effectiveness. Compared to the smallest ResNet-18 (11.13 million parameters), we (1.24 million parameters) reduced the 88.86% of the parameters and achieved better results, see Tables 1 and 6.

3.2. Principal characteristic

In a few-shot learning task, a small support set with N labeled samples $S = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ is given, where $x_i \in \mathbb{R}^D$ is the D-dimensional input vector of a sample, $y_i \in \{1, \cdots, K\}$ is the corresponding label. S_k denotes the support sample set labeled with class k, and N_S denotes the number of support samples in a class. First each input vector of a sample goes through the embedding network to learn an embedding function mapping $f_\theta : \mathbb{R}^D \to \mathbb{R}^M$, where θ is the learnable parameters of the embedding network, and the M-dimensional space is the embedded vector space after the mapping transformation. In the original prototypical network [9] they take a class's prototype to be the mean of its embedded support vectors:

Table 1
Architecture of embedding networks of eResNet and comparison. The representation of some symbols: "C" denotes a convolutional layer, "MP" denotes maximum pooling layer, "GAP" denotes global average pooling layer. Because of different size of the input images, we get rid of the first two down-sample layers of original ResNet-18. The bolds are the indicators of ours.

Model name	Simple CNN	ResNet-18	eResNet
Architecture	$\begin{bmatrix} C:3\times3,64 \\ MP:2\times2 \end{bmatrix}$	$\begin{bmatrix} C:3\times3,64 \\ C:3\times3,64 \end{bmatrix}\times2$	$\begin{bmatrix} C:3\times3,64 \\ C:3\times3,64 \\ C:3\times3,64 \\ MP:2\times2 \end{bmatrix}$
	$\begin{bmatrix} C:3\times3,64 \\ MP:2\times2 \end{bmatrix}$	$\begin{bmatrix} C:3\times3,128 \\ C:3\times3,128 \end{bmatrix}\times2$	$\begin{bmatrix} C:3\times3,96 \\ C:3\times3,96 \\ C:3\times3,96 \\ MP:2\times2 \end{bmatrix}$
	$\begin{bmatrix} C:3\times3,64 \\ MP:2\times2 \end{bmatrix}$	$\begin{bmatrix} C:3\times3,256 \\ C:3\times3,256 \end{bmatrix}\times2$	$\begin{bmatrix} C:3\times3,128 \\ C:3\times3,128 \\ C:3\times3,128 \\ MP:2\times2 \end{bmatrix}$
	$\begin{bmatrix} C:3\times3,64 \\ MP:2\times2 \end{bmatrix}$	$\begin{bmatrix} C:3\times3,512 \\ C:3\times3,512 \end{bmatrix}\times2$	$\begin{bmatrix} C:3\times3,128 \\ C:3\times3,128 \\ C:3\times3,128 \\ MP:2\times2 \end{bmatrix}$
		$[GAP]$	$\begin{bmatrix} C:1\times1,512 \\ GAP \end{bmatrix}$
Parameter layers	4	16	**13**
Parameters	0.11×10^6	11.13×10^6	$\mathbf{1.24\times10^6}$

$$c_{k\_src}=\frac{1}{N_S}\sum_{(x_i,y_i)\in S_k} f_\theta(x_i) \tag{4}$$

The method proposed in prototypical networks [9] using the mean as a prototype is based on the bregman divergence. It proves [13] that for a set of points in a particular space, when these points meet any probability distribution, the mean point of these points must be the point in this space that has the smallest average distance to these points. We believe that in the case of limited data space, the distribution of the embedding space is not sufficient to meet any probability distribution, especially in a testing environment. Therefore, we believe that the position of the prototype is not simply the mean of class embedded support vectors, and the embedded vectors in a class have the differentiation of closing to the prototype. Specifically, there exists some embedded vectors whose target in the sample is not obvious. For example, the target foreground is small; the background is large; the target is partially obstructed; the sample image contains only a part of the target and so on. Because the target is unobvious, the network cannot learn the corresponding target feature well in such a complicated situation, which causes the embedded vector to deviate from the class prototype. Therefore, the contribution of these embedded vectors to the prototype should not be consistent with the embedded vectors of more obvious target. Based on this, we propose principal characteristic to express prototype better. The sum of absolute difference (SAD) is used to measure the weight of contribution to the principal characteristic, which makes the embedded vector of unobvious target have a smaller weight, and the embedded vector of obvious target have a larger weight. Then the weighted sum of embedded vectors in the class is calculated as the class principal characteristic vector, as presented in Fig. 3. The SAD value of each embedded vector in its class is computed as:

$$D_i=\frac{1}{M}\sum_{(x_j,y_j)\in S_k}||f_\theta(x_i)-f_\theta(x_j)||_1 \tag{5}$$

Based on Eq. (5) the principal characteristic is calculated as:

$$p_k=\sum_{(x_i,y_i)\in S_k}\frac{\exp(-D_i)}{\sum_{i=1}^{N_S}\exp(-D_i)}f_\theta(x_i) \tag{6}$$

For the query image $\hat{x}$, the attention kernel computed on the principal characteristic is:

$$a(\hat{x},p_k)=\frac{\exp(c(f_\theta(\hat{x}),p_k))}{\sum_{k'}\exp\left(c\left(f_\theta(\hat{x}),p_{k'}\right)\right)} \tag{7}$$

where function c is the cosine distance function.

