A Post-Training Framework for Improving Heterogeneous Graph Neural Networks

Cheng Yang
yangcheng@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Xumeng Gong
xumeng1141@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi*
shichuan@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Philip S. Yu
psyu@cs.uic.edu
UNIVERSITY OF ILLINOIS AT CHICAGO
Chicago, United States

ABSTRACT

Recent years have witnessed the success of heterogeneous graph neural networks (HGNNs) in modeling heterogeneous information networks (HINs). In this paper, we focus on the benchmark task of HGNNs, *i.e.,* node classification, and empirically find that typical HGNNs are not good at predicting the label of a test node whose receptive field (1) has few training nodes from the same category or (2) has multiple training nodes from different categories. A possible explanation is that their message passing mechanisms may involve noises from different categories, and cannot fully explore task-specific knowledge such as the label dependency between distant nodes. Therefore, instead of introducing a new HGNN model, we propose a general post-training framework that can be applied on any pretrained HGNNs to further inject task-specific knowledge and enhance their prediction performance. Specifically, we first design an auxiliary system that estimates node labels based on (1) a global inference module of multi-channel label propagation and (2) a local inference module of network schema-aware prediction. The mechanism of our auxiliary system can complement the pretrained HGNNs by providing extra task-specific knowledge. During the post-training process, we will strengthen both system-level and module-level consistencies to encourage the cooperation between a pretrained HGNN and our auxiliary system. In this way, both systems can learn from each other for better performance. In experiments, we apply our framework to four typical HGNNs. Experimental results on three benchmark datasets show that compared with pretrained HGNNs, our post-training framework can enhance Micro-F1 by a relative improvement of 3.9% on average. Code, data and appendix are available at https://github.com/GXM1141/HGPF.

*Corresponding author

WWW '23, May 1–5, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583282

CCS CONCEPTS

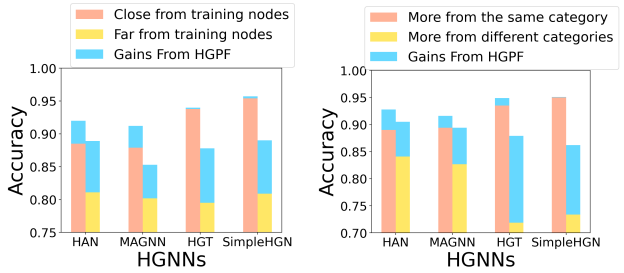
• **Computing methodologies** → **Machine learning**; • **Networks** → **Network algorithms**.

KEYWORDS

Heterogeneous Information Network, Graph Neural Network

ACM Reference Format:
Cheng Yang, Xumeng Gong, Chuan Shi, and Philip S. Yu. 2023. A Post-Training Framework for Improving Heterogeneous Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023 (WWW '23), May 1–5, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583282

1 INTRODUCTION



(a) Close (red) v.s. Far (yellow) (b) Same (red) v.s. Different (yellow)

Figure 1: Motivation verification on ACM dataset. The performance gaps between red and yellow columns indicate that all four HGNNs are not good at predicting the label of a test node: if (a) it's *far* from training nodes of the same category or (b) it has more training nodes from *different* categories than the same category in the receptive field. Blue columns are the performance gains from our post-training framework.

A heterogeneous information network (HIN) [17–19] can characterize the rich semantic relationship among a set of nodes and edges with different types. To effectively capture the semantics in an HIN, heterogeneous graph neural networks (HGNNs) were proposed in recent years [3, 9, 21, 26], and typical HGNNs will use meta-paths to enlarge the receptive field of each node in message passing [4].

The success of HGNNs facilitated the development of HIN-based applications such as recommendation system [1, 13, 23, 30] and malware detection system [2, 6, 8].

In this work, we focus on the benchmark task for evaluating HGNNs, *i.e.,* semi-supervised node classification on HINs. We empirically find that typical HGNNs are not good at predicting the label of a test node whose receptive field (1) has few training nodes from the same category or (2) has multiple training nodes from different categories. For example, we evaluate four typical HGNNs [3, 9, 14, 21] based on the benchmark setting [29] of ACM dataset. For the first observation, we compute the average distance from training nodes of the same category for every test node, and divide the test nodes into two groups depending on whether the average distance is larger than the range of receptive field or not. For the second observation, we build two node groups depending on whether a node's receptive field has more training nodes from the same category than different categories. As shown in Figure 1, the prediction accuracies of all four HGNNs drop significantly for the nodes that (1) are farther from training nodes of the same category or (2) have more training nodes from different categories than the same category in the receptive field. Note that such nodes occupy around 50 percent of all nodes. A possible reason is that the message passing mechanisms of typical HGNNs, which highly depend on the range of receptive fields, may involve noises from different categories, and cannot fully explore task-specific knowledge such as the label dependency between distant nodes. Moreover, this drawback cannot be addressed by simply enlarging the receptive field of HGNNs, *i.e.,* stacking more message passing layers, since more noises may be involved and the over-smoothing issue will be intensified.

As the above limitations generally exist in typical HGNNs, instead of introducing a new HGNN model, we propose a general post-training framework that can be applied on any pretrained HGNNs. Our Heterogeneous Graph Post-training Framework (HGPF) aims to alleviate the aforementioned limitations and enhance the prediction performance. Specifically, we first design an auxiliary system which predicts node labels based on a complementary mechanism: (1) a global inference module will diffuse node labels to distant nodes by multi-channel label propagation (usually 5 times farther than the receptive fields of typical HGNNs in our experiments); (2) a local inference module will predict node labels merely based on every node's network schema instance, and thus exclude the influence of meta-path-based neighbors from potentially different categories. During the post-training process, we will optimize system-level prediction consistency between pretrained HGNN and auxiliary system, and module-level prediction consistency between global and local modules. In this way, both systems can learn from each other, and our auxiliary system can complement HGNNs by injecting extra task-specific knowledge for better performance. After the post-training, either updated HGNN or learned auxiliary system can be used for prediction.

To fully evaluate our proposed framework HGPF, we conduct experiments on three benchmark datasets and test with four typical HGNN models, including HAN [21], HGT [9], Simple-HGN [14] and MAGNN [3]. Experimental results show that compared with a pretrained HGNN, both updated HGNN and learned auxiliary system can have consistent improvements by learning from each other, and the learned auxiliary system performs best among the three. In terms of Micro-F1, the relative improvement of learned auxiliary system against pretrained HGNN is 3.9% on average. We also compare our system against a broad range of state-of-the-art (SOTA) graph algorithms, showing that HGPF is the current SOTA method on this task.

Our contributions are summarized as follows:

• We focus on the semi-supervised node classification task, and point out a key limitation of typical HGNNs that they are not good at predicting the label of a node whose receptive field has few training nodes from the same category or has multiple training nodes from different categories.

• We propose a general and novel post-training framework HGPF that can be applied on any pretrained HGNNs to alleviate the above limitation and improve prediction accuracies. Specifically, we design an effective auxiliary system complementary with typical HGNNs, and encourage the two systems to learn from each other for better performance.

• Experimental results show that systems learned by HGPF consistently outperform pretrained HGNNs by a large margin, and achieve SOTA performance on all three benchmark datasets compared with a variety of baselines.

