Hierarchically Structured Network With Attention Convolution and Embedding Propagation for Imbalanced Few-Shot Learning

WEI XIONG, YU GONG, WEIHUA NIU, AND RUONAN WANG

1Department of Computer, North China Electric Power University, Baoding 071000, China
2Engineering Research Center of Intelligent Computing for Complex Energy Systems, Ministry of Education, Baoding 071000, China

Corresponding author: Yu Gong (gongyu0217@163.com)

ABSTRACT Generally, a few-shot distribution shift will lead to a poor generalization. Furthermore, while the number of instances of each class in the real world may significantly different, the existing few-shot classification methods are based on the assumption that the number of samples in each class is equal, which causes the trained classifier invalid. Moreover, through ResNet and WRN (Wide Residual Network) have achieved great success in the image processing field, the depth and width of CNNs constrain the conventional convolution layer performance. Thus, to overcome the above problems, the model of this paper proposes a novel few-shot classification model that uses learning balance variables to decide how much to learn from the imbalance dataset, which dynamically generates the convolution kernel based on each input. In our model, to extend the decision boundaries and enhance the class representations, this paper uses embedding propagation as a regularizer for manifold smoothing. Manifold smoothing can effectively solve the above problems of transductive learning. The interpolations between neural network features based on similarity graphs are used by embedding propagation. Experiments show that embedding propagation can produce a better embedding manifold and our model in standard few-shot datasets, such as miniImagenet, tieredImagenet, CUB has state-of-the-art results. It significantly outperforms the existing few-shot approaches, which consistently improves the accuracy of the models by about 11%.

INDEX TERMS Dynamic convolution, few-shot classification, imbalance, manifold smoothing.

I. INTRODUCTION Contemporary machine learning models usually use hand-designed fixed learning algorithms to train from scratch to a specific task. Compared with machine learning, deep learning solves the problem of manually labeling features to a certain extent. It inputs data into the network to make it automatically learn features, which significantly improves the model’s performance. Traditional machine learning and deep learning have achieved great success in many fields (image [1], [2], NLP [3], [4]).

However, these methods faced urgent problems. In many cases, it is impossible to obtain a large number of labeled samples and robust computing power. To solve the above limitations, few-shot learning [5]–[7] and semi-supervised learning [8] have been used all the time.

Like human learning of new concepts and skills, children can distinguish something after seeing it several times. People who can ride a bicycle may ride a motorcycle skillfully without learning how to ride a motorcycle. How to quickly learn new concepts and skills through several training examples. It is a problem to be solved by few-shot learning. Recently, some approaches have successful results in few-shot classification (FSC): metric learning and meta-learning. Prototypical Networks [9] and Relation Networks [10] are metric learning. Moreover, in [11], to distinguish the training data and testing data in the training stage and the actual testing data in the testing stage, it provides a novel name for the data in the training stage. Support data (training data in the training stage), query
data (testing data in the training stage). The details about the definition of Support and Query show in Fig.1. Another way in the few-shot learning is meta-learning [12], [13], which learns a set of initial network parameters that can effectively shift to new tasks through the two-stage optimization. Transfer learning [14], [15] leverages prior knowledge well and learns the class presentation.

However, a critical challenge in few-shot classification is that models cannot achieve a global parameter suit for all tasks through few-shot images. To solve this problem, some methods have been used, such as dropout [15], batch normalization [15], manifold mixup [14], and some task distribution algorisms [16]. In the above works, we found the manifold smoothing steadily improves the performance of a classifier. Our method is different from manifold [14] in that our method works between the embeddings, which can obtain higher representation information.

The EP (embedding propagation) method in our model can utilize features captured from the network and have a set of interpolations using their similarity based on a graph. This model uses the radial basis function (RBF) to handle the pairwise similarities in the constructed graph. EP applied on the top of the feature extractor. It can be seen as a layer in the training stage, and testing stage has a regularized manifold. The details about EP show in Fig.2.

The ResNet and other networks have achieved many successful results in the image field. Nevertheless, to improve the ability to capture the feature and deduce the reduction of the convolution network, our model use Dynamic Convolution [17] to replace the conventional convolution layer. It can dynamically yield the convolution kernel depending on inputs by utilizing the attentions [18].

Furthermore, most published few-shot ways have not considered the imbalance problems in each class (in the realistic world, some classes have a large number of instances, but others very seldom), which leads to the imbalance (class imbalance) appears in classes, causing the classification results to tend to the class with more samples. Moreover, in the training stage, this model needs to randomly choose classes from the train set to construct the n-way-k-shot proposed by [18]. In the end, some classes maybe are chosen more times than other classes, which causes the imbalance in class to choose (class chosen imbalance). In our work, this model learns the balancing variables. It can decide how much this model learns in the special classes to solve the above imbalance problems.

We refer to our network as EPCINet (Embedding Propagation Dynamic Imbalance Network). For few-shot classification, EPCINet leverages the unlabeled data in the testing stage to yield a better label of the query set. [18], [19] As productive semi-supervised learning is used in our work at the few-shot classification setup. Thus, for each unlabeled image, EPCINet chooses the class of the maximum predicted probability as the pseudo label. EPCINet then uses these pseudo labels and the support set to perform label propagation to predict the labels of the query set. This approach achieves...
significant improvement over the previous state-of-the-art in the 1-shot SSL setting. We hypothesize that EPCINet is effective in the SSL setting because of the properties of smoother manifolds[18].

