Expert Systems With Applications 289 (2025) 128332

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Meta-Tuner: Meta-trained node-specific transformations for Graph Few-Shot Class-Incremental Learning

Zhengnan Li [a], Jun Fang [b], Junbo Wang [c], Xilong Cheng [a], Yuting Tan [a], Yunxiao Qin [a,*]

[a] Communication University of China, Beijing, 100011, China
[b] Tsinghua University, Beijing, 100011, China
[c] China University of Geosciences, Beijing, 100011, China

ARTICLE INFO

Keywords:
Graph neural network
Meta-learning
Few-shot continual learning
Hypernetwork

ABSTRACT

Recommender systems leverage Graph Neural Networks (GNNs) to model user-item interactions and learn high-quality representations, incorporating category priors for personalized recommendations. However, real-world graphs continuously evolve, introducing new categories with scarce labeled data, such as emerging music genres or newly introduced product types. As new items and user preferences emerge, models must effectively adapt to novel classes with only a few labeled examples. This motivates us to investigate the **G**raph **F**ew-**S**hot **C**lass-**I**ncremental **L**earning (GFSCIL) problem, where graph data continuously evolve with the emergence of new classes, each containing only a few labeled nodes. To address this, we introduce Meta-Tuner, a novel framework that meta-trains a GNN-based encoder and a hypernetwork on simulated GFSCIL tasks. The encoder extracts initial node embeddings, while the hypernetwork applies node-specific transformations to refine these embeddings. We also propose an Equalized Prototypical Distribution Loss (EPDL) to constrain prototypes on a regular n-simplex, encouraging the model to fully utilize the high-dimensional embedding space. Our experimental results across three datasets show that Meta-Tuner significantly outperforms current state-of-the-art methods, achieving clearer decision boundaries and up to 30 % accuracy gains in the GFSCIL problem. Moreover, by applying Meta-Tuner to existing few-shot node classification (FSNC) methods, we also achieve improved FSNC performance, highlighting Meta-Tuner's effectiveness. Code is available.

1. Introduction

Graph Neural Networks (GNNs) (Hamilton, Ying, & Leskovec, 2017; Kipf & Welling, 2017; Veličković et al., 2017; Xu, Hu, Leskovec, & Jegelka, 2018; Yun, Jeong, Kim, Kang, & Kim, 2019) have garnered significant attention for their ability to effectively capture and exploit relational information within graph-structured data. These models have been widely applied across various domains, including traffic systems (He, Huang, Zhu, & Huang, 2025; Rahmani, Baghbani, Bouguila, & Patterson, 2023), social media (Guyan, Liu, Liu, & Zhang, 2025; Wu, Sun, Zhang, Xie, & Cui, 2022), e-commerce platforms (Feng et al., 2022), and financial systems (Wang, Zhang, Xiao, & Song, 2021), where complex relationships and structured interactions are prevalent. A key application of GNNs is *node classification*, which aims to infer missing labels for unlabeled nodes based on their features and graph context. In this area, existing GNN architectures (Kipf & Welling, 2017; Singh, Dar, Singh, & Kumar, 2025; Veličković et al., 2017; Yun et al., 2019; Zhao, Zhang, & Wang, 2021) have demonstrated strong performance. Moreover, as GNNs evolve, they are increasingly integrated into *expert systems*, where they serve as the reasoning engine in decision-critical applications. These expert systems leverage GNNs not only for accurate prediction but also for capturing domain-specific relational patterns, enabling robust support for complex decision-making tasks in dynamic and data-scarce environments.

However, graph data in real-world usually grow up with novel classes (Galke, Franke, Zielke, & Scherp, 2021; Wang, Song, Wu, & Wang, 2020; Zhou & Cao, 2021). For example, industrial development may lead to the introduction of new product categories, while social changes may lead to the emergence of new user groups in social media. The ever-changing nature of the real-world graph data necessitates GNNs to be able to handle a sequence of sessions, each of which introduces new classes. Due to practice such as insufficient prior-knowledge (Fredriksson, Mattos, Bosch, & Olsson, 2020), high labeling costs (Fredriksson et al., 2020), and long-tailed distributions (Downey, 2001), available labeled nodes within each novel class are commonly rare (Ding et al., 2020; Lu et al., 2022; Tan, Ding, Guo, & Liu, 2022;

* Corresponding author.
E-mail addresses: fmlyd@cuc.edu.cn (Z. Li), fangy23@mails.tsinghua.edu.cn (J. Fang), 1004211127@email.cugb.edu.cn (J. Wang), chengzhengyu330@cuc.edu.cn (X. Cheng), 202320081200023@cuc.edu.cn (Y. Tan), qinyunxiao@cuc.edu.cn (Y. Qin).

https://doi.org/10.1016/j.eswa.2025.128332
Received 20 April 2025; Received in revised form 10 May 2025; Accepted 23 May 2025
Available online 29 May 2025

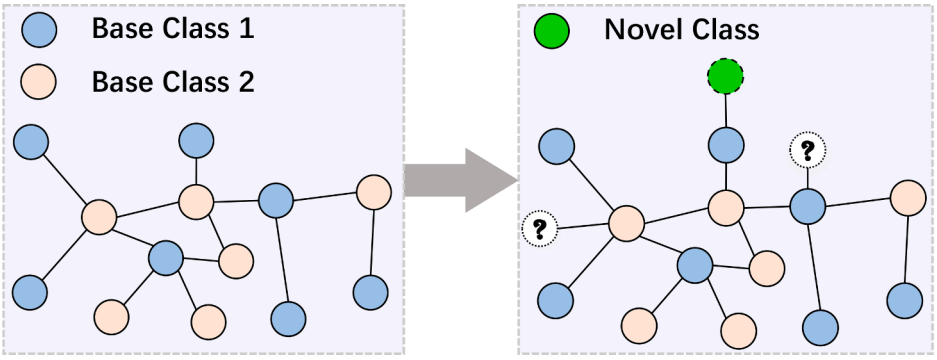

Fig. 1. Illustration of the GFSCIL problem. The graph evovles with the introduction of a novel class.

Zhou et al., 2019). As a result, GNNs need to learn novel classes on a few labeled nodes while maintaining performance on previously learned classes. This is a challenging task that is well known as **G**raph **F**ew-**S**hot **C**lass-**I**ncremental **L**earning (GFSCIL), as illustrated in (Fig. 1).

Unlike the case in FSCIL for computer vision (Wang, Zhou, Zhang, Zhan, & Ye, 2023a; Yang et al., 2023), where images are typically independent of each other, the representation of each node in a graph is inherently interdependent with other nodes. **This interdependence implies that the representation of a node, as derived from a Graph Neural Network (GNN), may change when a novel node is added to the graph, even if the parameters of the GNN remain fixed.** This introduces unique challenges in the context of GFSCIL. To address the GFSCIL problem, we identify two primary challenges (Lu et al., 2022; Tan et al., 2022). Firstly, the imbalance in the quantity of labeled nodes between novel and base classes poses a significant challenge. Models tend to overfit to base classes containing numerous labeled nodes, which negatively impacts their ability to learn new classes effectively. Secondly, similar to other deep neural networks, GNNs suffer from catastrophic forgetting (Kirkpatrick et al., 2017; Lu et al., 2022; Rebuffi, Kolesnikov, Sperl, & Lampert, 2017; Tan et al., 2022; , Song, Lin, Zheng, Pan & Xu, 2021; Zhou & Cao, 2021) when learning new classes, especially when these new classes contain few labeled nodes.

Recent GFSCIL methods (Lu et al., 2022; Tan et al., 2022) typically follow a two-step process: creating high-quality prototypes (Snell, Swersky, & Zemel, 2017) for each class and classifying nodes based on their proximity to these prototypes. Using techniques such as meta-learning (Sung et al., 2018) and attention (Vaswani et al., 2017), they form distinguishable prototypes and achieve impressive performance. However, distinguishable prototypes alone are not sufficient; node-level embedding distinguishability is also crucial. For instance, as shown in Fig. 2, methods like HAG-Meta (Tan et al., 2022) and Geometer (Lu et al., 2022) generate well-separated prototypes but still suffer from overlapping embeddings, leading to misclassifications. The failiure of the previous GFSCIL may be caused by the overlook the fact that each node in a graph is interdependent on each other.

In this paper, we unify the challenges of class imbalance and catastrophic forgetting as a single overfitting problem. Class imbalance involves overfitting on classes with a high number of nodes, while catastrophic forgetting pertains to overfitting on newly introduced classes.

To address these challenges, we propose a novel expert system called **Meta-Tuner** to simultaneously tackle both issues. It comprises of a GNN-based encoder, a hypernetwork, and a node-specific transformation controlled by the hypernetwork. The hypernetwork generates weights for a two-layer MLP based on node features, which are then used to transform each node's embedding. This soft weight sharing (where node-specific transformation weights are generated by the hypernetwork, and the hypernetwork's weights are shared rather than the weights of node-specific transformations directly) facilitates effective information sharing (Chauhan, Zhou, Ghosheh, Molaei, & Clifton, 2024; David, Andrew, & Quoc, 2016; Sun, Ozay, & Okatani, 2017), enhancing feature reuse, improving generalization, and preventing overfitting, ensuring a well-clustered embedding space. Experimental results in Section 5.3 and the visualizations in Fig. 2 demonstrate the importance of node-specific tuning for effective embedding clustering.

