doi:10.3772/j.issn.1006-6748.2022.02.006

Task-adaptation graph network for few-shot learning①

ZHAO Wencang(赵文仓), LI Ming②, QIN Wenqian
(College of Automation and Electronic Engineering, Qingdao University of Science and Technology, Qingdao 266061, P. R. China)

Abstract

Numerous meta-learning methods focus on the few-shot learning issue, yet most of them assume that various tasks have a shared embedding space, so the generalization ability of the trained model is limited. In order to solve the aforementioned problem, a task-adaptive meta-learning method based on graph neural network (TAGN) is proposed in this paper, where the characterization ability of the original feature extraction network is ameliorated and the classification accuracy is remarkably improved. Firstly, a task-adaptation module based on the self-attention mechanism is employed, where the generalization ability of the model is enhanced on the new task. Secondly, images are classified in non-Euclidean domain, where the disadvantages of poor adaptability of the traditional distance function are overcome. A large number of experiments are conducted and the results show that the proposed methodology has a better performance than traditional task-independent classification methods on two real-word datasets.

Key words: meta-learning, image classification, graph neural network (GNN), few-shot learning

0 Introduction

Plentiful machine learning problems have been solved by the development of deep learning. Based on large datasets and high-dimensional datasets, high prediction accuracy has been achieved in tasks such as computer vision, machine translation, sentiment analysis, as well as speech recognition, and the development of artificial intelligence has also been promoted. However, the following key problem still remains: a large number of labeled samples are required when training models; on the contrary, in the real world, the labeled samples are quite rare or undisclosed, which makes it impossible to train a highly accurate model. The problem mentioned is the few-shot classification problem[1].

In recent years, in order to solve the above few-shot learning problem, researchers have shifted their focus to meta-learning (learning to learn). Thanks to prior knowledge can be used by meta-learning to train a model with strong generalization ability, the most important thing is that it has weak demand for sample labels and has a wide range of applications. The existing meta-learning methods can be divided into the following four categories: meta-learning methods based on parameter optimization, such as MAML algorithm[2]; meta-learning methods based on external memory, such as MetaNet algorithm[3]; meta-learning methods based on data enhancement, such as the SGM algorithm[4]; meta-learning methods based on metric learning, such as RepMet algorithm[5]. Metric learning is also called similarity learning. Its essence is to complete the few-shot classification by comparing the similarity of two images.

Although stupendous progress has been made by these methods, limitations still remain. First of all, the feature vectors extracted by most methods are not sufficiently distinguishable for new tasks. The main reason is that they do not consider that different classification tasks have different differences between their features, and the feature representation should also be adjusted according to the task. In other words, the feature representation that is applicable to a certain task does not be applicable to other tasks. In order to overcome this shortcoming, a feasible solution is to link the training task with the test task, so that the feature extraction process is more focused on the task to be completed next, rather than extracting feature representations for all classification tasks. Secondly, Euclidean or other artificially defined distance functions are usually used by most of the existing few-shot learning methods based on metric learning, which are limited to a two-dimen-

① Supported by the National High Technology Research and Development Program of China(20-H863-05-XXX-XX), the National Natural Science Foundation of China(61171131), Shandong Province Key Research and Development Program(YD01033) and the China Scholarship Council Program(201608370049).

② To whom correspondence should be addressed. E-mail: Leonalm529@163.com.

Received on Mar. 29, 2021

sional or multi-dimensional space[6]. The solution is to use a neural network to train this metric, so that the distance function can be adjusted according to different tasks.

A self-attention learning module based on the metric learning algorithm is proposed in this paper, which is used to improve the characterization ability of the original algorithm feature extraction network, so that the extracted feature vectors can be applied to various tasks (task-adaptation). In addition, a graph neural network (GNN) is employed as a distance measurement method to overcome the shortcomings of traditional measurement methods and the relationship between samples is fully used to further improve the accuracy of few-shot classification.

1 Related work

The essence of the attention mechanism is to only focus on key information, that is, when multiple types of information appear, only a specific part is captured. A great success in natural language processing, semantic segmentation, image classification and other fields[7] has been achieved.

In recent years, the attention mechanism has been improved and the self-attention mechanism has been produced by researchers, where the ability to capture the internal correlation of features is referred to by attracting attention to each information in a set of information. Similarly, remarkable results in various fields have been achieved by attention mechanism. For example, when completing natural language processing, a self-attention mechanism instead of a recurrent neural network was used to excellently complete the learning of text representation[8]. In Ref. [9], the task of instance segmentation and target detection was completed through adding a non-local module (the modeling method is the self-attention mechanism) to the ResNet network, and the performance was greatly improved. In addition, the task of extracting biomedical relationships was completed through applying the self-attention mechanism in Ref. [10].