Overall, EPCINet has the state-of-the-art results on mini-Imagenet [11], CUB [20], tieredImagenet [21] in few-shot classification. In our experiments, we evaluate variations of embedding propagation and their impact on the smoothness of the decision boundaries. Experiment shows that, with EP, our model also achieves a clear improvement compared to the same model without EP. Our main contributions can be summarized as follows.

First, Leverage embedding interpolations to capture higher-order feature information.

Second, the model proposed in this paper considers a problem that is easily overlooked by previous few-shot works, the class imbalance and the class chosen imbalance in a real application scenario.

Third, Dynamic convolution is used to improve the ability to capture features and significantly reduce the reduction of the network.

Fourth, the effectiveness of the proposed methods is verified by ablation experiments.

II. RELATED WORK

Few-shot classification domains in our work. Nevertheless, manifold regularization, meta-learning, and dynamic convolution are also used in this model. This section discusses the recent related topics and finds their connections to our method.

A. FEW-SHOT CLASSIFICATION

The most existing meta-learning methods base on the bilevel optimization and the most famous is MAML [13], which proposed the ‘Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks’. However, the traditional MAML needs a lot of computation ability and often has a model shock. Thus, Reptile [22] is proposed. This algorithm and joint training are similar, the first-order is used instead of the second-order to update the parameters, which significantly reduces the calculation of the model, and there is no loss of gradient information. Experimental results show that the method of one-order approximation to two-order almost does not affect the performance of the model, and at the same time, dramatically reduces the computational complexity of the model. Based on previous work, other ways [23]–[25] that belong to meta-learning were proposed.

Most metrics use episode methods to train the model. It also can be seen as a part of meta-learning. Metric learning first learns a classifier over a feature space. Propagate labels are attained by [26] in EPCINet. Graph-based methods also belong to this category [27]–[29]. GNN [30] can produce the classification weights and leverage an autoencoder to regularize their representation. However, the unsupervised and non-parametric graph in our work is different from GNNs.

B. DYNAMIC CONVOLUTION

Traditional convolution has great success in image processing fields. In general, deeper network depth makes the network more productive. Nevertheless, it often has gradient disappears, gradient explosion problems. Previous work has proposed many productive networks are consist of CNN, such ResNet [34] and WRN [35]. Compared to a simple CNN network, these networks can capture more feature information, making the network perform better. However, the above methods use the traditional CNN. Light-weight CNN has performance degradation when suffers computation constrain. Dynamic convolution [17] is presented to overcome this issue, which improves the model performance without increasing the network depth. The general architecture shows in Fig.4 and Fig.5. Appealing attention to the input, it dynamically yields a special convolution kernel per layer. These experiments [36], [37] indicated that dynamic convolution could improve the result in the testing stage. Not surprisingly, our model achieves a more productive feature extractor experimentally and demonstrates the effectiveness of dynamic convolution by ablation experiments. The architecture of the Dynamic convolution generator shows in Fig.4.

C. REGULARIZATION FOR GENERALIZATION

Many methods are used to overcome the generalization performance problem, such as dropout, batch normalization. These methods improve the robustness of the model. But, a more effective regularization method called manifold regularization has better results in our work. [14], [32], [38] The
## Dynamic Convolution

### 1) Definition

Dynamic Convolution yields a dynamic convolution kernel for each input in each convolution layer instead of a fixed kernel for each convolution layer. To better understand this concept, here are some symbols. We denote the traditional perceptron as \( y = g(W^T x + b) \), where \( W \) is weight vector, \( b \) is bias vector, \( g \) as an activation function (e.g., ReLU). Our paper aggregates multiple (\( K \)) linear functions \( \{\widetilde{W}_k^T x + \tilde{b}_k\} \) as follows:

\[
y = g \left( W^T (x) x + \tilde{b} \right) \\
\widetilde{W} (x) = \sum_{k=1}^{K} \pi_k (x) \widetilde{W}_k \\
\tilde{b} = \sum_{k=1}^{K} \pi_k (x) \tilde{b}_k
\]

\[
\text{s.t} \ 0 \leq \pi_k (x) \leq 1, \sum_{k=1}^{K} \pi_k (x) = 1
\]

where \( \pi_k \) is the \( k^{th} \) attention weight in the \( \widetilde{W}_k^T x + \tilde{b}_k \), \( \widetilde{W} \) (x), and \( \tilde{b} \) (x) share the same attention score.

### 2) Attention

Each input \( x \) dynamically changes \( \pi_k (x) \). \( \widetilde{W}_k^T x + \tilde{b}_k \) is a non-linear function, which combines given input and has better performance. SENet [18] is used to gain attention score \( \pi_k (x) \). First, the global information squeezed by GAP (global average pooling). Following is the fully-connected layer, ReLU, and fully-connected layer, the softmax layer yields \( K \) attention weights for input. For input with \( H \times W \times C_{in} \), the attention with \( O (\pi (x)) = HWC_{in} + \frac{C_{in}}{4} + \frac{C_{out}K}{4} \), this is much less than the computational cost of convolution, \( O (\widetilde{W}^T (x) + \tilde{b}) = HWC_{in}C_{out}D_k^2 \), where \( D_k \) is kernel size, \( C_{out} \) is output channels.

From CNNs to Dynamic-CNNs. It can be used in a network that consists of convolution. Such as ResNet, WRN, and so on. From [17], we know it did not add too much computation. For the degree of performance improvement, the additional increase is worth it.

