DPGN: Distribution Propagation Graph Network for Few-shot Learning

Ling Yang[1]* Liangliang Li[2]*† Zilun Zhang[2] Xinyu Zhou[2] Erjin Zhou[2] Yu Liu[2]
Northwestern Polytechnical University[1] Megvii Technology[2]
yangling0818@163.com, liliangliang@megvii.com
zilun.zhang@mail.utoronto.ca, {zxy, zej, liuyu}@megvii.com

Abstract

Most graph-network-based meta-learning approaches model instance-level relation of examples. We extend this idea further to explicitly model the distribution-level relation of one example to all other examples in a 1-vs-N manner. We propose a novel approach named distribution propagation graph network (DPGN) for few-shot learning. It conveys both the distribution-level relations and instance-level relations in each few-shot learning task. To combine the distribution-level relations and instance-level relations for all examples, we construct a dual complete graph network which consists of a point graph and a distribution graph with each node standing for an example. Equipped with dual graph architecture, DPGN propagates label information from labeled examples to unlabeled examples within several update generations. In extensive experiments on few-shot learning benchmarks, DPGN outperforms state-of-the-art results by a large margin in 5% ∼ 12% under supervised settings and 7% ∼ 13% under semi-supervised settings. Code is available at https://github.com/megvii-research/DPGN

1. Introduction

The success of deep learning is rooted in a large amount of labeled data [19, 38], while humans generalize well after having seen few examples. The contradiction between these two facts brings great attention to the research of few-shot learning [7, 20]. Few-shot learning task aims at predicting unlabeled data (query set) given a few labeled data (support set).

Fine-tuning [4] is the defacto method in obtaining a predictive model from a small training dataset in practice nowadays. However, it suffers from overfitting issues [11]. Meta-learning [8] methods introduces the concept of *episode* to address the few-shot problem explicitly.

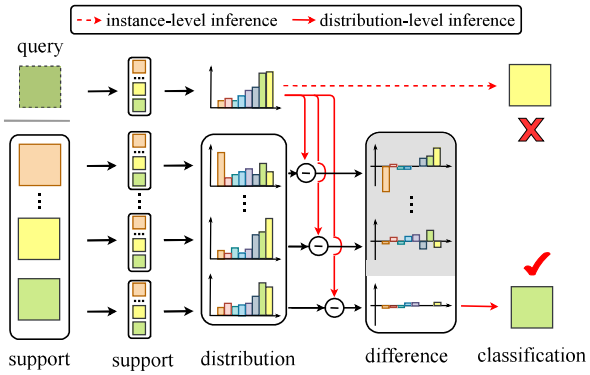

Figure 1: Our proposed DPGN adopts contrastive comparisons between each sample with support samples to produce distribution representation. Then it incorporates distribution-level comparisons with instance-level comparisons when classifying the query sample.

An episode is one round of model training, where in each episode, only few examples (e.g., 1 or 5) are randomly sampled from each class in training data. Meta-learning methods adopt a trainer (also called meta-learner) which takes the few-shot training data and outputs a classifier. This process is called *episodic training* [41]. Under the framework of meta-learning, a diverse hypothesis was made to build an efficient meta-learner.

A rising trend in recent researches was to process the training data with Graph Networks [2], which is a powerful model that generalizes many data structures (list, trees) while introduces a combinatorial prior over data. Few-Shot GNN [10] is proposed to build a complete graph network where each node feature is concatenated with the corresponding class label, then node features are updated via the attention mechanism of graph network to propagate the label information. To further exploit intra-cluster similarity and inter-cluster dissimilarity in the graph-based network, EGNN [18] demonstrates an edge-labeling graph neural network under the *episodic training* framework. It is noted that previous GNN studies in few-shot learning mainly fo-

*Contributed equally.
†Corresponding author.

cused on pair-wise relations like node labeling or edge labeling, and ignored a large number of substantial distribution relations. Additionally, other meta-learning approaches claim to make use of the benefits of global relations by *episodic training*, but in an implicitly way.

As illustrated in Figure 1, firstly, we extract the *instance feature* of support and query samples. Then, we obtain the *distribution feature* for each sample by calculating the instance-level similarity over all support samples. To leverage both instance-level and distribution-level representation of each example and process the representations at different levels independently, we propose a dual-graph architecture: a point graph (PG) and a distribution graph (DG). Specifically, a PG generates a DG by gathering 1-vs-n relation on every example, while the DG refines the PG by delivering distribution relations between each pair of examples. Such cyclic transformation adequately fuses instance-level and distribution-level relations and multiple generations (rounds) of this Gather-Compare process concludes our approach. Furthermore, it is easy to extend DPGN to semi-supervised few-shot learning task where support set containing both labeled and unlabeled samples for each class. DPGN builds a bridge connection between labeled and unlabeled samples in the form of similarity distribution, which leads to a better propagation for label information in semi-supervised few-shot classification.

Our main contributions are summarized as follows:

- To the best of our knowledge, DPGN is the first to **explicitly incorporate distribution propagation** in graph network for few-shot learning. The further ablation studies have demonstrated the effectiveness of distribution relations.
- We devise the **dual complete graph network** that combines instance-level and distribution-level relations. The cyclic update policy in this framework contributes to enhancing instance features with distribution information.
- Extensive experiments are conducted on four popular benchmark datasets for few-shot learning. By comparing with all state-of-the-art methods, the DPGN achieves a significant improvement of **5%∼12%** on average in few-shot classification accuracy. In semi-supervised tasks, our algorithm outperforms existing graph-based few-shot learning methods by **7%∼13 %**.