Recently, innumerable image classification work have shifted to the attention mechanism[11-13]. Different weights were assigned to different samples in the training set, so that important features were paid more attention by the classification model. In this way, irrelevant information was ignored, however, the sample features in the training set were weighed while the samples in the test set were not considered. That is to say, after the attention mechanism, feature vectors are more conducive to the classification of specific tasks, the generalization ability on other tasks is still not ideal. The reason is that feature vectors should be different for different tasks. For example, two tasks are given, one is to distinguish between lion and tiger, the other is to distinguish between lion and wolf, the difference of lion and tiger must be different from the difference of lion and wolf. If the feature vectors used for classification are the same, the accuracy will be definitely affected. In order to overcome this limitation, a task-adaptive few-shot learning method is proposed.

For the above considerations, the characteristics of the test set samples are merged when the attention weights of the training set samples are calculated, so that the trained model can adapt to various classification tasks. In addition, a large number of existing few-shot classification methods based on metric learning are limited to the use of cosine distance[14], Euclidean distance[15] or nonlinear neural network[16] to calculate similarity, where the correlation between the samples is not sufficiently considered. And the graph neural network model[17] is used as the classification module, that is, the distance measurement is transferred from the Euclidean space to the non-Euclidean space, so as to fully explore the relationship between the samples and further improve the classification accuracy.

2 Problem definition

In this section, the definition of the meta-learning problem and graph classification problem will be introduced.

2.1 Meta-learning problem

Firstly, the specific meaning of N-way K-shot will be presented: the unlabeled samples predicted by classification models belong to one of the N categories, and there are only K labeled samples in each category, where K is a very small number. $N \times K$ samples with known labels will be used to make predictions on samples with unknown labels. All the samples mentioned above are from the test set, denoted as D_{test}, and the set of labeled samples in the test set is also called the support set D_{support}, the set of unlabeled samples is called the query set D_{query}. And the samples in D_{support} are denoted as x_s, while the samples in D_{query} are denoted as x_q.

Since there are few samples with known labels, D_{train}, whose samples are all labeled, is used to simulate N-way K-shot tasks. The specific steps are: randomly select N classes from the training set, where there are K samples in each class, and these $N \times K$ samples form support set D_{support}, and then randomly select P samples from the remaining samples in each

category to form query set D_{query}, and repeat the above steps M times, M N-way K-shot tasks can be derived. In order to simulate the test task, what need to do is to use the samples in the support set to predict the labels of samples in the query set , that is, train the model on the M tasks in the training set, and then generalize it to the test set. Usually the classification model f is trained by minimizing the loss sum of these tasks (Eq. (1)). The specific steps are shown in Algorithm 1.

$$Loss = \sum_{(x_q, y_q) \in D_{query}} l(f(x_q), y_q) \tag{1}$$

where $f(\cdot)$ represents a classifier.

Algorithm 1 The formation process of meta-tasks

for all samples x in D_{train} **do**
  for each task T_i, $\forall i \in \{1, \cdots, M\}$ **do**
    Randomly sample N classes from D_{train};
    Randomly sample K instances from N classes to form D_{support};
    Randomly sample P instances from ($D_{\text{train}} - D_{\text{support}}$) to form D_{query};
  end for
end for

2.2 Graph classification problem

The definition of the graph is briefly introduced. Graph is a data structure consisting of two components: nodes and edges. Nodes represent the object of classifying and edges represent the specific relationship between the two objects. An undirected graph can be described as a set of nodes and edges, denoted as $G = (V, E)$, G is a two-tuple, where $V = \{v_1, v_2, \cdots, v_i, \cdots, v_n\}$ is a node set; $E = \{e_{i,j} = (v_i, v_j)\} \subseteq (V \times V)$ is an edge set. In addition, the matrix storing the data of the relationship (edge) between nodes is called the adjacency matrix ($\boldsymbol{A}$)[18].

The graph neural network used for classification in this paper contains two modules: weight update module and node update module. The weight update module is composed of 5 fully connected layers. Four of them have a batch normalization layer and an activation function leaky ReLU layer. In order to learn θ, a fully connected layer is added at the end of the network. When the nodes need to be classified, the softmax function is used to normalize the adjacency matrix $\boldsymbol{A}$ row by row, and outputs the adjacency matrix $\boldsymbol{B}$, whose elements represent the similarity between nodes. The node update module is composed of graph convolution block and cascading operation layers. Among them, the graph convolution block contains a batch normalization layer and a leaky ReLU layer. The network structure of the graph neural network and its detailed work process is shown in Section 3.

3 Methods

The task-adaptive few-shot learning method will be introduced in this section. Firstly, how the embedding module extracts the initial feature representations is explained. Then the initial feature vectors are operated based on the self-attention mechanism, so that the output of the self-attention module is a series of task-related feature vectors. Finally the working process of the classification module (graph neural network) is illustrated.

The visualization of the operation process is shown in Fig. 1.