## III. Proposed Method

This paper proposes a novel net called EPCINet with the following methods. The overview architecture of the dynamic convolution generator shows in Fig.6. Each input yields a dynamic convolution kernel by dynamic convolution layer and extracts the images features by the extractor. Then we leverage a new embedding propagation method to construct the map between features and embeddings. The classifier uses the above embeddings to label images. This section explains EPCINet in detail.

### A. Dynamic Convolution

#### 1) Definition

Dynamic Convolution shows in Fig.4. The general architecture of dynamic convolution.

\[ y = W^T x + b \]

\[
\pi (x) = \sigma (W^T x + b) \\
\widetilde{W} (x) = \sum_{k=1}^{K} \pi_k (x) \widetilde{W}_k \\
\tilde{b} = \sum_{k=1}^{K} \pi_k (x) \tilde{b}_k
\]

\[
\text{s.t} \ 0 \leq \pi_k (x) \leq 1, \sum_{k=1}^{K} \pi_k (x) = 1
\]

#### 2) Attention

Each input \( x \) dynamically changes \( \pi_k (x) \). \( \widetilde{W}_k^T x + \tilde{b}_k \) is a non-linear function, which combines given input and has better performance. SENet [18] is used to gain attention score \( \pi_k (x) \). First, the global information squeezed by GAP (global average pooling). Following is the fully-connected layer, ReLU, and fully-connected layer, the softmax layer yields \( K \) attention weights for input. For input with \( H \times W \times C_{in} \), the attention with \( O (\pi (x)) = HWC_{in} + \frac{C_{in}}{4} + \frac{C_{out}K}{4} \), this is much less than the computational cost of convolution, \( O (\widetilde{W}^T (x) + \tilde{b}) = HWC_{in}C_{out}D_k^2 \), where \( D_k \) is kernel size, \( C_{out} \) is output channels.

From CNNs to Dynamic-CNNs. It can be used in a network that consists of convolution. Such as ResNet, WRN, and so on. From [17], we know it did not add too much computation. For the degree of performance improvement, the additional increase is worth it.

#### B. Embedding Propagation

In an episode, after using a feature extractor (Dynamic CNN), we obtain the feature vectors \( Z_i \in \mathbb{R}^{m} \). Embedding propagation handles a set of vectors and outputs embeddings \( Z_i \in \mathbb{R}^{m} \) by two steps.

First, for each feature \( (i, j) \), our model produces the distance by \( d_{ij}^2 = \|Z_i - Z_j\|^2 \) and computes the adjacency \( A_{ij} = \exp(-d_{ij}^2/\sigma^2) \), where \( \sigma^2 \) is the controlling variable, \( A_{ii} = 0 \), as mentioned in TPN [22]. In our work, set \( \sigma^2 = Var(d_{ij}^2) \) to make a model more stable.

In the following section, we compute by (4) to obtain the Laplacian operator of the adjacency matrix,

\[
L = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, \quad D_{ii} = \sum_{j} A_{ij} \tag{4}
\]

and compute propagator matrix \( P \) by (5), as mentioned in [39] which proposes the label propagation formula,

\[
P = (I - aL)^{-1} \tag{5}
\]

where \( a \in \mathbb{R} \) is a controlling variable, and \( I \) is the identity matrix. Next, we compute embeddings by (6).

\[
\tilde{Z}_i = \sum_{j} P_{ij}Z_j \tag{6}
\]

Because the \( \tilde{Z}_i \) has its weight which can remove the useless noise from the feature vectors. It is easy to operate and can be compatible with a multi-function extractor and classifier. More significant, the computation cost can be ignored in few-shot episodes [22] due to there is a few-shot cases.
C. FEW-SHOT SETUP

Previous work [1] has proposed the procedure of few-shot setup. We have three datasets: train dataset \( (D_{\text{train}}) \), validation dataset \( (D_{\text{val}}) \), test dataset \( (D_{\text{test}}) \). Many labelled images in \( D_{\text{train}}, D_{\text{train}} = \{(X_i, y_i)\}_{i=1}^{N_{\text{train}}} \) each \( X_i \) is labelled by \( y_i, D_{\text{test}} = \{(X_j, y_j)\}_{j=1}^{N_{\text{novel}}} \), where \( X_j \) belongs to unseen classes \( y_j \), which is used to measure model transfer ability. The last dataset \( D_{\text{val}} \) is used for optimizing the model parameter.

Furthermore, each episode chooses \( n \) classes without repetition from all classes and \( K \) samples, called \( n \)-way-\( k \)-shot. In order to distinguish train set and test set in the training stage, we call them separately support set and query set.

D. INFERENCE PHASE

In the inference phrase, this model leverage EP to handle features from the input images and label propagation is applied. In general, we can obtain the matrix of propagated features from the input images and label propagation is carried out. In the first stage, we can get the propagated features \( \tilde{Z} \). We predict the labels for the query set by label propagation as described in [37].

E. TRAINING PROCEDURE

This model is trained through two stages. In the first stage, to learn a general representation, our method leverages the pre-training method over \( D_{\text{train}} \). The second stage is to generalize the novel classes. The episodes are constructed by \( D_{\text{train}} \).

In both stages, extractor \( F_\theta(X) \) parametrized by \( \theta \) is used to capture the features \( \tilde{Z} \) for image \( X \).

