Hybrid Graph Neural Networks for Few-Shot Learning

Tianyuan Yu[1,2], **Sen He**[1,3], **Yi-Zhe Song**[1,3], **Tao Xiang**[1,3]
[1]Center for Vision, Speech and Signal Processing, University of Surrey
[2]National University of Defense Technology
[3]iFlyTek-Surrey Joint Research Centre on Artificial Intelligence
{tianyuan.yu, sen.he, y.song, t.xiang}@surrey.ac.uk

Abstract

Graph neural networks (GNNs) have been used to tackle the few-shot learning (FSL) problem and shown great potentials under the transductive setting. However under the inductive setting, existing GNN based methods are less competitive. This is because they use an instance GNN as a label propagation/classification module, which is jointly meta-learned with a feature embedding network. This design is problematic because the classifier needs to adapt quickly to new tasks while the embedding does not. To overcome this problem, in this paper we propose a novel hybrid GNN (HGNN) model consisting of two GNNs, an instance GNN and a prototype GNN. Instead of label propagation, they act as feature embedding adaptation modules for quick adaptation of the meta-learned feature embedding to new tasks. Importantly they are designed to deal with a fundamental yet often neglected challenge in FSL, that is, with only a handful of shots per class, any few-shot classifier would be sensitive to badly sampled shots which are either outliers or can cause inter-class distribution overlapping. Extensive experiments show that our HGNN obtains new state-of-the-art on three FSL benchmarks. The code and models are available at *https://github.com/TianyuanYu/HGNN*.

Introduction

Deep convolutional neural networks (CNNs) have achieved great successes in various computer vision problems including image classification (Krizhevsky, Sutskever, and Hinton 2012), semantic segmentation (Chen et al. 2017), object detection (Ren et al. 2015) and image captioning (Xu et al. 2015). However, training deep neural networks requires a large amount of labeled data (e.g., hundreds of samples per class). Collecting and annotating them is often tedious, expensive and even infeasible for some rare classes. This thus hinders their deployments in real-world applications. One widely studied solution to this problem is few-shot learning (FSL) (Vinyals et al. 2016; Finn, Abbeel, and Levine 2017; Snell, Swersky, and Zemel 2017; Sung et al. 2018; Sun et al. 2019; Zhang et al. 2020), which aims to recognize a set of novel classes with only a handful of labeled samples/shots (e.g., 1-5 per class) by knowledge transfer from a set of base classes with abundant samples.

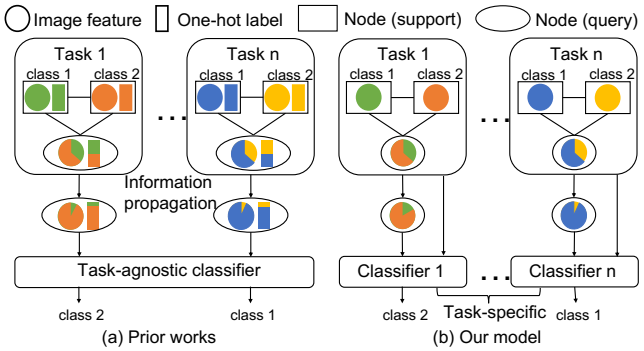
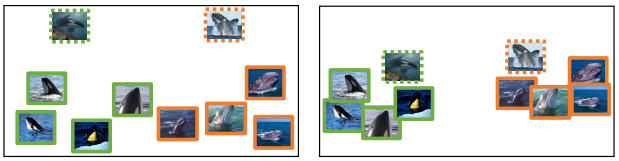


Figure 1: Illustration of the differences between our GNN and prior GNN in FSL using 2-way 1-shot tasks. (a) Previous methods, e.g. (Garcia and Bruna 2018), use GNN for label propagation, i.e., as a task-agnostic classifier. (b) We use GNN for feature adaptation and leave the query image label prediction job to task-specific classifiers.

Most recent FSL approaches follow the paradigm of meta-learning (Hospedales et al. 2020) with episodic training. Concretely, a model is trained in each episode with a few-shot classification task sampled from the base classes. Each task consists of a support set and a query set for inner and outer loop training respectively. This is to imitate the meta-test setting, under which only few labeled data are given for a novel task. The objective is to meta-learn a model capable of "learning to learn", that is, generalizing well to new tasks composed of unseen classes. Existing approaches differ primarily on what is meta-learned – a deep embedding/distance metric (Vinyals et al. 2016; Snell, Swersky, and Zemel 2017; Sung et al. 2018; Zhang et al. 2020) or an optimization algorithm (Finn, Abbeel, and Levine 2017).

Among various existing FSL approaches, graph neural network (GNN) (Kipf and Welling 2017) based FSL methods (Garcia and Bruna 2018; Luo et al. 2020; Liu et al. 2019; Kim et al. 2019; Yang et al. 2020) have received increasing attention due to their excellent performance under the transductive setting. These methods, as shown in Figure 1(a), employ GNN as a label propagation module whereby the graph is used for either node label prediction (Garcia and Bruna 2018; Liu et al. 2019; Luo et al. 2020) or edge label prediction (Kim et al. 2019; Yang et al. 2020). In other words, a GNN is used as a classifier which takes a feature embedding



(a) Feature embedding (b) HGNN-adapted embedding

Figure 2: (a) Illustration of two issues, i.e., intra-class outliers and inter-class overlapping, caused by badly sampled instances in 2-way 5 shot tasks. (b) With our HGNN, the outlier samples' effects are minimized and the two classes become well separated. More illustration with real data distributions can be found in Figure 4.

network's output as input and produces class labels. Both the classifier/GNN parameters and the feature embedding network parameters are learned jointly in the outer loop as two parts of a single model. According to (Garcia and Bruna 2018; Kim et al. 2019), a GNN is naturally suited for few-shot learning due to its ability to aggregate knowledge by message passing on a graph constructed on the limited support set instances as well as the query set instances. However, the efficacy of the existing methods (Garcia and Bruna 2018; Luo et al. 2020; Liu et al. 2019; Kim et al. 2019; Yang et al. 2020) under the inductive setting is still lagging behind the state-of-the-art (Zhang et al. 2020; Ye et al. 2020).