2 RELATED WORKS

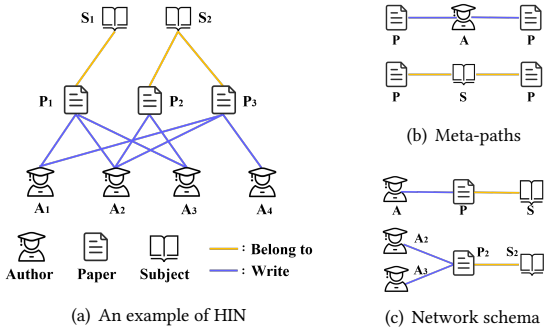
Heterogeneous graph neural networks: Recently, many researchers focus on developing HGNNs for HIN modeling. Specifically, HAN [21] employed two types of attention mechanism to learn the node-level and semantic-level structures. MAGNN [3] further considered the intermediate nodes along the meta-paths on the basis of HAN. RSHN [31] constructed a coarsened line graph to take edge features into account, and employed message passing neural network [5] to propagate information of both nodes and edges. HetGNN [26] used random walks with restart mechanism to find valuable neighbors for nodes. HetSANN [7] employed a type-specific graph attention mechanism to aggregate the direct neighbors of nodes without meta-paths. GTN [25] was proposed to learn useful meta-paths automatically for message passing. HGT [9] was a transformer-based HGNN for modeling web-scale HINs through a graph sampling method. In addition to pairwise proximity, NSHE [29] considered high-order proximity based on network schema instances. HGSL [28] was proposed to jointly perform HIN structure learning and GNN parameter learning for classification. HGNN-AC [10] employed attention mechanism to complete missing attributes with MAGNN [3] as its backbone. HGK-GNN [12] proposed Heterogeneous Graph Kernel (HGK) based on Mahalanobis distance, and incorporated it into HGNNs. Simple-HGN [14] designed a simple HGNN model based on GAT [20], and is the SOTA HGNN model.

Connections with relevant models: There are a few works that can be applied on homogeneous GNNs to boost their performance, though these methods did not use the term "post-training". For example, GMNN [15] adopted an EM framework to train two GCNs iteratively, which can be seen as a multi-stage post-training method. RDD [27] trained and ensembled multiple GCNs with a distillation framework to get better performance. CPF [24] used a knowledge distillation framework to extract the knowledge of GNNs and utilize more prior knowledge. These approaches are

only applicable to homogeneous graphs, and their motivations are also different from ours. In our experiments, we will adopt these methods as our baselines.

Note that our auxiliary system includes a global and a local inference module. In fact, there are also some HGNN methods emphasizing the usage of global and local information. But these methods are mainly proposed for unsupervised representation learning and their definitions of global/local information are quite different from ours. For example, HDGI [16] minimized the local-global mutual information between node-level representation and meta-path based graph-level representation. HeCo [22] conducted contrastive learning between meta-path and schema views of an HIN to learn node representations in an unsupervised manner. Besides, the key designs of our auxiliary system, such as the multi-channel label propagation, have never been explored in previous HGNNs.

3 PRELIMINARIES



(a) An example of HIN (b) Meta-paths (c) Network schema

Figure 2: A toy example of an HIN on ACM dataset [29].

Definition 1: Heterogeneous Information Network. A heterogeneous information network, denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \phi, \varphi\}$, is a special form of information networks, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. An HIN is also associated with a node type mapping function $\phi : \mathcal{V} \to \mathcal{T}$ and an edge type mapping function $\varphi : \mathcal{E} \to \mathcal{R}$, where $\mathcal{T}$ and $\mathcal{R}$ respectively denote the sets of node and edge types, with $|\mathcal{T}| + |\mathcal{R}| > 2$.

Fig. 2(a) illustrates an example of ACM dataset [29]. The HIN has three types of nodes, including author (A), paper (P) and subject (S). There are also two types of edges (*i.e.,* "write" relation between author and paper, "belong to" relation between paper and subject).

Definition 2: Meta-path. A meta-path P is defined as a path in the form of $T_1 \xrightarrow{R_1} T_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} T_{l+1}$ (abbreviated as $T_1T_2\cdots T_{l+1}$), which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between node types T_1 and T_{l+1}, where $\circ$ denotes the composition operator on relations. We denote the meta-path set as $\mathcal{P}$ which contains all the meta-paths in an HIN.

As shown in Fig. 2(b), two papers can be connected via two kinds of meta-paths, *i.e.,* paper-author-paper (PAP) and paper-subject-paper (PSP). Different meta-paths represent different semantics in an HIN. For example, PAP connects two papers written by the same author, while PSP connects two papers from the same subject.

Definition 3: Network Schema. The network schema $S_{\mathcal{G}} = (\mathcal{T}, \mathcal{R})$ is the blueprint of an HIN $\mathcal{G}$. Specifically, network schema is a directed graph defined on the set of node types $\mathcal{T}$, with edges as relations from the set of edge types $\mathcal{R}$. The network schema of ACM is shown in the upper half of Fig. 2(c). In addition, the lower half of Fig. 2(c) presents a network schema instance, which is defined as a local structure matching the paradigm of network schema.

Definition 4: Semi-supervised node classification on an HIN. Given an HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \phi, \varphi\}$ with node features $\mathcal{X}$, we aim to predict the labels for the nodes of a specific type $T \in \mathcal{T}$. We denote the the set of nodes with type T as the target node set $\mathcal{V}_T$. Each target node $v \in \mathcal{V}_T$ corresponds to a class label y_v from the label set $\mathcal{Y}$. For the semi-supervised setting, the labels of nodes in labeled set $\mathcal{V}_L \subset \mathcal{V}_T$ are known, and the task is to predict the labels for the unlabeled nodes in $\mathcal{V}_U = V_T \setminus \mathcal{V}_L$. Semi-supervised node classification is the most popular task for evaluating HGNN models [3, 9, 14, 21].

4 METHODOLOGY

In this section, we will first introduce our design of the auxiliary system. Then we will present our post-training algorithm which strengthens the two levels of consistencies. Finally, we will have a discussion about the proposed framework.

4.1 Formalization of HGNNs

As our framework is agnostic to the neural architecture of HGNNs, we simply treat them as black boxes. Formally, an HGNN is a layered network architecture, which takes an HIN $\mathcal{G}$ and the corresponding feature matrix $\mathcal{X}$ as input to calculate the representation of each node. The representations of the last layer will be normalized by a softmax operator to output the label distribution predictions. In this paper, we denote an HGNN model parameterized by Θ as f_Θ, where $f_\Theta(v) \in \mathbb{R}^{|\mathcal{Y}|}$ is the label distribution of node v predicted by the HGNN model.

Then the model parameters Θ can be optimized by minimizing the prediction error on labeled node set V_L:

$$\min_{\Theta} \sum_{v \in \mathcal{V}_L} \mathcal{L}(f_\Theta(v), y_v), \tag{1}$$

where $\mathcal{L}(\cdot, \cdot)$ denotes the loss function (*i.e.,* the cross entropy loss) between true label y_v and predicted label $f_\Theta(v)$.

4.2 Design of Auxiliary System

In order to complement with typical HGNNs, our auxiliary system consists of two modules to characterize the global and local dependency. As shown in Fig. 3, the two modules can infer the label distribution of a node separately, and their predictions can be further recombined by a learnable weighted average. Formally, we respectively denote the global and local inference modules as $g^G_{\Omega_1}$ and $g^L_{\Omega_2}$. The entire auxiliary system is represented as g_Ω where $\Omega = \{\Omega_1, \Omega_2\}$. Now we will introduce our global and local modules.