To ensure that the hypernetwork effectively controls node-specific transformations, the encoder and **hypernetwork are jointly trained on simulated GFSCIL tasks using metric-based Meta-Learning** (Lv et al., 2021; Sung et al., 2018). This meta-training strategy enhances the model's adaptability to new classes by continuously learning from diverse tasks (Finn, Abbeel, & Levine, 2017). It improves generalization, robustness to class imbalance, and retention of knowledge across old and new classes, thereby effectively mitigating catastrophic forgetting

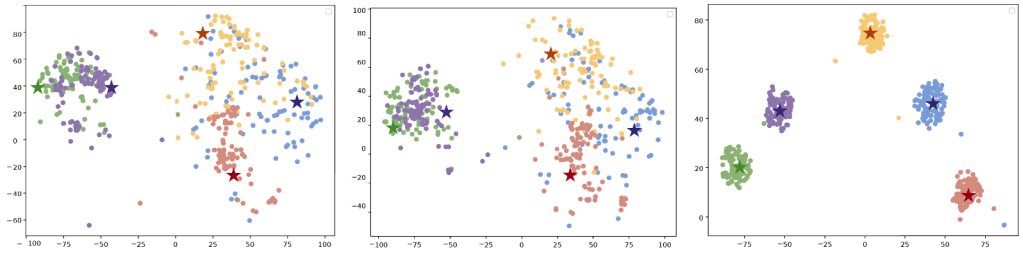

Fig. 2. Distributions of node embeddings and prototypes corresponding to five node classes. Different colors represent different classes. Points and stars represent embeddings and prototypes, respectively. t-SNE (Van der Maaten & Hinton, 2008) is used to visualize the embeddings and prototypes in the 2-D space. Obviously, the proposed Meta-Tuner clusters nodes from each class with much smaller inner-class distance, and separates prototypes of different classes with larger inter-class distance.

and adjusting to the dynamic nature of GFSCIL scenarios (Austerweil, Sanborn, & Griffiths, 2019; Hospedales, Antoniou, Micaelli, & Storkey, 2021; Li, Yang, Song, & Hospedales, 2018). Further details are provided in Section 4.

Recent studies (Hua et al., 2021; Jing, Vincent, LeCun, & Tian, 2022; Shi, Liang, Zhang, Tan, & Bai, 2024) have highlighted a problem in supervised learning known as dimensional collapse, where class prototypes crowd into subspaces, exacerbating overfitting in GFSCIL. Drawing from **n-simplex theory** (El-Gebeily* & Fiagbedzi, 2004; Gerber, 1975), we introduce the **E**qualized **P**rototypical **D**istribution **L**oss (EPDL). EPDL maintains equal distances between prototypes, preventing them from collapsing into a lower-dimensional space. By preventing them from collapsing into a lower-dimensional space, EPDL improves generalization and reduces overfitting (Hua et al., 2021; Jing et al., 2022). Unlike methods such as Center Loss (Wen, Zhang, Li, & Qiao, 2016) or SphereFace (Liu et al., 2018), which primarily address angular margins or distances from the center, EPDL promotes well-distributed prototypes across the embedding space. This enhanced distribution improves the handling of class imbalance and catastrophic forgetting. Fig. 2 shows that Meta-Tuner, with EPDL, achieves superior prototype distribution compared to methods like HAG-Meta and Geometer.

In summary, this work aims to develop a unified and generalizable framework that effectively addresses the challenges of Graph Few-Shot Class-Incremental Learning (GFSCIL), particularly overfitting caused by class imbalance and catastrophic forgetting. Our goal is threefold: (1) to design a node-level adaptation mechanism that enhances embedding separability while respecting the interdependence among graph nodes; (2) to build a meta-learned expert system that enables robust generalization to novel classes with limited supervision; and (3) to improve prototype distribution to prevent feature collapse and enhance long-term retention. Ultimately, we seek to bridge the gap between theoretical improvements and practical deployment of GNN-based expert systems in dynamic, evolving graph scenarios such as knowledge graphs, social networks, and recommendation systems.

2. Related work

2.1. Graph few-shot learning

In recent years, Graph Neural Networks (GNNs) have garnered significant attention and found wide-ranging applications across various domains, including recommendation systems (Chen et al., 2021; Yan, Li, Wang, Lin, & Yuan, 2024), traffic networks (Rahmani et al., 2023), and social networks (Liu et al., 2025; Wu et al., 2022). GCN (Kipf & Welling, 2017), a pioneering GNN, has set the stage for subsequent node classification methods, with variants like GraphSAGE (Hamilton et al., 2017) and GAT (Veličković et al., 2017) also achieving notable success on static graphs. However, conventional GNNs usually struggle to learn expressive node representations on few labeled nodes. Therefore, many methods were proposed to tackle this problem. Particularly, the episodic meta-learning paradigm (Finn et al., 2017) has emerged as a dominant strategy, leveraging knowledge transfer from numerous analogous Few-Shot Learning (FSL) tasks (Fan, Chen, Zeng, Zhou, & Zhang, 2025). G-Meta (Huang & Zitnik, 2020) employs subgraph representations as node embeddings for effective few-shot learning on graphs. Leveraging node importance and Prototypical Networks, GPN (Ding et al., 2020) enhances overall performance. RALE (Liu, Fang, Liu, & Hoi, 2021) innovatively learns node dependencies based on their positions within the graph. TENT (Wang, Ding, Zhang, Chen, & Li, 2022) tackles task variance across meta-tasks, facilitating task-adaptive few-shot node classification. On the other hand, COSMIC, introduced in (Wang, Tan, Liu, & Li, 2023b), employs a multi-step contrastive learning approach to bolster both intra-class and inter-class generalizability. However, these methods might encounter heavily performance degradation when applied to Class-incremental learning scenarios.

2.2. Graph Few-Shot Class-Incremental Learning

Graph Few-Shot Class-Incremental Learning (GFSCIL) aims to create a model capable of learning new classes with minimal labeled data while retaining knowledge from previous tasks (Lu et al., 2022; Tan et al., 2022). Unlike Few-Shot Class-Incremental Learning (CIL) in computer vision (CV), GFSCIL encounters additional challenges due to the dependence of node latent features on their neighbors. Therefore, even if the model parameters remain unchanged, it is susceptible to catastrophic forgetting.

The methodology of GFSCIL primarily employs prototype-based classifiers, which categorize samples by comparing their similarity to class prototypes. A groundbreaking work in the GFSCIL domain is HAG-Meta (Tan et al., 2022), built upon the pseudo-incremental learning paradigm. It alleviates task imbalances and catastrophic forgetting by employing attention mechanisms (Vaswani et al., 2017) and combining another GNN to adjust the contribution of each node to the prototypes, thereby enhancing the reliability of prototype. Another significant contribution in this field is Geometer (Lu et al., 2022), which addresses GFSCIL challenges by adjusting the distribution of attention-based prototypes in the geometric space and employs knowledge distillation (Hinton, Vinyals, & Dean, 2015) to mitigate catastrophic forgetting. However, these methods usually overlook the reliability of node-level embeddings, resulting in unsatisfactory GFSCIL performances.

2.3. Hypernetwork

Hypernetworks (Alaluf, Tov, Mokady, Gal, & Bermano, 2022; Chauhan, Zhou, Molaei, Ghosheh, & Clifton, 2023; David et al., 2016; Krueger et al., 2017), a type of neural networks that generate the weights of another neural network, have demonstrated impressive performance across various domains within deep learning, including continual learning (Von Oswald, Henning, Grewe, & Sacramento, 2019), weight pruning (Chauhan et al., 2023), uncertainty quantification (Krueger et al., 2017) and zero-shot learning (Zhao, Kobayashi, Sacramento, & von Oswald, 2020). Hypernetworks are typically classified into three categories based on their inputs: 1) Task-conditioned hypernetworks (Chauhan et al., 2023; David et al., 2016; Navon, Shamsian, Chechik, & Fetaya, 2020; Pan et al., 2018) that take task-specific information as input. 2) Data-conditioned hypernetworks (Alaluf et al., 2022; Balažević, Allen, & Hospedales, 2019; Littwin & Wolf, 2019) that generate network weights based on data-specific information. 3) Noise-conditioned hypernetworks (Deutsch, Nijkamp, & Yang, 2019; Krueger et al., 2017; Ratzlaff & Fuxin, 2019) that use randomly sampled noise to generate weights.

Previous work has applied hypernetworks for task incremental learning. Specifically, (Von Oswald et al., 2019) proposed a task-conditioned hypernetwork that outputs a task-specific neural network given a task-specific vector. Whereas the proposed Meta-Tuner utilizes a node-conditioned hypernetwork to output the weights of a transformation, which is then used to tune the junior node-level embedding.

3. Problem statement

Formally, a graph can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V}$, $\mathcal{E}$, and X represent the sets of nodes, edges, and original node features, respectively. It can be alternatively represented by $\mathcal{G} = (\mathcal{A}, X)$, where $\mathcal{A}$ is the adjacency matrix. Under the problem of GFSCIL, the learning of the GNN contains a base stage and a stream stage. In the base stage, the GNN learns an initial graph $\mathcal{G}^{\mathbf{B}}$ on abundant labeled nodes. In the stream stage, the GNN needs to adapt to evolving graph captured by T snapshots, denoted as $\mathbb{G}^{\mathbf{S}} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T\}$. A session refers to the process of learning a GNN model on a new snapshot of data. Each t-th session can be denoted as $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, X^t)$ where $\mathcal{V}^t$ and $\mathcal{E}^t$ can be detailed as $\mathcal{N}^t$ nodes $\{v^1, v^2, ..., v^{\mathcal{N}^t}\}$ and $\mathcal{M}^t$ edges $\{e^1, e^2, ..., e^{\mathcal{M}^t}\}$, respectively. As each node v^i associates with a feature vector x^i, X^t can be detailed as $\{x^1, ..., x^{\mathcal{N}^t}\}$.