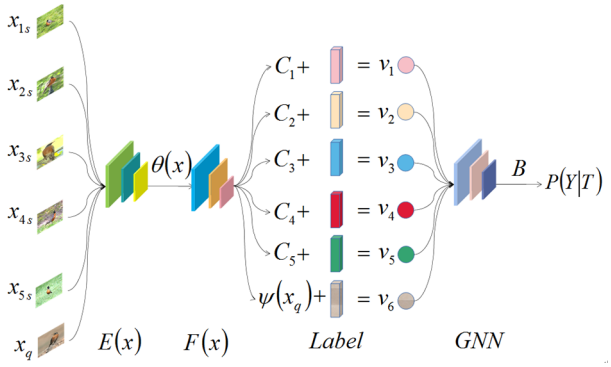


Fig. 1 The overview of task-adaptation graph network (TAGN) method

3.1 Feature embedding module

In order to learn the initial feature representations $\boldsymbol{\theta}(x)$ (Eq. (2)) of the inputs x, the prototype network[18] is used as the feature extraction network $E(x)$ (as visualized in Fig. 2). The core of this network is transforming the comparison between $\boldsymbol{\theta}(x_q)$ (the feature vectors of samples in D_{query}) and $\boldsymbol{\theta}(x_s)$ (the feature vectors of samples in D_{support}) (Eq. (3)) into the comparison between $\boldsymbol{\theta}(x_q)$ and $\boldsymbol{C}_i$ (the category center of D_{support}) (Eq. (5)). Among them, the calculation method of the category center of D_{support} is to take the average value of all the sample features in the category (Eq. (4)).

$$\boldsymbol{\theta}(x) = E(x), \quad \forall x \in D_{\text{train}} \tag{2}$$

$$y_q^p = f(\boldsymbol{\theta}(x_q), \boldsymbol{\theta}(x_s)) \propto \exp(sim(\boldsymbol{\theta}(x_q), \boldsymbol{\theta}(x_s)))y_s \quad \forall (x, y) \in D_{\text{train}} \tag{3}$$

where $f(\cdot)$ represents the classifier, y_q^p represents the predicted label of the sample x_q, y represents the true label of the sample x, and $sim(\cdot)$ represents the similarity between samples.

$$C_i = \frac{1}{M} \sum_{y_s = i} \boldsymbol{\theta}(x_s), \quad \forall i = 1, \cdots, N \tag{4}$$

where M represents the number of samples in category i, and N represents the number of categories in D_{support}.

$$y_q^p = f(\boldsymbol{\theta}(x_q), C_i) \propto \exp(sim(\boldsymbol{\theta}(x_q), C_i))y_i \tag{5}$$

where, $\forall i = 1, \cdots, N$.

In order to further improve the classification accuracy, a temperature scalar r[19] is added to the similarity. In this paper, the range of this value is [0.1, 1, 16, 32, 64, 128], at this time the predicted value y_q^p of the query set sample x_q can be calculated by Eq. (6).

$$y_q^p = \frac{\exp(r \cdot sim(\boldsymbol{\theta}(x_q), C_i))}{\sum_{i'=1}^{N} \exp(r \cdot sim(\boldsymbol{\theta}(x_q), C_{i'}))}y_i \tag{6}$$

where, $\forall i = 1, \cdots, N$.

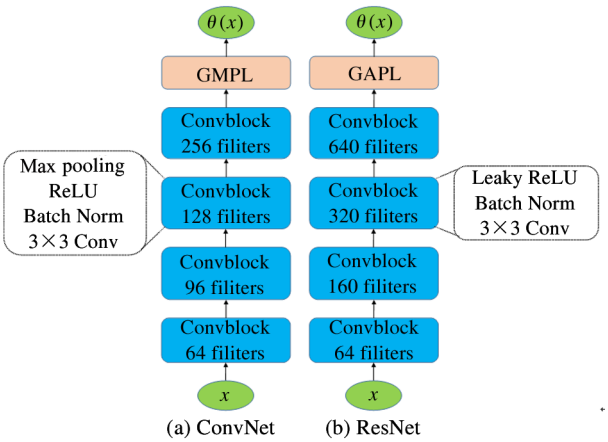


Fig. 2 The overview of the network structure of feature embedding module

3.2 Self-attention module

The purpose of self-attention module is to transform the initial feature representation $\boldsymbol{\theta}(x)$ unrelated to the task into a task - adaptive feature representation $\boldsymbol{\psi}(x)$ (Eq. (7)). The specific operation steps are based on the self-attention mechanism, that is, a converter F is introduced, whose function is to calculate the correct value corresponding to the query point. It is a store of triplets, which is composed of query, key and value[8]. Among them, the query vector represents the feature of the sample, the key vector represents the feature of the information, and the value vector represents the content of the information. The way of defining them will be described later.

$$\boldsymbol{\psi}(x) = F(\boldsymbol{\theta}(x)), \ \forall x \in D_{\text{train}} \tag{7}$$

In order to calculate the similarity between query points and keys, these query points are first linearly mapped to a certain space (Eq. (8)).