4.2.1 Global Inference Module. For global-level inference, we design a novel Multi-Channel Label Propagation (MCLP) process based on meta-paths. The basic assumption of MCLP is that nodes linked by a meta-path instance tend to have similar labels. In MCLP, labels will iteratively propagate from nodes to their meta-path based neighbors for inference. After several layers of propagation, the

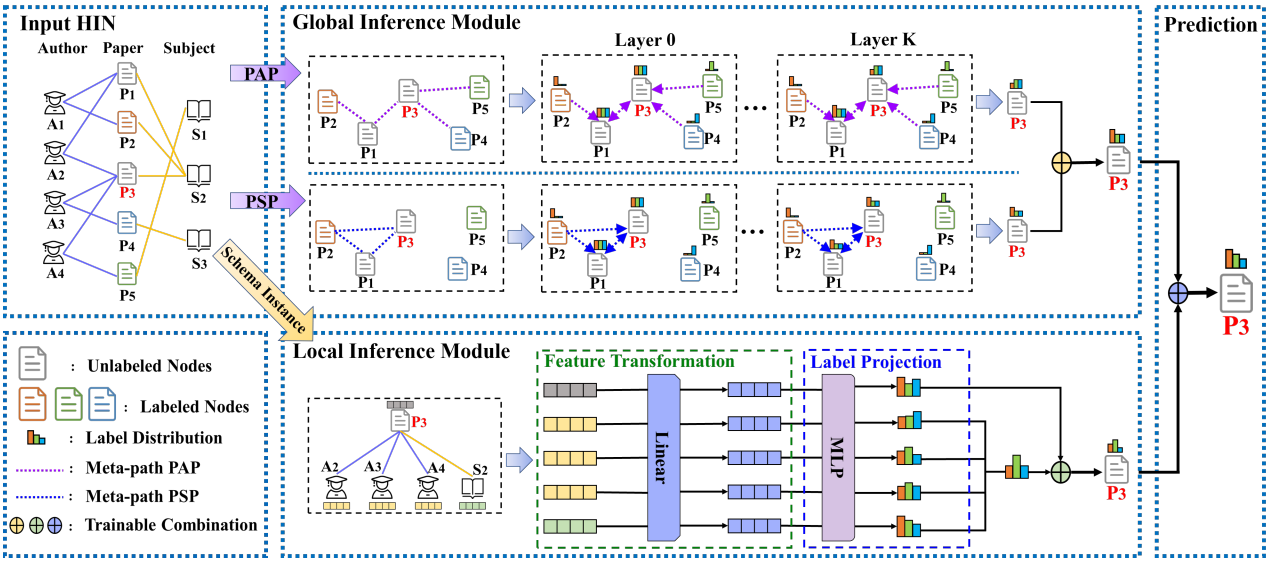


Figure 3: The architecture of our auxiliary system. The global module based on multi-channel label propagation and the local module based on network schema-aware prediction will infer the label of node P_3 separately. Then their predictions will be recombined as the final prediction of the auxiliary system.

labels eventually propagate from labeled nodes to unlabeled ones as predictions. Note that an HIN usually has multiple meta-paths that define different semantic relations between nodes. Hence we will perform label propagation along different meta-paths in independent channels, and then recombine the predictions from different channels as the output of the global module. In our implementation, the global module will diffuse labels to the nodes around 5 times farther than the receptive fields of typical HGNNs, and thus capture the label dependency between distant nodes.

Formally, we use l_P^k to denote the predictions of MCLP based on meta-path $P \in \mathcal{P}$ after k layers of propagation. We initialize the label prediction l_P^0 before propagation as follows:

$$l_P^0(v) = \begin{cases} (0, \dots 1, \dots 0) \in \mathbb{R}^{|\mathcal{Y}|}, & \forall v \in \mathcal{V}_L \\ (\frac{1}{|\mathcal{Y}|}, \dots \frac{1}{|\mathcal{Y}|}, \dots \frac{1}{|\mathcal{Y}|}) \in \mathbb{R}^{|\mathcal{Y}|}, & \forall v \in \mathcal{V}_U \end{cases}, \tag{2}$$

where each labeled node corresponds to a one-hot label vector and all unlabeled nodes correspond to the uniform distribution.

We assume that the importance (*i.e.*, propagation intensities) of different neighbors to a node should be different. Thus, for each meta-path instance connecting node u and v, we parameterize its propagation weight $w_{uv}^P \in [0, 1]$ as follows:

$$w_{uv}^P = \frac{\exp(s_{uv}^P)}{\sum_{u' \in \mathcal{N}_v^P} \exp(s_{u'v}^P)}, \tag{3}$$

where $\mathcal{N}_v^P$ is the set of node v's neighbors connected by meta-path instance of P, and $s_{uv}^P \in \mathbb{R}$ is a learnable parameter representing the propagation intensity between node u and v in channel P.

At each layer of label propagation, we will update the label distribution predictions of unlabeled nodes, and fix those of labeled nodes as one-hot vectors. Specifically, the update function of node label predictions in the $k+1$-th layer can be formalized as:

$$l_P^{k+1}(v) = \sum_{u \in \mathcal{N}_v^P} w_{uv}^P l_P^k(u), \tag{4}$$

where all the meta-path based neighbors of node v will compete to propagate their labels to v.

Then we recombine the predictions from different channels to calculate the final predictions of MCLP as below:

$$g_{\Omega_1}^G(v) = \sum_{P \in \mathcal{P}} \alpha_v^P l_P^K(v), \tag{5}$$

where K is the number of layers in MCLP, $\alpha_v^P \in [0, 1]$ is a trainable weight parameter for each pair of node v and meta-path P with $\sum_{P \in \mathcal{P}} \alpha_v^P = 1$, and $g_{\Phi_1}^G(v)$ denotes the final predictions for node v by the global inference module.

4.2.2 Local Inference Module. We also design a local-level module to characterize each node's local neighborhood. The local module will predict node labels merely based on every node's network schema instance, and thus exclude the influence of meta-path-based neighbors from potentially different categories.

According to the network schema proximity [29], all the nodes with different types in a network schema instance tend to be similar. To take advantage of this proximity, we assume that besides the nodes with type T, all the nodes with other types can also be projected into the same label space $\mathcal{Y}$. Then the nodes in a network schema instance will have similar labels. Therefore, in order to help predict the label of node v, we will average the label distributions of all the nodes that are in the same network schema instances with v for assistance.

Formally, we first apply a node type-specific transformation to project the features of different types of nodes into the same space:

$$\mathbf{h}_u = W_{\phi(u)} \cdot \mathbf{x}_u, \forall u \in \mathcal{V}, \tag{6}$$

where $\mathbf{h}_u$ is the projected feature of node u, $\mathbf{x}_u$ is the original feature of u, $\phi(u)$ is the type of node u, and $W_{\phi(u)}$ is the type-specific projection matrix.

Afterward, each node u will be projected into a label distribution $p_u \in \mathbb{R}^{|\mathcal{Y}|}$ by

$$p_u = \text{softmax}(\text{MLP}(\mathbf{h}_u)), \tag{7}$$

where $\text{MLP}(\cdot)$ is a multi-layer perceptron.

Then we denote the set of nodes that are in the same network schema instances with v as $\mathcal{N}_v$. The labels of every node $u \in \mathcal{N}_v$ will be averaged and then combined with p_v as the label distribution of node v predicted by the local module:

$$g^L_{\Omega_2}(v) = \beta_v p_v + (1-\beta_v)\frac{\sum_{u\in\mathcal{N}_v} p_u}{|\mathcal{N}_v|}, \tag{8}$$

where $\beta_v \in [0,1]$ is a node-specific trainable parameter to balance the two parts of predictions.