We believe that it is the joint meta-learning of the classifier and feature embedding that impedes the effectiveness of existing GNN-based FSL methods under the inductive setting. Specifically, there is an on-going debate (Raghu et al. 2020; Oh et al. 2020; Tian et al. 2020) on what meta-learning is truly about: rapid learning or feature reuse, or both? There seems to be little doubt on the importance of learning a good feature embedding, to the extent that it was argued recently that a good embedding is all one needs (Tian et al. 2020). Meanwhile, the ability to rapid adaptation to the new task at hand can also be critical (Ye et al. 2020). Nevertheless, there is an emerging consensus that jointly meta-learning both the classifier which has to be adapted quickly to each new task, and the feature embedding network which is intrinsically task-agnostic for feature reuse, is perhaps not a good idea due to their contradictory nature (Raghu et al. 2020; Oh et al. 2020).

To overcome this limitation, in this paper, a novel ***Hybrid Graph Neural Network*** (HGNN) based FSL framework is proposed. As shown in Figure 1(b), different from existing GNN-FSL methods (Garcia and Bruna 2018; Luo et al. 2020; Liu et al. 2019; Kim et al. 2019; Yang et al. 2020) whereby GNNs are used as classifiers via label propagation from support to query, our GNNs are used as feature embedding task adaptation modules to address a specific problem that is often ignored in FSL. That is, when only few support set instances are available to represent each class in a support set, any classifier built on them would be sensitive to badly drawn samples. More specifically, as shown in Figure 2(a), two issues can be caused by bad samples: outliers and inter-class overlapping. Outliers, caused by unusual pose/-lighting etc (Yu et al. 2019), are problematic for any learning tasks, but particular so in FSL – with few samples per class, a single outlier could have an immense effect on the class distribution. The issue of inter-class overlapping is also commonplace when the training samples of different classes have very similar background or object pose, or just being visually similar. Through message passing across the whole supports set containing all classes, our GNNs are learned to identify these outliers, minimize their negative effects, and re-adjust the class distributions so that each class' distribution is compact and further apart from each other (see Figure 2(b)).

Concretely, our HGNN is integrated with a feature embedding network meta-learned using the popular prototypical network (ProtoNet) (Snell, Swersky, and Zemel 2017). As shown in Figure 3, it consists of two GNNs, namely an ***Instance Graph Neural Networks*** (IGNN) constructed with the whole support set samples/instances plus a single query sample as nodes, and a ***Prototypical Graph Neural Network*** (PGNN) whose nodes correspond to class prototypes. They are designed to address the two bad sampling issues respectively. In particular, the IGNN focuses on outlier identification and neutralization through instance-level message passing, while the PGNN operates at the class prototype level to make sure that different classes are well separable in the embedding space adapted by the GNN. These two objectives are clearly complementary and our HGNN exploits this using two graph-specific losses and an inter-graph consistency loss.

Our contributions are as follows: (1) We propose a new framework for using GNNs for FSL which differs from existing GNN-FSL methods in that it uses the GNNs for feature embedding adaptation for new tasks, rather than label prediction. (2) As an instantiation, we propose an instance GNN which is designed to address the outlying sample problem. (3) We further propose a prototype GNN to deal with overlapping classes, which, as far as we know, has never been used before for FSL. (4) These two GNNs are integrated into a hybrid graph model which produces new state-of-the-art on three widely used FSL datasets.

Related Work

Meta-learning

Most FSL methods are based on meta-learning, which aims to learn, through episodic training a model that can generalize to unseen new tasks. Depending on what is meta-learned, existing methods can be divided broadly into two categories, *optimization-based* and *metric-based*.

Optimization-based Methods target at learning a good optimization algorithm that can adapt a deep CNN to a new task represented with few samples. Early works focused on meta-learning an optimizer. These include the LSTM-based meta-learner (Ravi and Larochelle 2017) and the external-memory assisted weight updating (Munkhdalai and Yu 2017). Later, the focus shifted to meta-learning a model initialization suitable for fast adapting the model to a new task by fine-tuning on few support samples with few iterations. The representative works are MAML (Finn, Abbeel, and Levine 2017) and its many variants (Nichol, Achiam, and Schulman 2018; Rajeswaran et al. 2019). These methods are faced with a difficult bi-level optimization problem

due to the inter-dependency of the parameters updated in the inner and outer loops in each episode. To overcome this challenge, (Sun et al. 2019) proposed to learn task-relevant scaling and shifting functions to dynamically adjust the CNN weights.

Metric-based Methods aim to learn a transferable feature embedding or distance metric. In the early works, MatchingNet (Vinyals et al. 2016) used an attention mechanism over the learned representations and applied nearest neighbor search for classification. ProtoNet (Snell, Swersky, and Zemel 2017) used the mean of each class's support set in the embedding space as a prototype without any classifier parameter and meta-learned the feature embedding network in the outer loop using a query set. RelationNet (Sung et al. 2018) learned a distance metric through a neural network on top of a feature embedding network. Many recent works, similar to our HGNN, were formulated with the ProtoNet (Snell, Swersky, and Zemel 2017) as basis, due to its simplicity and competitive performance. For example, (Allen et al. 2019) represented each class as a set of clusters rather than a single cluster. (Li et al. 2019) replaced the global feature by a local descriptor. (Afrasiyabi, Lalonde, and Gagné 2020) introduced the idea of associative alignment for leveraging each informative part of the support data. Most existing works used holistic image features. In contrast, DeepEMD (Zhang et al. 2020) showed that image-patch features can be useful in addressing the spatial misalignment issue.