$$\begin{cases} \boldsymbol{Q}' = \boldsymbol{W}_Q^{\mathrm{T}}[\boldsymbol{\theta}_{x_q}; \ \forall x_q \in \boldsymbol{Q}] \in R^{d \times n} \\ \boldsymbol{K}' = \boldsymbol{W}_K^{\mathrm{T}}[\boldsymbol{\theta}_{x_k}; \ \forall x_k \in \boldsymbol{K}] \in R^{d \times n} \\ \boldsymbol{V}' = \boldsymbol{W}_V^{\mathrm{T}}[\boldsymbol{\theta}_{x_v}; \ \forall x_v \in \boldsymbol{V}] \in R^{d \times n} \end{cases} \tag{8}$$

where n is the number of sample points in the set $\boldsymbol{Q}$, which is the union of D_{support} and D_{query}, and $\boldsymbol{K} = \boldsymbol{V} = \boldsymbol{Q}$ (their mapping matrices are different).

To learn three feature maps $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, $\boldsymbol{W}_V$, the contrast loss (regularization term) is added to the loss function (Eq. (9)).

$$L(y_q^p, y_q) = l(y_q^p, y_q) + \lambda \sum_{x_q \in Q} \sum_{y_k = y_q} a_{qk} \log a_{qk} \tag{9}$$

where a_{qk} represents a certain degree of similarity, which will be explained in detail below, and the value of λ is discussed in the experimental part.

Next, the scale dot product[8] is used to get the attention score (Eq. (10)), the self-attention score a_{qk} can be obtained. That is, the degree of correlation or similarity between each sample and each piece of information.

$$a_{qk} = \frac{\exp\left\{\left(\frac{\boldsymbol{\theta}_{x_q}^{\mathrm{T}} \boldsymbol{W}_Q \boldsymbol{K}'}{\sqrt{d}}\right)\right\}}{\sum_{i=1}^{n}\left(\frac{\boldsymbol{\theta}_{x_i}^{\mathrm{T}} \boldsymbol{W}_Q \boldsymbol{K}'}{\sqrt{d}}\right)} \tag{10}$$

where d is the dimension of the vector $\boldsymbol{K}$ (which is set as 64 for ConvNet, and 640 for ResNet), which plays an adjustment role in scaling the dot product so that the inner product is not too large; the query point $x_q \in \boldsymbol{Q}$.

Then, the self-attention score obtained above becomes the weight of the vector $\boldsymbol{V}$, the weighted sum of the value vector $\boldsymbol{V}$, plus the initial feature representation $\boldsymbol{\theta}(x_q)$, a new task-adaptive embedding representation $\boldsymbol{\psi}(x_q)$ can be obtained, as shown in Eq. (11).

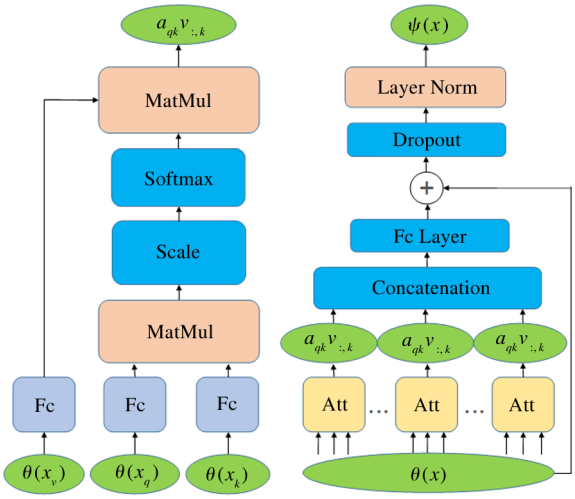
$$\boldsymbol{\psi}(x_q) = \boldsymbol{\theta}(x_q) + \sum_k a_{qk} \boldsymbol{V}_{:,k} \tag{11}$$

where $V_{:,k}$ is the k-th column of $\boldsymbol{V}$.

To convert the task-independent feature vectors $\boldsymbol{\theta}(x_q)$ to the task-adaptive feature vectors $\boldsymbol{\psi}(x_q)$, the 'self-attention' operation is utilized, and the scaling dot is used to get the self-attention score. In addition, the self-attention module consists of a fully connected layer, a matrix multiplication layer, and a softmax normalization layer. In order to improve the generalization ability of the model and prevent overfitting, a dropout layer (which rate is set as 0.5) is added to the network structure. In addition, in order to make the final model more stable and effective, layer normalization layer instead of batch normalization layer is utilized at the end, and its network structure is shown in Fig. 3.