4.2.3 The Combination of Global and Local Modules. Finally, the label distribution of node v predicted by the auxiliary system is computed as

$$g_\Omega(v) = \gamma_v g^G_{\Omega_1}(v) + (1-\gamma_v) g^L_{\Omega_2}(v), \tag{9}$$

where $\gamma_v \in [0,1]$ is a node-specific trainable parameter to balance the global module and local module. The pseudo code of our auxiliary system is provided in Appendix A.

4.3 Post-training Algorithm

To facilitate the cooperation of the two systems, we propose to optimize system-level and module-level consistencies, *i.e.,* minimizing the prediction gap between HGNN f_Θ and auxiliary system g_Ω, as well as the gap between global module $g^G_{\Omega_1}$ and local module $g^L_{\Omega_2}$.

Given a pretrained HGNN model learned by its original training objective in Eq. 1, we will then optimize the two systems alternatively. Formally, we will update parameter $\Omega = \{\Omega_1, \Omega_2\}$ in the auxiliary system by

$$\min_{\Omega} \sum_{v\in\mathcal{V}_U} \text{dist}(f_\Theta(v), g_\Omega(v)) + \lambda \text{dist}(g^G_{\Omega_1}(v), g^L_{\Omega_2}(v)), \tag{10}$$

where $\text{dist}(\cdot,\cdot)$ denotes the distance function between two label distributions such as KL divergence or Euclidean distance, and λ is a hyper-parameter. We empirically set $\lambda = 0.3$ for all our experiments.

Afterward, we will update parameter Θ of HGNN by

$$\min_{\Theta} \sum_{v\in\mathcal{V}_U} \text{dist}(f_\Theta(v), g_\Omega(v)) + \sum_{v\in\mathcal{V}_L} \mathcal{L}(f_\Theta(v), y_v), \tag{11}$$

where the second term is the prediction error on labeled set to avoid trivial solutions, *e.g.,* both systems always predict a specific label $y \in \mathcal{Y}$.

By iteratively optimizing the two systems, they can learn from each other and both achieve better generalization ability. Note that both systems can be used for evaluation after training. We empirically find that the learned auxiliary system has better prediction accuracies. We name our overall framework as Heterogeneous Graph Post-training Framework, abbreviated as HGPF. The pseudo code of HGPF is provided in Appendix A. Note that the unlabeled node set $\mathcal{V}_U$ is further divided into the validation set $\mathcal{V}_D$ for model selection and test set $\mathcal{V}_S$ for final evaluation. We empirically set the maximum number of iterations and epochs $N = 5$ and $M = 150$ for all experiments.

4.4 Discussion

4.4.1 Computational Complexity. The time and space complexity of our framework is linear to the scale of an HIN, *i.e.,* the number of nodes, edges, features and meta-path instances. In fact, the training of HGPF is very time efficient. For example, if we apply HGPF on HAN [21], the running time of HGPF on ACM dataset is about 5 minutes with a single GPU device of GeForce RTX 3090.

4.4.2 System Complementarity. Existing HGNN models can only have one or two stacked message passing layers to avoid the over-smoothing issue. In contrast, the MCLP process of our auxiliary system can be stacked for $8 \sim 10$ layers to achieve better results. Thus, our auxiliary system can utilize more global information than typical HGNNs. Besides, many meta-path based HGNN models such as HAN [21] failed to consider the relationship between nodes with different types but in the same network schema instances. Hence the local module of our auxiliary system can also provide complementary knowledge with HGNNs.

4.4.3 Model Compatibility. Our proposed framework is agnostic to the architecture of HGNNs, and thus can be integrated with any HGNN models for implementation. After the post-training phase, we can offer more accurate predictions and have a high compatibility in improving HGNN models.

5 EXPERIMENTS

In this section, we will conduct experiments to answer the following research questions: • **RQ1**: Can HGPF improve HGNNs on semi-supervised node classification task? • **RQ2**: How does HGPF perform compared with SOTA graph algorithms? • **RQ3**: How does HGPF perform under different settings, *i.e.,* training ratios, ablated models, and hyper-parameters? • **RQ4**: How about the interpretability of learned parameters in the auxiliary system?

5.1 Datasets

We adopt three benchmark datasets including ACM [29], DBLP [3] and IMDB [3] for evaluation. The detailed statistics and descriptions are listed in Appendix B. For semi-supervised node classification task, we randomly choose 20 or 50 labeled nodes per class as the training set, 50 nodes per class as the validation set, and the remaining nodes as the test set. Following previous HGNNs [3, 14, 21], we also employ Micro-F1 and Macro-F1 as evaluation metrics. For auxiliary experiments in Section 5.5, we only report the Micro-F1 metric on ACM for brevity. For the analysis related to interpretability in Section 5.6, we will focus on DBLP, which has richer semantics with the largest numbers of node types, edge types and meta-paths.

5.2 Experimental Settings

Here we only present some key settings and more detailed experimental settings are provided in Appendix D.

5.2.1 System Settings. To prove the effectiveness of HGPF, we explore four typical HGNN models, including HAN [21], HGT [9], Simple-HGN [14] and MAGNN [3]. Since the implementation with

MAGNN is much slower than that with the other three HGNNs, we only evaluate MAGNN in main experiments. All the HGNNs are carefully tuned according to our dataset splits, and a sufficient training can be guaranteed during the pretraining phase. For the hyper-parameter setting of our auxiliary system, we explore MCLP layers K from 6 to 14, and use a 2-layer MLP with hidden size as 128 in our local module.

5.2.2 Optimization Settings. We will alternatively run the two systems for 150 epochs in each iteration, and choose the best epoch and iteration according to the performance on the validation set.

5.2.3 Models for Comparisons. In the experiments, we will consider three models in each group of comparison:

• **Pretrain**. The pretrained HGNN, *i.e.,* a standard HGNN.

• **HGPF$_{\text{self}}$**. A variant of HGPF where the auxiliary system is replaced by the same HGNN without sharing parameters.

• **HGPF**. The learned auxiliary system HGPF$_{\text{AS}}$ in our HGPF. We omit the subscript of AS for brevity. Note that the performance of updated HGNN HGPF$_{\text{HGNN}}$ will be discussed in RQ3, and HGPF will refer to learned auxiliary system unless specified.

5.3 Performance on Node Classification (RQ1)

Experimental results on three datasets with four HGNNs are presented in Table 1 and 2. We bold the best result in each group of comparison, and have the following observations:

(1) Comparing HGPF with Pretrain, we can find that our learned auxiliary system consistently outperforms the pretrained HGNN in all 48 groups of comparisons. The relative improvements of Micro-F1 and Macro-F1 are respectively 3.90% and 3.94% on average. Also, the standard derivation of HGPF is about ±0.3 and thus the improvements are significant. Hence our proposed framework can be successfully integrated with all four HGNNs, and offer more accurate predictions.

(2) Comparing HGPF with HGPF$_{\text{self}}$ where the auxiliary system has the same architecture with pretrained HGNN, we can see that our carefully designed auxiliary system has better performance in 47 out of 48 groups of comparisons. The relative improvements of Micro-F1 and Macro-F1 are respectively 1.87% and 2.04% on average. This observation shows that our auxiliary system can better complement with typical HGNNs, which demonstrates the effectiveness of our model design.

(3) Comparing HGPF$_{\text{self}}$ with Pretrain, though two systems have the same architecture, we can still get some improvement by our optimization framework. A possible reason is that the two systems are optimized into different local minimums, and thus can also complement with each other to some extent. Different from ensemble, the learned system won't increase the complexity at test stage.