2. Related Work

2.1. Graph Neural Network

Graph neural networks were first designed for tasks on processing graph-structured data [34, 41]. Graph neural networks mainly refine the node representations by aggregating and transforming neighboring nodes recursively. Recent approaches [10, 25, 18] are proposed to exploit GNN in the field of few-shot learning task. TPN [25] brings the transductive setting into graph-based few-shot learning, which performs a Laplacian matrix to propagate labels from support set to query set in the graph. It also considers the similarity between support and query samples through the process of pairwise node features affinities to propagate labels. EGNN [18] uses the similarity/dissimilarity between samples and dynamically update both node and edge features for complicated interactions.

2.2. Metric Learning

Another category of few-shot learning approaches focus on optimizing feature embeddings of input data using metric learning methods. Matching Networks [41] produces a weighted nearest neighbor classifier through computing embedding distance between support and query set. Prototypical Networks [36] firstly build a prototype representation of each class in the embedding space. As an extension of Prototypical Networks, IMF [1] constructs infinite mixture prototypes by self-adaptation. RelationNet [40] adopts a distance metric network to learn pointwise relations in support and query samples.

2.3. Distribution Learning

Distribution Learning theory was first introduced in [17] to find an efficient algorithm that determines the distribution from which the samples are drawn. Different methods [16, 5, 6] are proposed to efficiently estimate the target distributions. DLDL [9] is one of the researches that has assigned the discrete distribution instead of one-hot label for each instance in classification and regression tasks. CPNN [44] takes both features and labels as the inputs and produces the label distribution with only one hidden layer in its framework. LDLFs [35] devises a distribution learning method based on the decision tree algorithm.

2.4. Meta Learning

Some few-shot approaches adopt a meta-learning framework that learns meta-level knowledge across batches of tasks. MAML [8] are gradient-based approaches that design the meta-learner as an optimizer that could learn to update the model parameters (e.g., all layers of a deep network) within few optimization steps given novel examples. Reptile [28] simplifies the computation of meta-loss by incorporating an L2 loss which updates the meta-model parameters towards the instance-specific adapted models. SNAIL [27] learn a parameterized predictor to estimate the parameters in models. MetaOptNet [21] advocates the use of linear classifier instead of nearest-neighbor methods which can be optimized as convex learning problems. LEO [33] utilizes an encoder-decoder architecture to mine the latent genera-

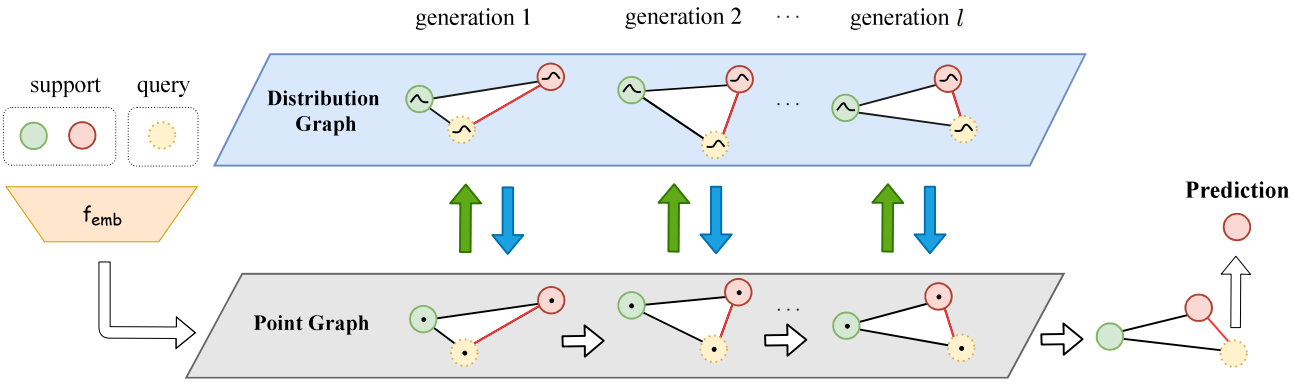

Figure 2: The overall framework of DPGN. In this illustration, we take a 2way-1shot task as an example. The support and query embeddings obtained from feature extractor are delivered to the dual complete graph (a point graph and a distribution graph) for transductive propagation generation after generation. The **green arrow** represents a edge-to-node transformation (P2D, described in Section 3.2.1) which aggregates instance similarities to construct distribution representations and the **blue arrow** represents another edge-to-node transformation (D2P, described in Section 3.2.2) which aggregates distribution similarities with instance features. DPGN makes the prediction for the query sample at the end of generation l.

tive representations and predicts high-dimensional parameters in extreme low-data regimes.

3. Method

In this section, we first provide the background of few-shot learning task, then introduce the proposed algorithm in detail.

3.1. Problem Definition

The goal of few-shot learning tasks is to train a model that can perform well in the case where only few samples are given.

