Few-shot Learning for Heterogeneous Information Networks

YANG FANG, National University of Defense Technology, Changsha, China
XIANG ZHAO*, Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China
WEIDONG XIAO, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China
MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands

Heterogeneous information networks (HINs) are a key resource in many domain-specific retrieval and recommendation scenarios, and in conversational environments. Current approaches to mining graph data often rely on abundant supervised information. However, supervised signals for graph learning tend to be scarce for a new task and only a handful of labeled nodes may be available. Meta-learning mechanisms are able to harness prior knowledge that can be adapted to new tasks.

In this paper, we design a meta-learning framework, called META-HIN, for few-shot learning problems on HINs. To the best of our knowledge, we are among the first to design a unified framework to realize the few-shot learning of HINs and facilitate different downstream tasks across different domains of graphs. Unlike most previous models, which focus on a single task on a single graph, META-HIN is able to deal with different tasks (node classification, link prediction, and anomaly detection are used as examples) across multiple graphs. Subgraphs are sampled to build the support and query set. Before being processed by the meta-learning module, subgraphs are modeled via a structure module to capture structural features. Then, a heterogeneous GNN module is used as the base model to express the features of subgraphs. We also design a GAN-based contrastive learning module that is able to exploit unsupervised information of the subgraphs.

In our experiments, we fuse several datasets from multiple domains to verify META-HIN's broad applicability in a multiple-graph scenario. META-HIN consistently and significantly outperforms state-of-the-art alternatives on every task and across all datasets that we consider.

CCS Concepts: • **Information systems** → **Network data models**.

Additional Key Words and Phrases: Heterogeneous information network; Few-shot learning; Meta-learning; Graph mining

1 INTRODUCTION

Heterogeneous information networks (HINs) are ubiquitous. Social networks, knowledge graphs, and interactions between users and items in search and recommender systems can be modeled as networks with multiple types of nodes and edges [12, 30, 42, 45]. Unlike homogeneous networks, which assume that every node is of a single type, HINs have a richer repertoire of means to describe networks. This leads to more effective solutions to a wide spectrum of information retrieval, data mining, and knowledge discovery tasks, e.g., node classification,

*Corresponding author.

Authors' addresses: Yang Fang, National University of Defense Technology, Changsha, China, fangyang12@nudt.edu.cn; Xiang Zhao, Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China, xiangzhao@nudt.edu.cn; Weidong Xiao, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China, wdxiao@nudt.edu.cn; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands, m.derijke@uva.nl.

ACM 1046-8188/2024/2-ART
https://doi.org/10.1145/3649311

link prediction, and recommendation [4, 9].

Representation learning is essential for mining a HIN [28, 29, 60]. Recent efforts resort to graph neural networks (GNNs) to achieve promising results [24, 55]. During the representation learning process, it is taken for granted that the majority of labels in the network is available, and the GNNs are trained in a supervised manner. In practice, however, it is common that only a handful of labels are given, which poses serious challenges to keeping up the performance. In order to effectively mine HINs with scarce labels, we investigate *few-shot learning* problems on HINs in this paper.

Inspired by meta-learning approaches that have been studied extensively in computer vision, there is an emerging line of research that applies meta-learning to few-shot learning of graph data [2, 8, 13, 22, 61]. In this line of work, well-trained initial parameters of a base GNN are learned and then the learned base-learner is adapted to new tasks, following a standard *model-agnostic meta-learning* (MAML) framework [10].

Current limitations. Various limitations hinder the application of meta-learning approaches to HINs. First, most of them are designed for *homogeneous* networks, with relatively few prior publications so far trying to solve few-shot learning problems on HINs. Second, most of them can only deal with one task on a single graph. For example, Meta-GNN [61] and Meta-MGNN [13] are only designed for a node classification task and cannot be transferred across different graphs, especially those having different distributions; Meta-Graph [2] is only designed for a link prediction task on a single graph; Meta-GDN [8] is devised for anomaly detection on a single graph; and Meta-HIN [32] is designed for cold-start recommendation on a single graph. Third, most previous methods overlook the unlabeled information in graphs; making full use of abundant unsupervised information could help improve the performance. Finally, the question of how to leverage the structural information of HINs has not been studied extensively by existing approaches.