5.4 Comparison with SOTA GNNs (RQ2)

Besides the four HGNNs used in our HGPF, in this subsection we conduct experiments to compare with more state-of-the-art GNNs, including two HGNNs (HGSL [28], HeCo [22]) and three frameworks (RDD [27], CPF [24], HGNN-AC [10]) that can be applied on any GNNs. For a fair comparison, we use MAGNN [3] as the backbone in the three frameworks and our HGPF. Also, since a recent work [14] mentioned that typical homogeneous GNNs perform well for HINs in practice, we add GCN [11] and GAT [20] for comparison as well. Figure 4 shows the comparison results on the three datasets with 50 labeled nodes per class. The relative improvements of Micro-F1 against best performed baselines on three datasets are respectively 1.10%, 1.13%, 3.12%, which demonstrates that our HGPF is the SOTA method for this task. Detailed settings of the seven baselines are provided in Appendix D.

5.5 Performance under Different Settings (RQ3)

5.5.1 Analysis of Different Training Ratios. In this subsection, we conduct experiments under different training ratios to further prove the robustness of our framework. Specifically, we report the classification results with 10, 20, 50, and 100 labeled nodes per class. From Figure 5, we can see that the results of HGPF are consistently better than both the pretrained HGNN models and HGPF$_{\text{self}}$ when the number of labeled nodes increases. Note that HGPF can even outperform pretrained HGNNs with only 1/5 or 1/10 labeled data. For example, the performance of HGPF with 10 labeled nodes per class is better than that of pretrained HAN with 100 labeled nodes per class. Hence our framework is stable and effective with different training ratios.

5.5.2 Ablation Study of HGPF. To further demonstrate the effectiveness of our global and local inference modules, we design two variants of HGPF:

• **HGPF$_{\mathbf{G}}$**: The variant of HGPF with only global inference module in the auxiliary system.

• **HGPF$_{\mathbf{L}}$**: The variant of HGPF with only local inference module in the auxiliary system.

Experimental results on ACM dataset are shown in Figure 6. We can see that HGPF always has better performance than two ablated models. Compared with HGPF$_{\text{L}}$, HGPF has 1.23% relative improvement on average. Hence, although HGPF$_{\text{G}}$ performs worst among the three, the global inference module is still an indispensable part of our auxiliary system. This experiment demonstrates the necessity of both global and local modules.

5.5.3 Analysis of Hyper-parameters. In this subsection, we will investigate the influence of two key hyper-parameters of HGPF

(1) **Maximum Number of Iterations in HGPF**: Figure 7 presents the classification results of HGPF$_{\text{HGNN}}$ and HGPF$_{\text{AS}}$ in first 5 iterations. Compared with pretrained HGNN in iteration 0, we can see that the performance of both updated HGNN and learned auxiliary system can be effectively improved within 5 iterations. Hence we empirically set the maximum iteration number $N = 5$.

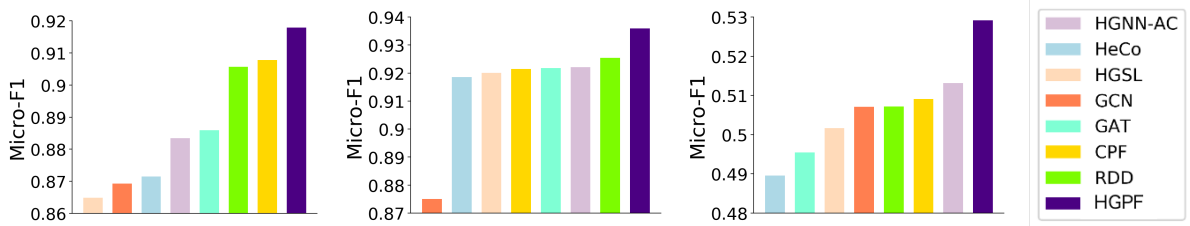
(2) **Number of Layers in MCLP**: Figure 8 shows the performance of HGPF with label propagation layers $K \in \{6, 8, 10, 12, 14\}$. We can observe that the performance of HGPF is stable as the layer number K changes within a reasonable range. Even the worst performance with K from 6 to 14 has already outperformed pretrained HGNN and HGPF$_{\text{self}}$. Also, HGPF usually achieves the best performance with K from 6 to 10, which is much larger than the number of stacked layers in typical HGNNs and thus enables a much wider receptive field. By propagating labels instead of representations, we barely encounter the over-smoothing issue.

Table 1: Classification performance with HAN [21] and HGT [9].

Models		HAN						HGT					
		Pretrain		$\text{HGPF}_{\text{self}}$		HGPF		Pretrain		$\text{HGPF}_{\text{self}}$		HGPF	
# Labeled Nodes		20	50	20	50	20	50	20	50	20	50	20	50
ACM	Micro-F1	0.8826	0.8838	0.8949	0.9091	**0.9163**	**0.9271**	0.8693	0.8701	0.8835	0.8852	**0.9173**	**0.9177**
	Macro-F1	0.8785	0.8864	0.8901	0.9054	**0.9165**	**0.9280**	0.8679	0.8699	0.8827	0.8807	**0.9173**	**0.9174**
DBLP	Micro-F1	0.9092	0.9217	0.9251	0.9300	**0.9280**	**0.9349**	0.8941	0.9256	0.9011	0.9317	**0.9084**	**0.9342**
	Macro-F1	0.9038	0.9165	0.9220	0.9242	**0.9258**	**0.9287**	0.8871	0.9229	0.8925	0.9239	**0.8995**	**0.9275**
IMDB	Micro-F1	0.4581	0.4809	0.4629	0.5060	**0.4879**	**0.5149**	0.4600	0.5067	0.4672	0.5117	**0.4724**	**0.5286**
	Macro-F1	0.4346	0.4817	0.4574	0.5059	**0.4792**	**0.5189**	0.4540	0.5123	**0.4623**	0.5139	0.4611	**0.5328**

Table 2: Classification performance with Simple-HGN [14] and MAGNN [3].

Models		Simple-HGN						MAGNN					
		Pretrain		$\text{HGPF}_{\text{self}}$		HGPF		Pretrain		$\text{HGPF}_{\text{self}}$		HGPF	
# Labeled Nodes		20	50	20	50	20	50	20	50	20	50	20	50
ACM	Micro-F1	0.8816	0.8865	0.8945	0.8994	**0.9179**	**0.9216**	0.8776	0.8831	0.8917	0.9022	**0.9157**	**0.9179**
	Macro-F1	0.8815	0.8881	0.8895	0.8943	**0.9179**	**0.9210**	0.8715	0.8824	0.8923	0.9014	**0.9112**	**0.9173**
DBLP	Micro-F1	0.9108	0.9253	0.9245	0.9315	**0.9279**	**0.9366**	0.9121	0.9223	0.9271	0.9322	**0.9291**	**0.9359**
	Macro-F1	0.9026	0.9249	0.9152	0.9286	**0.9177**	**0.9316**	0.9056	0.9228	0.9234	0.9269	**0.9252**	**0.9295**
IMDB	Micro-F1	0.4698	0.5109	0.4798	0.5318	**0.4925**	**0.5412**	0.4518	0.5090	0.4877	0.5173	**0.4962**	**0.5292**
	Macro-F1	0.4562	0.5141	0.4522	0.5296	**0.4874**	**0.5396**	0.4515	0.5119	0.4880	0.5204	**0.4921**	**0.5256**



(a) ACM (b) DBLP (c) IMDB

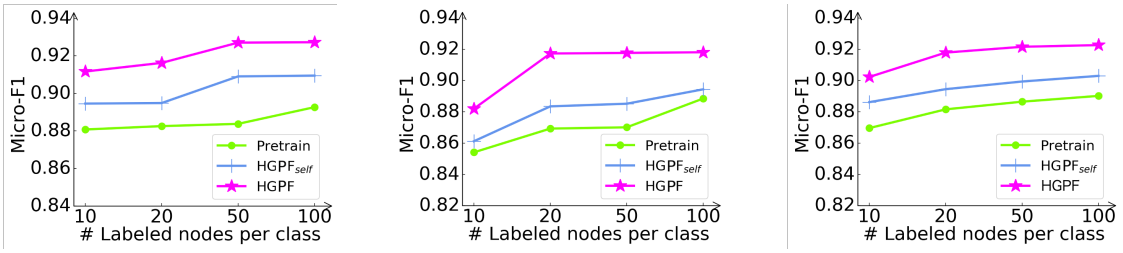
Figure 4: Performance of HGPF and seven state-of-the-art GNNs and HGNNs on the three datasets.