Then the prediction label of the query sample $\hat{x}$ is calculated as:

$$\hat{y}=\sum_{k=1}^{h}a(\hat{x},p_k)y_k \tag{8}$$

We know that an embedded vector that has a smaller SAD is closer to the principal characteristic. In other words, the weight of this embedded vector should be larger when calculating the principal characteristic. This is a negative correlation, so we use negative SAD value to input softmax to get the weight, and then compute the weighted sum to get our principal characteristic.

3.3. Mixture loss function

As for a query image $\hat{x}$, its class is k, the basic loss function is:

$$\mathrm{J}(\theta)=\log P_\theta(\mathrm{y}=\mathrm{k}|\hat{x},\mathrm{P}) \tag{9}$$

where P represents the class principal characteristic set of the entire embedded support vectors.

We find that there are some similar classes in classification tasks. For example, the walker hound class and the Gordon setter class in the miniImageNet dataset, both of these two classes belong to the family of Canidae. Same as the hamster class and rat class in the Cifar100 dataset, they belong to the same species of rodents and are very similar to each other. In addition, because of few samples, it is easier to cause misclassification of similar classes. Therefore, we believe that increasing distances among various classes can effectively reduce the probability of misclassification of similar classes. Specifically, the distance between the embedded query vector and the correct class principal characteristic is basically unchanged before and after optimization. However, because the distances among the classes is getting larger, the distance to the similar class principal characteristic is also larger than that before the optimization. As a result, the distances from an embedded query vector to several similar class principal characteristics which are similar before optimization become relatively close to the correct class principal characteristic, thus avoiding misclassification. Based on this, we propose the mixture loss function, which adds the relative error loss term to the basic loss. For a few-shot classification task with n classes of support set in an episode, there is:

$$\mathrm{J}(\theta)=-\log P_\theta(\mathrm{y}=\mathrm{k}|\hat{x},\mathrm{P})+\frac{1}{n-1}\sum_{i=1}^{n}(1-z_i)a(\hat{x},p_i)+z_k(1-a(\hat{x},p_k)) \tag{10}$$

where $z_i=\begin{cases}0, y_i\neq k\\ 1, y_i=k\end{cases}$, that is to say, if the query image $\hat{x}$ belongs to the class k: $y_{\hat{x}}=k$, when the label y_i of class principal characteristic p_i is also k, we have $z_i=1$. Thus the second item becomes 0, and the third item becomes $(1-a(\hat{x},p_k))$. To make the third term smaller (minimize the loss), it is required that $a(\hat{x},p_k)$ becomes larger. Therefore, $\hat{x}$ is required to be closer to the k-class principal characteristic p_k. When the label y_i of class principal characteristic p_i is not k, we have $z_i=0$. Thus the second item becomes $\frac{1}{n-1}\sum_{i=1}^{n}a(\hat{x},p_i)$, and the third item becomes 0. To make the second term smaller, it is required that $a(\hat{x},p_i)$ becomes smaller. Therefore, it is required to keep $\hat{x}$ away from the non-k-class principal characteristics p_i. Here $a(\hat{x},p_i)$ is the attention kernel of embedded query vector $\hat{x}$ and the i-class principal characteristic p_i. Based on the above, it can be seen that our mixture loss function can close the distance between the query image $\hat{x}$ and principal characteristic of the same