F. PRE-TRAINING STAGE

As describe in Fig 6., we train \( F_\theta(X) \) by simple linear classifiers with activations parametrized by \( W_l \) and \( W_r \) respectively. At the start of the training stage, the classifier is to predict the class labels. We use a cross-entropy loss to optimize our model.

\[
\mathcal{L}_c (X_i, y_i; W_l, \theta) = -\ln p(y_i | \tilde{Z}_i, W_l)
\]

where \( y_i \in Y_b \) and it can use softmax to compute the probabilities.

G. EPISODIC LEARNING STAGE

In our model, we use the learned classifier to generalize the new classes. In this stage, our model also uses the two classifiers to handle images. The first uses label propagation through computing class probabilities over the query set label.

\[
L_p (X_i, y_i; \theta) = -\ln p(y_i | \tilde{Z}_i, \tilde{Z}, Y_S) \quad (8)
\]

The second classifier and the \( W_l - \text{based} \) classifier in the pretraining stage are the same. Therefore, we try our best to reduce the following losses:

\[
\begin{align*}
\arg \min_{\theta, W_l} \left[ \frac{1}{|Q|} \sum_{(X_i, y_i) \in Q} \frac{1}{|S \cup Q|} \sum_{(X_j, y_j) \in S \cup Q} \mathcal{L}_c (X_i, y_i; W_l, \theta) + \frac{1}{2} L_p (X_i, y_i; \theta) \right] \\
\arg \min_{\theta, W_l} \left[ \frac{1}{|Q|} \sum_{(X_i, y_i) \in Q} \frac{1}{|S \cup Q|} \sum_{(X_j, y_j) \in S \cup Q} \mathcal{L}_c (X_i, y_i; W_l, \theta) + \frac{1}{2} L_p (X_i, y_i; \theta) \right]
\end{align*}
\]

IV. VARIATIONAL INFERENCE

The log likelihood of the joint Dataset \( D^T \) and \( D^T \), which is needed to maximize each task. However, computing the posterior \( p (\phi^T | D^T, D^T; \psi) \) is difficult, as shown in Fig 7. In the end, variational inference is used to solve this limitation. This model introduces a tractable approximate posterior \( q (\phi^T | D^T, D^T; \psi) \), parameterized by \( \psi \). The same network generates the two variables to minimize the computational cost.

For consistency, we observe the whole testing dataset in the training stage and the other in the testing stage where it does not have the labels. Thus, the new approximate posterior is \( q (\phi^T | D^T; \psi) \), which can reduce the computational difficulty of the inference framework. Meanwhile, the lower bound of the log [43] evidence is valid. To make the log-likelihood only contain test samples with the appropriate scaling factor, this method modifies the objective function a little bit. Our excepted objective is as follows:

\[
L_{\theta, \psi} = \frac{N_T + M_T}{M_T}
\]
Inference network.

FIGURE 7. Graphical model (a) Generative process. (b) Inference. Inference network.

\[ \sum_{m=1}^{M_T} \mathbb{E}_q(\phi^T | D_T^c; \psi) \times \log p(\tilde{Y}_m^T | \tilde{X}_m^T, \phi_s^T; \theta) \]

\[ -KL \left[ q(\phi^T | D_T^c; \psi) \middle\| p(\phi^T) \right] \] (11)

\[ q(\phi^T | D_T^c; \psi) = \prod_c \mathbb{E}_q(\tilde{\omega}_c^T | D_T^c; \psi) \prod_i q(\tilde{z}_i^T | D_T^c; \psi) \] (12)

Follows univariate Gaussian having trainable mean and variance. Moreover, each dimension of prior \( p(\psi^T) \) is \( N(0, 1) \). Thus, KL-divergence is a very especially simple form from [41].

The final form objective with Monte-Carlo (MC) approximation is as follows:

\[ \min_{\theta, \psi} \frac{1}{M_T} \sum_{m=1}^{M_T} \frac{1}{S} \sum_{s=1}^{S} -\log p(\tilde{y}_m^T | \tilde{X}_m^T, \phi_s^T; \theta) \]

\[ + \frac{1}{N_T + M_T} KL \left[ q(\phi^T | D_T^c; \psi) \middle\| p(\phi^T) \right] \] (13)

To make gradient estimate \( \psi \) stable and unbiased, assuming the reparameterization trick for \( \phi^T \) [41]. For the sake of computational efficiency, \( S = 1 \) in MC. In the testing stage, set \( S = 10 \).

\[ p(\tilde{y}_m^T | \tilde{X}_m^T; \theta) = \mathbb{E}_q \left[ p(\tilde{y}_m^T | \tilde{X}_m^T, \phi_s^T; \theta) \right] \approx \frac{1}{S} \sum_{s=1}^{S} p(\tilde{y}_m^T | \tilde{X}_m^T, \phi_s^T; \theta), \phi_s^T \sim q(\phi^T | D_T^c; \psi) \] (14)

V. DATASET ENCODING

In modeling variational, this paper constructs the training dataset \( D_T^c \) into designated representation, which is very important. Some statistical methods (mean, var, and cardinality) can capture the statistics information to identify the class imbalance based on Dataset \( D_T^c \). Papers often use the sum-pooling or mean-pooling as an efficient set-encoder, where each instance is shared and then generates a vector summarizing the set. However, in our classification dataset, the traditional non-hierarchical structure will ignore the label information and perform poorly. Thus, this paper uses the two-layer hierarchy. The first is used to encode classes as a series of samples, then encodes the set as the set of classes. Therefore, in our work, a two-layer hierarchical set-encoder is introduced. From [26], the experiment shows that mean pooling has a limitation in that it does not compute the set number. It leads to a wrong impression of the imbalanced Dataset. Thus, our model takes the number of instances into the encoder. However, the set cardinality is not enough to capture important information about a dataset. As a supplement, this paper encodes the variance of the set as well.