Rapid Learning vs Feature Reuse Despite their theoretical attractiveness, optimization-based methods are in general less competitive compared to metric-based methods. This triggered discussions on the merits of rapid learning and feature reuse (Raghu et al. 2020; Oh et al. 2020; Tian et al. 2020). The optimization-based methods obviously focus on rapid learning, while metric-based methods are mostly about feature reuse. Yet, it was discovered that even those optimization-based methods also heavily rely on learning a good stable feature embedding that can be used for any new task (Raghu et al. 2020; Oh et al. 2020). Based on this understanding, it was suggested that the classifier parameters, which must be adapted rapidly to new tasks, should not be meta-learned jointly with the feature embedding parameters, a principle adopted by our HGNN. Recently, (Tian et al. 2020) suggested that feature embedding learning is all one needs - pre-training a feature embedding network together with model distillation can bypass the episodic training stage altogether. A similar finding was reported in (Liu et al. 2020). However, it was argued that a task adaptation module (not a classifier), jointly learned with a feature embedding, is the best trade-off between rapid learning and feature reuse (Oreshkin, López, and Lacoste 2018; Ye et al. 2020). Our HGNN provides novel task adaptation modules in the form of both instance and prototype GNNs and yields clearly better results than (Oreshkin, López, and Lacoste 2018; Ye et al. 2020).

Graph Neural Networks in Few-Shot Learning

Garcia *et al.* (Garcia and Bruna 2018) were the first to use GNNs to address few-shot classification tasks. In their GNNs, each node corresponds to one instance (support or query) and is represented as the concatenation of a feature embedding and a label embedding. The final layer in their model is a linear classification layer which directly outputs the prediction scores for each query node. (Liu et al. 2019) proposed to learn to propagate labels from support nodes to query nodes under the transductive setting, by learning a graph construction module that exploits the manifold of data in a latent space. Similarly focusing on the transductive setting but different from these node label prediction modules, EGNN (Kim et al. 2019) learned to predict the edge-labels on the graph. Based on EGNN, (Luo et al. 2020) jointly modeled the long-term inter-task correlations and short-term intra-class adjacency with the derived continual graph neural networks, which can retain and then access important prior information associated with newly encountered episodes. Recently, Yang *et al.* (Yang et al. 2020) proposed DPGN, a dual graph network consisting of a point graph and a distribution graph, in which each instance node is used to combine the distribution-level and instance-level relations.

Most of these GNN based FSL methods focus on the transductive setting, under which the full test query set can be injected into the graph for label propagation to alleviate the lack of training sample problem. However, their inductive setting performance is lagging behind the state-of-the-art. As mentioned early, we hypothesize that this is because label propagation means that these GNNs are essentially classifiers, and jointly meta-learning a classifier and a feature embedding confuses the model on whether to emphasize on rapid learning or feature reuse. In contrast, our HGNN removes the label propagation functionality and focuses on feature embedding task adaptation. Further, different from these instance GNN only methods, we additionally introduce a prototype GNN. As a result, our HGNN produces the new state-of-the-art under the inductive setting on several benchmarks.

Methodology

Problem Definition

We follow the standard FSL formulation (Vinyals et al. 2016; Finn, Abbeel, and Levine 2017; Snell, Swersky, and Zemel 2017). Concretely, a task is a N-way K-shot classification problem sampled from a meta-test set $\mathcal{D}_{\text{TST}}$. Adopting an episodic-training based meta-learning strategy, N-way K-shot tasks are sampled from a meta-training dataset $\mathcal{D}_{\text{TRN}}$, in order to imitate the few-shot test setting. Note that there is no overlapping between the class label spaces of the two sets, i.e., $\mathcal{C}_{\text{TRN}} \cap \mathcal{C}_{\text{TST}} = \varnothing$. In each episode, we first sample N classes $\mathcal{C}_N$ from the training class space randomly. Training instances are then sampled from these classes to form a support set $\mathcal{S}_N$ and a query set $\mathcal{Q}_N$ consisting of K and Q samples from each sampled class in $\mathcal{C}_N$ respectively. The sampled training instances are denoted as $\mathcal{S}_N = \{(x_i, y_i) \mid y_i \in \mathcal{C}_N, i = 1, \ldots, N \times K\}$ and $\mathcal{Q}_N = \{(x_i, y_i) \mid y_i \in \mathcal{C}_N, i = 1, \ldots, N \times Q\}$, where $\mathcal{S}_N \cap \mathcal{Q}_N = \varnothing$. During training, the model uses the support set $\mathcal{S}_N$ to build a classifier in an inner loop, and then the query set $\mathcal{Q}_N$ is used in an outer loop to evaluate and update the model parameters.