For a specific session, we formulate the problem of learning N novel classes with K labeled nodes for each class as the "N-way K-shot Graph Few-Shot Classification with Incremental Learning (GFSCIL)" problem. The labeled training nodes are represented as support sets $\mathbf{S}$ containing $N \times K$ nodes, while the other nodes from the novel classes, and the testing nodes come from all seen classes are referred to as query set $\mathbf{Q}$. The GFSCIL problem requires the model to first learn on the support set $\mathbf{S}$ of the t-th session and then predict all query nodes in $\mathbf{Q}$.

4. Methodology

To solve the above mentioned GFSCIL problem, we propose a novel framework termed Meta-Tuner to episodically meta-train GNNs on simulated GFSCIL tasks.

Specifically, instead of directly training the model on the entire graph $\mathcal{G}^{\mathbf{B}}$, we firstly simulate lots of GFSCIL tasks on $\mathcal{G}^{\mathbf{B}}$ and then iteratively meta-trains the model on all the simulated tasks. All simulated tasks share an initial graph $\mathcal{G}^{b_0}$ containing $\mathbb{N}$ simulated seen classes with rich labeled nodes sampled from $\mathcal{G}^{\mathbf{B}}$. Our meta-learning phase is structured into two hierarchies: tasks and sessions. A session represents a phase of class increment, while a task consists of multiple sessions. In each GFSCIL task, the graph grows from the initial graph $\mathcal{G}^{b_0}$ to the entire graph $\mathcal{G}^{\mathbf{B}}$. For each i-th GFSCIL task, we start from $\mathcal{G}^{b_0}$ and grow $\mathcal{G}^{b_0}$ up to the 1-th session $\mathcal{G}_i^{b_1}$ by introducing N randomly sampled unseen classes with $K+Q$ labeled nodes (and the related edges) for each of the unseen classes from the base graph $\mathcal{G}^{\mathbf{B}}$. Similarly, we can iteratively grow each $j-1$-th session up to j-th session, resulting in the i-th GFSCIL task with the sequence of $\mathcal{G}_i^b = \{\mathcal{G}^{b_0}, \mathcal{G}_i^{b_1}, \mathcal{G}_i^{b_2}, ..., \mathcal{G}_i^{b_T}\}$, where $\mathcal{G}_i^{b_T}$ is the entire graph $\mathcal{G}^{\mathbf{B}}$. Each j-th session contains $N \times (K+Q)$ novel nodes compared with the $j-1$-th session. In our work, we use $N \times K$ nodes and the corresponding edges to construct the support set. The query set of the j-th session is composed of the other $N \times Q$ novel nodes and the query set of the j-1-th session containing $C \times Q$ query nodes. C stands for the number of old classes, equals to $\mathbb{N} + (j-1) \times N$ in the j-th session.

Let's consider the DBLP dataset under a 5-way 5-shot setting as an example. As Table 1 shows, the initial graph $\mathcal{G}^{b_0}$ contains 37 classes ($\mathbb{N} = 37$). By iterative introducing 5 new classes containing $K+Q$ labeled nodes and corresponding edges to the graph, we obtain $\mathcal{G}_i^{b_{10}}$ containing 87 classes after 10 iterations. The support set of $\mathcal{G}_i^{b_{10}}$ are 5×K novel nodes while the query set contains 5×Q novel nodes plus 82×Q query nodes inherent from $\mathcal{G}_i^{b_0}$.

Table 1
Details of three GFSCIL benchmarks.

Dataset	Amazon-C	DBLP	Reddit
Number of nodes	24,919	40,672	232,965
Number of edges	91,680	288,270	11,606,919
Feature dimension	9,034	7 ,602	602
Number of classes	77	137	41
Number of classes in $\mathcal{G}^{b_0}$	20	37	11
Number of classes in $\mathcal{G}^{\mathbf{B}}$	50	92	23
Number of classes in $\mathcal{G}^{T}$	77	137	41
N	3	5	2
K	5	5	3

4.1. Episodic meta-training

The structure of the proposed Meta-Tuner contains a GNN based encoder, a hypernetwork, and a transformation controlled by the hypernetwork. Details of the model structure are shown in Fig. 3(c). Here we detail the metric-based meta-training of Meta-Tuner on each simulated GFSILC task. Suppose the model is learning on $\mathcal{G}_i^{b_j}$, the j-th session of the i-th task. The model processes the input graph through the following steps.

4.1.1. GNN based encoder

First, the GNN based encoder extracts embeddings from the graph data, which can be formulated as

$$\mathbf{Z}_i^{b_j} = g_\theta\left(A_i^{b_j}, X_i^{b_j}\right), \tag{1}$$

where g_θ is the GNN encoder parameterized by θ. $A_i^{b_j}$ represents the adjacency matrix and $X_i^{b_j}$ is node feature set in the current graph $\mathcal{G}_i^{b_j}$. $\mathbf{Z}_i^{b_j}$ are the extracted junior node embeddings of $\mathcal{G}_i^{b_j}$.

4.1.2. Hypernetwork

As Fig. 2 shows, the previous methods cluster node embeddings poorly, which may explain the unsatisfactory GFSCIL performances of these methods. Using another learn-able transformation may is a straightforward idea to refine the embeddings and improve the clustering of node embeddings. However, as shown by the experiments in Table 4, naively applying a transformation to the embeddings can hardly improve the GFSCIL performance. The reason may be that different node embedding should be tuned differently. Motivated by this, we propose a hypernetwork to customize node-specific transformation for

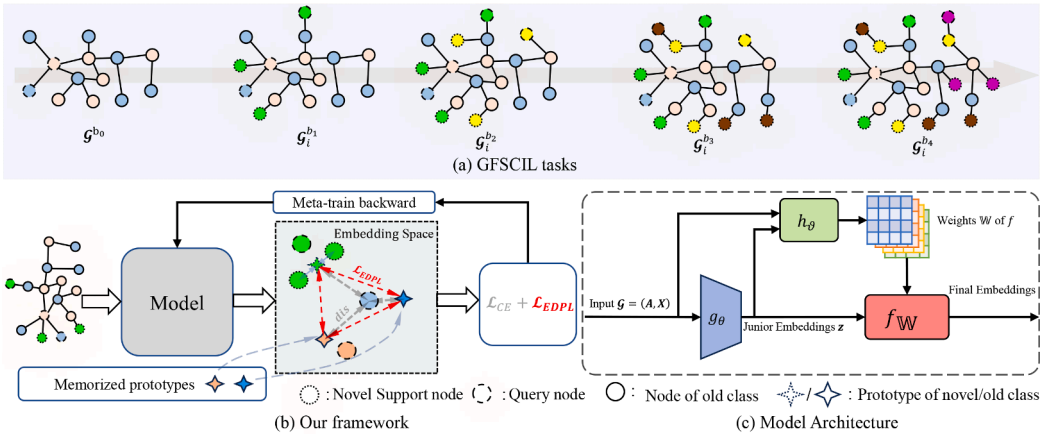

Fig. 3. Illustration of our framework. In this 1-way 2-shot GFSCIL task illustrated in (a), the evolving graph structure introduces new nodes, heightening complexity and giving rise to novel node classes, each distinguished by a unique color. Our approach, showcased in (b), leverages prototype-based classifiers for effective node classification. The model architecture, detailed in (c), begins by generating embeddings for nodes through a Graph Neural Network (GNN). Subsequently, node classification is achieved by assessing proximity to the nearest prototype. Our model follows a robust pipeline that involves encoding nodes using a GNN, learning the transformation of each node, and ultimately transforming them.

each node embedding, thus greatly improves the GFSCIL performance. In this work, we use a two layer MLP to serve as the transformation, whose weight is generated by the hypernetwork, as Fig. 3(c) shows. We formulate the generation of node-specific transformation weight as:

$$\mathbb{W} = h_\vartheta\left(X_i^{b_j}, \mathbf{Z}_i^{b_j}\right), \tag{2}$$

where h_ϑ is the hypernetwork parameterized by ϑ. $\mathbb{W}$ is the set of transformation weights, each one in $\mathbb{W}$ corresponds to a node in the graph $\mathcal{G}_i^{b_j}$. With the node-specific transformation, we refine each node embedding in the embedding space via

$$\xi_k = f_{\mathbb{W}_k}(z_k), \tag{3}$$

where $f_{\mathbb{W}_k}$ is the transformation with weight $\mathbb{W}_k \in \mathbb{W}$ for the k-th node v_k. $z_k \in \mathbf{Z}_i^{b_j}$ and ξ_k are the initial extracted embedding and the refined embedding for the node v_k, respectively.

Algorithm 1 The meta-training framework of Meta-Tuner.

Input: Initial graph $\mathcal{G}^{\mathbf{B}}$, GNN model g_θ, hypernetworks h_ϑ, the number of GFSCIL tasks M, the number of sessions per task T, the number of novel classes N per session, the number of shot K per class and the number of query Q.
Output: Trained GNN model g_θ and hypernetworks h_ϑ

while $i \leq M$ **do**
  while $j \leq T$ **do**
    Compute the junior embeddings $\mathbf{Z}_i^{b_j}$ by Eq. (1).
    Compute the weight set $\mathbb{W}$ by Eq. (2).
    Compute the final embedding $\xi_k = f_{\mathbb{W}_k}(z_k)$ of each node v_k in current graph by Eq. (3).
    Form prototypes of each class by Eq. (4).
    Predict each node in $\mathcal{Q}^i$ by Eq. (5).
    Compute $\mathcal{L}_{EPDL}$ shown in Eq. (6) and $\mathcal{L}_{CE}$.
    Optimize θ and ϑ via minimizing $\mathcal{L}_{EPDL}$ and $\mathcal{L}_{CE}$.
  end while
end while

4.1.3. *Prototype-base classifier*

In the embedding space, we classify each node by prototype-base classifier. Specifically, we first compute the prototype of each class by averaging the corresponding node embeddings within the class in the support set. For simplicity, we use S^j and S_l^j to denote the support set of $\mathcal{G}_i^{b_j}$ and the l-th novel class in the support set, respectively. Then we can formulate the prototype of each l-th novel class as

$$\mathbf{p}_l = \frac{1}{|S_l^j|} \sum_{v_k \in S_l^j} \xi_k, \tag{4}$$

where $\mathbf{p}_l$ is the prototype for the l-th class. As shown in Fig. 3(b), we classify a node by finding its nearest prototype in the embedding space. For old classes, we use their prototypes stored in memory.