In summary, this model calls the set encoder as StatisticsPooling (*), which generates a series of set statistics, mean, variance, and cardinality.

In Table 1, we talk about the effectiveness of the dataset encoding by using an ablation study. Mean and var represent the mean and variance of the data set, respectively. Moreover, \( N \) is the set cardinality. We can find that element-wise sample skewness and kurtosis hardly improve the performance, but the set cardinality is better.

\[ V_T = \text{StatisticsPooling}(\{NN_2(S_c)\}_{c=1}^C) \] (15)

\[ S_c = \text{StaticsPooling}(\{NN_1(x)\}_{x \in X_T^c}) \] (16)

where \( c = 1, \ldots, C \). \( X_T^c \) is the instances of class \( c \) in task \( T \). \( NN_1 \) and \( NN_2 \) are neural networks. \( V_T \) contains the dataset statistics information. The balancing variables \( \omega_c^T \) are generated from \( S_c \) (see Fig. 8.).

VI. LEARN TO BALANCE

In [11], it enlarges the definition of n-way-k-shot and the process of generating tasks. \( D \) contains train set \( D_T^c \) and a test set \( D_T^c = \{X_T^c, Y_T^c\} \). In each episode, this paper randomly chooses \( n \) classes, and \( k \) instances belong to these \( n \) classes. Here, we find two kinds of imbalances (class imbalance and class chosen imbalance). However, the existing few-shot approaches do not take these imbalances into account, preventing the model from efficiently solving realistic problems.

1) Class imbalance: The few-shot methods do not have any structure to handle class imbalance. Thus, the class with more samples may dominate the task-specific learning in the training stage, and the fewer shots classes have poor representation. (Class A has 500 samples; B class has ten samples)

| Encoding | Train / Test | Encoding |
|----------|--------------|----------|
| MiniImagenet/CUB | √ | Mean | 73.22 ±0.32 |
| Mean + Var. | 73.53 ±0.28 |
| Mean + Var. + Skew | 76.23 ±0.07 |
| Mean + Var. + Kurt. | 74.25 ±0.22 |
| Mean + Var. + N | 78.67 ±0.11 |

TABLE 1. Dataset encoding ablation study N: Set cardinality, skew: Skewness, kurt: Kurtosis.
2) Class chosen imbalance: In previous work [1], this model has learned that in each episode, the model randomly chose n-way-k-shot to construct tasks, which leads to some classes be chosen more times but others are fewer. When our Dataset is extensive, and there are many episodes, this imbalance will become more serious. The categories that are chosen more often learn more fully, and vice versa. In the end, there are always some categories whose selection times are far more significant than others, which will lead to insufficient training and affect the performance of classifiers.

**A. TACKLING IMBALANCE**

In view of the above two kinds of imbalanced problems, this paper proposes the following solutions.

1) Tackling class imbalance: Through the understanding of the above imbalance problem. For each inner-optimization step, we can set a specific learning rate for each category. For category \( c = 1, \ldots, C \), we use a set of class-specific scalars \( \omega_c^T = (\omega_{c,1}^T, \ldots, \omega_{c,C}^T) \in [0, 1]^C \), for each gradient descent process, the model can obtain the following parameters \( \nabla_{\theta_c} \mathcal{L}(\theta; D_c^T), \ldots, \nabla_{\theta_c} \mathcal{L}(\theta; D_c^T) \). When the sample size of a certain category is very small, \( \omega_{c,c}^T \) will be large.

2) Tackling class chosen imbalance: In order to control the selection times of the current task and the proportion of the iteration number of the whole model, we introduce task-rate learning rate \( \gamma^T = (\gamma_{1}^T, \ldots, \gamma_{C}^T) \in [0, 1]^C \), category \( c = 1, \ldots, C \), the scalars become \( (\gamma_{1}^T \alpha, \ldots, \gamma_{C}^T \alpha) \). The smaller the number of selected categories, the larger the task-rate. By controlling the relationship between variables and learning rate, the learning degree of the model is determined.

Finally, we assume that all the variables proposed are integrated into the original meta-learning framework, and the update rules are changed as follows:

\[
\theta_k = \theta_{k-1} - \gamma^T \alpha^T \sum_{c=1}^{C} \omega_{k,c}^T \nabla_{\theta_c} \mathcal{L}(\theta_{k-1}; D_c^T) 
\]

where \( \alpha \) is the global learning rate, \( \gamma^T \) is the multiplication operator. \( k = 1, \ldots, K \). The last step \( \theta_k \) corresponds to the task-specific predictor \( \theta^T \).

**B. GENERATING IMBALANCE VARIABLE**

As shown in Fig.8, our model introduces \( \gamma, \omega \) to reduce the impact of data imbalance. Firstly, each input is sent to the \( NN_1 \) network for processing. The structure of the \( NN_1 \) network shows in Fig.9. (a). The obtained results are sent to statistics pooling for processing (SP (statistics pooling) principle is shown in V. DATASET ENCODING), and then some useful statistical data, such as Mean, Var, and Set cardinality, are obtained. Then, the obtained results are processed by the \( NN_2 \) network (Fig.9. (b)), and the filtered statistical data are obtained by SP processing again. By using the original statistical data of Mean, Var, and SC, as well as the relevant data after filtering, the variables were generated \( \mu, \theta \). Processing global variables \( \theta \).