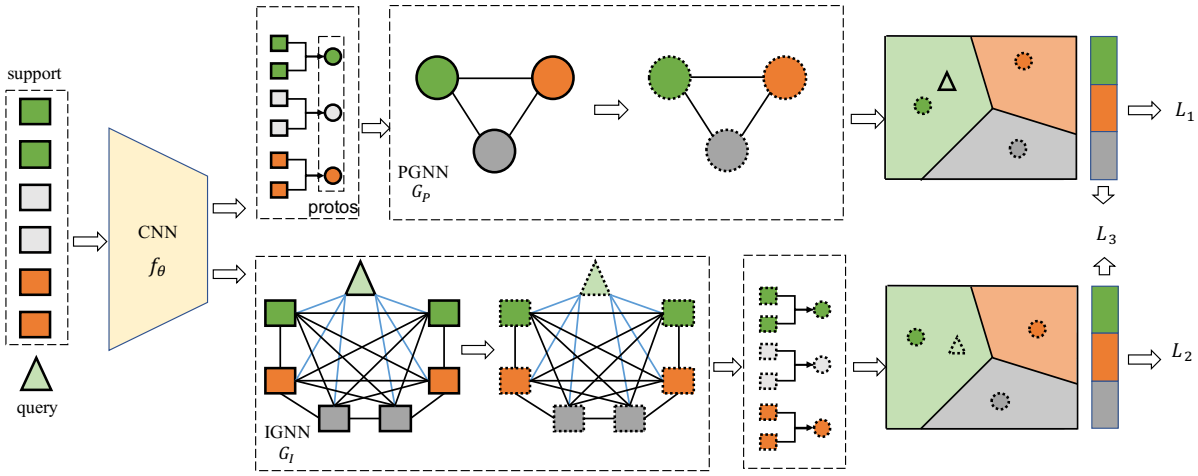


Figure 3: Overview of our HGNN model in a 3-way 2-shot case. Features extracted by a feature embedding CNN are fed into a PGNN and an IGNN for task adaptation. In the PGNN, each node represents a class prototype, which is initialized by averaging the support set features for that class. The nodes of the IGNN, on the other hand, include all the instances in the support set as well as a single query instance. After instance feature adaptation using the IGNN, the updated instance features are used to produce another set of class prototypes. Note that in the IGNN, the edge/message passing from instances in the support set to the query instance is one-directional from support to query (blue), while the edges between instances in the support set are bidirectional (black). These two sets of GNN-updated prototypes are used to predict the class of the query instances and the predictions are evaluated using cross-entropy losses (L_1 and L_2), and a consistency loss (L_3) is used to enforce the prediction consistency between the two GNNs.

Label Propagation GNNs for FSL

Before introducing our framework, we first briefly describe the formulation of existing GNN-based FSL methods in order to highlight the differences in our design. As mentioned earlier, all existing GNN-based FSL methods (Garcia and Bruna 2018; Luo et al. 2020; Liu et al. 2019; Kim et al. 2019; Yang et al. 2020) use a GNN as a label propagation module, i.e., a label classifier. To explain this, we use the model in (Garcia and Bruna 2018) as an example. In this model, the GNN is a fully-connected graph composed of M nodes. Each node represents an instance from either a support set or a query set. Under the inductive setting, only one query instance is included in the graph. Therefore for a N-way K-shot task, we have $M = N \times K + 1$. In contrast, all query instances are used to construct the graph under the transductive setting; we thus have $M = N \times (K + Q)$. For a graph G with L layers, the input with M entries from the l^{th} layer is denoted as $F^l \in \mathbb{R}^{M \times (d_f + d_l)}$, where d_f is the dimension of the instance feature obtained from a feature embedding network, and d_l is the dimension of label embedding. In other words, each node is represented as a concatenation of a visual feature embedding and a label embedding indicating which class each support set instance belongs to. The output of the $(l+1)^{\text{th}}$ layer is given by

$$F^{l+1} = G^l(F^l) = \rho\left(\mathcal{A}^l \phi^l(F^l)\right), \tag{1}$$

where ρ is an element-wise non-linear activation function, ϕ^l is a linear transformation layer , and $\mathcal{A}^l \in \mathbb{R}^{M \times M}$ is an adjacency operator in layer l. Each entry in $\mathcal{A}^l$ is computed as

$$\mathcal{A}^l_{ij} = \psi(\phi^l(F_i), \phi^l(F_j)), \tag{2}$$

where ψ is a neural network. $\mathcal{A}^l$ is normalized along each row.

The final class prediction score of a query node is computed as

$$\mathbf{p}_i = F_i^L \mathbf{w}, \tag{3}$$

where $\mathbf{w} \in \mathbb{R}^{(d_f + d_l) \times N}$ parameterizes a linear classification layer.

It is clear from the formulation of this GNN that, its objective is to take the support set labels as part of the node representation at the input layer of the graph and perform label propagation on the graph. As a result, the query sample nodes at the output layer can be used directly for label prediction. The alternative designs in (Kim et al. 2019; Luo et al. 2020; Yang et al. 2020) encode the support set labels on the edges, instead of nodes of the graph, but the main functionality of the GNN as a label classifier remains the same. In contrast, our HGNN does not encode the label information anywhere in the graph and it serves as a feature embedding task adaptation module as described in detail next.

Hybrid GNN for FSL

The proposed GNN based FSL framework is illustrated in Figure 3. It consists of a feature extraction backbone and a hybrid graph neural network (HGNN) that is further composed of an instance GNN (IGNN) and a prototype GNN (PGNN). In each training episode, we extract the features for the images in the support set $\mathcal{S}_N$ and the query set $\mathcal{Q}_N$ using a feature embedding CNN. The extracted features are then fed into the GNNs as node features for task adaptation. A key difference between the proposed GNNs and the prior ones in (Garcia and Bruna 2018; Luo et al. 2020; Liu et al.

2019; Kim et al. 2019; Yang et al. 2020) is that our GNNs contain no label information.

Prototypical Graph Neural Network (PGNN) As illustrated in Figure 2(a), as each class in the support set $\mathcal{S}_N$ is only represented by K samples, a FSL model is challenged by two issues caused by badly sampled instances, namely outliers and class overlapping. Our PGNN is designed to address the class overlapping problem. More specifically, since our HGNN is integrated with a ProtoNet FSL model where each class in $\mathcal{S}_N$ is represented by the class mean or prototypes, we feed the prototypes into the PGNN and use message passing between them to ensure that class overlapping is minimized.