Then, a distance function **dis** is employed to generate the probability distribution over all classes for each query node v_q. The probability that node v_q comes from the $\boldsymbol{a}$-th class is

$$p(y = \mathbf{a}|v_q) = \frac{exp\left(-\mathbf{dis}\left(\boldsymbol{\xi}_\mathbf{q}, \mathbf{p}_\mathbf{a}\right)\right)}{\sum_b exp\left(-\mathbf{dis}\left(\boldsymbol{\xi}_\mathbf{q}, \mathbf{p}_\mathbf{b}\right)\right)}, \tag{5}$$

where $\mathbf{dis}(\boldsymbol{\xi}_\mathbf{q}, \mathbf{p}_{\boldsymbol{a}})$ denotes the distance between the feature $\boldsymbol{\xi}_\mathbf{q}$ and the prototype $\mathbf{p}_\mathbf{a}$ corresponding to the $\boldsymbol{a}$-th class. In this work, the distance metric **dis** is the popularly used Euclidean distance.

4.1.4. *Equalized prototypical distribution loss*

Since we classify nodes by prototype, the cross-entropy loss function $\mathcal{L}_{CE}$, which aims to make query nodes closest to their class prototypes, can be regarded as an intra-class clustering function. Recent studies (Hua et al., 2021; Jing et al., 2022; Shi et al., 2024) have highlighted a problem in supervised learning known as dimensional collapse, where class prototypes crowd into subspaces, exacerbating overfitting in GFSCIL. To address this, we propose the **Equalized Prototypical Distribution Loss** (EPDL) alongside cross-entropy. EPDL enforces equal distances between prototypes, promoting the use of high-dimensional space and preventing dimensional collapse. Based on regular n-simplex theory (El-Gebeily* & Fiagbedzi, 2004; Gerber, 1975), EPDL ensures that $n+1$ points form a simplex with equal edge lengths, requiring a space with at least n dimensions. This constraint prevents prototypes from collapsing into lower-dimensional subspaces, thereby reducing overfitting and improving generalization.

To illustrate why EPDL encourages high-dimensional utilization, consider this analogy: In 2D, placing three points equidistant from each other forms a triangle, but adding a fourth point while maintaining equal distances requires 3D space, forming a tetrahedron. Minimizing EPDL drives the model to use higher dimensions, as it enforces equal distances between prototypes. The formulation of EPDL is:

$$\mathcal{L}_{EPDL} = \mathbf{var}(\{d(\mathbf{p_a}, \mathbf{p_b})^2 | \forall a \in P, \forall b \in P\}) \tag{6}$$

where d represents the distance metric and P is the set of the class prototype encountered. Here, we employ euclidean distance as our choice of d. Finally, we optimize the encoder and the hypernetwork via the formula:

$$\arg\min_{\theta,\vartheta}\left(\mathcal{L}_{EPDL} + \mathcal{L}_{CE}\right) \tag{7}$$

Algorithm 1 summarizes all training details of our framework. By meta-training Meta-Tuner across all sessions on all simulated GFSCIL tasks, we cultivate a model proficient to solve GFSCIL problems.

4.2. *Inference*

As described in Section 3, the model will be tested on $\mathbb{G}^{\mathbf{S}} = \{\mathcal{G}^1, \mathcal{G}^2, ..., \mathcal{G}^T\}$ containing T snapshots of a growing graph. Each $\mathcal{G}^i$ snapshot contains an additional $N \times K$ nodes and the corresponding edges compared to $\mathcal{G}^{i-1}$. The model is sequentially tested on these snapshots. Here we take the inference on the i-th snapshot $\mathcal{G}^i$ as an example to detail the inference pipeline. Initially, embeddings are extracted using Eq. (1). Then, we calculate the node-conditioned transformation and transform the junior embedding of each node in $\mathcal{G}^i$ by Eqs. (2) and (3). Afterward, we perform node classification on query set by Eqs. (4) and (5). The pseudocode for inference is illustrated in Algorithm 2.

Algorithm 2 The inference of Meta-Tuner.

Input: Initial graph $\mathcal{G}^{\mathbf{S}}$, GNN model g_θ, hypernetworks h_ϑ, the number of sessions per task T, the number of novel classes N per session, the number of shot K per class and the number of query Q for each encountered class.
Output: The prediction for each node in query set $\mathcal{Q}$

while $i \leq T$ **do**
  Compute the junior embeddings $\mathbf{Z}^i$ by Eq. (1).
  Compute the weight set $\mathbb{W}$ by Eq. (2).
  Compute the final embedding $\xi_k = f_{\mathbb{W}_k}(z_k)$ of each node v_k in current graph by Eq. (3).
  Form prototypes of each class by Eq. (4).
  Predict each node in the i-th query set $\mathcal{Q}^i$ by Eq. (5).
end while

5. Experiments

5.1. *Experimental setup*

5.1.1. *Benchmark*

To rigorously evaluate the proposed Meta-Tuner, we conduct comprehensive experiments on N-way K-shot GFSCIL benchmarks well-

Table 2
Experimental results on three GFSCIL benchmarks.

Benchmark	Method	Accuracy in each session (%)										Average (%)
		0	1	2	3	4	5	6	7	8	9	
Amazon-C (3-way 5-shot)	Proto-GCN	60.10	42.81	42.43	40.53	36.67	33.76	31.65	29.92	27.02	25.06	36.99
	GPN	81.33	67.28	65.24	62.47	60.33	55.38	51.73	50.13	47.87	46.26	58.80
	iCaRL	80.20	73.36	66.43	62.81	60.69	56.00	53.74	50.38	48.45	46.09	59.82
	CEC	82.26	73.03	71.12	65.06	61.45	60.33	54.78	51.76	50.53	48.07	61.84
	TEEN	83.41	74.66	70.01	69.24	68.11	65.43	59.64	56.03	54.21	51.58	65.23
	NC-FSCIL	85.49	83.18	80.18	75.19	72.48	70.82	67.85	65.54	61.82	60.65	72.32
	HAG-Meta	83.47	75.51	71.15	66.93	63.97	61.66	55.94	54.24	52.19	50.78	63.58
	Geometer	86.64	85.56	81.42	78.05	75.64	73.15	69.92	68.02	67.50	64.67	75.06
	Ours	**94.28**	**93.58**	**89.28**	**86.27**	**83.46**	**81.38**	**79.70**	**77.04**	**73.98**	**73.50**	**83.25** (+8.19)
DBLP (5-way 5-shot)	Proto-GCN	38.43	33.8	24.78	19.81	19.46	18.55	18.6	19.14	18.84	18.32	22.97
	GPN	42.06	34.47	33.02	32.25	31.48	30.38	29.54	27.38	25.35	23.36	30.93
	iCaRL	45.14	40.7	37.24	36.25	34.57	33.68	31.57	29.95	28.41	28.05	34.56
	CEC	46.46	40.13	37.85	37.22	25.01	34.34	32.52	32.51	30.94	28.8	34.58
	TEEN	60.45	59.18	53.48	50.12	48.45	46.18	43.58	40.85	39.22	37.42	47.89
	NC-FSCIL	65.18	60.01	55.43	52.42	50.25	46.57	45.14	42.59	40.27	37.37	49.52
	HAG-Meta	55.03	47.29	46.28	45.06	44.67	41.04	39.51	39.02	37.86	36.78	43.25
	Geometer	62.28	60.10	55.61	52.64	50.84	46.74	45.47	42.7	40.82	39.43	49.66
	Ours	**94.08**	**92.39**	**88.91**	**85.63**	**82.24**	**78.57**	**75.05**	**72.22**	**70.15**	**67.91**	**80.72 (+31.05)**
Reddit (2-way 3-shot)	Proto-GCN	48.02	42.82	37.78	35.38	34.30	31.04	27.68	26.58	24.62	22.66	33.09
	GPN	54.95	51.81	46.57	43.59	41.54	39.78	38.67	34.97	32.49	32.08	41.64
	iCaRL	55.28	51.43	48.70	46.16	43.93	42.02	40.16	39.33	36.63	33.51	43.72
	CEC	57.85	53.43	50.06	48.12	45.91	42.87	40.73	39.73	37.08	36.75	45.25
	TEEN	72.4	71.08	68.83	64.81	61.58	58.71	55.48	53.87	50.87	48.18	60.58
	NC-FSCIL	76.87	70.19	69.48	67.18	65.82	60.47	57.18	52.48	50.74	49.28	61.97
	HAG-Meta	60.52	53.58	52.56	50.83	50.04	46.97	45.58	43.42	42.61	40.51	48.66
	Geometer	75.64	72.28	70.84	67.19	64.31	59.19	56.00	52.72	51.41	51.21	62.08
	Ours	**90.86**	**88.54**	**85.67**	**80.37**	**78.10**	**76.61**	**70.45**	**69.28**	**68.10**	**65.64**	**77.36 (+15.28)**

established in Tan et al. (2022). These benchmarks are detailed in Table 1 and are built based on three popularly used graph datasets: Amazon-Clothing (shorten as Amazon-C in the rest paper) (McAuley, Pandey, & Leskovec, 2015), DBLP (Tang et al., 2008), and Reddit (Hamilton et al., 2017). These datasets exhibit diverse characteristics, enabling us to assess performance across a wide range of conditions. Specifically, Amazon-C possesses high feature dimensionality, DBLP features a large number of classes, and Reddit represents a large-scale graph with over 11 million edges. For fair comparisons between our method and existing methods, we strictly follow the experimental settings outlined in Tan et al. (2022) and the benchmark settings detailed in Table 1.