Our proposal. We aim to address the shortcomings listed above, and propose (i) a heterogeneous GNN module as base model to fully capture heterogeneous information, (ii) a general framework that can be applied across different tasks and graphs, (iii) a GAN-based contrastive module to leverage unsupervised information, and (iv) a structure module to employ a graph's structural information. We refer to the above meta-learning framework for few-shot learning problems on HINs as META-HIN. While the tasks addressed by each of the components may have been studied individually in the context of previously proposed models, our work is among the first to incorporate different modules to address all of the limitations under a single framework and to demonstrate the benefits of this unified approach. In other words, the main novelty of our paper is that we propose a unified framework to realize the few-shot learning of HINs and facilitate different downstream tasks across different domains of graphs.

Concretely, we first sample subgraphs from the original graphs to form a support set and query set for both meta-training and meta-testing datasets. Before sampling, we first rank the nodes of a certain neighborhood based on three importance evaluation metrics, i.e., betweenness centrality, eigencentrality, and closeness centrality. Then we adopt rank-guided heterogeneous walks to sample subgraphs in which influential nodes and every type of node will be collected. Note that traditional meta-structure is domain-specific and pre-defined, which is not applicable in meta-learning to transfer to unseen tasks and labels. Next, we apply a structure module on the subgraphs to learn the structural embeddings. Specifically, we adopt an auto-encoder to encode the structural information, based on the intuition that nodes with similar structure will share similar embeddings. Then we apply a heterogeneous GNN module to encode the input subgraphs. We first group the nodes based on their types and apply a Bi-LSTM to each group. The structural information loss by this operation has been preserved by the above structure module. Then we aggregate different type embeddings via a self-attention mechanism to generate the final node embeddings. To make use of unsupervised information, we incorporate a contrastive module in a meta-learning setting before calculating the support loss and query loss. During training, positive samples are the nodes from the given subgraph while negative samples are nodes from other subgraphs. Then

we maximize the mutual information between the node embedding and subgraph embedding. Additionally, a GAN-based mechanism is introduced to generate high-quality negative samples that are hard to identify, which further improves the model performance.

Meta-learning. Different tasks may have different effects on the meta-learner; we employ a self-attention mechanism to calculate the weights of tasks that will be incorporated into the meta-learning process to boost the model performance. Unlike previous models that view a task as a batch of node representations, for META-HIN, a task is a batch of subgraphs. By doing so, META-HIN can adapt to new tasks rapidly and thus have a broader applicability. This explains META-HIN's ability to handle different tasks across different graphs. We simultaneously conduct the following three tasks: (i) node classification, (ii) link prediction, and (iii) anomaly detection.[1] For the link prediction and anomaly detection tasks, we use a contrastive loss to distinguish positive samples (existing links and abnormal nodes) and negative samples (non-existing links and normal nodes).

Experiments. To demonstrate that META-HIN is transferable across HINs, aside from using the open academic graph (OAG) dataset, we combine the OAG, DBLP and Aminer dataset to build a new dataset named ODA. These three datasets are all bibliographic networks; we further introduce two datasets from different domains: YELP and YAGO. META-HIN consistently and significantly outperforms the state-of-the-art models on the three tasks across different datasets.

Contributions. In short, our main contributions can be summarized as follows:

- We design a unified framework META-HIN to realize the few-shot learning of HINs and facilitate different downstream tasks across different domains of graphs.
- We sample subgraphs to be trained so that META-HIN is applicable to three tasks and transferable across different HINs.
- We adopt a structural module, a heterogeneous module, and a GAN-based contrastive module to capture the structural information, heterogeneous features, and unlabeled information of a subgraph, respectively.
- We show that META-HIN significantly and consistently outperforms state-of-the-art alternatives on three tasks across multiple HINs.

2 RELATED WORK

2.1 Graph neural network

To learn the representation of a network, many researchers propose variants of GNN models which have shown promising performance.

Some models are based on spectral graph theory. One line of GNN research is based on spectral graph theory [3, 6, 19, 24, 26, 27]. For example, Bruna et al. [3] conduct spectral convolution operations on the whole graph. To improve scalability, graph convolutional networks (GCNs) [24] leverage the first-order approximation of spectral graph convolutions. Such spectral operations are always conducted on the whole graph which may cause efficiency issues. Therefore, another line of research named spatial GNN has been proposed [11, 16, 35, 37]. For example, GraphSAGE [16] adopts a random walk to sample the neighboring nodes to be processed further via CNN or LSTM operation. Gao et al. [11] sample subgraphs and adapt them to be processed via a normal convolution layer.