Each few-shot task has a *support set* $\mathcal{S}$ and a *query set* $\mathcal{Q}$. Given training data $\mathbb{D}^{train}$, the support set $\mathcal{S} \subset \mathbb{D}^{train}$ contains N classes with K samples for each class (i.e., the N-way K-shot setting), it can be denoted as $\mathcal{S} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{N\times K}, y_{N\times K})\}$. The *query* set $\mathcal{Q} \subset \mathbb{D}^{train}$ has $\bar{T}$ samples and can be denoted as $\mathcal{Q} = \{(x_{N\times K+1}, y_{N\times K+1}), \ldots, (x_{N\times K+\bar{T}}, y_{N\times K+\bar{T}})\}$. Specifically, in the training stage, data labels are provided for both support set $\mathcal{S}$ and query set $\mathcal{Q}$. Given testing data $\mathbb{D}^{test}$, our goal is to train a classifier that can map the query sample from $\mathcal{Q} \in \mathbb{D}^{test}$ to the corresponding label accurately with few support samples from $\mathcal{S} \in \mathbb{D}^{test}$. Labels of support sets and query sets are mutually exclusive.

3.2. Distribution Propagation Graph Networks

In this section, we will explain the DPGN that we proposed for few-shot learning in detail. As shown in Figure 2. The DPGN consists of l generations and each generation consists of a point graph $G_l^p = (V_l^p, E_l^p)$ and a distribution graph $G_l^d = (V_l^d, E_l^d)$. Firstly, the feature embeddings of all samples are extracted by a convolutional backbone, these embeddings are used to compute the instance similarities E_l^p. Secondly, the instance relations E_l^p are delivered to construct the distribution graph G_l^d. The node features V_l^d are initialized by aggregating E_l^p following the position order in G_l^p and the edge features E_l^d stand for the distribution similarities between the node features V_l^d. Finally, the obtained E_l^d is delivered to G_l^p for constructing more discriminative representations of nodes V_l^p and we repeat the above procedure generation by generation. A brief introduction of generation update for the DPGN can be expressed as $E_l^p \rightarrow V_l^d \rightarrow E_l^d \rightarrow V_l^p \rightarrow E_{l+1}^p$, where l denotes the l-th generation.

For further explanation, we formulate V_l^p, E_l^p, V_l^d and E_l^d as follows: $V_l^p = \{v_{l,i}^p\}$, $E_l^p = \{e_{l,ij}^p\}$, $V_l^d = \{v_{l,i}^d\}$, $E_l^d = \{e_{l,ij}^d\}$ where $i, j = 1, \cdots T$. $T = N \times K + \bar{T}$ denotes the total number of examples in a training episode. $v_{0,i}^p$ is first initialized by the output of the feature extractor f_{emb}. For each sample x_i:

$$v_{0,i}^p = f_{emb}(x_i)\,, \tag{1}$$

where $v_{0,i}^p \in \mathbb{R}^m$ and m denotes the dimension of the feature embedding.

3.2.1 Point-to-Distribution Aggregation

Point Similarity Each edge in the point graph stands for the instance (point) similarity and the edge $e_{0,ij}^p$ of the first generation is initialized as follows:

$$e_{0,ij}^p = f_{e_0^p}((v_{0,i}^p - v_{0,j}^p)^2)\,, \tag{2}$$

where $e_{0,ij}^p \in \mathbb{R}$. $f_{e_0^p} : \mathbb{R}^m \rightarrow \mathbb{R}$ is the encoding network that transforms the instance similarity to a certain scale. $f_{e_0^p}$

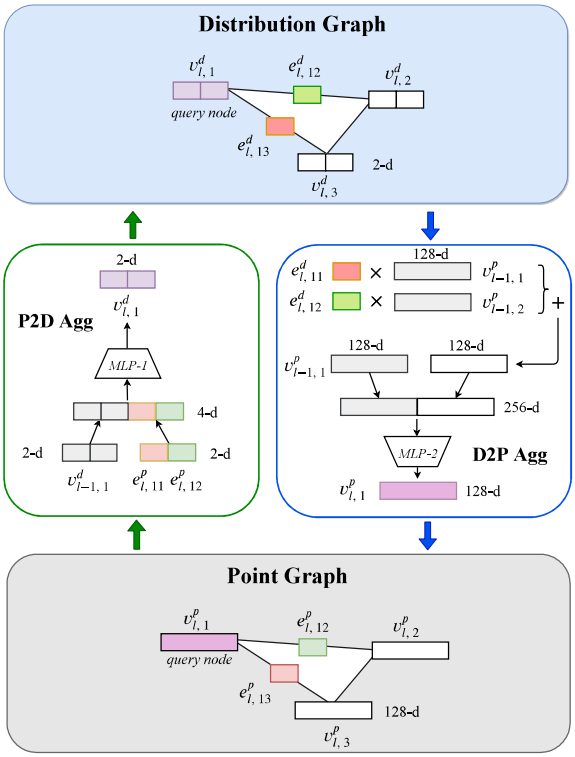

Figure 3: Details about **P2D aggregation** and **D2P aggregation** in DPGN. A 2way-1shot task is presented as an example. MLP-1 is the FC-ReLU blocks mentioned in P2D Aggregation and MLP-2 is the Conv-BN-ReLU blocks mentioned in D2P Aggregation. The green arrow denotes the P2D aggregation while the blue arrow denotes the D2P aggregation. Both aggregation processes integrate the node or edge features of their previous generation.

contains two Conv-BN-ReLU [13, 15] blocks with the parameter set $\theta_{e_0^p}$ and a sigmoid layer.