Formally, as shown in Figure 3, our PGNN G_P receives the prototypes' features $F_P \in \mathbb{R}^{N\times d}$ of N classes in the support set as nodes' features, where $F_{P_n} = \frac{1}{K}\sum_{i=1}^{K} f_\theta(x_{n_i})$, where $f_\theta(\cdot)$ is a feature embedding network producing features of d dimensions, and x_{n_i} is the i^{th} image from the n^{th} class in the support set. To stabilize the training, as per standard, we adopt the residual connection (He et al. 2016) and the layer norm (Ba, Kiros, and Hinton 2016) in our GNNs. Thus the output of an one-layer PGNN is computed as

$$\hat{F}_P = \text{LayerNorm}(F_P + \varphi_P(G_P(F_P))), \tag{4}$$

where φ is a linear transformation layer to improve the expressive power of the adapted features with the same dimension d, and G_P denotes the same operations in Equation 1. Finally, the refined prototypes are used to classify the query samples in $\mathcal{Q}_\mathcal{N}$. Concretely, the probability of the ith query belonging to the class j is first computed as

$$p^P_{i,j} = \frac{\exp(-Ed(f_\theta(x_i), \hat{F}_{P_j}))}{\sum_{k=1}^{N}\exp(-Ed(f_\theta(x_i), \hat{F}_{P_k})}, \tag{5}$$

where $Ed(\cdot,\cdot)$ is the Euclidean distance. To maximize the probability $p^P_{i,j}$, the learned PGNN has the incentive to rearrange the relative position of the N prototypes so that they become more separable. This way, it becomes easier to assign each query image to the correct class with high confidence.

Instance Graph Neural Network (IGNN) The PGNN focuses on the inter-class relationship with the class mean or prototypes as graph node. It thus has limited ability to deal with the outliers that are best identified when intra-class instance relationships are examined. To that end, an IGNN is formulated.

Since we focus on the inductive setting, that is, only one query instance is available for inference at a time, our IGNN consists of the whole support set instances and one query set instance as nodes (see Figure 3). For a N-way K-shot task, there are $N \times K + 1$ nodes in the graph. This means that for Q query set samples in a training episode, Q graph needs to be constructed. Formally, the i^{th} instance graph takes the i^{th} query raw feature together with all support set samples' feature extracted by the feature embedding network $f_\theta(\cdot)$ as the nodes $F_{I_i} \in \mathbb{R}^{(N\times K+1)\times d}$. Similar to the PGNN, the features of nodes in our IGNN are refined by:

$$\hat{F}_{I_i} = \text{LayerNorm}(F_{I_i} + \varphi_I(G_I(F_{I_i}))). \tag{6}$$

Note that the updated nodes include all support nodes and a single query node. However, as shown in Figure 3, the message passing between support and query is one-directional (from support to query only) and the query has no effect on the support set node updating. This means that though we have Q graphs, the support set instances only need to be updated once and the additional computation is only on the query set instances. As a result, once trained the inference on our IGNN is very efficient.

With the updated support set features, to adhere to the ProtoNet pipeline, we compute another set of prototypes for each class by computing the updated support set instance class means. Finally, the probability of the i^{th} query node belonging to class j is

$$p^I_{i,j} = \frac{\exp(-Ed(\hat{F}_{I_{iq}}, \tilde{F_{I_{iP_j}}}))}{\sum_{k=1}^{N}\exp(-Ed(\hat{F}_{I_{iq}}, \tilde{F_{I_{iP_k}}}))}, \tag{7}$$

where $\hat{F}_{I_{iq}}$ is the updated query node feature in the i^{th} IGNN, and $\tilde{F_{I_{iP_j}}}$ is the prototype for the j^{th} class. To maximize the probability $p^I_{i,j}$, the learned IGNN is encouraged to identify the outlying support set samples and use the other instances of the same class to 'pull' it closer in the updated embedding space. Together with PGNN, IGNN can create a more friendly embedding space for classifying a query sample by comparing it with the support set samples. This is illustrated in Figure 2(b) and verified both quantitatively and qualitatively in our experiments.

Training Objectives

During training, the two GNNs in our HGNN make predictions for each query using their respective prototypes, and trained together with the shared feature embedding network f_θ via cross entropy losses. All the parameters in f_θ, PGNN G_P, and IGNN G_I are end-to-end trained.

Specifically, the classification losses on PGNN and IGNN are

$$L_1 = \sum_{i}^{Q}\sum_{j}^{N} -\mathbb{I}(y_i == j)\log(p^P_{i,j}), \tag{8}$$

$$L_2 = \sum_{i}^{Q}\sum_{j}^{N} -\mathbb{I}(y_i == j)\log(p^I_{i,j}), \tag{9}$$

where y_i is the ground truth label of the i^{th} query and $\mathbb{I}(x)$ is an indicator function: $\mathbb{I}(x) = 1$ when x is true and 0 otherwise.

In addition, to make the prediction scores for each query from the GNNs to be consistent in our HGNN, a symmetric Kullback–Leibler (KL) divergence loss is used:

$$L_3 = \sum_{j}^{N} p^I_{i,j}\log\frac{p^I_{i,j}}{p^P_{i,j}} + \sum_{j}^{N} p^P_{i,j}\log\frac{p^P_{i,j}}{p^I_{i,j}}. \tag{10}$$

Thus, during training, the total loss is:

$$L(\theta, \phi_I, \phi_P) = L_1 + L_2 + L_3. \tag{11}$$

Method	Backbone	1-shot	5-shot
ProtoNet*	Conv4	52.78 ± 0.45	71.26 ± 0.36
MAML	Conv4	48.70 ± 1.84	63.10 ± 0.92
Centroid	Conv4	53.14 ± 1.06	71.45 ± 0.72
Neg-Cosine	Conv4	52.84 ± 0.76	70.41 ± 0.66
FEAT	Conv4	55.15 ± 0.20	71.61 ± 0.16
GNN*	Conv4	52.21 ± 0.20	67.03 ± 0.17
EGNN	Conv4	51.65 ± 0.55	66.85 ± 0.49
EGNN*	Conv4	48.99 ± 0.59	61.99 ± 0.43
TPN †	Conv4	$53.75 \pm$ n/a	$69.43 \pm$ n/a
BGNN*	Conv4	52.35 ± 0.42	67.35 ± 0.35
DPGN*	Conv4	53.22 ± 0.31	65.34 ± 0.29
HGNN	Conv4	$\mathbf{55.63 \pm 0.20}$	$\mathbf{72.48 \pm 0.16}$
ProtoNet*	ResNet-12	62.41 ± 0.44	80.49 ± 0.29
Neg-Cosine	ResNet-12	63.85 ± 0.81	81.57 ± 0.56
Distill	ResNet-12	64.82 ± 0.60	82.14 ± 0.43
DSN-MR	ResNet-12	64.60 ± 0.72	79.51 ± 0.50
DeepEMD	ResNet-12	65.91 ± 0.82	82.41 ± 0.56
FEAT	ResNet-12	66.78 ± 0.20	82.05 ± 0.14
E^3BM	ResNet-25	$64.3 \pm$ n/a	$81.0 \pm$ n/a
MABAS	ResNet-12	65.08 ± 0.86	82.70 ± 0.54
APN	CapsuleNet	66.43 ± 0.26	82.13 ± 0.23
PSST	WRN-28-10	64.16 ± 0.44	80.64 ± 0.32
FRN	ResNet-12	66.45 ± 0.19	82.83 ± 0.13
HGNN	ResNet-12	$\mathbf{67.02 \pm 0.20}$	$\mathbf{83.00 \pm 0.13}$

Table 1: 5-way 1/5-shot classification accuracy (%) and 95% confidence interval on *Mini*ImageNet. $*$ indicates our reproduced results with the same pre-trained backbone, and † means transductive setting.

During meta-test, the class prediction of a query is given by the mean of two prediction scores from the two GNNs in our HGNN.

Experiments

Datasets and Settings

Datasets Three widely used FSL benchmarks, *Mini*ImageNet (Vinyals et al. 2016), *Tiered*ImageNet (Ren et al. 2018) and CUB-200-2011 (Wah et al. 2011) are used in our experiments. *Mini*ImageNet contains a total of 100 classes and 600 images per class. We follow the standard splits provided in (Vinyals et al. 2016), consisting of 64 classes for training, and 16 classes and 20 classes for validation and testing respectively. *Tiered*ImageNet is a larger subset of the ImageNet ILSVRC-12, comprising 779,165 images from 608 classes. They are divided into 351, 97, and 160 classes for training, validation and testing respectively (Chen et al. 2018). Different from the other two datasets, CUB-200-2011 is a fine-grained classification dataset. It includes 11,778 images from 200 different bird classes. The 200 classes are divided into 100, 50, 50 classes for training, validation and testing respectively as in (Liu et al. 2020; Afrasiyabi, Lalonde, and Gagné 2020). In all datasets, images are downsampled to 84×84 as per standard.

Method	Backbone	1-shot	5-shot
ProtoNet*	Conv4	50.89 ± 0.21	69.26 ± 0.18
MAML	Conv4	51.67 ± 1.81	70.30 ± 0.08
E^3BM	Conv4	$52.1 \pm$ n/a	$70.2 \pm$ n/a
GNN*	Conv4	42.37 ± 0.20	62.54 ± 0.19
EGNN*	Conv4	47.40 ± 0.43	62.66 ± 0.57
BGNN*	Conv4	49.41 ± 0.43	65.27 ± 0.35
DPGN*	Conv4	53.99 ± 0.31	69.86 ± 0.28
HGNN	Conv4	$\mathbf{56.05 \pm 0.21}$	$\mathbf{72.82 \pm 0.18}$
ProtoNet*	ResNet-12	69.63 ± 0.53	84.82 ± 0.36
Distill	ResNet-12	71.52 ± 0.69	86.03 ± 0.49
DSN-MR	ResNet-12	67.39 ± 0.82	82.85 ± 0.56
DeepEMD	ResNet-12	71.16 ± 0.87	86.03 ± 0.58
FEAT	ResNet-12	70.80 ± 0.23	84.79 ± 0.16
E^3BM	ResNet-12	$70.0 \pm$ n/a	$85.0 \pm$ n/a
APN	ResNet-12	69.87 ± 0.32	86.35 ± 0.41
FRN	ResNet-12	71.16 ± 0.22	86.01 ± 0.15
HGNN	ResNet-12	$\mathbf{72.05 \pm 0.23}$	$\mathbf{86.49 \pm 0.15}$

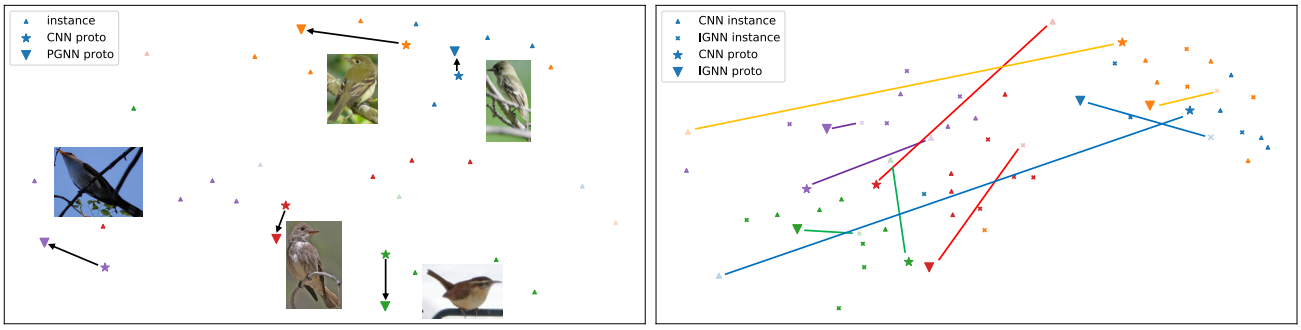
Table 2: Results on *Tiered*ImageNet

Methods	Backbone	1-shot	5-shot
ProtoNet*	Conv4	51.25 ± 0.21	72.26 ± 0.18
Adversarial	Conv4	63.30 ± 0.94	81.35 ± 0.67
HGNN	Conv4	$\mathbf{69.02 \pm 0.22}$	$\mathbf{83.20 \pm 0.15}$
ProtoNet*	ResNet-12	68.11 ± 0.21	87.33 ± 0.13
DeepEMD	ResNet-12	77.14 ± 0.29	88.98 ± 0.49
Neg-Cosine	ResNet-18	72.66 ± 0.85	89.40 ± 0.43
Centroid	ResNet-18	74.22 ± 1.09	88.65 ± 0.55
HGNN	ResNet-12	$\mathbf{78.58 \pm 0.20}$	$\mathbf{90.02 \pm 0.12}$