**VII. EXPERIMENTS**

In order to verify the effectiveness of our proposed model. Experiments are conducted on the following standard FSC datasets, miniImagenet, CUB, and tiredImagenet. We also use ablation experiments to verify the effectiveness of our proposed properties. At the end of this section, we show the result compared to previous state-of-the-art models.

**A. DATASETS**

MiniImagenet is part of the Imagenet dataset, which has 100 classes and 600 instances per class. Training dataset, testing dataset and validation dataset are divided respectively 64 classes, 16 classes, and 16 classes.

CUB contains 200 classes and 11,788 images divided into 100 train classes, 50 validation classes, and 50 novel classes.

tiredImagenet, similarly, it’s also a part of the Imagenet dataset. It is divided into three parts: train, test, and validation. For each part, it includes 193 classes, 196 classes plus 300 novel classes, and 196 classes plus 300 novel classes.
B. IMPLEMENTATION DETAILS

In our work, we use three kinds of feature extractors: a 4-layer convolution layer [9], [11] with 32 channels per layer, resnet-12 [34], and a WRN-28-10 [35]. The image size is resized to 84 \times 84.

1) IN THE TRAINING STAGE
we use SGD to optimize our model with a learning rate of 0.01 for 100 epochs. Furthermore, it is reduced by 0.001 per 10 times. If the model has more than thirty epochs and the performance has not been improved, the training will be terminated in advance.

2) FOR THE FINE-TUNING STAGE
As mentioned above, per episode, n classes and k samples per class are selected in the support set, and select 15 per class for the query set. We use SGD with a learning rate 0.001 to optimize the model. The rest is the same as the training stage. In addition, all conventional convolution layers in our network are replaced with a dynamic convolution layer.

3) FOR WRN (WIDE RESIDUAL NETWORKS)
we only use the simple data augmentation approaches [44] in these architectures.

C. IMBALANCE SETUP
To simulate natural imbalance conditions, we randomly chose 40 classes in the miniImagenet dataset, which are set the number of instances in each class to 100, and other classes still contain 600 instances.

D. THE EXPERIENCE RESULTS
In Table 2. Our model is compared with previous state-of-the-art models. Ours is applied to the miniImagenet and CUB with 1-shot, 5-shot. Params are shown in the second column, and the third and the fourth column show the accuracy result about 1-shot, 5-shot in the corresponding dataset.

This is in line with our conjecture that as a substitute for conventional convolution and dynamic convolution can significantly improve the network’s performance.

E. EXTRA RESULTS ABOUT A SIMILAR NETWORK
EPNet and our network are evaluated, the results obtained with tiredImagenet, 84 \times 84 images. EPNet uses a 12-layer residual network. MMNet is a Manifold mixup. The details about comparison are in TABLE 3.

F. SEMI-SCENARIOS
SSL with 100 unlabeled samples, *NET and EPCINet are the same but without EP (embedding propagation). As shown in Table 4, Table 5, Table 6.
TABLE 3. Comparison with the EPNET and manifold mixup (MMNet) on tired Imagenet. Moreover, accuracies were obtained with 84*84 images.

| backbone       | 1-shot | 5-shot |
|----------------|--------|--------|
| EPNet          | 67.18 ±0.02 | 74.65 ±0.07 |
| Ours           | 74.65 ±0.29 | 86.18 ±0.23 |
| MMNet          | 70.68 ±0.81 | 80.33 ±0.58 |
| Ours           | 77.17 ±0.02 | 82.25 ±0.87 |

TABLE 4. The results of SSL results with CONV-4.

| triedImagenet | CUB |
|---------------|-----|
| 1-shot        | 5-shot | 1-shot       | 5-shot |
| TPNSSL        | 52.78 ±32.66 | 68.81 ±23.73 | -        | -        |
| K-Means       | 50.70 ±31.13 | 63.11 ±22.22 | 51.67 ±12.68 | 70.30 ±14.22 |
| EPCINet(ours) | 61.25 ±26.76 | 76.20 ±26.26 | 53.31 ±0.01 | 63.11 ±0.02 |
| * NETSSL(ours)| 63.75 ±26.76 | 75.68 ±26.35 | -        | -        |
| EPCINetSSL(ours) | 66.32 ±26.76 | 76.98 ±26.36 | 67.36 ±0.01 | 78.91 ±0.01 |

TABLE 5. The results of SSL results with ResNet-12.

| triedImagenet | CUB |
|---------------|-----|
| 1-shot        | 5-shot | 1-shot       | 5-shot |
| LST           | 70.70 ±28.12 | 71.71 ±22.21 | 77.56 ±0.07 | 85.30 ±0.07 |
| EPCINet(ours) | 68.25 ±20.20 | 81.20 ±26.26 | 76.31 ±0.01 | 87.32 ±0.02 |
| * NETSSL(ours)| 74.42 ±20.62 | 82.68 ±26.15 | -        | -        |
| EPCINetSSL(ours) | 75.81 ±20.82 | 84.82 ±26.36 | 82.66 ±0.01 | 88.95 ±0.01 |

TABLE 6. The results of SSL results with wrn-28-10.

| triedImagenet | CUB |
|---------------|-----|
| 1-shot        | 5-shot | 1-shot       | 5-shot |
| K-Means       | 52.50 ±32.20 | 66.42 ±22.22 | 59.36 ±0.06 | 71.66 ±0.06 |
| EPCINet(ours) | 70.25 ±20.30 | 77.20 ±26.36 | 78.36 ±0.02 | 88.38 ±0.02 |
| * NETSSL(ours)| 77.75 ±20.66 | 86.68 ±26.32 | -        | -        |
| EPCINetSSL(ours) | 79.32 ±20.82 | 88.64 ±26.32 | 79.86 ±0.01 | 89.91 ±0.01 |

VIII. ABLATION STUDIES

To verify the effectiveness of our proposed methods. This paper applies ablation studies to investigate dynamic convolution, EP (embedding propagation), balance variables, and LP (label propagation) on 1-shot miniImagenet. As shown in Table 7.