5.6 Interpretability Analysis (RQ4)

To answer RQ4, we conduct experiments on DBLP dataset, and investigate whether the learned auxiliary system can properly assign label recombination weights to different components. Specifically, we focus on analyzing the learned balance parameters in Eq. 9, 5 and 8, *i.e.*, γ_v between global and local modules, α_v^P among meta-paths, β_v between a node itself and its network schema-based neighbors. We will average the balance parameters over all nodes, and report the results in Figure 9. From the results, we have the following observations:

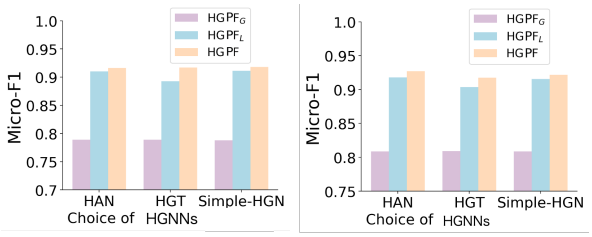
(1) As shown in Figure 9(a), we can see that average γ_v will increase with the number of labeled nodes, which indicates that the global module becomes more important as training ratio increases. Note that the global module will propagate labels from labeled nodes to unlabeled ones for inference. Thus, more labeled nodes will help propagate more accurately and improve the importance of the global module. In contrast, the local module is based on node features and local network schema instances, which are less sensitive to the number of labeled nodes. Therefore, our proposed HGPF can automatically adjust the importance of global/local modules.

(2) As shown in Figure 9(b), we find that the meta-path (APVPA) always gets the largest weight α_v^P among the three. The partial order APVPA>APA>APTPA is consistent and independent with the choice of HGNNs. Note that the labels of authors in DBLP dataset are determined by their research areas. Hence meta-path APVPA which links two authors publishing papers in the same venue will be more suitable to propagate labels than the other two



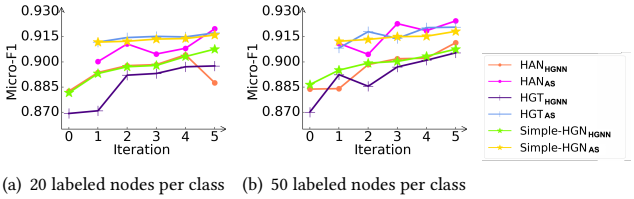
(a) HAN (b) HGT (c) Simple-HGN

Figure 5: Classification performance on ACM dataset under different training ratios.



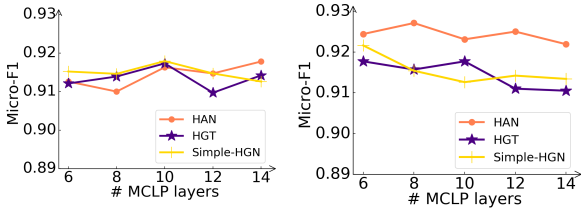
(a) 20 labeled nodes per class. (b) 50 labeled nodes per class.

Figure 6: Ablation study on global/local modules on ACM.



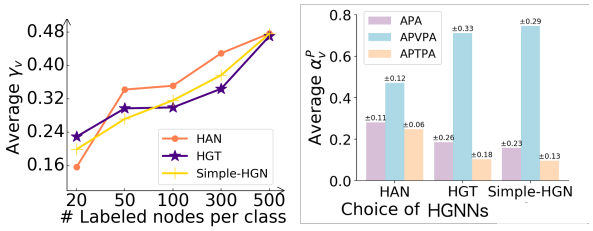
(a) 20 labeled nodes per class (b) 50 labeled nodes per class

Figure 7: Performance of HGPF$_{HGNN}$ and HGPF$_{AS}$ in 5 iterations on ACM dataset. Iteration 0 indicates the pretrained HGNN. Legend example: HAN$_{AS}$ represents the learned auxiliary system HGPF$_{AS}$ based on HAN.



(a) 20 labeled nodes per class (b) 50 labeled nodes per class

Figure 8: Performance of HGPF with different layer numbers in MCLP. Legend example: HAN represents the learned auxiliary system HGPF$_{AS}$ based on HAN.



(a) Average γ_v (b) Average α_v^P

Figure 9: Analysis of learned balance parameters, including γ_v between global and local modules and α_v^P among meta-paths. We report the averages of γ_v and α_v^P over all nodes.

meta-paths. This observation demonstrates that our global module is able to automatically adjust the weights of different meta-paths.

(3) When we apply HGPF on HAN, HGT and Simple-HGN, the averages of β_v are 0.4734, 0.4555, 0.4221, respectively. Hence a node and its network schema-based neighbors are almost equally important in our local module, which reflects the necessity of utilizing nodes in the same network schema instances for modeling.

6 CONCLUSION

In this paper, we propose a post-training framework HGPF to improve HGNNs for semi-supervised node classification. To alleviate the limitations of HGNNs, we design an auxiliary system with a complementary prediction mechanism: a global inference module based on multi-channel label propagation and a local inference module based on network schema-aware prediction. Two levels of consistency can encourage the cooperation between two systems. Experimental results on three benchmark datasets with four typical HGNNs demonstrate the effectiveness of HGPF. For future work, an interesting direction is to generalize our framework for other graph tasks (*e.g.*, link prediction and clustering) or graph types (*e.g.*, dynamic and heterophily graphs) with proper auxiliary systems.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments. This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 62002029, 62192784, 62172052, 62172052, U1936104).

REFERENCES

[1] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of SIGKDD*, 2019.

[2] Y. Fan, S. Hou, Y. Zhang, Y. Ye, and M. Abdulhayoglu. Gotcha-sly malware! scorpion a metagraph2vec based malware detection system. In *Proceedings of SIGKDD*, 2018.

[3] X. Fu, J. Zhang, Z. Meng, and I. King. MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*, 2020.

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of ICML*, 2017.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of ICML*, 2017.

[6] Y. Hei, R. Yang, H. Peng, L. Wang, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun. Hawk: Rapid android malware detection through heterogeneous graph attention networks. *IEEE TNNLS*, 2021.

[7] H. Hong, H. Guo, Y. Lin, X. Yang, Z. Li, and J. Ye. An attention-based graph neural network for heterogeneous structural learning. In *AAAI*, 2020.

[8] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu. Hindroid: An intelligent android malware detection system based on structured heterogeneous information network. In *KDD*, 2017.

[9] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of WWW*, 2020.

[10] D. Jin, C. Huo, C. Liang, and L. Yang. Heterogeneous graph neural network via attribute completion. In *Proceedings of WWW 2021*, 2021.