3.3 Classification module

As mentioned in subsection 3.1, the center of task-adaption feature representation $\boldsymbol{\psi}(x)$ will be obtained firstly according to Eq. (12), and then the category center $\boldsymbol{C}_i$ and query set sample representations will



(a) Self-attention extraction (b) Feature embedding

Fig. 3 The overview of the network structure of self-attention module

be connected with their labels. For test samples that do not know the labels, $\boldsymbol{h}(l)$ is defined as a uniform distribution function (for the sake of simplicity, the label is filled with all 0) (Eq. (13)), and the spliced feature vectors are regarded as nodes, denoted as $V = \{v_1, v_2, \cdots, v_i, \cdots, v_n\}$, where $n = N + P$ (P is the number of sample points in D_{query}), which will be the inputs of the graph neural network.

$$\boldsymbol{C}_i = \frac{1}{M}\sum_{y_s=i}\boldsymbol{\psi}(x_s) \quad \forall i = 1, \cdots, N \tag{12}$$

where M represents the number of samples in category i, and N represents the total number of categories in D_{support}.

$$\begin{cases} \boldsymbol{V}_i = (\boldsymbol{C}_i, \boldsymbol{h}(l_i)) & \forall i = 1, \cdots, N \\ \boldsymbol{V}_j = (\boldsymbol{\psi}(x_j), K^{-1}1_K) & \forall j = 1, \cdots, P \end{cases} \tag{13}$$

The edges between nodes represent the degree of similarity between classes and samples. Therefore, the absolute difference between nodes is utilized to measure the similarity, and multi-layer perceptrons is used to construct the adjacency matrix $\boldsymbol{A}$ and Eq. (14) is used to update the adjacency matrix.

$$\boldsymbol{A}^l_{i,j} = MLP_\theta(abs(\boldsymbol{V}^l_i - \boldsymbol{V}^l_j)) \tag{14}$$

where A^l_{ij} represents the element in the i-th row and j-th column in the l-th matrix $\boldsymbol{A}$, and θ is a series of training parameters.

In addition to updating the weights, Eq. (15) is also used by the graph neural network to update the nodes, so that the relationship between samples can be fully explored by the graph model, thereby classification accuracy is capable to be improved. Its network structure is shown in Fig. 4. The specific process is shown in Algorithm 2.

$$\boldsymbol{V}^{l+1}_i = Gc(\boldsymbol{V}^l) = \rho\left(\sum_{B\in A}\boldsymbol{B}\boldsymbol{V}^l\theta^l_{B,i}\right)$$

$$\forall i = d_1, \cdots, d_{l+1} \tag{15}$$

where $Gc(\cdot)$ represents a neural network layer, $\rho(\cdot)$ represents the activation function leaky ReLU, d is the number of rows of the node vector, and $\boldsymbol{B}$ is the adjacency matrix normalized by softmax for each row of the adjacency matrix $\boldsymbol{A}$.

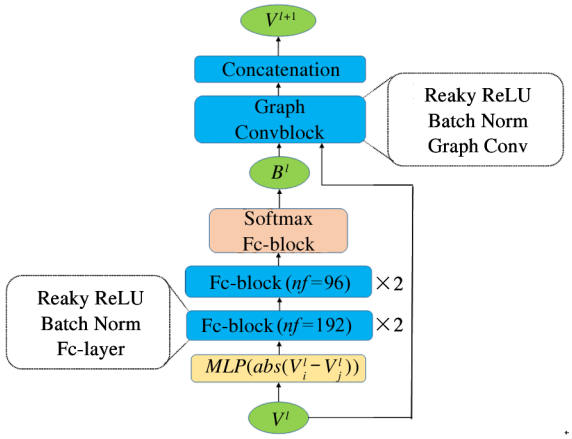


Fig. 4 The overview of the network structure of classification module

Algorithm 2 Training strategy of task-adaption graph network

Inputs: $D_{\text{Support}} = \{(x_1, y_1), \cdots, (x_{N\times K}, y_{N\times K})\}$; $D_{\text{query}} = \{x_1, \cdots, x_P\}$

Outputs: labels y of samples in D_{query}

for all M tasks **do**

Calculate $\boldsymbol{\theta}(x)$ using Eq. (2);

Calculate a_{qk} using Eq. (10);

Calculate $\boldsymbol{\psi}(x)$ using Eq. (11);

Calculate $\boldsymbol{C}_i$ using Eq. (12);

Connect $\boldsymbol{C}_i$, $\boldsymbol{\psi}(x_j)$ with its labels using Eq. (14) $\rightarrow \boldsymbol{V} = (v_1, \cdots, v_{N+P})$;

for all layers $= 1, \cdots, l$ **do**

Calculate $\boldsymbol{A}^l$ using Eq. (14);

Calculate $\boldsymbol{V}^{l+1}$ using Eq. (15);

Predict the labels y^p_q of x_q using adjacency matrix $\boldsymbol{B}$;

Calculate the loss function $l(y^p_q, y_q)$ using Eq. (16);

end for

Calculate the gradient of *Loss* using Eq. (1);

Update E, F and GNN with the gradient of *Loss* using stochastic gradient descent;

end for

The adjacency matrix $\boldsymbol{B}$ stores the weight value between any connected nodes, namely, the similarity between nodes (in the form of probability P).