For generation $l > 0$, given $e^p_{l-1,ij}$, $v^p_{l-1,i}$ and $v^p_{l-1,j}$, $e^p_{l,ij}$ can be updated as follows:

$$e^p_{l,ij} = f_{e^p_l}((v^p_{l-1,i} - v^p_{l-1,j})^2) \cdot e^p_{l-1,ij}\,. \tag{3}$$

In order to use edge information with a holistic view of the graph G^p_l, a normalization operation is conducted on the $e^p_{l,ij}$.

P2D Aggregation After edge features E^p_l in point graph G^p_l are produced or updated, the distribution graph $G^d_l = (V^d_l, E^d_l)$ is the next to be constructed. As shown in Figure 3, G^d_l aims at integrating instance relations from the point graph G^p_l and process the *distribution-level* relations. Each distribution feature $v^d_{l,i}$ in G^d_l is a NK dimension feature vector where the value in j-th entry represents the relation between sample x_i and sample x_j and NK stands for the total number of support samples in a task. For first initialization:

$$v^d_{0,i} = \begin{cases} ||_{j=1}^{NK}\, \delta(y_i, y_j) & \text{if } x_i \text{ is labeled,} \\ [\frac{1}{NK}, \cdots, \frac{1}{NK}] & \text{otherwise,} \end{cases} \tag{4}$$

where $v^d_{0,i} \in \mathbb{R}^{NK}$ and $||$ is the concatenation operator. $\delta(\cdot)$ is the Kronecker delta function which outputs one when $y_i = y_j$ and zero otherwise (y_i and y_j are labels).

For generations $l > 0$, the distribution node $v^d_{l,i}$ can be updated as follows:

$$v^d_{l,i} = \text{P2D}(\, ||_{j=1}^{NK}\, e^p_{l,ij}\,,\, v^d_{l-1,i})\,, \tag{5}$$

where $\text{P2D} : (\mathbb{R}^{NK}, \mathbb{R}^{NK}) \to \mathbb{R}^{NK}$ is the aggregation network for distribution graph. P2D applies a concatenation operation between two features. Then, P2D performs a transformation : $\mathbb{R}^{2NK} \to \mathbb{R}^{NK}$ on the concatenated features which is composed of a fully-connected layer and ReLU [13], with the parameter set $\theta_{v^d_l}$.

3.2.2 Distribution-to-Point Aggregation

Distribution Similarity Each edge in distribution graph stands for the similarity between distribution features of different samples. For generation $l = 0$, the distribution similarity $e^d_{0,ij}$ is initialized as follows:

$$e^d_{0,ij} = f_{e^d_0}((v^d_{0,i} - v^d_{0,j})^2)\,, \tag{6}$$

where $e^d_{0,ij} \in \mathbb{R}$. The encoding network $f_{e^d_0} : \mathbb{R}^{NK} \to \mathbb{R}$ transforms the distribution similarity using two Conv-BN-ReLU blocks with the parameter set $\theta_{e^d_0}$ and a sigmoid layer in the end. For generation $l > 0$, the update rule for $e^d_{l,ij}$ in G^d_l is formulated as follows:

$$e^d_{l,ij} = f_{e^d_l}((v^d_{l,i} - v^d_{l,j})^2) \cdot e^d_{l-1,ij}\,. \tag{7}$$

Also, we apply a normalization to $e^d_{l,ij}$.

D2P Aggregation As illustrated in Figure 3, the encoded distribution information in G^d_l flows back into the point graph G^p_l at the end of each generation. Then node features $v^p_{l,i}$ in G^p_l captures the distribution relations through aggregating all the node features in G^p_l with edge features $e^d_{l,i}$ as follows:

$$v^p_{l,i} = \text{D2P}(\sum_{j=1}^{T}(e^d_{l,ij} \cdot v^p_{l-1,j}), v^p_{l-1,i})\,, \tag{8}$$

where $v^p_{l,i} \in \mathbb{R}^m$ and $\text{D2P} : (\mathbb{R}^m, \mathbb{R}^m) \to \mathbb{R}^m$ is the aggregation network for point graph in G^p_l with the parameter set

$\theta_{v_l^p}$. D2P concatenates the feature which is computed by $\sum_{j=1}^{T}(e_{l,ij}^d \cdot v_{l-1,j}^p)$ with the node features $v_{l-1,i}^p$ in previous generation and update the concatenated feature with two Conv-BN-ReLU blocks. After this process, the node features can integrate the distribution-level information into the instance-level feature and prepares for computing instance similarities in the next generation.

3.3. Objective

The class prediction of each node can be computed by feeding the corresponding edges in the final generation l of DPGN into softmax function:

$$P(\hat{y}_i|x_i) = \text{Softmax}(\sum_{j=1}^{NK} e_{l,ij}^p \cdot \textit{one-hot}(y_j)) \,, \quad (9)$$

where $P(\hat{y}_i|x_i)$ is the probability distribution over classes given sample x_i, and y_j is the label of jth sample in the support set. $e_{l,ij}^p$ stands for the edge feature in the point graph at the final generation.