There are also many variants of GNNs [25, 49, 58], fusing neighboring sub-structures to learn the network embedding. For example, GAT [49] harnesses masked self-attention layers to learn the weights of nodes in a neighborhood. GIN [54] applies injective multiset functions for neighborhood aggregation by parameterizing universal multiset functions with neural networks.

[1]With minor additional efforts, META-HIN can be adapted to handle more downstream tasks under the proposed framework, which is left to future work.

However, these models are all devised for homogeneous networks by aggregating nodes or walks regardless of their types. Few methods focus on the representation of HINs. Therefore, we propose a heterogeneous GNN encoder to fully capture the type information behind a HIN.

2.2 Contrastive learning

Contrastive learning is a line of self-supervised learning that can capture the unlabeled information of a HIN. In natural language processing, to learn word embeddings, word2vec [34] uses co-occurring words and negative sampling to capture similarity from data. In computer vision, self-supervised image representation is learned by minimizing the distance between two views of the same image [14, 18, 46]. For example, He et al. [18] present momentum contrast (MoCo) for visual representation learning by constructing a dynamic dictionary and a moving-averaged encoder.

Recently, contrastive learning approaches devised for graph data are also being proposed. GCC [41] adopts the InfoNCE loss [48] and uses the instance discrimination task [53] to distinguish the nodes from the same subgraph and other subgraphs. DGI [50] aims to maximize the mutual information between node representation and graph representation. GMI [40] proposes to make a contrast between a center node representation and its local patch representation learned from structural information and node features. Infograph [44] is based on DGI, which maximizes the mutual information between the local representation and global representation, and uses GIN as the base GNN encoder. GraphCL [15] generates two randomly perturbed subgraphs of a given node, and maximizes the mutual information between them. Hassani and Ahmadi [17] propose structure-space augmentation to increase the number of views and contrast node and graph embeddings across views. Mu et al. [36] propose an alignment strategy based on mutual information maximization, which aligns the explicit disentangled representations based on knowledge with the implicit disentangled representations learned from user-item interaction

We incorporate contrastive learning with a meta-learning setting to leverage unsupervised information.

2.3 Few-shot learning

Few-shot learning models can be grouped into two types, i.e., (i) based on metric-based learning, and (ii) based on gradient-based learning. The former is to learn a generative metric that is able to compare and match few-samples, while the latter leverages a specific meta-learner to learn the well-initialized parameters of the base model, which will be further adapted to new tasks.

In this paper, we mainly focus on gradient-based learning for graph data. Cui et al. [5] address cold-start issues in sequential knowledge graphs within a meta-learning framework. Niu et al. [38] address a few-shot knowledge graph completion problem, using relational learning incorporated with the gated and attentive neighbor aggregator. Jiang et al. [23] also focus on knowledge graph completion, and propose a meta pattern learning framework to predict new facts of relations under a few-shot setting. Meta-GNN [61] deals with the node classification problem; it obtains prior knowledge by training similar few-shot learning tasks and classifies nodes from new classes with a few labeled samples. It applies a GNN encoder on the entire graph, while META-HIN trains subgraphs individually via a GNN encoder. AMM-GNN [51] and RALE [31] also only deal with the node classification task; the former employs attribute matching and the latter adopts the relative location of a hub-node. Meta-Graph [2] focuses on the link prediction task; it introduces a graph signature function to bootstrap fast adaptation to new graphs. For Meta-GNN and Meta-Graph, a task is a batch of node representations while for META-HIN it is a batch of subgraphs. This important difference allows META-HIN to be adapted to new tasks rapidly, leading to broader applicability on different meta-learning problems and graphs. Meta-GNN and Meta-Graph are only applicable to a single graph.

G-META [22] is applied to node classification and link prediction; it follows a standard MAML framework and

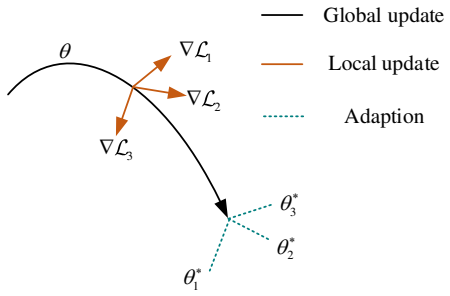


Fig. 1. The meta-learning framework.