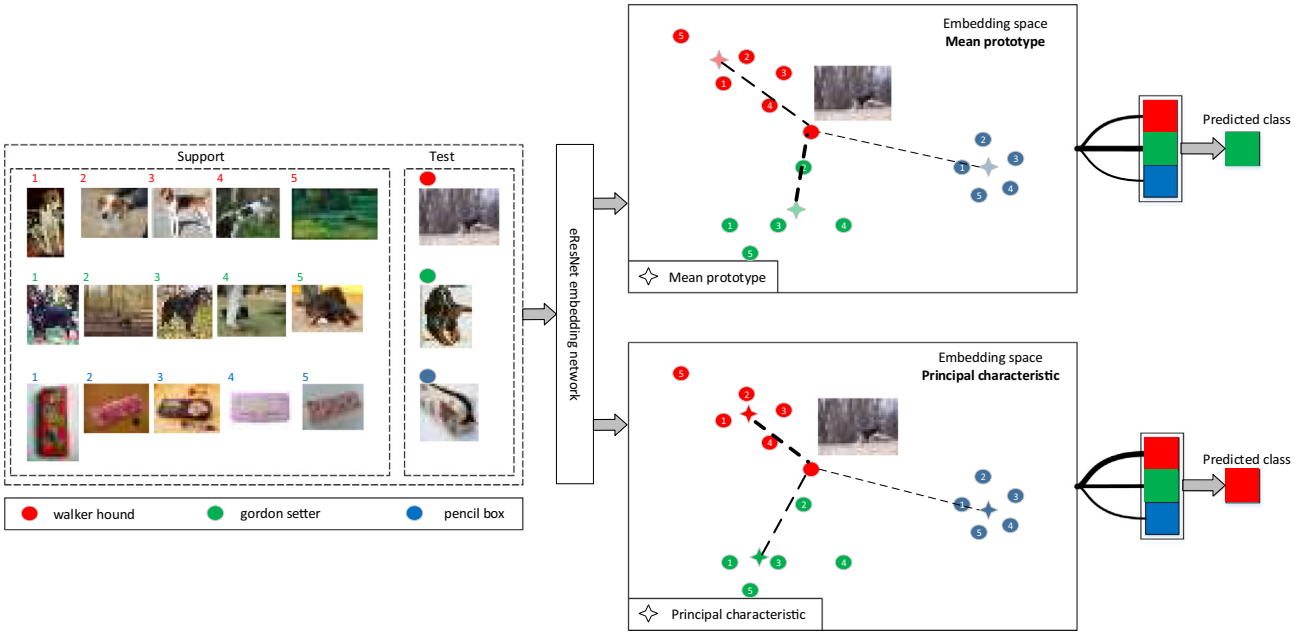

Fig. 3. A diagram of the principal characteristic method. The eResNet embedding network first maps samples to the embedding space, and the circle in the embedding space denotes the embedded vector of each sample. The thickness of dashed lines indicates distances. There are three classes in this figure: walker hound, Gordon setter, pencil box (from miniImageNet dataset). Each class has five support samples, one query image. It shows that when a support vector of unobvious target deviates from the class prototype, the principal characteristic computed by our approach is more representative than mean prototype, and avoids misclassification.

class, and further distances to the principal characteristics of different classes. By principal characteristics, then tracing back to embedded support vectors, distances among various classes are resulted in furthering in whole embedding space, as shown in Fig. 4.

The mixture loss function consists of a basic loss term (cross entropy loss term) and a relative error loss term. Benefit from designing a limited range per term, the relative importance of each term to the total loss does not require adding additional weighting factors to learn. The principle of allocating share per term: the relative error loss is used to optimize the misclassification of similar categories, which is an auxiliary task, so it should be relatively small. The cross entropy loss term is the main task, so it should be relatively large. Here we achieve the purpose of distinguishing importance by limiting the range of each term.

First, the range of the cross-entropy loss term is as follows:

$$-\log P_\theta(\mathrm{y}=\mathrm{k}|\hat{x},\mathrm{P}) \in (0,+\infty) \tag{11}$$

From Eq. (7) we know:

$$a(\hat{x},p_i) \in (0,1) \tag{12}$$

Therefore, the range of the relative error loss term is as follows:

$$\frac{1}{n-1}\sum_{i=1}^{n}(1-z_i)a(\hat{x},p_i) \in (0,1) \tag{13}$$

$$z_k(1-a(\hat{x},p_k)) \in (0,1) \tag{14}$$

From Eqs. (13) and (14) we can see that the relative error loss term has a limited contribution to the total loss, and cannot cause a large deviation to the total loss. The total loss is mainly contributed by the cross entropy loss. Thereby, we achieve the purpose of distinguishing importance.

We use gradient descent to minimize the mixture loss function during training as usual. Specifically, a set of samples is chosen randomly from the training set without replacement to construct an episode according to the requirements of different few-shot learning tasks, and then the episode sample set is divided into a support set and a query set. The query set is classified based on the support set, and the prediction is compared with the ground

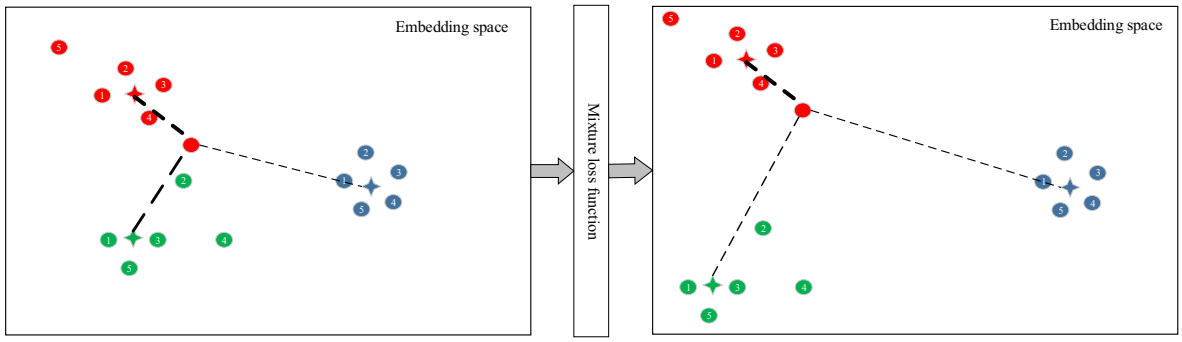

Fig. 4. A diagram of the effect of the mixture loss function method. The left part is the original embedding space, and the right part is the embedding space adjusted by our mixture loss function. The thickness of dashed lines indicates the distance. It can be seen that the distance between walker hound and Gordon setter embedded support vectors which close to each other is further enlarged, which makes it easier to get correct classification result of the query image walker hound (the red circle without tag in this diagram). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Few-shot classification accuracies on Omniglot dataset. The bolds are the indicators of ours and the best, same as below.