Table 3: Results on CUB-200-2011

Feature Embedding Network As in many other CNN-based visual recognition tasks, a feature embedding network is required in a FSL model and the choice of its backbone has a major impact on its performance. For fair comparisons with prior works, two widely used backbones are adopted in our experiments, namely Conv4 and ResNet-12. Following (Snell, Swersky, and Zemel 2017), the Conv4 backbone has four convolutional blocks and its final output feature dimension is 64. The ResNet-12 backbone is used in most of the state-of-the-art models (Zhang et al. 2020; Ye et al. 2020; Liu et al. 2020). It consists of four residual blocks, and the output feature dimension is 640. Following the common practice (Liu et al. 2020; Ye et al. 2020; Zhang et al. 2020), we pre-train our feature embedding network with supervised learning on the whole training set before the episodic meta-learning stage.

Baselines Three types of baselines are chosen for comparisons: (1) representative FSL methods including MAML (Finn, Abbeel, and Levine 2017) (optimization-based), ProtoNet (Snell, Swersky, and Zemel 2017) (embedding-based), FEAT (Ye et al. 2020) (task-adaptation), and Distill (Tian et al. 2020) (without episodic meta-learning), (2) GNN-based methods includ-



(a) PGNN (b) IGNN

Figure 4: Qualitative results on PGNN and IGNN alleviating the class overlapping and outlier issues respectively. The t-SNE projection of instances and prototypes of 5 classes in CUB-200-2011 are shown here. In (a), we use vectors to indicate how class prototypes moves from before (CNN protos) to after PGNN updates (PGNN Protos). It can be seen that after the PGNN update, the 5 class prototypes are clearly more separable. In (b), the distance between an outlying instance and its class prototypes (before and after IGNN updates) are highlighted using color-coded straight lines (same color means the same outlier). IGNN clearly neutralizes the negative impact of outliers by pulling those outliers closer to their prototypes, indicated by the shorter lines to IGNN protos.

ing GNN (Garcia and Bruna 2018), EGNN (Kim et al. 2019), TPN (Liu et al. 2019), DPGN (Yang et al. 2020) and BGNN (Luo et al. 2020), and (3) the state-of-the-art (SOTA) methods published in 2020 and 2021, including Centroid (Afrasiyabi, Lalonde, and Gagné 2020), Neg-Cosine (Liu et al. 2020), DSN-MR (Simon et al. 2020), DeepEMD (Zhang et al. 2020), E^{3}BM (Liu, Schiele, and Sun 2020), MABAS (Kim, Kim, and Kim 2020), APN (Lu, Pang, and Zhang 2020), Adversarial (Afrasiyabi, Lalonde, and Gagné 2020), ArL (Zhang et al. 2021), PSST (Chen et al. 2021), FRN (Wertheimer, Tang, and Hariharan 2021).

Main Results

The comparative results on *Mini*ImageNet, *Tiered*ImageNet and CUB-200-2011 are shown in Tables 1, 2 and 3 respectively. The following observations can be made. **(1)** Our HGNN achieves the new SOTA on all three datasets under both the 1-shot and 5-shot settings and with both the Conv4 and ResNet-12 backbones, validating its effectiveness. **(2)** Our GNN-FSL model significantly outperforms all five existing GNN-FSL models (Garcia and Bruna 2018; Luo et al. 2020; Liu et al. 2019; Kim et al. 2019; Yang et al. 2020) under the inductive setting[1]. This verifies our hypothesis that jointly meta-learning a GNN-based label propagation/classification module with a feature embedding network confuses the objectives of rapid learning and feature reuse. This results in inferior performance under the inductive setting where, without the access to the full query set, the usefulness of label propagation is limited. **(3)** Overall, the advantages of our HGNN over the SOTA alternatives under the more challenging 1-shot setting and with the fine-grained CUB-200-2011 dataset are more pronounced. This is expected: our IGNN and PGNN are designed to address the badly sampled shots problems in the support set. With fewer shots and more inter-class overlapping in the fine-grained cases, these problems are more acute and hence the clearer advantages of our HGNN.

[1]For fair comparison, we use the same backbone pre-trained on the base training dataset.

Are PGNN and IGNN Doing Their jobs?

Our PGNN and IGNN are designed to solve the inter-class overlapping and outlier issues caused by badly sampled instances in each few-shot classification task. These two issues severely impact the performance of a few-shot learning method, but are rarely discussed before. Figure 4 visualizes how the feature embedding distributions of the support set instances in a task sampled from CUB-200-2011 under the 5-way 5-shot setting. This qualitative result aims to show how the 5 prototypes and outlying instances are distributed before and after the output of the feature embedding CNN is updated by the two GNNs. We can see clearly that these two GNNs are indeed addressing the two issues as we anticipated: the PGNN pushes the class prototypes further away from each other to tackle the class overlapping problem, while the IGNN produces a more compact intra-class distribution by shortening the distance between the outliers and prototypes, mitigating the outlier problem.