uses a GNN as its base model. It also trains on subgraphs, so it can be applied across different graphs. However, in its own setting, for the multiple-graph scenario, it chooses different protein networks that are from the same domain. GFL [56] faces the same issue as it employs auxiliary graphs of the same distribution to facilitate its performance. In our experiments, the multiple-graph scenario is composed of graphs from the same domain as well as different domains. Meta-MGNN [13] focuses on the molecular property prediction task on a single graph, which can be regarded as node classification. It also makes use of unlabeled information via generative learning. This unsupervised module is incorporated with a MAML framework to deal with a few-shot problem. Meta-GDN [8] focuses on anomaly detection on a single graph. It introduces a new graph neural network, namely a graph deviation network (GDN), as its base model. Zhuang et al. [62] propose a dataset designed for few-shot learning on HINs. However, it is not practical as it creates 80 classes for every node, which is quite rare in real-world settings. MetaHIN [32] regards HINs as auxiliary information of a user-item network, and it only uses meta-paths with at most three steps including user and item to help model network features. Unlike other graph few-shot learning methods that explore the whole network, MetaHIN leverages limited network features and is limited to recommendation task on a single graph.

The main novelty of our paper is that we propose a unified framework to address all the limitations the previous studies have, realize the few-shot learning of HIN and facilitate different downstream tasks across different domains of graphs. We also equip our model with a structural module to fully capture the structural information and a heterogeneous module to capture network heterogeneity. And we also incorporate contrastive learning into meta-learning to leverage unsupervised information.

3 THE PROPOSED MODEL META-HIN

3.1 Preliminaries and overview

Let $G = (V, E, T)$ denote a *heterogeneous information network* (HIN), in which V and E denote the node set and edge set, respectively; T_V and T_E denote the node type set and edge type set, respectively. A HIN is a network where $|T_V| > 1$ and/or $|T_E| > 1$. We use $\mathcal{G} = \{G_1, G_2, \ldots, G_N\}$ to denote a set of graphs and $\mathcal{Y} = \{y_1, y_2, \ldots, y_M\}$ to denote the label set. Only a handful of labeled nodes are given, and the goal of our work is to learn the initial parameters θ of a meta-learner, and then adapt the learner to new graphs, tasks, and labels, that is, to correctly map new nodes to new labels given the limited number of labeled nodes.

The gradient based meta-learning is shown as Fig. 1. During the meta-learning process, there exist two kinds of update operations, that is, a local update and a global update. Specifically, firstly the model locally updates the parameter θ to θ_i^* on the support set (learning process), and it searches for the task-specific desired parameters. Based on the task-specific parameter θ_i^*, the model further updates its global parameters to minimize the loss of query set over different tasks (learning-to-learn process). Finally, the learned global parameters θ can fit into various tasks.