5.1.2. *Non-GFSCIL baselines*

Include Proto-GCN, iCaRL, GPN, and CEC. Prototypical-Net (Snell et al., 2017), iCaRL (Rebuffi et al., 2017), and CEC (Zhang et al., 2021) are three classical methods designed for few-shot, class-incremental, and few-shot class-incremental image classification, respectively. Prototypical Net (Snell et al., 2017) is primarily designed for few-shot image classification. We adapted it for the GFSCIL problem by replacing its CNN-based encoder with GCN, and denote it as Proto-GCN in our experiment. iCaRL (Rebuffi et al., 2017) is known for its class-incremental learning capabilities. It employs a prototype-based classifier, an experience replay strategy, and a knowledge distillation strategy to alleviate catastrophic forgetting. NC-FSCIL (Yang et al., 2023) addresses the problem by initially assigning class prototypes before training. TEEN (Wang et al., 2023a) enhances the discriminability of new classes by integrating new prototypes (mean features of a class) with weighted base prototypes. GPN (Ding et al., 2020) meta-trains a GNN encoder and performs node classification based on prototypes. CEC (Zhang et al., 2021) is a prominent few-shot class-incremental learning method in computer vision, boasting continually evolved classifiers. We replaced the encoders of these methods with GCN to serve as baselines.

5.1.3. *Implemental details*

In line with the foundational principles outlined by Tan et al. (2022), we adopted a two-layer Graph Convolutional Network (GCN) as the encoder. The first GCN layer housing 32 hidden neurons and the second layer containing 16 hidden neurons. The transformation in Meta-Tuner is implemented as a two-layer MLP with ReLU activation (Fukushima, 1969). ReLU activation functions is deployed between the two layers. During the training phase, we meta-train the proposed Meta-Tuner on 3,000 GFSCIL sessions to ensure convergence. The learning rate is meticulously chosen from the range of 1×10^{-3} to 5×10^{-4} to foster efficient learning.

5.1.4. *Evaluation metric*

We follow the inference pipeline introduced in Section 4.2 to evaluate all methods on the GFSCIL problem. All reported GFSCIL performances are the averaged accuracy on **10** randomly runs.

5.2. *Experimental results*

In this section, we compare our proposed framework with the baselines described in Section 5.1. The experimental results are presented in Table 2.

Given that non-GFSCIL baselines are not tailored for GFSCIL, it is unsurprising that most of them perform poorly in this context, aligning with our expectations. For instance, simply replacing the CNN-based encoder with a GCN results in Proto-GCN struggling to tackle the GFSCIL problem, achieving only 36.99% in average accuracy. Recent non-GFSCIL baselines such as NC-FSCIL and TEEN, however, achieve performance comparable to GFSCIL methods. This might be because non-GFSCIL methods have received more attention and development, yet they still fall short of Meta-Tuner. In contrast, GFSCIL baselines (HAG-Meta and Geometer) incorporating techniques tailored for GFSCIL, exhibit noticeable performance enhancements compared to their

Table 3
T-test results comparing Meta-Tuner with other methods over 10 random runs ($n = 10$ per method). Statistical significance is determined at the 0.05 level with $df = 18$ and a critical value of $t > 2.101$. Results show statistically significant improvements on three GFSCIL benchmarks.

	HAG-Meta	Geometer	NC-FSCIL	TEEN	CEC	iCaRL	GPN	Proto-GCN
Amazon-C	5.91	14.71	7.57	12.54	16.59	17.07	14.89	37.04
DBLP	26.53	24.78	26.69	20.46	28.15	43.11	31.49	35.08
Reddit	9.66	24.71	13.17	14.90	24.68	25.20	33.35	36.08

Table 4
Ablation study results of core components (EDPL and Hypernetwork) on DBLP 5-way 5-shot GFSCIL benchmark.

MLP	Hypernetwork	EDPL	Accuracy in each session (%)										Average (%)
			0	1	2	3	4	5	6	7	8	9	
			45.28	43.86	40.10	37.28	36.12	34.76	35.18	34.42	33.28	29.76	37.01
✓			46.32	44.02	41.34	39.01	36.49	35.49	35.29	33.60	33.34	30.22	37.51
✓	✓		92.34	91.95	86.70	74.31	65.84	63.33	62.82	57.13	55.86	53.25	70.35
✓	✓	✓	**94.08**	**92.39**	**88.91**	**85.63**	**82.24**	**78.57**	**75.05**	**72.22**	**70.15**	**67.91**	**80.72**

Non-GFSCIL counterparts. Notably, the state-of-the-art method, Geometer, achieves at least 13 % improvement over non-GFSCIL methods. As a comparison, the proposed method, Meta-Tuner, outshines all others, consistently demonstrating superior performance across all GFSCIL benchmarks. For instance, under the 5-way 5-shot class incremental learning setting on DBLP, Geometer achieves an average accuracy of 49.66 %. Whereas the proposed Meta-Tuner achieves a remarkable 80.72 %, showcasing an exceptional relative **improvement of approximately 62.5 %**. We also provide the results of a t-test, conducted at the 0.05 significance level. The critical t-value from the t-distribution table (with $df = 18$) is 2.101. As shown in Table 3, the performance improvements of our method are statistically significant.

5.2.1. *The effect of EPDL*

Here we verify how the proposed EPDL affect the GFSCIL performance by removing it from the training loss outlined in Eq. (7). Removal of the EPDL results in the hypernetwork relying solely on the cross-entropy loss to guide the node-specific transformation. The third line in Table 4 reports the experimental results of the EPDL-absence version. The last two lines in Table 4 show that the default version of Meta-Tuner clearly outperforms the EPDL-absence version, proving the effectiveness of the proposed EPDL. We also calculate the volume of the *hyperrectangle on the class prototypes embeddings, which increases with higher dimensions*, as formulated:

$$Volume = \prod_{k=1}^{d} (max_{z \in Z}(z_k) - min_{z \in Z}(z_k)), \tag{8}$$

where d represents the dimension of the features space, while Z represents the set of embeddings. The default version of Meta-Tuner yields a volume of **0.24**, contrasting sharply with the EPDL-absence version, which has a volume of only **0.04**. This notable disparity indicates that the absence of EPDL results in a crowded prototype space, where prototypes are distributed in a low-dimensional manifold in the embedding space. In contrast, the inclusion of *EPDL explicitly compels the prototypes to be distributed across the entire embedding space*, transcending a confined manifold. This not only prevents crowding but also enhances the representation robustness. We also compare EPDL with SphereFace, demonstrating that EPDL is better suited for the GFSCIL problem. Further details are available in the Table 10.

5.2.2. *The effect of hypernetwork*

We investigate the effectiveness of hypernetwork by further removing it from the above EPDL-absence version while keep all other experimental settings consistent. We term the new castration version as hyper-absence version. Without hypernetwork, the transformation in this experiment turns to a universal transformation (MLP) shared by all nodes rather than node-specific transformation. As the comparison between the EPDL-absence and hyper-absence version in Table 4 demonstrates, **the removal of hypernetwork further damages the average classification accuracy by almost a half**.

The first line in Table 4 is the experiment where we remove the universal transformation and turns the model to be an encoder-only version. The comparison between the first three lines reveals that **simply applying a universal transformation hardly benefit accuracy** while the proposed node-specific transformation can.

5.3. *Ablation study*

In this section, we investigate the effectiveness of core components in our framework via ablation experiments. We perform the corresponding experiments on DBLP, under the 5-way 5-shot setting, with all experiments share the same data splits.

To investigate what kind of nodes can benefit from the node-specific transformation controlled by the hypernetwork, we perform another experiment where we first employ the encoder-only model (first line in Table 4) to provide prediction distribution for each node by Eq. (5). By evaluating the prediction distributions, we select the easiest 500 nodes and the hardest 500 nodes for classification from the query set on DBLP dataset, respectively. We then verify how these nodes benefit from node-specific transformations by comparing the classification accuracy on these 500 nodes between the EPDL-absence and hyper-absence versions. As shown in Table 5, we observed a **25.07% performance enhancement for the easiest nodes while a remarkable 71.37% improvement for the hardest nodes**. This observation underscores the utility of such transformations in enhancing the interpretability of classification outcomes.

5.3.1. *The effect of meta-learning*

By default, we train all components in our network architecture via metric-based meta-learning. Here we investigate how meta-training affect the performance by two experiments. In the **first** experiment, we

Table 5
The classification accuracy comparison between the EPDL-absence and hyper-absence versions on easy and hard classification nodes. Both easy and hard nodes benefit from the node-specific transformation controlled by the hypernetwork, especially the hard nodes.

Node	w/o (%)	w/ (%)	Improvement(%)
Easist	68.06	93.13	25.07
Hardest	13.99	85.36	71.37

Table 6
Ablation study results of meta-learning on DBLP 5-way 5-shot GFSCIL benchmark.