Model	FT	5-way Acc.		20-way Acc.	
		1-shot	5-shot	1-shot	5-shot
MATCHING NETS [8]	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS [8]	Y	97.9%	98.7%	93.5%	98.7%
PROTOTYPYCAL NETS [9]	N	98.8%	99.7%	96.0%	98.9%
MAML [14]	Y	98.7 ± 0.4%	99.9 ± 0.1%	95.8 ± 0.3%	98.9 ± 0.2%
RELATION NET [30]	N	**99.6 ± 0.2%**	99.8 ± 0.1%	97.6 ± 0.2%	99.1 ± 0.1%
PRINCIPAL CHARACTERISTIC NETS (OURS)	N	98.6%	99.6%	96.3%	98.9%
PRINCIPAL CHARACTERISTIC NETS (OURS)	Y	**99.2%**	**99.8%**	**98.4%**	**99.5%**

truth to form a loss, finally back-propagation is used to compute gradients to update network parameters.

4. Experiments

This section describes our experimental results in detail and compares against various baselines. The tasks are based on N-way, k-shot classification tasks. Specifically, it refers to: k labeled samples per class provided, total N classes which are new classes that have not seen in the training set. Then, based on this sample set, the network predicts whether a query image belongs to which class in these N classes. We use the same number of classes during training as testing. For example, we use 5 classes, 5 labeled samples per class to form the support set of training episodes on 5-way, 5-shot classification task. In addition, we use the cosine distance function to compute the distances from the embedded query vector to class principal characteristics.

Three datasets are used here for extensive experimentation. The first two datasets are Omniglot [18] and miniImageNet [8,10] version of the ImageNet Large Scale Visual Recognition Competition 2012 [20] (ILSVRC-2012). Besides, we find that the Cifar100 [19] dataset is also suitable for this few-shot leaning tasks. We also perform experiments on this dataset, which further validates the capabilities of our principal characteristic networks.

All of our experiments use Adam [21] optimization algorithm with initial learning rate 0.001; 1000 episodes construct an epoch; the learning rate is reduced by half for every 10 epochs, and we totally train 100 epochs. We use L2 regularization in the optimization algorithm Adam [21] to reduce overfitting. The experiment is implemented on the NVIDIA GeForce GTX TITANX 12 GB graphics card, Intel Xeon E3-1231V3 processor, 32G memory, and Ubuntu 15.04 operating system through the PyTorch deep learning framework. The following is divided into three subsections that describe the performance of our principal characteristic networks on each dataset.

4.1. Omniglot few-shot classification

The Omniglot [18] dataset consists of 50 alphabets, totally 1623 handwritten characters. Each character class has 20 samples, which are handwritten by different people. Because of the large number of classes in this dataset and the small number of samples per class, it is suitable for few-shot learning tasks. We use the same dataset settings as the matching network [8]: resizing the sample size to 28×28 and using a data augmentation method that rotates multiples of 90 degrees. We use 1200 characters for training, 123 character classes for evaluation, and remaining 300 classes for testing.

Experiments show that our results are much better than matching networks [8] and prototypical networks [9]. We achieve currently known state-of-the-art performance on most of the tasks, except 5-way 1-shot where our architecture is 0.4% lower in accuracy than the relation network [30]. It uses a more complex network design for a learnable metric method, while we use a fixed metric, although the result is lower on this task, our architecture is simpler and more efficient by contrast. The specific comparison data are shown in Table 2. Since few-shot classification tasks have little space for improvement on this Omniglot dataset, further detailed item-by-item comparisons of our approach are given on the miniImageNet and Cifar100 datasets.

4.2. miniImageNet few-shot classification

The miniImageNet dataset originally proposed in the matching networks [8], it selects 100 classes with 600 samples per class (a total of 60,000 samples) from the ILSVRC-2012 [20] dataset, and resizes to 84×84 size. Since the exact splits used in matching networks [8] were not released, we follow the split introduced by the Ravi and Larochelle [10] experiments, with 64 classes for training, 16 classes for validation, and remaining 20 classes for testing.

This miniImageNet dataset is basically a benchmark dataset for few-shot classification tasks. Because it uses complex data of ImageNet (a benchmark dataset in the field of image classification), it has high complexity and can test the capabilities of model approaches well. And the number of samples per class is very suitable for few-shot classification tasks. So once it is proposed, it is very popular until now. We conduct detailed comparative experiments on this dataset to verify the capabilities of our various optimization methods.

It should be noted that the number of query images per class used in our experimental episode is 10, while 15 is adopted by most existing few-shot learning work. The reason is that using our proposed eResNet embedding network consumes more memory than the common simple CNNs structure, which leads to insufficient memory for a single graphics card. In order to keep other important hyperparameters unchanged, we choose to reduce the number of query images. In the vanilla matching networks [8], we experimentally proved that the accuracy of 10 query images per class is about 1% lower than 15 samples. Our experimental results did not add this 1% accuracy, which is the actual data by using 10 query images per class. In addition, the accuracy results are shown with 95% confidence interval and averaged of 5 times experiments to ensure the accuracy of the experimental results.