Point Loss It is noted that we make classification predictions in the point graph for each sample. Therefore, *the point loss* at generation l is defined as follows:

$$\mathcal{L}_l^p = \mathcal{L}_{CE}(P(\hat{y}_i|x_i), y_i) \,, \quad (10)$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss function, T stands for the number of samples in each task $(S, Q) \in D^{train}$. $P(\hat{y}_i|x_i)$ and y_i are model probability predictions of sample x_i and the ground-truth label respectively.

Distribution Loss To facilitate the training process and learn discriminative distribution features , we incorporate the *distribution loss* which plays a significant role in contributing to faster and better convergence. We define the *distribution loss* for generation l as follows:

$$\mathcal{L}_l^d = \mathcal{L}_{CE}(\text{Softmax}(\sum_{j=1}^{NK} e_{l,ij}^d \cdot \textit{one-hot}(y_j)), y_i) \,, \quad (11)$$

where $e_{l,ij}^d$ stands for the edge feature in the distribution graph at generation l.

The total objective function is a weighted summation of all the losses mentioned above:

$$\mathcal{L} = \sum_{l=1}^{\hat{l}} (\lambda_p \mathcal{L}_l^p + \lambda_d \mathcal{L}_l^d) \,, \quad (12)$$

where $\hat{l}$ denotes total generations of DPGN and the weights λ_p and λ_d of each loss are set to balance their importance. In most of our experiments, λ_p and λ_d are set to 1.0 and 0.1 respectively.

4. Experiments

4.1. Datasets and Setups

4.1.1 Datesets

We evaluate DPGN on four standard few-shot learning benchmarks: *mini*ImageNet [41], *tiered*ImageNet [31], CUB-200-2011 [42] and CIFAR-FS [3]. The *mini*ImageNet and *tiered*ImageNet are the subsets of ImageNet [32]. CUB-200-2011 is initially designed for fine-grained classification and CIFAR-FS is a subset of CIFAR-100 for few-shot classification. As shown in Table 1, we list details for images number, classes number, images resolution and train/val/test splits following the criteria of previous works [41, 31, 4, 3].

Table 1: Details for few-shot learning benchmarks.

Dataset	Images	Classes	Train-val-test	Resolution
*mini*ImageNet	60000	100	64/16/20	84x84
*tiered*ImageNet	779165	608	351/97/160	84x84
CUB-200-2011	11788	200	100/50/50	84x84
CIFAR-FS	60000	100	64/16/20	32x32

4.1.2 Experiment Setups

Network Architecture We use four popular networks for fair comparison, which are ConvNet, ResNet12, ResNet18 and WRN that are used in EGNN [18], MetaOptNet [21], CloserLook [4] and LEO [33] respectively. ConvNet mainly consists of four Conv-BN-ReLU blocks. The last two blocks also contain a dropout layer [37]. ResNet12 and ResNet18 are the same as the one described in [14]. They mainly have four blocks, which include one residual block for ResNet12 and two residual blocks for ResNet18 respectively. WRN was firstly proposed in [46]. It mainly has three residual blocks and the depth of the network is set to 28 as in [33]. The last features of all backbone networks are processed by a global average pooling, then followed by a fully-connected layer with batch normalization [15] to obtain a 128-dimensions instance embedding.

Training Schema We perform data augmentation before training, such as horizontal flip, random crop, and color jitter (brightness, contrast, and saturation), which are mentioned in [11, 43]. We randomly sample 28 meta-task episodes in each iteration for meta-training. The Adam optimizer is used in all experiments with the initial learning rate of 10^{-3}. We decay the learning rate by 0.1 per 15000 iterations and set the weight decay to 10^{-5}.

Evaluation Protocols We evaluate DPGN in 5way-1shot/5shot settings on standard few-shot learning datasets,

*mini*ImageNet, *tiered*ImageNet, CUB-200-2011 and CIFAR-FS. We follow the evaluation process of previous approaches [18, 33, 43]. We randomly sample 10,000 tasks then report the mean accuracy (in %) as well as the 95% confidence interval.

4.2. Experiment Results

Main Results We compare the performance of DPGN with several state-of-the-art models including graph and non-graph methods. For fair comparisons, we employ DPGN on *mini*ImageNet, *tiered*ImageNet, CIFAR-FS and CUB-200-2011 datasets, which is compared with other methods in the same backbones. As shown in Table 2, 3 and 4, the proposed DPGN is superior to other existing methods and achieves the state-of-the-art performance, especially compared with the graph-based methods.

Table 2: Few-shot classification accuracies on *miniImageNet*. † denotes thatit is implemented by public code. [10, 25, 18] and DPGN are tested in transduction.