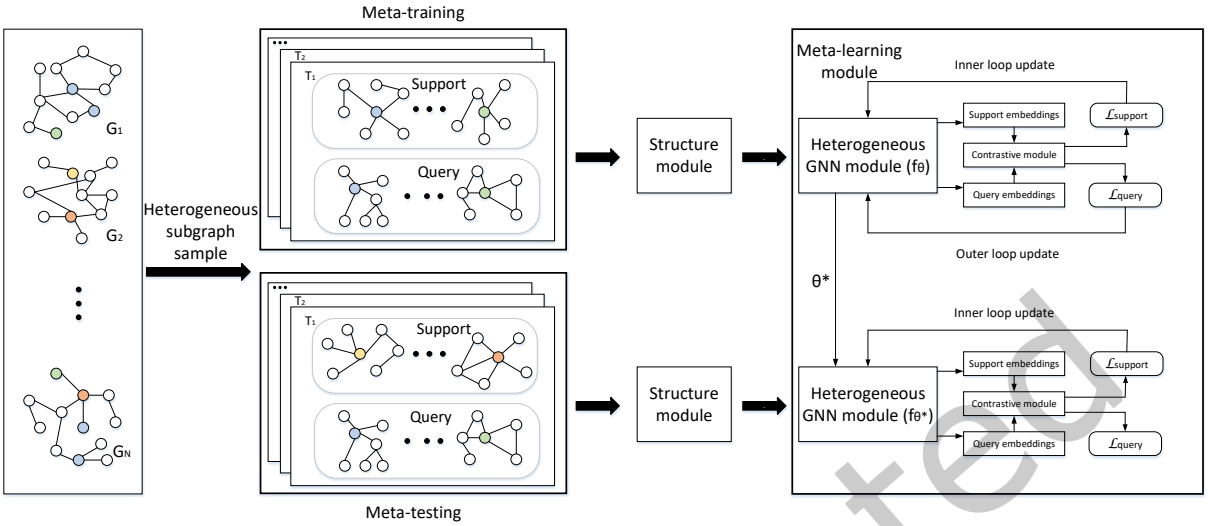


Fig. 2. The META-HIN framework. The heterogeneous subgraph sampling strategy (left) is detailed in Section 3.2; the structure module (center, top) in Section 3.3; the meta-learning module (right) in Section 3.4, with the heterogeneous GNN module (right, bottom) described in Section 3.4.1 and the contrastive module (right) in Section 3.4.2.

A visual sketch of the proposed meta-learning framework for heterogeneous information networks (META-HIN) is given in Fig. 2. The original input HINs will first be processed by a heterogeneous subgraph sampling strategy, so as to form the meta-training and meta-testing subgraph datasets. Then the subgraphs will first be sent to a structure module, capturing the structural information of these subgraphs. The learned structural embeddings will be sent to a meta-learning module as input. In the meta-learning module, the base model is the heterogeneous GNN module which is used to capture network heterogeneity. The contrastive module is also introduced during meta-learning, so that unsupervised information can also be leveraged. Under the meta-learning framework, in a k-shot meta-training phase, for each task $\mathcal{T}_\tau$ sampled from distribution $p(\mathcal{T})$, only k data samples will be used for training (i.e., the *support* set, denoted as $\mathcal{G}_\tau$), and the remainder will be used for evaluation (i.e., the *query* set, denoted as $\mathcal{G}'_\tau$). During meta-training, the parameters θ will first be updated on the support set, and then further optimized on the query set using some loss function. After a sufficient amount of training, meta-testing employs the learned parameters θ^* as initialization to adapt to new tasks quickly with only k samples (as the support set).

3.2 Sampling strategy

To construct meta-training and meta-testing datasets, we sample the subgraphs from the given graph sets $\mathcal{G}$. As illustrated in [22], sampling local subgraphs will not cause a loss of necessary information compared to the entire graph. Since it is tricky to propagate information through the whole graph using only a handful of nodes, in this paper, we also choose to sample subgraphs for training and testing. In addition, sampling subgraphs also enables our model to transfer knowledge across different graphs [22].

Specifically, to form the subgraph, we first rank the neighboring nodes of the given labeled node based on their structural importance. To measure structural importance, we adopt the concept of node centrality from [1]. We include three commonly used centrality metrics, i.e., (i) betweenness centrality, (ii) eigencentrality, and (iii) closeness centrality. *Betweenness centrality* is used to measure the fraction of shortest paths passing through a given node, it evaluates a node's ability to connecting the network. *Eigencentrality* calculates a node's importance based on its neighboring nodes' importance. *Closeness* centrality is used to compute the total length of the shortest

paths between the given node and others, which measures how close a node to other nodes in a network. We assign learnable weights to these metrics.[2]

Next, we adopt a so-called *rank-guided heterogeneous walk* to construct the subgraph, which is meant to capture heterogeneous and structural features of a node's neighbor. For a given node v_1, the sampling walk first reaches out to node v_2 having the highest rank of v_1's adjacent nodes. Then it will search for node v_3 having the highest rank of v_1 and v_2's adjacent nodes. This walk will not stop until it collects a pre-determined number of nodes. In order to equip the model with a sense of heterogeneity, we constrain the number of different types to be collected in the sequence so that every type of node can be included.

The nodes are sorted based on each node's rank, which serves as a kind of sequential information.

Unlike traditional sampling strategies like random walks, breadth-first search or depth-first search, our sampling strategy is able to extract important and influential neighboring nodes for each node by selecting nodes with a higher rank; this allows us to capture more representative structural information of a neighborhood. Our sampling strategy collects all types of nodes for each neighborhood while traditional strategies ignore the nodes' types. Empirical results with an analysis are provided in Section 5.2.

Note that meta-paths, meta-graphs, or network schemas can also be used to explore the heterogeneity of a HIN. However, they are domain-specific and usually pre-defined by domain experts, which makes it hard to apply them in the meta-learning setting to transfer the prior knowledge to new tasks.