In addition, the cross-entropy loss Eq. (16) is used to train the parameters of the GNN model.

$$l(y^p_q, y_q) = -\sum_k y_k \log P(y^p_q = y_q \mid T) \tag{16}$$

where k is the number of samples in D_{query}.

4 Experiments

The performance of the task-adaptation method on two general datasets (MiniImageNet and CUB200-2011) is shown in this section. Firstly, a brief introduction to the two datasets and their category allocation is given. Then the accuracy of the proposed method and baselines is compared to verify the advancement of the proposed method based on two backbone networks. Finally ablation experiments are conducted to further test the significance of each module in TAGN.

4.1 Datasets

MiniImageNet(Mini) dataset[15] is excerpted from the ImageNet dataset and it is the benchmark dataset in the field of meta-learning and few-shot learning. It contains 100 categories of 60 000 images. And there are 600 samples in each category.

CUB200-2011(CUB) dataset[15] is a fine-grained dataset presented in 2010, and it contains 200 categories of 11 788 images. More about the setting methods of these two datasets are shown in Table 1.

Table 1 Descriptive statistics of datasets

Datasets	Images	Classes	Train	Verify	Test
Mini	60 000	100	64	16	20
CUB	11 788	200	100	50	50

4.2 Backbone networks

Two traditional feature extraction networks are used as $E(x)$, namely four-layer convolution network (ConvNet-4) and residual network (ResNet-12). Next, they will be introduced in detail.

ConvNet-4 consists of 4 identical convolutional blocks, and each convolutional block contains a 3×3 convolutional layer, a batch normalization layer, an activation function layer and the maximum pooling layer for compression. And in order to reduce the amount of calculation for subsequent operations, a global max pooling layer (GMPL) that reduces the dimension of feature representation is added at the end. Its network structure is shown in Fig. 2(a).

ResNet-12 is composed of 4 convolutional blocks, and each convolutional block contains a 3×3 convolutional layer, a batch normalization layer, and an activation function layer. Also in order to reduce the amount of calculation, a global average pooling layer (GAPL) is added at the end. Its network structure is shown in Fig. 2(b).

4.3 Results and discussion

Table 2 and Table 3 indicate the results of the performance comparison between task-adaptive method and baselines, where the model of this paper achieves the best performance on two datasets. Among all models, two non-linear neural networks-based models, relation network and GNN, have shown better results, compared with traditional distance function-based models matching network and prototypical network. For example, in 5-way 1-shot setting the accuracy of GNN improved by at least 0.16% (50.46% vs. 50.62%) on MiniImageNet and 1.27% (62.45% vs. 63.72%) on CUB200-2011, while in 5-way 5-shot setting the accuracy of GNN improved by 0.62% (65.85% vs. 66.47%) on MiniImageNet and 5.44% (76.12% vs. 81.56%) on CUB200-2011. To further demonstrate the advancement of the method of this paper, it is compared with EGNN[20] and TPN[21], where the accuracy is improved by at least 0.97% (52.46% vs. 53.43%) and 1.05% (81.64% vs. 82.69%), under 5-way 1-shot setting and 5-way 5-shot setting compared with EGNN, and the accuracy is improved by at least 0.08% (53.35% vs. 53.43%) and 0.97% (81.72% vs. 82.69%), under 5-way 1-shot setting and 5-way 5-shot setting compared with TPN (as shown in Table 2 and Table 3).

Table 2 Few-shot classification accuracy on MiniImageNet dataset

Methods	5-way 1-shot		5-way 5-shot	
	Conv	ResNet	Conv	ResNet
MatchNet	43.84%	-	56.32%	-
ProtoNet	49.54%	-	65.21%	-
RelationNet	50.46%	-	65.85%	-
GNN	50.62%	-	66.47%	-
EGNN	52.46%	-	66.85%	-
TPN	53.35%	-	69.43%	-
MatchNet(TA)	52.84%	61.55%	68.46%	75.91%
ProtoNet(TA)	52.31%	62.10%	71.35%	76.58%
Proposed method	**53.43%**	**62.60%**	**72.17%**	**77.95%**

In addition, in order to verify the effectiveness of the task-adaptation module (TA), a self-attention module is added to the original structure of the matching network and the prototype network. The results show that due to the addition of the self-attention mechanism on MiniImageNet, the accuracy of matching network has increased from 43.84% to 52.84% (5-way 1-shot setting), the accuracy of prototype network increased from 49.54% to 52.31% (5-way 1-shot setting).