From the comparison experiment, we know that, on the 5-way, 5-shot task, our weighted principal characteristic method is more effective than the original matching networks [8]. It has an improvement of about 6 percentage point; the mixture loss has about 1 percentage point; the improvement of eResNet embedding network is also about 1 percentage point, and finally all of these optimization methods bring 11 percentage point improvement after fine-tuning on the support set during testing. Analyzing the reasons, we can see that the principal characteristic idea is based on two points: one is the prototype idea, and the second is to distinguish the contribution of support vectors whether the target is obvious to the prototype, which expresses the prototype better. At first, because the prototype idea based on the thought that the

abstract representation vector of class support vector set is used to classify query vector instead of the original support vectors. This simple inductive bias idea is very helpful for avoiding the misclassification caused by the distance based on the support vectors in the case of limited labeled samples. Then this kind of idea needs a good implementation way to explore its ability. We use weighted principal characteristic to express the prototype, fully considering the different contributions of different support vectors to the prototype, and get a prototype expression that is better than mean prototype, which makes the result improve more obvious. For the mixture loss function, it is more prone to reduce the misclassification caused by similar classes in the embedding space, while this dataset exists similar classes but is not common, so this loss optimization method does not greatly enhance the performance. For the improved embedding network eResNet, it is used to increase the degree of abstraction of embedded vectors in complex datasets, while the original common four-layer CNN structure has a simple structure and shallower layers, which cannot extract embedded vectors of high class representation. Our eResNet network that is large, deep, and has shortcut connections structure (which can make the network deeper without degradation) proves its ability, but the fact of few data limits the generalization of deeper networks, so the improvement is restricted.

However, our approach showed a greater advantage when fine-tuning on the support set during testing. In the original matching networks [8], fine-tuning had about 3% improvement, while our approach improved nearly 11% after fine-tuning. At first, in the definition of tasks, fine-tuning means that during an iteration (an episode) of testing, only the support set in the episode is used to compute loss and then backpropagation update the network parameters, and the updated network is then used to accept the support set and query set inputs in episode for predicting. After the predicting, the network is returned to the state of last training, and then the next iteration test is performed. That is to say, parameters can be updated only once in the corresponding episode during the entire testing phase.

The reason is analyzed to see that, firstly our eResNet network shows its advantage. In fine-tuning, we use the new class support set to update the network parameters only once, instead of iterating multiple times. Because the network has been basically formed after training thousands of episodes, the network parameters will not be greatly changed. As a result, it is not easy to over-fit. At this time, the problem that should be considered is whether the network can effectively learn the new class information while be updated only once. We simply compare the gradient of a normal network and a network with a shortcut connection structure in the back propagation. In the normal network:

$$z_k = w_k a_{k-1} + b_k \tag{15}$$

$$a_k = g(z_k) = g(w_k a_{k-1} + b_k) \tag{16}$$

Here w_k, b_k is the weight parameter and bias term of layer k; a_k is the output of layer k; and the function g represents the corresponding activation function. At this time, the back propagation gradient formula is:

$$\frac{\partial J}{\partial a_{k-1}} = \frac{\partial J}{\partial a_k}\frac{\partial a_k}{\partial a_{k-1}} = (w_k)^T \frac{\partial J}{\partial a_k} g'(a_{k-1}) \tag{17}$$

When it comes to shortcut connection:

$$z_k = w_k a_{k-1} + b_k \tag{18}$$

$$\begin{aligned} a_k &= g(z_k + a_{k-1}) = g(w_k a_{k-1} + b_k + a_{k-1}) \\ &= g((w_k + 1)a_{k-1} + b_k) \end{aligned} \tag{19}$$

Then the back propagation gradient formula is:

$$\frac{\partial J}{\partial a_{k-1}} = \frac{\partial J}{\partial a_k}\frac{\partial a_k}{\partial a_{k-1}} = (w_k + 1)^T \frac{\partial J}{\partial a_k} g'(a_{k-1}) \tag{20}$$

Comparing Eqs. (17) and (20), it can be seen that the skip connection structure makes the gradient larger; so it is easier to make the parameters adjust to the new sample. In a normal network, because the gradient needs transfer through layers one by one, the gradient becomes smaller and smaller from back to front. It results in ineffective update.