[11] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*, 2017.

[12] Q. Long, L. Xu, Z. Fang, and G. Song. HGK-GNN: heterogeneous graph kernel based graph neural networks. In *Proceedings of SIGKDD*, 2021.

[13] Y. Lu, Y. Fang, and C. Shi. Meta-learning on heterogeneous information networks for cold-start recommendation. In R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, editors, *KDD*, 2020.

[14] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of SIGKDD*, 2021.

[15] M. Qu, Y. Bengio, and J. Tang. Gmnn: Graph markov neural networks. In *ICML*, 2019.

[16] Y. Ren, B. Liu, C. Huang, P. Dai, L. Bo, and J. Zhang. Heterogeneous deep graph infomax. *arxiv*, 2019.

[17] C. Shi, Y. Fang, Y. Ye, and J. Zhang. The 4th workshop on heterogeneous information network analysis and applications (HENA 2021). In F. Zhu, B. C. Ooi, and C. Miao, editors, *KDD*, 2021.

[18] Y. Sun and J. Han. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 2013.

[19] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Heterogeneous information networks: the past, the present, and the future. *VLDB*, 2022.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *Proceedings of ICLR*, 2018.

[21] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *Proceedings of WWW*, 2019.

[22] X. Wang, N. Liu, H. Han, and C. Shi. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of SIGKDD*, 2021.

[23] F. Xu, J. Lian, Z. Han, Y. Li, Y. Xu, and X. Xie. Relation-aware graph convolutional networks for agent-initiated social e-commerce recommendation. In *CIKM*, 2019.

[24] C. Yang, J. Liu, and C. Shi. Extract the knowledge of graph neural networks and go beyond it: An effective knowledge distillation framework. In *WWW*, 2021.

[25] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. In *Proceedings of NeurIPS*, 2019.

[26] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of SIGKDD*, 2019.

[27] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui. Reliable data distillation on graph convolutional network. In *Proceedings of SIGMOD*, 2020.

[28] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye. Heterogeneous graph structure learning for graph neural networks. In *Proceedings of AAAI*, 2021.

[29] J. Zhao, X. Wang, C. Shi, Z. Liu, and Y. Ye. Network schema preserving heterogeneous information network embedding. In *Proceedings of IJCAI*, 2020.

[30] J. Zheng, Q. Ma, H. Gu, and Z. Zheng. Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation. In F. Zhu, B. C. Ooi, and C. Miao, editors, *KDD*, 2021.

[31] S. Zhu, C. Zhou, S. Pan, X. Zhu, and B. Wang. Relation structure-aware heterogeneous graph neural network. In *Proceedings of ICDM*, 2019.

A PSEUDO CODE

Alg. 1 and 2 show the pseudo code of our auxiliary system and the entire framework HGPF, respectively.

Algorithm 1: The Implementation of auxiliary system

Input : HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \phi, \varphi\}$,
node features $\mathcal{X} = \{\mathbf{x}_v | \forall v \in \mathcal{V}\}$,
target node type $T \in \mathcal{T}$,
labeled node set $\mathcal{V}_L \subset \mathcal{V}_T$,
unlabeled node set $\mathcal{V}_U = \mathcal{V}_T \setminus \mathcal{V}_L$,
metapath set $\mathcal{P}$,
number of layers K in the global module;
Output : Predicted label distribution $g_\Omega(v)$ for every unlabeled node $v \in \mathcal{V}_U$;

```
// Global Inference Module:
for P ∈ 𝒫 do
    Initialize the label prediction l_P^0(v) for v ∈ 𝒱_T by Eq. 2;
    for k = 1, 2 ... K do
        for v ∈ 𝒱_U do
            Update the label prediction of v by Eq. 4;
        end
    end
end
for v ∈ 𝒱_U do
    Combine the predictions from different channels as the
      global-level output by Eq. 5;
end
// Local Inference Module:
for v ∈ 𝒱 do
    Perform type-specific feature transformation by Eq. 6;
    Perform label projection by Eq. 7;
end
for v ∈ 𝒱_U do
    Compute the local-level output by Eq. 8:
end
// Combination of Two Modules:
for v ∈ 𝒱_U do
    Compute the prediction of auxiliary system by Eq. 9;
end
```

Algorithm 2: The Overall Framework of HGPF

Input : HIN $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \phi, \varphi\}$,
node features $\mathcal{X}$,
labeled node set $\mathcal{V}_L \subset \mathcal{V}_T$,
unlabeled node set $\mathcal{V}_U = \mathcal{V}_T \setminus \mathcal{V}_L$,
validation node set $\mathcal{V}_D \subset \mathcal{V}_U$,
test node set $\mathcal{V}_S = \mathcal{V}_U \setminus \mathcal{V}_D$,
number of iterations N, number of epochs M;
Output : Trained node label predictors f_Θ, g_Ω.

```
Pretrain the HGNNs with labeled node set 𝒱_L by Eq. 1;
for i = 1, 2 ... N do
    for epoch = 1, 2 ... M do
        Update parameter Ω of auxiliary system by Eq. 10;
        Evaluate Ω on validation set 𝒱_D;
    end
    Select Ω as the epoch with the best performance on 𝒱_D;
    for epoch = 1, 2 ... M do
        Update parameter Θ of HGNNs by Eq. 11;
        Evaluate Θ on validation set 𝒱_D;
    end
    Select Θ as the epoch with the best performance on 𝒱_D;
end
```

B DETAILS AND STATISTICS OF DATASETS

Table 3 shows the statistics of three datasets, and the detailed descriptions about the three datasets are as follows:

• **ACM**[1] [29] is a citation network with the target node type as Paper (P). All the papers in the HIN are divided into three classes: database, wireless communication, and data mining. We use the bag-of-words representations of keywords as the node features of paper nodes.

• **DBLP**[2] [3] is a bibliography website of computer science. The target node type is Author (A), and the authors are divided into four classes according to their research areas (database, data mining, artificial intelligence, and information retrieval). The features of authors are also described by bag-of-words representations of their paper keywords.

• **IMDB**[3] [3] is a website about movies and television programs. We use an HIN with the target node type as Movie (M). All the movies are labeled as one of three classes (action, comedy, and drama) according to their genres. The features of movies are described by a bag-of-words representation of its plot keywords.

For all three datasets, the features of nodes with other types are one-hot vectors.

C IMPLEMENTATION DETAILS

We implement HGPF based on PyTorch and Deep Graph Library (DGL)[4]. For all experiments, we employ the GPU device of GeForce RTX 2080 and 3090.

D DETAILS OF EXPERIMENTAL SETTINGS

In this section, we will describe detailed settings of our HGPF.

D.1 HGNNs Settings

In this subsection, we provide the detailed settings of the four HGNNs used in our HGPF.

• **HAN** [21] is an HGNN model which learns meta-path-specific node representations by leveraging node-level attention and semantic-level attention mechanism. We use 8 attention heads and 64-dimensional hidden size in our experiments.

• **HGT** [9] is a transformer-based HGNN model with heterogeneous subgraph sampling. We employ a 2-layer HGT with 4 to

[1] https://github.com/Andy-Border/NSHE
[2] https://github.com/cynricfu/MAGNN
[3] https://github.com/cynricfu/MAGNN
[4] https://github.com/dmlc/dgl

Table 3: Dataset statistics with more details.