Conclusions

We have proposed a novel GNN-based FSL model. Different from the existing GNN-FSL methods which utilize GNN as a label propagation tool to be jointly meta-learned with a feature embedding network, we argue that a GNN is best used in FSL as a feature embedding task adaptation module. In particular, it should address the outlying samples and class overlapping problems commonly existing in FSL through the task adaptation. To that end, an instance GNN and a prototype GNN are formulated and their complementarity is exploited in a hybrid GNN framework. Extensive experiments demonstrate that our HGNN is indeed effective in addressing the poor shot sampling problems, yielding new state-of-the-art on three benchmarks.

References

Afrasiyabi, A.; Lalonde, J.-F.; and Gagné, C. 2020. Associative Alignment for Few-shot Image Classification. In *ECCV*.

Allen, K. R.; Shelhamer, E.; Shin, H.; and Tenenbaum, J. B. 2019. Infinite mixture prototypes for few-shot learning. In *ICML*.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *CoRR*, 1607.06450.

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 40(4): 834–848.

Chen, W.-Y.; Liu, Y.-C.; Kira, Z.; Wang, Y.-C. F.; and Huang, J.-B. 2018. A Closer Look at Few-shot Classification. In *ICLR*.

Chen, Z.; Ge, J.; Zhan, H.; Huang, S.; and Wang, D. 2021. Pareto Self-Supervised Training for Few-Shot Learning. In *CVPR*, 13663–13672.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*.

Garcia, V.; and Bruna, J. 2018. Few-shot learning with graph neural networks. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2020. Meta-Learning in Neural Networks: A Survey. arXiv:2004.05439.

Kim, J.; Kim, H.; and Kim, G. 2020. Model-Agnostic Boundary-Adversarial Sampling for Test-Time Generalization in Few-Shot learning. In *ECCV*.

Kim, J.; Kim, T.; Kim, S.; and Yoo, C. D. 2019. Edge-labeling graph neural network for few-shot learning. In *CVPR*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Li, W.; Wang, L.; Xu, J.; Huo, J.; Gao, Y.; and Luo, J. 2019. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*.

Liu, B.; Cao, Y.; Lin, Y.; Li, Q.; Zhang, Z.; Long, M.; and Hu, H. 2020. Negative Margin Matters: Understanding Margin in Few-shot Classification. In *ECCV*.

Liu, Y.; Lee, J.; Park, M.; Kim, S.; Yang, E.; Hwang, S. J.; and Yang, Y. 2019. Learning to Propagate Labels: Transductive Propagation Network for Few-shot Learning. In *ICLR*.

Liu, Y.; Schiele, B.; and Sun, Q. 2020. An ensemble of epoch-wise empirical Bayes for few-shot learning. In *ECCV*.

Lu, W.; Pang, C.; and Zhang, B. 2020. Attentive Prototype Few-shot Learning with Capsule Network-based Embedding. In *ECCV*.

Luo, Y.; Huang, Z.; Zhang, Z.; Wang, Z.; Baktashmotlagh, M.; and Yang, Y. 2020. Learning from the Past: Continual Meta-Learning via Bayesian Graph Modeling. In *AAAI*.

Munkhdalai, T.; and Yu, H. 2017. Meta Networks. In *ICML*.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On First-Order Meta-Learning Algorithms. *CoRR*, abs/1803.02999.

Oh, J.; Yoo, H.; Kim, C.; and Yun, S.-Y. 2020. Does MAML really want feature reuse only? *CoRR*, abs/2008.08882.

Oreshkin, B.; López, P. R.; and Lacoste, A. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*.

Raghu, A.; Raghu, M.; Bengio, S.; and Vinyals, O. 2020. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *ICLR*.

Rajeswaran, A.; Finn, C.; Kakade, S.; and Levine, S. 2019. Meta-Learning with Implicit Gradients. In *NeurIPS*.

Ravi, S.; and Larochelle, H. 2017. Optimization as a Model for Few-Shot Learning. In *ICLR*.

Ren, M.; Triantafillou, E.; Ravi, S.; Snell, J.; Swersky, K.; Tenenbaum, J. B.; Larochelle, H.; and Zemel, R. S. 2018. Meta-learning for semi-supervised few-shot classification. In *ICLR*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*.

Simon, C.; Koniusz, P.; Nock, R.; and Harandi, M. 2020. Adaptive Subspaces for Few-Shot Learning. In *CVPR*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *NeurIPS*.

Sun, Q.; Liu, Y.; Chua, T.-S.; and Schiele, B. 2019. Meta-transfer learning for few-shot learning. In *CVPR*.

Sung, F.; Yang, Y.; Zhang, L.; Xiang, T.; Torr, P. H.; and Hospedales, T. M. 2018. Learning to compare: Relation network for few-shot learning. In *CVPR*.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J. B.; and Isola, P. 2020. Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need? In *ECCV*.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *NeurIPS*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wertheimer, D.; Tang, L.; and Hariharan, B. 2021. Few-Shot Classification With Feature Map Reconstruction Networks. In *CVPR*, 8012–8021.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Yang, L.; Li, L.; Zhang, Z.; Zhou, X.; Zhou, E.; and Liu, Y. 2020. DPGN: Distribution Propagation Graph Network for Few-shot Learning. In *CVPR*.

Ye, H.-J.; Hu, H.; Zhan, D.-C.; and Sha, F. 2020. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*.

Yu, T.; Li, D.; Yang, Y.; Hospedales, T. M.; and Xiang, T. 2019. Robust person re-identification by modelling feature uncertainty. In *ICCV*, 552–561.

Zhang, C.; Cai, Y.; Lin, G.; and Shen, C. 2020. DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover's Distance and Structured Classifiers. In *CVPR*.

Zhang, H.; Koniusz, P.; Jian, S.; Li, H.; and Torr, P. H. 2021. Rethinking Class Relations: Absolute-relative Supervised and Unsupervised Few-shot Learning. In *CVPR*, 9432–9441.