Table 3 Few-shot classification accuracy on CUB200-2011 dataset

Methods	5-way 1-shot		5-way 5-shot	
	Conv	ResNet	Conv	ResNet
MatchNet	61.26%	-	72.68%	-
ProtoNet	51.31%	-	70.87%	-
RelationNet	62.45%	-	76.12%	-
GNN	63.72%	68.09%	81.56%	83.65%
EGNN	64.23%	-	81.64%	-
TPN	65.57%	-	81.72%	-
MatchNet(TA)	66.98%	73.69%	80.08%	85.68%
ProtoNet(TA)	68.65%	77.13%	80.78%	85.54%
Proposed method	**68.94%**	**77.26%**	**82.69%**	**86.71%**

As for CUB200-2011, the accuracy of the matching network increased from 62.26% to 66.98% (5-way 1-shot setting), the accuracy of prototype network has risen from 51.31% to 68.65% (5-way 1-shot setting). This proves that the motivation using self-attention module as supplement to feature extraction module gets benign results. Moreover, the task-adaptive graph neural network has at least improved accuracy compared with the task-unknown baselines 0.5% (62.10% vs. 62.60%) and 0.13% (77.13% vs. 77.26%) for MiniImageNet and CUB200-2011 respectively. It implies that the model that uses self-attention mechanism to get task-adaptation feature vector and then utilizes GNN as classification module has received well performance.

4.4 Ablation experiments

To predict how the regularization λ impacts the results, another experiment is implemented, where the value of λ is changed and all other parameters are fixed to the values that produce the best results. In addition, the value of λ is set as 0, 1, 10, 100, and the changes of classification accuracy are observed based on ConvNet-4 on the dataset MiniImageNet. The results show that when the value of λ changes within a certain range, the accuracy is increased. And models perform well when the value of $\lambda = 10$. The overall operational results is depicted in Fig. 5.

In order to prove the advancement of TAGN, comparative experiments are conducted on the MiniImageNet dataset and CUB200-2011 dataset based on ConvNet-4. Among them, the comparison baselines are the matching network using the cosine distance, the prototype network using the squared Euclidean distance, and GNN which changes the distance measurement from the Euclidean domain to the non-Euclidean domain. Results are shown in Fig. 6, where MatchNet, ProtoNet, GNN and TA represents matching network,

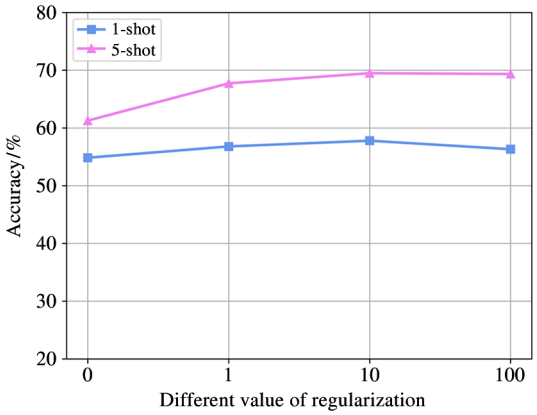
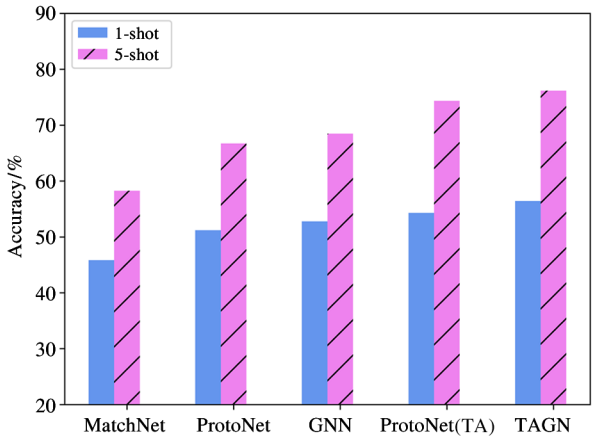
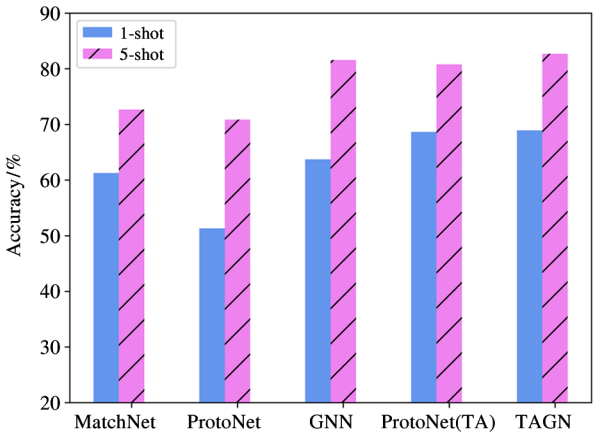


Fig. 5 Line chart of the accuracy for different value of regularization



(a) Different network structures on MiniImageNet



(b) Different network structures on CUB200-2011

Fig. 6 Histogram of the accuracy for different network structure

prototype network, graph neural network and the self-attention (task-adaptation) module, respectively. The results prove that the methods of training the metric function with neural network are better than the traditional methods of fixed distance function.