A. ALGORITHM ABLATION

In Table 7, through ablation algorithm, this paper separately shows the improvement of image classification accuracy by dynamic convolution, LP (label propagation), EP (embedding propagation), and balance variables on 1-shot miniImagenet.

DC: Dynamic convolution, EXP: Episode. LP (label propagation), and EP (embedding propagation), and balance variable, BV1(class balance variable), BV2(class chosen imbalance variable).

Our model apply algorithm ablation to conv-4, 1-shot miniImagenet.

Interestingly, we can find that when our model leverages the dynamic convolution layer to replace the conventional convolution layer, the improvement is the biggest among the proposed methods in our model.

B. ALGORITHM ABLATION

BV1 is used to handle a class imbalance. Table 8 shows the accuracy varies with the degree of imbalance. We find that BV1-EPNet has a better result than EPNet. The improvement shows in Fig.10 demonstrates that our balance variable is influential, and BV1+EPNet learns the appropriate balance variable, which has better performance on 1-shot miniImagenet, using WRN-28-10.
In order to verify the effectiveness of BV1 and BV2, the EPCNet is used as the benchmark, and the ablation experiment is used for BV1 and BV2. For the sake of simplicity, we only use 500 episodes on 1-shot miniImagenet, using WRN-28-10. As shown in Fig.11., the experiment shows that when BV1 and BV2 are used for EPCNet respectively, the accuracy of the model is improved by about 3% and 4%.

**IX. CONCLUSION**

Our work proposes a novel few-shot classification model—EPCINet that handles the distribution limitations of shift few-shot learning by using EP and LP. Moreover, we have fully considered the imbalance problem in the real world: none of the existing few-shot classification methods takes this into account through learning balance variables to control the degree of imbalance. Furthermore, as far as we know, dynamic convolution is introduced into a few-shot classification for the first time. Empirically, we leverage ablation studies to prove the effectiveness of the above methods drawn from our model, and in the end, our model achieves the state-of-the-art results in a 1-shot and a 5-shot case in three standard few-shot classification datasets (miniImagenet, tiredlImageNet, CUB). We also have interests in few-shot tasks with multiple labels, which is a direction of our future efforts, and we are also trying to see if we can apply it to NLP or other fields.

**REFERENCES**

[1] H. Xu, C. Li, M. M. Rahaman, Y. Yao, Z. Li, J. Zhang, F. Kulwa, X. Zhao, S. Qi, and Y. Teng, “An enhanced framework of generative adversarial networks (EF-GANs) for environmental microorganism image augmentation with limited rotation-invariant training data,” IEEE Access, vol. 8, pp. 187455–187469, 2020, doi: 10.1109/ACCESS.2020.3031059.

[2] D. Xue, X. Zhou, C. Li, Y. Yao, M. M. Rahaman, J. Zhang, H. Chen, J. Zhang, S. Qi, and H. Sun, “An application of transfer learning and ensemble learning techniques for cervical histopathology image classification,” IEEE Access, vol. 8, pp. 104603–104618, 2020, doi: 10.1109/ACCESS.2020.2999816.

[3] Y. Cao, S. Ma, and H. Pan, “Bilevel-A: Fully convolutional scene text detection with text attention,” IEEE Access, vol. 8, pp. 155441–155449, 2020, doi: 10.1109/ACCESS.2020.3018784.

[4] M. Cordova, A. Pinto, H. Pedrini, and R. D. S. Torres, “Peele-text++: A tiny neural network for scene text detection,” IEEE Access, vol. 8, pp. 223172–223188, 2020, doi: 10.1109/ACCESS.2020.3043813.

[5] Y. Yang, Y. Li, R. Zhang, J. Wang, and Z. Miao, “Robust compare network for few-shot learning,” IEEE Access, vol. 8, pp. 137966–137974, 2020, doi: 10.1109/ACCESS.2020.3012720.

[6] Y. Fung, X. Tian, H. Wu, S. Gu, Z. Wang, F. Wang, J. Li, and Y. Weng, “Few-shot learning for Chinese legal controversial issues classification,” IEEE Access, vol. 8, pp. 75022–75034, 2020, doi: 10.1109/ACCESS.2020.2988493.

[7] X. Sun, B. Wang, Z. Wang, H. Li, H. Li, and K. Fu, “Research progress on few-shot learning for remote sensing image interpretation,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 8, pp. 187455–187469, 2020, doi: 10.1109/ACCESS.2020.3018784.

[8] X. Wang, B. Ma, Z. Yu, F. Li, and Y. Cai, “Multi-scale decision network for feature fusion and weighting for few-shot learning,” IEEE Access, vol. 8, pp. 92172–92181, 2020, doi: 10.1109/ACCESS.2020.2994805.

[9] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in Proc. Adv. Neural Inf. Process. Syst., Dec. 2017, pp. 4078–4088.