Dataset	# Nodes	# Edges	# Features	# Meta-paths	# Classes	# Training	# Validation	# Test
ACM	Author (7167) Paper (4019) Subject (60)	P-A (13407) P-S (4019)	1902	P-A-P P-S-P	3	20×3 50×3	50×3 50×3	3809 3719
DBLP	Author (4057) Paper (14328) Term (7723) Venue (20)	P-A (19645) P-T (85810) P-V (14328)	334	A-P-A A-P-V-P-A A-P-T-P-A	4	20×4 50×4	50×4 50×4	3777 3657
IMDB	Movie (4278) Director (2081) Actor (5257)	M-D (4278) M-A (12828)	3066	M-A-M M-D-M	3	20×3 50×3	50×3 50×3	4068 3978

8 attention heads and 256-dimensional hidden size in our experiments.

• **Simple-HGN** [14] is an improved version of GAT [20] with learnable edge-type embedding and residual connections. In our experiments, we use a 2-layer Simple-HGN with attention heads from 4 to 8, hidden size from $\{64, 128, 256\}$ and 32-dimensional edge-type embedding.

• **MAGNN** [3] is an improved HAN with several meta-path encoders to embed all the nodes along a meta-path. In our experiments, we use 8 attention heads, hidden size from $\{64, 128, 256\}$, attention vector dimension from $\{32, 64, 128\}$, and batch size from $\{16, 32, 64, 128, 256, 512\}$.

For the pretraining settings of HGNNs, we will run HGNN models for 150 epochs, and choose the best epoch according to the performance on the validation set. For hyper-parameter settings, we use dropout rate from 0.2 to 0.8, learning rate from 0.001 to 0.05, and weight decay of Adam optimizer from $\{0, 0.0001, 0.0005, 0.001\}$.

We also tried GTN [25], but omit its results since GTN performs worse (either with or without HGPF) than the above four HGNNs under our experimental settings.

D.2 Auxiliary System Settings

We explore dropout rate of local module from $\{0.4, 0.5, 0.6, 0.7\}$, and use Adam optimizer with learning rate as 0.01 for training, the weight decay of the optimizer is set as 0.0005 for ACM/IMDB and 0 for DBLP.

D.3 Optimization Settings

For distance functions, we use Euclidean distance between two systems, and KL-divergence between global and local modules in Eq. 10. For Eq. 11, we employ the KL-divergence function. We use Xavier normal distribution to initialize parameters.

D.4 Detailed Settings in Section 5.4

In this subsection, we provide the detailed settings of the models in Section 5.4.

• **GCN** [11]: We set hidden size $k = 128$ for all datasets. For ACM, we set layers $L = 2$, we set layers $L = 3$ for DBLP and IMDB.

• **GAT** [20]: We set hidden size $k = 64$, layers $L = 3$, the number of attention heads $n = 4$ and negative slope $s = 0.05$ for all datasets.

• **CPF** [24]: We set hidden size $k = 64$, LP layers $L = 5$, MLP layers $l = 2$ for all datasets.

• **RDD** [27]: We set hidden size $k = 64$, parameters $\gamma = 3$ and $\beta = 10$ for all datasets.

• **HGNN-AC** [10]: We employ MAGNN as the backbone of HGNN-AC, and we set hidden size $k = 128$, attention vector size $a = 128$, and the number of attention heads $n = 8$ for all datasets. For DBLP, we set the batch size $bz = 8$.

• **HeCo** [22]: We set hidden size $k = 256$ for ACM and IMDB, for DBLP, we set $k = 64$.

• **HGSL** [28]: We set hidden size $k = 64$ for DBLP, for ACM and IMDB, we set $k = 256$. We set the number of heads $h = 2$ for all datasets.

D.5 Detailed Settings in Section 5.6

For γ_v between global and local modules in Eq. 9: We conduct experiments by gradually increasing the number of labeled nodes, and report the changing trend of the average of γ_v over all node $v \in \mathcal{V}_T$.

For α_v^P among meta-paths in Eq. 5: To prove that our global inference module is capable of identifying the importance of different meta-paths, we report the average of α_v^P over all node $v \in \mathcal{V}_T$ for every meta-path P. Here we use 100 labeled nodes per class for training, because there are larger weights of the global module (*i.e.,* γ_v) under this setting and the difference of α_v^P will contribute more to the final prediction of auxiliary system. In addition, we also mark the standard deviation of α_v^P on each column in Figure 9(b). We can see that the weights of different nodes can vary greatly, which means that it is necessary to use node-specific balance parameters for modeling.

For β_v which balances the importance of the label distribution of a node itself and its network schema-based neighbors in Eq. 8: We also use 100 labeled nodes per class as α_v^P for training for consistency.

E TIME AND MEMORY COST OF HGPF

We report the time cost to run a single epoch for the four HGNNs and AS on the ACM dataset in Table 4. The running speed of AS is second only to SimpleHGN and much faster than that of MAGNN and HGT. In addition, we also compared the memory consumption

of HGPF and pretrained HGNNs on the ACM dataset in Table 5. The memory cost of HGPF is similar to that of pretrained HGNNs, which indicates that the memory overhead of HGPF is negligible.

Table 4: Training time per epoch.

	HAN	MAGNN	SimpleHGN	HGT	AS
Time	0.185s	0.581s	0.093s	0.331s	0.171s

Table 5: Memory cost during training.

	HAN	MAGNN	SimpleHGN	HGT
Pretrain	4591MB	9108MB	3865MB	5791MB
HGPF	4665MB	9243MB	3873MB	5846MB

F CASE STUDY

As illustrated in the introduction section, there are two types of nodes difficult to predict by HGNNs: (1) nodes far from training nodes of same category; (2) nodes with more training nodes from different categories than the same category in the receptive field. Here we present case study for such "hard nodes" to show the effectiveness of HGPF. We select two representative nodes from each type on ACM dataset for analysis: HAN predicts these cases incorrectly, while HGPF can predict their accurate labels.

Case 1 (ID 2883): The node has a distance greater than 4 from all training nodes of the same category. In HGPF, the node-specific balance parameter $\gamma_v = 0.59$, which means that the prediction of this case is more dependent on the global module to capture more information from remote training nodes. And the meta-path balance parameter $\alpha_v^{PAP} = 0.41$, $\alpha_v^{PSP} = 0.59$, which means meta-path PSP is more helpful for this case.

Case 2 (ID 2948): Similar to case 1, there is no training node with a distance less than or equal to 4 from this node. The parameter γ_v of this case is 0.73, the prediction of global module also has larger weight than local module. And the meta-path balance parameter $\alpha_v^{PAP} = 0.49$, $\alpha_v^{PSP} = 0.51$, which means that both meta-paths PAP and PSP are important for this node.

Case 3 (ID 3499): Among the 4-hop neighbors of this case, the numbers of training nodes with label 0, 1, and 2 are 4, 20, and 3 respectively, and HAN incorrectly predicts it as label 1, since the training node neighbors with label 1 disturb the prediction of HGNNs. For HGPF, the γ_v of this case is 0.31, which means that the local module play a more important role in the prediction of this case than global module. The network schema based local module can effectively reduce the noise cause by training node neighbors from different categories.

Case 4 (ID 3032): Similar to case 3, in its 4-hop neighborhood, the number of training nodes with label 0, 1, and 2 are 15, 1, and 12 respectively. In this case, the γ_v of this case is only 0.13, the prediction highly relies on the local module to reduce the noise from training node neighbors from different categories.

As shown from the above case studies, for HGPF, the prediction of case 1 and case 2 (type 1) is more dependent on the global inference module, while the prediction of case 3 and case 4 (type 2) is more dependent on the local inference module, which is consistent with our motivation.