Secondly, when only one-time update can be performed during testing, the model obtains a better principal characteristic expression than a mean expression for the corresponding support set, and furthers distances among classes in the embedding space is particularly important. At this time, updating network only depends on the support set in an episode before testing, compared to 1000 episodes during training. Therefore, the new classes concepts are severely influenced by this one support set. If there are certain support samples which targets are unobvious, our principal characteristic is needed to reduce sensitivity to them and close to the real prototype. If the mean prototype is used at this time, misclassification is more likely to occur because of the deviation from the real prototype. If the network can be updated based on multiple episodes, since the sample count per class increases, and the sample vector space increases, the centroid (mean prototype) of them gains certain robustness, and the sensitivity to the samples of

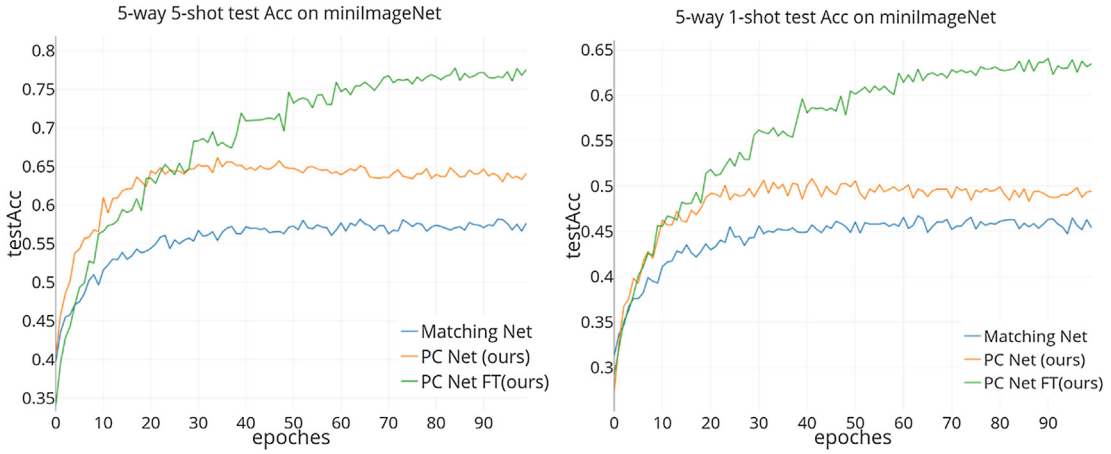

Fig. 5. Testing accuracy on the miniImageNet dataset of 5-way 5-shot, 1-shot few-shot classification task.

Table 3
Few-shot classification accuracies on miniImageNet dataset.

Model	FT	5-way Acc.	
		1-shot	5-shot
BASELINE-NEAREST-NEIGHBOR [10]	N	41.08 ± 0.70%	51.04 ± 0.65%
MATCHING NETS [8]	N	41.2%	56.2%
MATCHING NETS [8]	Y	42.4%	58.0%
MATCHING NETS FCE [8]	N	44.2%	57.0%
MATCHING NETS FCE [8]	Y	46.6%	60.0%
META-LEARNER LSTM [10]	N	43.44 ± 0.77%	60.60 ± 0.71%
MAML [14]	T	48.70 ± 1.84%	63.11 ± 0.92%
PROTOTYPYCAL NETS [9] (cosine)	N	42.48 ± 0.74%	51.23 ± 0.63%
PROTOTYPYCAL NETS [9]	N	49.42 ± 0.78%	68.20 ± 0.66%
RALATION NET [30]	N	57.02 ± 0.92%	71.07 ± 0.69%
PRINCIPAL CHARACTERISTIC NETS (PC)	N	47.42 ± 0.68%	63.67 ± 0.74%
PRINCIPAL CHARACTERISTIC NETS (PC + ML)	N	48.18 ± 0.70%	64.35 ± 0.69%
PRINCIPAL CHARACTERISTIC NETS (PC + ML + eResNet)	N	50.08 ± 0.73%	65.87 ± 0.75%
PRINCIPAL CHARACTERISTIC NETS (PC + ML + eResNet)	Y	**63.29 ± 0.76%**	**77.08 ± 0.68%**

Table 4
Effect of principal characteristic on miniImageNet dataset.

Model	5-way Acc.	
	1-shot	5-shot
MATCHING NETS [8]	41.2%	56.2%
PRINCIPAL CHARACTERISTIC NETS (PC)	**47.42 ± 0.68%**	**63.67 ± 0.74%**

Table 5
Effect of mixture loss function on miniImageNet dataset.

Model	5-way Acc.	
	1-shot	5-shot
PRINCIPAL CHARACTERISTIC NETS (PC)	47.42 ± 0.68%	63.67 ± 0.74%
PRINCIPAL CHARACTERISTIC NETS (PC + ML)	**48.18 ± 0.70%**	**64.35 ± 0.69%**

unobvious targets is reduced, which results in more accessible to real prototype. At this point, the advantage of principal characteristic is not obvious. Therefore, when fine-tuning based on only one support set in an episode during testing, principal characteristic has obvious advantages. The analysis of furthering distance is similar, so far, we achieved currently known state-of-the-art results: 5-way, 1-shot task 64.05%, and 5-shot task 77.76% classification accuracy, as shown in Fig. 5 and Table 3.

As can be seen from Table 6, ResNet-18 is less effective than the simple 4-layer CNNs when applied directly to few-shot learning task. Since the number of parameters is very large, it is 100 times bigger than the simple CNNs. The number of training data is only 38,400, and we need to predict the new class, which results in a large degree of overfitting. There is a higher precision than the simple CNNs on the training set, but the test is lower than the simple CNNs. Moreover, the excessive number of parameters leads to slower training. It can be seen that the training time of ResNet-18 is 2.74 times longer than that of simple CNNs. Our well-designed eResNet has been experimentally proven to be significantly better than ResNet-18 in both performance and time-consuming, and achieves better results than simple CNNs with a little time-consuming sacrifice.