Method	Backbone	5way-1shot	5way-5shot
MatchingNet [41]	ConvNet	43.56±0.84	55.31± 0.73
ProtoNet [36]	ConvNet	49.42±0.78	68.20±0.66
RelationNet [40]	ConvNet	50.44±0.82	65.32±0.70
R2D2 [3]	ConvNet	51.20±0.60	68.20±0.60
MAML [8]	ConvNet	48.70±1.84	55.31±0.73
Dynamic [11]	ConvNet	56.20±0.86	71.94±0.57
GNN [10]	ConvNet	50.33±0.36	66.41±0.63
TPN [25]	ConvNet	55.51±0.86	69.86±0.65
Global [26]	ConvNet	53.21±0.40	72.34±0.32
Edge-label [18]	ConvNet	59.63±0.52†	76.34±0.48
DPGN	**ConvNet**	**66.01±0.36**	**82.83±0.41**
LEO [33]	WRN	61.76±0.08	77.59±0.12
wDAE [12]	WRN	61.07±0.15	76.75±0.11
DPGN	**WRN**	**67.24±0.51**	**83.72±0.44**
CloserLook [4]	ResNet18	51.75±0.80	74.27±0.63
CTM [22]	ResNet18	62.05±0.55	78.63±0.06
DPGN	**ResNet18**	**66.63±0.51**	**84.07±0.42**
MetaGAN [47]	ResNet12	52.71±0.64	68.63±0.67
SNAIL [27]	ResNet12	55.71±0.99	68.88±0.92
TADAM [29]	ResNet12	58.50±0.30	76.70±0.30
Shot-Free [30]	ResNet12	59.04±0.43	77.64±0.39
Meta-Transfer [39]	ResNet12	61.20±1.80	75.53±0.80
FEAT [43]	ResNet12	62.96±0.02	78.49±0.02
TapNet [45]	ResNet12	61.65±0.15	76.36±0.10
Dense [24]	ResNet12	62.53±0.19	78.95±0.13
MetaOptNet [21]	ResNet12	62.64±0.61	78.63±0.46
DPGN	**ResNet12**	**67.77±0.32**	**84.60±0.43**

Semi-supervised Few-shot Learning We employ DPGN on semi-supervised few-shot learning. Following [25, 18], we use the same criteria to split *mini*ImageNet dataset into labeled and unlabeled parts with different ratios. For a 20% labeled semi-supervised scenario, we split the support samples with a ratio of 0.2/0.8 for labeled and unlabeled data in each class. In semi-supervised few-shot learning, DPGN uses unlabeled support samples to explicitly construct similarity distributions over all other samples and the distributions work as a connection between queries and labeled sup-

Table 3: Few-shot classification accuracies on *tieredImageNet*. † denotes that it is implemented by public code. * denotes that it is reported from [21]. [25, 18] and DPGN are tested in transduction.

Method	backbone	5way-1shot	5way-5shot
MAML* [8]	ConvNet	51.67±1.81	70.30±1.75
ProtoNet* [36]	ConvNet	53.34±0.89	72.69±0.74
RelationNet* [40]	ConvNet	54.48±0.93	71.32±0.78
TPN [25]	ConvNet	59.91±0.94	73.30±0.75
Edge-label [18]	ConvNet	63.52±0.52†	80.24±0.49
DPGN	**ConvNet**	**69.43±0.49**	**85.92±0.42**
CTM [22]	ResNet18	64.78±0.11	81.05±0.52
DPGN	**ResNet18**	**70.46±0.52**	**86.44±0.41**
TapNet [45]	ResNet12	63.08±0.15	80.26±0.12
Meta-Transfer [39]	ResNet12	65.62±1.80†	80.61±0.90†
MetaOptNet [21]	ResNet12	65.81±0.74	81.75±0.53
Shot-Free [30]	ResNet12	66.87±0.43	82.64±0.39
DPGN	**ResNet12**	**72.45±0.51**	**87.24±0.39**

Table 4: Few-shot classification accuracies on CUB-200-2011 and CIFAR-FS. * denotes that it is reported from [21] or [4]. DPGN are tested in transduction.

Method	backbone	**CUB-200-2011**	
		5way-1shot	5way-5shot
ProtoNet* [36]	ConvNet	51.31±0.91	70.77±0.69
MAML* [8]	ConvNet	55.92±0.95	72.09±0.76
MatchingNet* [41]	ConvNet	61.16±0.89	72.86±0.70
RelationNet* [40]	ConvNet	62.45±0.98	76.11±0.69
CloserLook [4]	ConvNet	60.53±0.83	79.34±0.61
DN4 [23]	ConvNet	53.15±0.84	81.90±0.60
DPGN	**ConvNet**	**76.05±0.51**	**89.08±0.38**
FEAT [43]	ResNet12	68.87±0.22	82.90±0.15
DPGN	**ResNet12**	**75.71±0.47**	**91.48±0.33**
Method	backbone	**CIFAR-FS**	
		5way-1shot	5way-5shot
ProtoNet* [36]	ConvNet	55.5±0.7	72.0±0.6
MAML* [8]	ConvNet	58.9±1.9	71.5±1.0
RelationNet* [40]	ConvNet	55.0±1.0	69.3±0.8
R2D2 [3]	ConvNet	65.3±0.2	79.4±0.1
DPGN	**ConvNet**	**76.4±0.5**	**88.4±0.4**
Shot-Free [30]	ResNet12	69.2±0.4	84.7±0.4
MetaOptNet [21]	ResNet12	72.0±0.7	84.2±0.5
DPGN	**ResNet12**	**77.9±0.5**	**90.2±0.4**

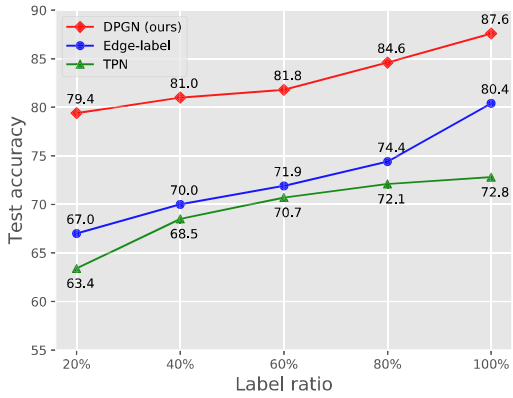

Figure 4: Semi-supervised few-shot learning accuracy in 5way-10shot on *mini*ImageNet. DPGN surpass TPN and EGNN by a large margin consistently.