Encoder	Hypernetwork	Accuracy in each session (%)										Average (%)
		0	1	2	3	4	5	6	7	8	9	
		52.36	49.92	47.40	44.88	42.85	41.43	39.61	37.63	36.51	35.70	42.82
	✓	71.56	70.83	66.58	64.72	61.31	58.86	56.55	54.42	52.91	50.72	60.84
✓	✓	**94.08**	**92.39**	**88.91**	**85.63**	**82.24**	**78.57**	**75.05**	**72.22**	**70.15**	**67.91**	**80.72**

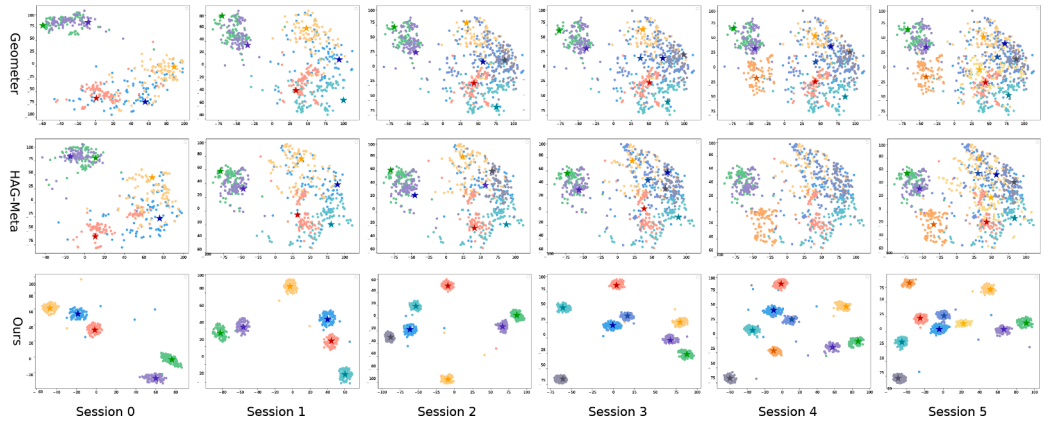

Fig. 4. A t-SNE visualization of the query node embeddings and prototypes of three methods on DBLP(1-way 5-shot). We compute intra-class distances on normalized embeddings. Our method effectively clusters the embeddings, as evidenced by the mean intra-class distance of 0.519, compared to 1.076 for the geometer's approach.

only meta-train the hypernetwork while train the encoder (in Fig. 3(c)) via supervised learning, shown as the second line in Table 6. The corresponding results reveals that meta-learning is crucial for the encoder. In the **second** experiment, both the encoder and the hypernetwork are supervised trained on $\mathcal{G}^{\mathbf{B}}$. This model, shown as the first line in Table 4, further decreases the performances compared with the second line. **The two experiments support the indispensable of meta-training in Meta-Tuner**. The reason for this result may is that meta-training on simulated GFSCIL tasks makes the Meta-Tuner be familiar with dynamic graph structure, thus enhancing the GFSCIL performance when evaluating.

5.4. Visualization

To further demonstrate the effectiveness of the proposed Meta-Tuner, we conduct a visual analysis of the embeddings from the initial six GFSCIL sessions on the DBLP dataset. The visual analysis, as depicted in Fig. 4, reveals that our model adeptly captures the emergence of novel classes, resulting in well-clustered representations. These representations are characterized by discerning decision boundaries that effectively segregate each class. In contrast, the Geometer model fails to learn separable representations and clustering patterns, consequently leading to performance degradation. In the face of evolving data and the introduction of novel classes, our model consistently showcases its adaptability by learning representations that maintain clear separations between different classes. This adaptability is a critical aspect of its effectiveness and generalizability.

5.5. Few-shot learning

Here we demonstrate that the ideas of hypernetwork and node-specific transformation in the proposed Meta-Tuner are general effective for not only GFSCIL, but also for few-shot node classification (FSNC).

5.5.1. *Experimental settings*

To show the effectiveness of our idea(node-conditioned hypernetwork and node transformation), we integrated our node-conditioned hypernetwork with two Graph Few-Shot methods GPN (Ding et al., 2020) and TENT (Wang et al., 2022), and use Hyper-GPN and Hyper-TENT to denote them, respectively.

5.5.2. *Benchmark*

We carried out experiments on four widely-used real-world graph datasets: DBLP (Tang et al., 2008), OGBN-arxiv (He, Fan, Wu, Xie, & Girshick, 2020), Cora-full (Bojchevski & Günnemann, 2017), and Amazon-E (McAuley et al., 2015). Details of these datasets are summarized in Table 8.

5.5.3. *Baseline*

We compare Hyper-TENT and Hyper-GPN with the six following baselines. Proto-GCN (Snell et al., 2017) focuses on learning class prototypes for query matching. G-Meta (Huang & Zitnik, 2020) employs subgraph representations as node embeddings for few-shot learning on graphs. GPN (Ding et al., 2020) combines node importance and Prototypical Networks for higher graph few-shot learning performance. RALE (Liu et al., 2021) introduces a method to learn node dependencies based on their locations in the graph. TENT (Wang et al., 2022) proposes to reduce the task variance among various meta-tasks and conduct task-adaptive few-shot node classification from different levels. COSMIC (Wang et al., 2023b) employs multiple contrastive learning losses to learn generalized representations. Through the comparisons across Hyper-GPN, Hyper-TENT, and the baselines, we aim to showcase the efficacy of the hypernetwork and node-specific transformation.

5.5.4. *Implemental details*

During the training phase, we randomly sample 2000 meta-training tasks from the training classes. The number of the training classes is reported in Table 8. All methods are trained using these meta-tasks for fairness. Following Wang et al. (2022), we assess the model's performance through an evaluation conducted on a set of 50 randomly sampled meta-test tasks from the test classes (i.e., the novel classes). To ensure comparison fairness, the class division remains identical for all methods. The final result, represented by the average classification accuracy, is derived from the outcomes of these meta-test tasks. All baselines' experimental results are copied from Wang et al. (2022) except COSMIC because we re-implement COSMIC with hidden size 16 for fairness.

Table 7
N-way *K*-shot node classification experimantal results on DBLP and Amazon-E. *N* represents the number of classes in each task and *K* represents the number of samples per class in support set. The metric used here is average classification accuracy. Note that we re-implemented COSMIC with the hidden size consistent to the other methods for fair comparison.

Dataset	DBLP				OGBN-arxiv			
Setting	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot
Proto-GCN	41.51	46.17	28.98	36.71	37.99	49.71	31.44	35.79
G-Meta	73.49	78.56	60.77	66.26	47.66	49.81	35.93	40.13
RALE	75.38	79.85	62.81	67.61	53.90	56.99	37.60	41.42
COSMIC	76.74	81.20	66.88	71.65	54.73	62.38	41.09	44.38
GPN	76.42	80.85	63.14	69.55	49.16	53.06	37.28	43.33
TENT	79.04	82.84	65.47	72.38	55.62	62.96	41.14	44.73
Hyper-GPN	78.97	82.51	68.48	72.49	54.08	57.71	40.18	43.68
Hyper-TENT	**80.86**	**83.91**	**70.38**	**72.65**	**58.51**	**63.66**	**42.13**	**45.79**
Dataset	Cora-full				Amazon-E			
Setting	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot	5-way 3-shot	5-way 5-shot	10-way 3-shot	10-way 5-shot
Proto-GCN	41.51	46.17	28.98	36.71	56.80	62.53	44.26	48.20
G-Meta	73.49	78.56	60.77	66.26	64.56	68.36	59.75	63.02
RALE	75.38	79.85	62.81	67.61	69.55	74.97	63.27	64.85
COSMIC	76.74	81.20	66.88	71.65	76.25	78.83	67.79	67.04
GPN	76.42	80.85	63.14	69.55	65.16	71.89	62.52	63.98
TENT	79.04	82.84	65.47	72.38	75.76	79.38	67.59	69.77
Hyper-GPN	78.97	82.51	68.48	72.49	70.23	73.71	65.93	66.81
Hyper-TENT	**80.86**	**83.91**	**70.38**	**72.65**	**76.83**	**80.12**	**68.56**	**70.29**

Table 8
Details of the four node classification datasets Amazon-E, DBLP, Cora-full, and OGBN-arxiv.

	Amazon-E	DBLP	Cora-full	OGBN-arxiv
Nodes	42, 318	40, 672	19, 793	169, 343
Edges	43, 556	288, 270	65, 311	1, 166, 243
Features	8, 669	7, 202	8, 710	128
Training classes	90	80	25	15
Val classes	37	27	20	5
Testing classes	40	30	25	20

Table 9
Comparison of Runtime Costs (in seconds) per Epoch during Meta-Training Phase for GFSCIL Baselines and Meta-Tuner on Three Datasets.

	HAG-Meta	Geometer	Ours	Ours (w/o EDPL)
DBLP	0.82	0.69	0.80	0.65
Amazon Clothing	0.32	0.31	0.30	0.28
Reddit	2.62	1.75	1.74	1.69

Table 10
Performance comparison between Meta-Tuner with EPDL and SphereFace versions.