We also make detailed ablation experiments of each proposed method, as showed below. Principal characteristic vs. no principal characteristic, see Table 4.

Mixture loss function vs. traditional loss function, see Table 5.

eResNet vs. ResNet, see Table 6.

4.3. Cifar100 few-shot classification

The Cifar100 [19] dataset has 100 classes, each class contains 600 samples of 32×32 size, which are divided into 500 training samples and 100 testing samples. The 100 classes are divided from 20 coarse classes, and each sample comes with a fine label (the fine class it belongs to) and a coarse label (the coarse class it belongs to). There are large differences between different coarse classes, such as "flowers", "fish", "household", etc.; while differences among fine classes in a same coarse class are small, such as "orchids", "poppies", "roses", etc. in coarse class "flowers". So we think this task is very suitable for few-shot learning. We reorganize the dataset and combine the training samples and test samples per class, and then randomly pick one fine class from each coarse class to compose 20 classes for validation, and likewise select 20 classes for testing without replacement, and use the remaining 60 classes for training.

Our experiments also used results with 95% confidence interval and averaged of 5 times experiments as the final experimental results, showing in Table 7. As can be seen from the table, our approach also has an excellent performance on the Cifar100 dataset. We also performed a detailed comparison. Compared to the matching networks [8], we increased by 24% on the 5-way, 5-shot task. As a result, the effectiveness of our approach has been verified on various datasets, and we can see the generalization ability of our approach.

Table 6
Effect of eResNet on miniImageNet dataset.

Model	5-way Acc.		Parameters	Cost
	1-shot	5-shot		
PRINCIPAL CHARACTERISTIC NETS (PC + CNN)	47.42 ± 0.68%	63.67 ± 0.74%	0.11×10^6	405.04 s/epoch
PRINCIPAL CHARACTERISTIC NETS (PC + ResNet-18)	47.00 ± 0.65%	60.58 ± 0.73%	11.13×10^6	1108.01 s/epoch
PRINCIPAL CHARACTERISTIC NETS (PC + eResNet)	**49.30 ± 0.75%**	**64.84 ± 0.70%**	$\mathbf{1.24 \times 10^6}$	**616.02 s/epoch**

Table 7
Few-shot classification accuracies on Cifar100 dataset.

Model	FT	5-way Acc.	
		1-shot	5-shot
MATCHING NETS [8]	N	53.82 ± 0.71%	65.55 ± 0.74%
MATCHING NETS [8]	Y	54.52 ± 0.61%	67.13 ± 0.71%
PRINCIPAL CHARACTERISTIC NETS (PC)	N	53.45 ± 0.65%	69.02 ± 0.67%
PRINCIPAL CHARACTERISTIC NETS (PC)	Y	59.97 ± 0.66%	76.38 ± 0.76%
PRINCIPAL CHARACTERISTIC NETS (PC + ML)	N	53.95 ± 0.73%	69.57 ± 0.70%
PRINCIPAL CHARACTERISTIC NETS (PC + ML + eResNet)	N	55.53 ± 0.72%	70.13 ± 0.73%
PRINCIPAL CHARACTERISTIC NETS (PC + ML + eResNet)	Y	**76.04 ± 0.73%**	**90.15 ± 0.69%**

5. Conclusion

We have proposed principal characteristic networks based on the training method of matching networks [8] and inspired by the prototype [9] idea. Aiming at the problem that common embedding networks extract low level of abstraction of samples, we propose an improved embedding network eResNet to improve extracting the high-level feature representation of samples. Aiming at the situation that the target in a sample is not obvious, we propose a weighted principal characteristic method based on the sum of absolute difference (SAD) of embedded vectors to reduce the influence of the unobvious data on the class principal characteristic. Aiming at the problem that networks misclassify similar classes, we put forward the mixture loss function to increase the distances among classes in the embedding space. We conducted detailed experiments to verify that our approach performed well on various datasets. At the same time, we also realize the shortcoming of our method: for embedding networks, there exist networks with better representation and fewer parameters theoretically (like densely connected structure [44]), which has further space for improvement. In future work, we hope to explore a better embedding network to achieve better results.

Conflict of interest

We declare that we have no conflicts of interest to this work.

Acknowledgment

We express our sincere thanks to the anonymous reviewers for their helpful comments and suggestions to raise the standard of our paper. We would also like to thank Yiqing Hu, Xuchen Yao, Tingting Yu for helpful discussions on the presentation. This work is partly supported by the National Natural Science Foundation of China under Grant No. 61672202.