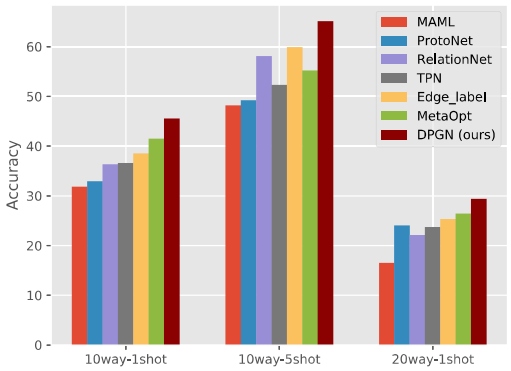

Figure 5: High-way few-shot classification accuracies on *mini*ImageNet.

port samples, which could propagate label information from labeled samples to queries sufficiently.

Table 5: Trasductive/non-transductive experiments on *mini*ImageNet. "BN" means information is shared among test examples using batch normalization. † denotes that it is implemented by public code released by authors.

Method	Transduction	5way-5shot
Reptile [28]	No	62.74
GNN [10]	No	66.41
Edge-label [18]	No	66.85†
DPGN	**No**	**72.83**
MAML [8]	BN	63.11
Reptile [28]	BN	65.99
RelationNet [40]	BN	67.07
MAML [8]	Yes	66.19
TPN [25]	Yes	69.86
Edge-label [18]	Yes	76.37
DPGN	**Yes**	**84.62**

In Figure 4, DPGN shows the superiority to exsisting semi-supervised few-shot methods and the result demonstrates the effectiveness to exploit the relations between labeled and unlabeled data when the label ratio decreases. Notably, DPGN surpasses TPN [25] and EGNN [18] by $11\% \sim 16\%$ and $7\% \sim 13\%$ respectively in few-shot average classification accuracy on *mini*ImageNet.

Transductive Propagation To validate the effectiveness of the transductive setting in our framework, we conduct the transductive and non-transductive experiments on *mini*ImageNet dataset in 5way-5shot setting. Table 5 shows that the accuracy of DPGN increases by a large margin in the transductive setting (comparing with non-transductive setting). Compared to TPN and EGNN which consider instance-level features only, DPGN utilizes distribution similarities between query samples and adopts dual graph architecture to propagate label information in a sufficient way.

High-way classification Furthermore, the performance of DPGN in high-way few-shot scenarios is evaluated on *mini*ImageNet dataset and its results are shown in Figure 5. The observed results show that DPGN not only exceeds the powerful graph-based methods [25, 18] but also surpasses the state-of-the-art non-graph methods significantly. As the number of ways increasing in few-shot tasks, it can broaden the horizons of distribution utilization and make it possible for DPGN to collect more abundant distribution-level information for queries.

4.3. Ablation Studies

Impact of Distribution Graph The distribution graph G_l^d works as an important component of DPGN by propagating distribution information, so it is necessary to investigate the effectiveness of G_l^d quantitatively. We design the experiment by limiting the distribution similarities which flow to G_l^p for performing aggregation in each generation during the inference process. Specifically, we mask out the edge features E_l^d through keeping a different number of feature dimensions and set the value of rest dimensions to zero, since zero gives no contribution. Figure 6 shows the result for our experiment in 5way-1shot on *mini*ImageNet. It is obvious that test accuracy and the number of feature dimensions kept in E_l^d have positive correlations and accuracy increment (area in blue) decreases with more feature dimen-

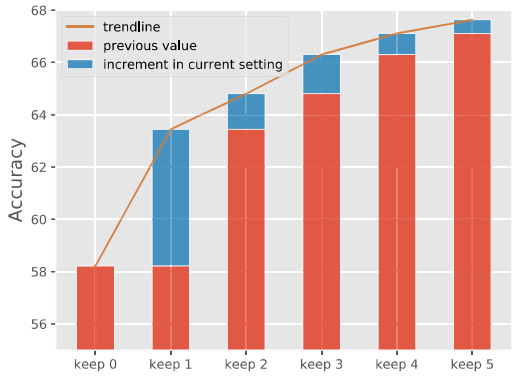

Figure 6: Effectiveness of G_l^d through keeping n dimensions in 5way-1shot on *mini*ImageNet.