	Amazon Clothing	DBLP
Ours(EPDL Version)	83.25	80.72
Ours(SphereFace Version)	78.91	73.64

5.5.5. *Experimental results*

Table 7 shows the corresponding experimental results on the four benchmarks. The results on the four datasets demonstrate the superior performance of Hyper-GPN and Hyper-TENT compared to their counterparts, GPN and TENT. This substantiates the effectiveness of the proposed idea involving hypernetworks and node-specific transformations in the context of few-shot node classification. Notably, Hyper-TENT surpasses all baseline models, achieving state-of-the-art (SOTA) performance. For instance, on the 10-way 5-shot node classification setting, Hyper-TENT outperforms the base baseline COSMIC by about **3.5%**. Particularly intriguing is the observation that Hyper-TENT outperforms TENT more prominently when confronted with fewer samples. This phenomenon can be attributed to the hypernetwork enhancing the robustness of node representations through node-conditioned transformations. Comparing Hyper-TENT with Hyper-GPN, it's evident that Hyper-GPN exhibits more improvement than its original version, GPN. This could be attributed to GPN addressing the node importance assignment, a task naturally suited for hypernetworks. Overall, these experiments indicate that the ideas (hypernetwork and node-specific transformation) in the proposed Meta-Tuner are effective in not only GFSCIL but also few-shot node classification.

5.6. *Complexity & scalability*

5.6.1. *Complexity*

The primary additional complexity in Meta-Tuner is introduced by EPDL, which has a complexity of $\mathcal{O}(|P|^2)$, where P represents the set of all seen classes, including both base and novel classes. Despite this quadratic complexity, it does not significantly impact the time cost. We conducted experiments to compare the runtime costs (in seconds) per epoch during the meta-training phase. The results reported in Table 9 (the lower the better) show that our method is more efficient than others in certain scenarios and does not require extensive training time.

5.6.2. *Scalability*

Reddit is a large graph dataset with over 200,000 nodes and more than 10 million edges. The experimental results in Table 2 demonstrate that our method outperforms others on the Reddit dataset, highlighting its scalability and efficiency in handling large-scale graphs.

5.7. *EPDL vs SphereFace*

In addition, we replaced EPDL in Meta-Tuner with SphereFace (Liu et al., 2018), and the results in Table 10 highlight the superior effectiveness of EPDL. While SphereFace focuses on angular margins to improve class separability, EPDL provides a more holistic approach by equalizing prototype distances. This regularization addresses the issue of dimensional collapse-where class prototypes can crowd into lower-dimensional subspace and exacerbate overfitting-as discussed in recent studies (Hua et al., 2021; Jing et al., 2022; Shi et al., 2024). By enforcing equal distances between prototypes, EPDL ensures better utilization of the embedding space, leading to improved generalization and more effective handling of class imbalance and catastrophic forgetting.

Table 11
Performance of Meta-Tuner with varying learning rates.

	0.005	0.001	0.0005
DBLP	79.18	79.64	80.72
Amazon Clothing	82.69	82.58	83.25

Table 12
The results of hidden size sensitivity.

	4	8	16	32	64	128	256
Amazon Clothing	76.75	78.65	81.54	83.25	84.18	84.40	82.98
DBLP	57.14	69.23	74.16	80.72	81.18	80.31	77.24

Table 13
The results of layers of hypernetwork sensitivity.

	1	2	3	4
Amazon Clothing	76.19	83.25	83.45	81.06
DBLP	68345	80.72	80.27	77.38

5.8. Hyperparameter analysis

5.8.1. The sensitive of learning rate

Here we verify how the learning rate effect the performance. The results with different learning rates on DBLP and Amazon Clothing are shown in Table 11. The results indicate that the proposed Meta-Tuner demonstrates robustness performances across various learning rates.

5.8.2. The sensitive of hidden size

We varify the sensitivity of the proposed Meta-Tuner to the hidden size of the hypernetwork. Table 12 reports the results, indicating an insensitivity property of the proposed Meta-Tuner to the hidden size of the hypernetwork when the hidden size is larger than 16. We default set the hidden size to 32, achieve balance between effectiveness and efficiency.

5.8.3. The sensitive of layers

We validate the sensitivity of the proposed Meta-Tuner to the number of layers in the hypernetwork. The results are shown in Table 13, indicating our default setting of 2 achieves balance between effectiveness and efficiency.

6. Discussion

6.1. Practical implications

Beyond its performance improvements on GFSCIL benchmarks, Meta-Tuner also offers practical value in real-world graph-based learning scenarios. Many graph-structured systems, such as e-commerce platforms and knowledge graphs (KGs), frequently evolve with the introduction of new classes, entities, or relations. In such settings, labeled data for novel classes are typically scarce due to high annotation costs or class imbalance. Meta-Tuner enables fast adaptation to these novel classes with minimal supervision, while preserving previously acquired knowledge. This makes it particularly useful for dynamic applications like KG completion or user modeling. Moreover, by explicitly modeling the learning dynamics across incremental sessions, Meta-Tuner provides insights into how GNNs update and retain knowledge over time, contributing to the interpretability and manageability of class-wise behavior in evolving graph environments.

6.2. Limitation

In this section, we discuss the limitations of our proposed Meta-Tuner. As demonstrated in previous experiments, Meta-Tuner achieves significant improvements on the GFSCIL benchmark, marking a substantial step toward mitigating catastrophic forgetting. The GFSCIL setting is transductive, meaning that knowledge learned from old classes is transferred to novel classes within the same graph. This setting closely resembles real-world scenarios-for example, when new products emerge on an existing e-commerce platform, or when new papers or authors are added to an academic citation network.

However, like previous works such as HAG-Meta and Geometer, Meta-Tuner does not yet support inductive settings. In an inductive setting, new nodes may not be connected to the existing graph, and thus knowledge transfer from the old graph to the new graph becomes more challenging. This presents a limitation, as many real-world applications-such as those in dynamic environments or rapidly evolving domains-require the ability to transfer knowledge across different graphs.

6.3. Future direction

In the future, we will try to further improve Meta-Tuner's performance within the existing transductive setting. While the model already demonstrates substantial improvements in the GFSCIL benchmark, there is potential for further optimization. Specifically, future work could focus on improving the speed and learning efficiency of the model, enabling faster adaptation to new classes while maintaining performance. Additionally, refining the knowledge transfer mechanisms to better mitigate catastrophic forgetting and improving scalability for handling larger, more complex graphs would increase Meta-Tuner's applicability to real-world, large-scale applications, such as e-commerce platforms, academic citation networks, and recommendation systems.

Another crucial future direction is to extend Meta-Tuner to support inductive generalization. Currently, Meta-Tuner operates in a transductive setting, where knowledge is transferred within the same graph. However, many real-world applications involve scenarios where new nodes and classes are not connected to the existing graph, making inductive generalization essential. By enabling Meta-Tuner to transfer knowledge across different graphs, it could address dynamic and evolving domains, such as real-time systems and rapidly changing knowledge graphs. This would significantly broaden its applicability and make it more versatile for handling new, unseen data in a variety of practical settings.

7. Conclusion

In this paper, we introduced Meta-Tuner, a novel framework designed to address the challenges of Graph Few-Shot Class-Incremental Learning (GFSCIL). Our approach effectively tackles the dual challenges of class imbalance and catastrophic forgetting by reconceptualizing them as manifestations of an underlying overfitting problem. Through the synergistic combination of a GNN-based encoder, a hypernetwork that generates node-specific transformations, and the innovative Equalized Prototypical Distribution Loss (EPDL), we create a robust solution for learning on continuously evolving graph data.

The experimental results across multiple benchmarks demonstrate that Meta-Tuner achieves remarkable improvements in embedding space organization, resulting in better-clustered node representations with clearer decision boundaries. Our analysis reveals that the hypernetwork-controlled node-specific transformations are particularly effective for handling difficult-to-classify nodes, while EPDL prevents dimensional collapse by maintaining equidistant prototypes in the high-dimensional embedding space. The meta-learning strategy further enhances the model's adaptability to new classes with limited labeled data-a critical capability in real-world dynamic graph environments.

In summary, the contributions of this work can be concluded as follows:

- We propose Meta-Tuner, a framework combining a GNN encoder, a hypernetwork, and node-specific transformations to address GFSCIL. Meta-Tuner features meta-training on simulated tasks and a

novel Equalized Prototypical Distribution Loss that ensures equidistant prototypes in the embedding space. As the experimental results in Tables 4 and 6 demonstrate, all the proposed node-specific transformation, meta-training strategy, and EPDL are crucial for Meta-Tuner's performance.

- Meta-Tuner achieves state-of-the-art results on GFSCIL benchmarks such as Reddit (Hamilton et al., 2017), DBLP (Tang et al., 2008), and Amazon Clothing (McAuley et al., 2015), outperforming the best existing method, Geometer (Lu et al., 2022), by up to 30%. Extensive analyses, including intra-class distance evaluation, further demonstrate the robustness of our framework.
- Meta-Tuner also improves performances in Few-Shot Node Classification (FSNC) when applied to existing FSNC methods (Ding et al., 2020; Liu et al., 2021; Wang et al., 2022, 2023b), highlighting its versatility and effectiveness across different node classification tasks. Please refer to Section 5.5 for details.

CRediT authorship contribution statement

Zhengnan Li: Conceptualization, Methodology, Writing - original draft; **Jun Fang:** Investigation, Resources; **Junbo Wang:** Software; **Xilong Cheng:** Validation, Formal analysis; **Yuting Tan:** Data curation, Visualization; **Yunxiao Qin:** Writing - review & editing, Supervision, Project administration, Funding acquisition.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that no competing financial interests or personal relationships that may influence the work reported in this paper.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62206259) and in part by the Fundamental Research Funds for the Central Universities.

References

Alaluf, Y., Tov, O., Mokady, R., Gal, R., & Bermano, A. (2022). Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 18511–18521).

Austerweil, J. L., Sanborn, S., & Griffiths, T. L. (2019). Learning how to generalize. *Cognitive Science*, *43*(8), e12777.