References

[1] K. Yip, G.J. Sussman, Sparse representations for fast, one-shot learning, in: AAAI'97/IAAI'97 Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence, 1997, pp. 521–527.
[2] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, et al., Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016. ArXiv Preprint. ArXiv:1609.08144.
[3] S. Ren, K. He, R.B. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, vol. 2015, 2015, pp. 91–99.
[4] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: ICML Deep Learning Workshop, vol. 2, 2015.
[5] Y. Bengio, Deep learning of representations for unsupervised and transfer learning, in: Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 2012, pp. 17–36.
[6] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks, Neural Inform. Process. Syst. (2014) 3320–3328.
[7] A. Graves, G. Wayne, I. Danihelka, Neural Turing Machines, 2014. ArXiv Preprint ArXiv:1410.5401.
[8] O. Vinyals, C. Blundell, T.P. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, Neural Inform. Process. Syst. (2016) 3630–3638.
[9] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, Neural Inform. Process. Syst. (2017) 4077–4087.
[10] S. Ravi, H. Larochelle, Optimization as a model for few-shot learning, in: ICLR 2017: International Conference on Learning Representations 2017, 2017.
[11] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning for control, Artif. Intell. Rev. 11 (1997) 75–113.
[12] N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel, Meta-Learning with Temporal Convolutions, 2017. ArXiv Preprint ArXiv:1707.03141
[13] A. Banerjee, S. Merugu, I.S. Dhillon, J. Ghosh, Clustering with Bregman divergences, J. Mach. Learn. Res. 6 (2005) 1705–1749.
[14] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, 2017, pp. 1126–1135.
[15] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, T.P. Lillicrap, Meta-learning with memory-augmented neural networks, in: In ICML'16 Proceedings of the 33rd International Conference on International Conference on Machine Learning, vol. 48, 2016, pp. 1842–1850.
[16] S. Fort, Gaussian Prototypical Networks for Few-Shot Learning on Omniglot, 2018. ArXiv Preprint ArXiv:1708.02735.
[17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[18] B.M. Lake, R. Salakhutdinov, J. Gross, J.B. Tenenbaum, One shot learning of simple visual concepts, Cognit. Sci. 33 (33) (2011).
[19] A. Krizhevsky, V. Nair, G. Hinton, Cifar-10 (Canadian Institute for Advanced Research), 2010. http://www.cs.toronto.edu/kriz/cifar.html.
[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vision 115 (3) (2015) 211–252.
[21] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2014. arXiv preprint arXiv:1412.6980.
[22] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[23] T. Munkhdalai, H. Yu, Meta Networks, 2017. arXiv preprint arXiv:1703.00837.
[24] A. Wong, A.L. Yuille, One shot learning via compositions of meaningful patches, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1197–1205.
[25] B.M. Lake, R. Salakhutdinov, J.B. Tenenbaum, Human-level concept learning through probabilistic program induction, Science 350 (6266) (2015) 1332–1338.
[26] L. Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Trans. Pattern Anal. Mach. Intell. 28 (4) (2006) 594–611.
[27] M. Dixit, R. Kwitt, M. Niethammer, N. Vasconcelos, Aga: attribute-guided augmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7455–7463.
[28] B. Hariharan, R.B. Girshick, Low-shot visual recognition by shrinking and hallucinating features, in: ICCV, 2017, pp. 3037-3046.
[29] A. Mehrotra, A. Dukkipati, Generative Adversarial Residual Pairwise Networks for One Shot Learning, 2017. arXiv preprint arXiv:1703.08033.
[30] F.S.Y. Yang, L. Zhang, T. Xiang, P.H. Torr, T.M. Hospedales, Learning to Compare: Relation Network for Few-shot Learning, 2018.
[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, et al., Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[32] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[33] J. Lee, K. Cho, T. Hofmann, Fully Character-level Neural Machine Translation without Explicit Segmentation, 2016. arXiv preprint arXiv:1610.03017.
[34] J. Chung, K. Cho, Y. Bengio, A Character-level Decoder without Explicit Segmentation for Neural Machine Translation, 2016. arXiv preprint arXiv:1603.06147.
[35] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: European Conference on Computer Vision, Springer, Cham, 2016, pp. 21–37.

[37] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359.
[38] E. Parisotto, J.L. Ba, R. Salakhutdinov, Actor-mimic: Deep Multitask and Transfer Reinforcement Learning, 2015. arXiv preprint arXiv: 1511.06342.
[39] G. Cheng, P. Zhou, J. Han, Duplex metric learning for image set classification, IEEE Trans. Image Process. 27 (1) (2018) 281–292.
[40] G. Cheng, C. Yang, X. Yao, L. Guo, J. Han, When deep learning meets metric learning: remote sensing image scene classification via learning discriminative CNNs, IEEE Trans. Geosci. Remote Sens. 56 (5) (2018) 2811–2821.
[41] K. Li, G. Cheng, S. Bu, X. You, Rotation-insensitive and context-augmented object detection in remote sensing images, IEEE Trans. Geosci. Remote Sens. 56 (4) (2018) 2337–2348.
[42] A.A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, R. Hadsell, Meta-Learning with Latent Embedding Optimization, 2018. arXiv preprint arXiv:1807.05960.
[43] Y. Yu, Z. Ji, J. Guo, Z. Zhang, Zero-shot learning via latent space encoding, IEEE Trans. Cybernet. 99 (2018) 1–12.
[44] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: CVPR, vol. 1, no. 2, 2017, p. 3.