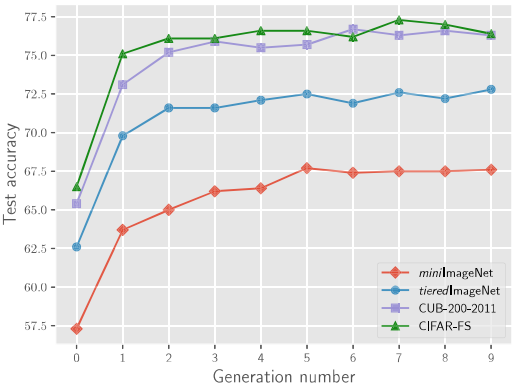

Figure 7: Generation number in DPGN on *mini*ImageNet, *tiered*ImageNet, CUB-200-2011 and CIFAR-FS.

sions. Keeping dimensions from 0 to 5, DPGN boosts the performance nearly by 10% in absolute value and the result shows that the distribution graph has a great impact on our framework.

Generation Numbers DPGN has a cyclic architecture that includes point graph and distribution graph, each graph has node-update and edge-update modules respectively. The total number of generations is an important ingredient for DPGN, so we perform experiments to obtain the trend of test accuracy with different generation numbers in DPGN on *mini*ImageNet, *tiered*ImageNet, CUB-200-2011, and CIFAR-FS. In Figure 7, with the generation number changing from 0 to 1, the test accuracy has a significant rise. When the generation number changes from 1 to 10, the test accuracy increases by a small margin and the curve becomes to fluctuate in the last several generations. Considering that more generations need more iterations to converge, we choose generation 6 as a trade-off between the test accuracy and convergence time. Additionally, to visualize the procedure of cyclic update, we choose a test scenario where the ground truth classes of five query images are [1, 2, 3, 4, 5] and visualize instance-level similarities which is used for predictions of five query samples as shown in Figure 8. The heatmap shows DPGN refines the instance-level similarity matrix after several generations and makes the right predictions for five query samples in the final generation. Notably, DPGN not only contributes to predicting more accurately but also enlarge the similarity distances between the samples in different classes through making instance features more discriminative, which cleans the prediction heatmap.

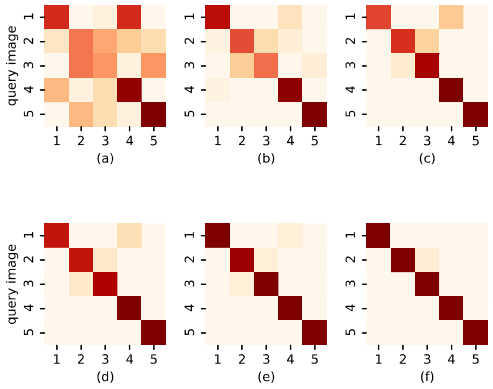

Figure 8: The visualization of edge prediction in each generation of DPGN. (a) to (f) denotes generation 1 to 6. The dark denotes higher score and the shallow denotes lower confidence. The left axis stands for the index of 5 query images and the bottom axis stands for 5 support class.

5. Conclusion

In this paper, we have presented the Distribution Propagation Graph Network for few-shot learning, a dual complete graph network that combines instance-level and distribution-level relations in an explicit way equipped with label propagation and transduction. The point and distribution losses are used to jointly update the parameters of the DPGN with *episodic training*. Extensive experiments demonstrate that our method outperforms recent state-of-the-art algorithms by 5%∼12% in the supervised task and 7%∼13% in semi-supervised task on few-shot learning benchmarks. For future work, we aim to focus on the high-order message propagation through encoding more complicated information which is linked with task-level relations.

6. Acknowledgement

This research was supported by National Key R&D Program of China (No. 2017YFA0700800).

References

[1] Kelsey R Allen, Evan Shelhamer, Hanul Shin, and Joshua B Tenenbaum. Infinite mixture prototypes for few-shot learning. *arXiv preprint arXiv:1902.04552*, 2019.

[2] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[3] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[4] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[5] Sanjoy Dasgupta. Learning mixtures of gaussians. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 634–644. IEEE, 1999.

[6] Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A Servedio. Learning poisson binomial distributions. *Algorithmica*, 72(1):316–357, 2015.

[7] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[9] Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. Deep label distribution learning with label ambiguity. *IEEE Transactions on Image Processing*, 26(6):2825–2838, 2017.

[10] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

[11] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[12] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. *arXiv preprint arXiv:1905.01102*, 2019.

[13] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. pages 315–323, 2011.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Efficiently learning mixtures of two gaussians. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 553–562. ACM, 2010.

[17] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. On the learnability of discrete distributions. In *STOC*, volume 94, pages 273–282. Citeseer, 1994.

[18] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011.

[21] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[22] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[23] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7260–7268, 2019.

[24] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9258–9267, 2019.

[25] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[26] Tiange Luo, Aoxue Li, Tao Xiang, Weiran Huang, and Liwei Wang. Few-shot learning with global class representations. *arXiv preprint arXiv:1908.05257*, 2019.

[27] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[28] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[29] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pages 721–731, 2018.

[30] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. *arXiv preprint arXiv:1905.04398*, 2019.

[31] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[33] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[35] Wei Shen, Kai Zhao, Yilu Guo, and Alan L Yuille. Label distribution learning forests. In *Advances in Neural Information Processing Systems*, pages 834–843, 2017.

[36] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[38] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[39] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019.

[40] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1199–1208, 2018.

[41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[42] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[43] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, 2018.

[44] Chao Yin and Xin Geng. Facial age estimation by conditional probability neural network. In *Chinese Conference on Pattern Recognition*, pages 243–250. Springer, 2012.

[45] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. *arXiv preprint arXiv:1905.06549*, 2019.

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[47] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.