As for the accuracy on the MiniImageNet dataset, the addition of self-attention module improves the accuracy from 49.54% and 65.21% to 52.31% and 71.35%,

under 1-shot and 5-shot settings, respectively. And the method TAGN increases the accuracy by at least 2.21% (52.31% vs. 54.43%) and 0.82% (71.35% vs. 72.17%) under 1-shot and 5-shot settings. As for the accuracy on the CUB200-2011 dataset, the addition of self-attention module improves the accuracy from 51.31% and 68.65% to 70.87% and 80.78%, under 1-shot and 5-shot settings, respectively. And the method TAGN increases the accuracy by at least 0.29% (68.65% vs. 68.94%) and 1.13% (81.56% vs. 82.69%) under 1-shot and 5-shot settings.

5 Conclusions

A feature learning model based on the metric learning algorithm is proposed in this paper, where the self-attention mechanism is utilized to produce task-adaption feature vectors and adaptable model. In addition, a classification framework based on graph neural network is established, where the relationship between samples can be fully explored by the model, and the effect of improving the accuracy of few-shot classification can be achieved. The experimental results on the datasets MiniImageNet and CUB200-2011 prove that the method of this paper is better than task-independent methods. In the future, the model will be extended to the two settings including zero-shot classification and generalized few-shot classification.

References

[1] PAPERNOT N, McDANIEL P, JHA S, et al. The limitations of deep learning in adversarial settings[C]//2016 IEEE European Symposium on Security and Privacy, Saarbruecken, Germany, 2016: 372-387

[2] FINN C, ABBEEL P, LEVINE S. Model-agnostic meta-learning for fast adaptation of deep networks[C]//International Conference on Machine Learning, Proceedings of Machine Learning Research, Sydney, Australia, 2017: 1126-1135

[3] MUNKHDALAI T, YU H. Meta networks[C]//International Conference on Machine Learning, Sydney, Australia, 2017: 2554-2563

[4] HARIHARAN B, GIRSHICK R. Low-shot visual recognition by shrinking and hallucinating features[C]//Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 2017: 3018-3027

[5] KARLINSKY L, SHTOK J, HARARY S, et al. Repmet: representative-based metric learning for classification and few-shot object detection[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019: 5197-5206

[6] HU J, LU J, TAN Y P. Discriminative deep metric learning for face verification in the wild[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, USA, 2014: 1875-1882

[7] ZHU X, CHENG D, ZHANG Z, et al. An empirical study of spatial attention mechanisms in deep networks[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, USA, 2019: 6688-6697

[8] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 6000-6010

[9] WANG X, GIRSHICK R, GUPTA A, et al. Non-local neural networks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 7794-7803

[10] VERGA P, STRUBELL E, McCallum A. Simultaneously self-attending to all mentions for full-abstract biological relation extraction[C]//Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orieans, USA, 2018: 872-884

[11] ZHENG Y, WANG R, YANG J, et al. Principal characteristic networks for few-shot learning[J]. *Journal of Visual Communication and Image Representation*, 2019, 59: 563-573

[12] WANG F, JIANG M, QIAN C, et al. Residual attention network for image classification[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy, 2017: 3156-3164

[13] CAO C, LIU X, YANG Y, et al. Look and think twice: capturing top-down visual attention with feedback convolutional neural networks[C]//Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 2015: 2956-2964

[14] VINYALS O, BLUNDELL C, LILLICRAP T, et al. Matching networks for one shot learning[C]//Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 2016: 3637-3645

[15] SNELL J, SWERSKY K, ZEMEL R. Prototypical networks for few-shot learning[C]//Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 4080-4090

[16] SUNG F, YANG Y, ZHANG L, et al. Learning to compare: relation network for few-shot learning[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018:1199-1208

[17] GARCIA V, BRUNA J. Few-shot learning with graph neural networks[EB/OL]. https://arxiv.org/pdf/1711.04043.pdf:arxiv, (2018-02-20), [2021-03-29]

[18] WU Z, PAN S, CHEN F, et al. A comprehensive survey on graph neural networks[J]. *IEEE Transactions on Neural Networks and Learning Systems*, 2021, 32(1): 4-24

[19] ORESHKIN B N, RODRIGUEZ P, LACOSTE A. TADAM: task dependent adaptive metric for improved few-shot learning[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, Canada, 2018: 719-729

[20] KIM J, KIM T, KIM S, et al. Edge-labeling graph neural network for few-shot learning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019: 11-20

[21] LIU Y, LEE J, PARK M, et al. Learning to propagate labels: transductive propagation network for few-shot learning[EB/OL]. https://arxiv.org/pdf/1805.1002.pdf:arxiv, (2019-02-08), [2021-03-29]

ZHAO Wencang, born in 1973. He received his Ph.D degree in Information Science Department of Ocean University of China in 2005. He also received his B.E. and M.E. degrees from Qingdao University of Science and Technology and Shandong University in 1995 and 2002 respectively. He is mainly engaged in image processing and pattern recognition, cognitive informatics, neural computing, machine learning.