[10] F. Sung, Y. Yang, and L. Zhang, “Learning to compare: Relation network for few-shot learning Queen Mary University of London(639),” in Proc. CVPR, 2018, pp. 1199–1208.

[11] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, “Matching networks for one shot learning,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3637–3645.

[12] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, “Bilevel programming for hyperparameter optimization and meta-learning,” in Proc. 35th Int. Conf. Mach. Learn. (ICML), vol. 4, 2018, pp. 2537–2548.

[13] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in Proc. 34th Int. Conf. Mach. Learn. (ICML), vol. 3, 2017, pp. 1856–1866.

[14] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitriagkas, D. Lopez-Paz, and Y. Bengio, “Manifold mixup: Better representations by interpolating hidden states,” in Proc. 36th Int. Conf. Mach. Learn. (ICML), Jun. 2019, pp. 11196–11205.

[15] W. Y. Chen, Y. C. Liu, Z. Kira, Y. C. F. Wang, and J. B. Huang, “A closer look at few-shot classification,” in Proc. ICLR, 2019.

[16] H. Yao, Y. Wei, J. Huang, and Z. Li, “Hierarchically structured meta-learning,” in Proc. 36th Int. Conf. Mach. Learn. (ICML), Jun. 2019, pp. 12189–12209.

[17] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, “Dynamic convolution: Attention over convolution kernels,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2020, pp. 11027–11036, doi: 10.1109/CVPR42600.2020.01104.

[18] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-excitation networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 8, pp. 2011–2023, Aug. 2020, doi: 10.1109/TPAMI.2019.2913372.

[19] H.-U. Kim, Y. J. Koh, and C.-S. Kim, “Online multiple object tracking based on open-set few-shot learning,” IEEE Access, vol. 8, pp. 190312–190326, 2020, doi: 10.1109/ACCESS.2020.3032252.

[20] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, “Caltech-UCSD birds 200,” California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2010-001, 2010.

[21] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[22] A. Nichol and J. Schulman, “Reptile: A scalable metalearning algorithm,” 2018, arXiv:1803.02999. [Online]. Available: https://arxiv.org/abs/1803.02999.

[23] A. Nichol, J. Achiam, and J. Schulman, “On first-order meta-learning algorithms,” pp. 1–15, 2018, arXiv:1803.02999. [Online]. Available: http://arxiv.org/abs/1803.02999.
[24] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, “Online meta-learning,” in Proc. 36th Int. Conf. Mach. Learn. (ICML), Jun. 2019, pp. 3398–3410.

[25] A. Antoniou, A. Starkey, and H. Edwards, “How to train your MAML,” in Proc. 7th Int. Conf. Learn. Represent. (ICLR), 2019, pp. 1–11.

[26] Z. Wang, Y. Zhao, P. Yu, R. Zhang, and C. Chen, “Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks,” in Proc. ICLR, 2020.

[27] V. Garcia and J. Bruna, “Few-shot learning with graph neural networks,” in Proc. ICLR, 2017.

[28] L. Yang, L. Li, Z. Zhang, X. Zhou, E. Zhou, and Y. Liu, “DPGN: Distribution propagation graph network for few-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 13387–13396, doi: 10.1109/CVPR42600.2020.01340.

[29] Y. Hu, V. Gripon, and S. Pateux, “Exploiting unsupervised inputs for accurate few-shot classification,” 2020.

[30] S. Gidaris and N. Komodakis, “Generating classification weights with GNN denoising autoencoders for few-shot learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 21–30, doi: 10.1109/CVPR.2019.00011.

[31] M. Bauer, M. Rojas-Carulla, J. B. Swiątkowski, B. Schölkopf, and R. E. Turner, “Discriminative k-shot learning using probabilistic models,” 2017, arXiv:1706.00326. [Online]. Available: https://arxiv.org/abs/1706.00326

[32] P. Mangla, M. Singh, A. Sinha, N. Kumari, V. N. Balasubramanian, and B. Krishnamurthy, “Charting the right manifold: Manifold mixup for few-shot learning,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2020, pp. 2207–2216, doi: 10.1109/WACV45572.2020.0093338.

[33] Z. Yuan, W. Huang, L. Li, and X. Luo, “Few-shot scene classification with multi-attention deepemd network in remote sensing,” IEEE Access, vol. 9, pp. 19891–19901, 2021, doi: 10.1109/ACCESS.2020.3044192.

[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[35] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in Proc. Proceedings Brit. Mach. Vis. Conf., 2016, pp. 87.1–87.12, doi: 10.5244/C.30.87.

[36] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, and Y. Yang, “Learning to propagate labels: Transductive propagation network for few-shot learning,” in Proc. ICLR, 2019.

[37] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in Proc. Adv. Neural Inf. Process. Syst., vol. 17, 2006, pp. 321–328.

Wei Xiong was born in Heishan, Liaoning, China, in 1976. He is currently pursuing the Ph.D. degree. He is also currently a Senior Engineer. His main research interests include computer network technology, intelligent computing, and trusted cloud technology.

Yu Gong was born in Pingquan, Hebei, China, in 1997. He is currently pursuing the master’s degree. His research interests include intelligent computing, target detection, and few-shot learning.

Weihua Niu was born in Tianjin, China, in 1978. Her main research interests include digital image processing and computer vision.

Ruonan Wang was born in Hebei, China, in 1996. She is currently pursuing the master’s degree. Her research interests include intelligent computing and target detection.