Balažević, I., Allen, C., & Hospedales, T. M. (2019). Hypernetwork knowledge graph embeddings. In *Artificial neural networks and machine learning–ICANN 2019: Workshop and special sessions: 28th international conference on artificial neural networks, Munich, Germany, September 17–19, 2019, proceedings 28* (pp. 553–565). Springer.

Bojchevski, A., & Günnemann, S. (2017). Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. arxiv preprint arXiv:1707.03815.

Chauhan, V. K., Zhou, J., Ghosheh, G., Molaei, S., & Clifton, D. A. (2024). Dynamic intertreatment information sharing for individualized treatment effects estimation. In *International conference on artificial intelligence and statistics* (pp. 3529–3537). PMLR.

Chauhan, V. K., Zhou, J., Molaei, S., Ghosheh, G., & Clifton, D. A. (2023). Dynamic intertreatment information sharing for heterogeneous treatment effects estimation. arxiv preprint arXiv:2305.15984.

Chen, X., Tang, T., Ren, J., Lee, I., Chen, H., & Xia, F. (2021). Heterogeneous graph learning for explainable recommendation over academic networks. In *IEEE/WIC/ACM International conference on web intelligence and intelligent agent technology* (pp. 29–36).

David, H., Andrew, D., & Quoc, V. L. (2016). Hypernetworks. *arXiv preprint arXiv*, *1609*.

Deutsch, L., Nijkamp, E., & Yang, Y. (2019). A generative model for sampling high-performance and diverse weights for neural networks. arxiv preprint arXiv:1905.02898.

Ding, K., Wang, J., Li, J., Shu, K., Liu, C., & Liu, H. (2020). Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 295–304).

Downey, A. B. (2001). Evidence for long-tailed distributions in the internet. In *Proceedings of the 1st ACM SIGCOMM workshop on internet measurement* (pp. 229–241).

El-Gebeily*, M. A., & Fiagbedzi, Y. A. (2004). On certain properties of the regular n-simplex. *International Journal of Mathematical Education in Science and Technology*, *35*(4), 617–629.

Fan, L., Chen, B., Zeng, X., Zhou, J., & Zhang, X. (2025). Knowledge-enhanced meta-transfer learning for few-shot ECG signal classification. *Expert Systems with Applications*, *263*, 125764.

Feng, C., He, Y., Wen, S., Liu, G., Wang, L., Xu, J., & Zheng, B. (2022). DC-GNN: Decoupled graph neural networks for improving and accelerating large-scale e-commerce retrieval. In *Companion proceedings of the web conference 2022* (pp. 32–40).

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135). PMLR.

Fredriksson, T., Mattos, D. I., Bosch, J., & Olsson, H. H. (2020). Data labeling: An empirical investigation into industrial challenges and mitigation strategies. In *International conference on product-focused software process improvement* (pp. 202–216). Springer.

Fukushima, K. (1969). Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, *5*(4), 322–333. https://doi.org/10.1109/TSSC.1969.300225

Galke, L., Franke, B., Zielke, T., & Scherp, A. (2021). Lifelong learning of graph neural networks for open-world node classification. In *2021 International joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.

Gerber, L. (1975). The orthocentric simplex as an extreme simplex. *Pacific Journal of mathematics*, *56*(1), 97–111.

Guyan, Q., Liu, Y., Liu, J., & Zhang, P. (2025). PEGNN: Peripheral-enhanced graph neural network for social bot detection. *Expert Systems with Applications*, *278*, 127294.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*, (pp. 1024-1034).

He, G., Huang, W., Zhu, Y., & Huang, M. (2025). Adaptive spatial-temporal dependence graph convolution neural network for traffic flow prediction. *Expert Systems with Applications*, (p. 127564).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9729–9738).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arxiv preprint arXiv:1503.02531.

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(9), 5149–5169.

Hua, T., Wang, W., Xue, Z., Ren, S., Wang, Y., & Zhao, H. (2021). On feature decorrelation in self-supervised learning. arXiv:2105.00470.

Huang, K., & Zitnik, M. (2020). Graph meta learning via local subgraphs. *Advances in Neural Information Processing Systems*, *33*, 5862–5874.

Jing, L., Vincent, P., LeCun, Y., & Tian, Y. (2022). Understanding dimensional collapse in contrastive self-supervised learning. arXiv:2110.09348.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., & Courville, A. (2017). Bayesian hypernetworks. arxiv preprint arXiv:1710.04759.

Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. (2018). Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*. (*vol. 32*), (pp. 3490-3497).

Littwin, G., & Wolf, L. (2019). Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1824–1833).

Liu, G., Sun, X., Li, H., Guo, Z., Li, Y., & Pi, S. (2025). Knowledge-enhanced heterogeneous graph attention networks for privacy co-disclosure detection in online social network. *Expert Systems with Applications*, *268*, 126266.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2018). Sphereface: Deep hypersphere embedding for face recognition. arXiv:1704.08063.

Liu, Z., Fang, Y., Liu, C., & Hoi, S. C. H. (2021). Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4267–4275). (*vol. 35*).

Lu, B., Gan, X., Yang, L., Zhang, W., Fu, L., & Wang, X. (2022). Geometer: Graph few-shot class-incremental learning via prototype representation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1152–1161).

Lv, H., Chen, C., Cui, Z., Xu, C., Li, Y., & Yang, J. (2021). Learning normal dynamics in videos with meta prototype network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 15425–15434).

McAuley, J., Pandey, R., & Leskovec, J. (2015). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794).

Navon, A., Shamsian, A., Chechik, G., & Fetaya, E. (2020). Learning the pareto front with hypernetworks. arxiv preprint arXiv:2010.04104.

Pan, Z., Liang, Y., Zhang, J., Yi, X., Yu, Y., & Zheng, Y. (2018). HyperST-Net: Hypernetworks for spatio-temporal forecasting. arxiv preprint arXiv:1809.10889.

Rahmani, S., Baghbani, A., Bouguila, N., & Patterson, Z. (2023). Graph neural networks for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*. (pp. 8846-8885).

Ratzlaff, N., & Fuxin, L. (2019). Hypergan: A generative model for diverse, performant neural networks. In *International conference on machine learning* (pp. 5361–5369). PMLR.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). ICARL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2001–2010).

Shi, Y., Liang, J., Zhang, W., Tan, V. Y. F., & Bai, S. (2024). Towards understanding and mitigating dimensional collapse in heterogeneous federated learning. arXiv:2210.00226.

Singh, A., Dar, S. S., Singh, R., & Kumar, N. (2025). A hybrid similarity-aware graph neural network with transformer for node classification. *Expert Systems with Applications*, *279*, 127292.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, *30*, (pp. 4080-4090).

Sun, Z., Ozay, M., & Okatani, T. (2017). Hypernetworks with statistical filtering for defending adversarial examples. arXiv:1711.01791.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1199–1208).

Tan, Z., Ding, K., Guo, R., & Liu, H. (2022). Graph few-shot class-incremental learning. In *Proceedings of the fifteenth ACM international conference on web search and data mining* (pp. 987–996).

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 990–998).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11), (pp. 2579-2605).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*, (pp. 5998-6008).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arxiv preprint arXiv:1710.10903.

Von Oswald, J., Henning, C., Grewe, B. F., & Sacramento, J. (2019). Continual learning with hypernetworks. arxiv preprint arXiv:1906.00695.

Wang, J., Song, G., Wu, Y., & Wang, L. (2020). Streaming graph neural networks via continual learning. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1515–1524).

Wang, J., Zhang, S., Xiao, Y., & Song, R. (2021). A review on graph neural network methods in financial applications. arxiv preprint arXiv:2111.15367.

Wang, Q.-W., Zhou, D.-W., Zhang, Y.-K., Zhan, D.-C., & Ye, H.-J. (2023a). Few-shot class-incremental learning via training-free prototype calibration. arXiv:2312.05229.

Wang, S., Ding, K., Zhang, C., Chen, C., & Li, J. (2022). Task-adaptive few-shot node classification. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1910–1919).

Wang, S., Tan, Z., Liu, H., & Li, J. (2023b). Contrastive meta-learning for few-shot node classification. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 2386–2397).

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *Computer vision–ECCV 2016: 14th European conference, Amsterdam, the Netherlands, October 11–14, 2016, proceedings, part VII 14* (pp. 499–515). Springer.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, *55*(5), 1–37.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? arxiv preprint arXiv:1810.00826.

Yan, S., Li, C., Wang, H., Lin, B., & Yuan, Y. (2024). Feature interactive graph neural network for KG-based recommendation. *Expert Systems with Applications*, *237*, 121411.

Yang, Y., Yuan, H., Li, X., Lin, Z., Torr, P., & Tao, D. (2023). Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. arXiv:2302.03004.

Yun, S., Jeong, M., Kim, R., Kang, J., & Kim, H. J. (2019). Graph transformer networks. *Advances in Neural Information Processing Systems*, *32*.

Zhang, C., Song, N., Lin, G., Zheng, Y., Pan, P., & Xu, Y. (2021). Few-shot incremental learning with continually evolved classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12455–12464).

Zhao, D., Kobayashi, S., Sacramento, J., & von Oswald, J. (2020). Meta-learning via hypernetworks. In *4th Workshop on meta-learning at neurIPS 2020 (Metalearn 2020)*. NeurIPS.

Zhao, T., Zhang, X., & Wang, S. (2021). Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 833–841).

Zhou, F., & Cao, C. (2021). Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4714–4722). (*vol. 35*).

Zhou, F., Cao, C., Zhang, K., Trajcevski, G., Zhong, T., & Geng, J. (2019). Meta-GNN: On few-shot node classification in graph meta-learning. In *Proceedings of the 28th ACM international conference on information and knowledge management* (pp. 2357–2360).