3.3 Structure module

Before applying the meta-learning module to transfer knowledge to different tasks and graphs, we first introduce a structure module to preserve the structural information of a subgraph. Specifically, we adopt the autoencoder mechanism to preserve the graph structure. For each subgraph, we have its adjacency matrix $A = \{a_1, a_2, \ldots, a_n\}$, in which a_i is a row of A representing the nodes that are adjacent with node i. Then it is fed into the encoder to obtain the latent representation of node i as follows:

$$h_i^{(1)} = \delta(W^{(1)} a_i + b^{(1)}), \tag{1}$$

$$h_i^{(k)} = \delta(W^{(k)} h_i^{(k-1)} + b^{(k)}), \tag{2}$$

in which k is the number of layers of the encoder, which we set to 2 in practice; $W^{(k)}$ is the parameter matrix of the k-th layer; $b^{(k)}$ is the bias of the k-th layer; and δ is the activation function.

In the decoder we reverse the above process to obtain the reconstructed output $\hat{a}_i$. The objective function of the autoencoder is to minimize the reconstruction error of the input and output. Mathematically,

$$\mathcal{L}_{structure} = \sum_{i=1}^{n} \|(\hat{a}_i - a_i)\|_2^2. \tag{3}$$

This reconstruction criterion forces nodes with a similar structure to have similar representations. To alleviate the sparsity issue, we impose more penalty on non-zero elements than on zero elements. Then the loss function is defined as:

$$\mathcal{L}_{structure} = \sum_{i=1}^{n} \|(\hat{a}_i - a_i) \odot b_i\|_2^2 \tag{4}$$

$$= \|(\hat{A} - A) \odot B\|_F^2, \tag{5}$$

in which $b_i = \{b_{i,j}\}_{j=1}^{n}$, and $b_{i,j} = 1$ if there exists no edge between nodes i and j, otherwise $b_{i,j} = \beta > 1$; $\odot$ is

[2]Additional metrics that are of particular interest to the user could also be added to the above three metrics. In our implementation, we use the above three metrics to illustrate the effectiveness of META-HIN.

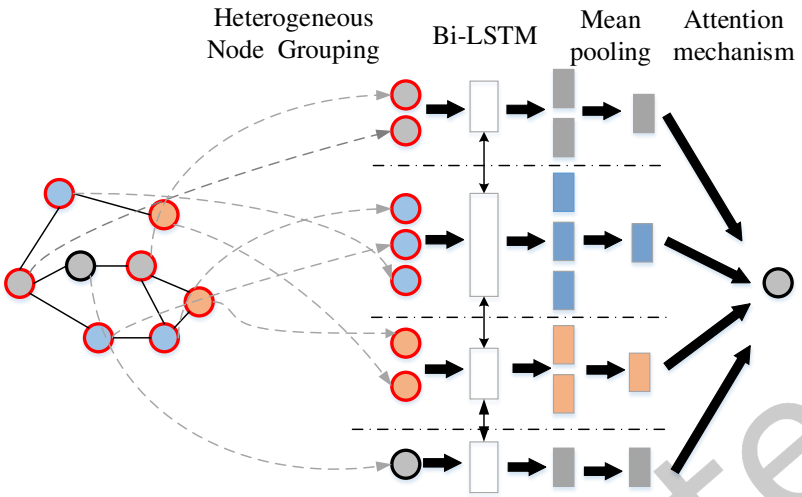


Fig. 3. Framework of the heterogeneous GNN module.

element-wise product.

The optimized latent representations of nodes in the structural module will be sent to meta-learning module as the input node embeddings.

3.4 Meta-learning module

After representing the structural information of the subgraphs, we introduce the meta-learning framework to deal with the few-shot setting. We first introduce a heterogeneous GNN module to encode subgraphs. Then comes the contrastive module to make use of unlabeled information.

3.4.1 Heterogeneous GNN module. Fig. 3 presents the framework of the heterogeneous GNN module. We first group the nodes of the subgraph based on their types, that is, nodes with the same type are grouped together. This operation transforms a subgraph into a node sequence. As shown in the Fig. 3, after heterogeneous node grouping, it is simply a node sequence with no edges in between. This operation will corrupt the structure of the subgraph, however, the structural information has already been preserved by the structure module. And this operation also allows us to model the heterogeneity by processing type-based feature information. Then we apply a Bi-LSTM on these groups to extract type-specific features. A Bi-LSTM is able to process sequence-like data, learn deep feature interactions between nodes in the subgraphs, and obtain larger expressive capability for node representation. The representation of the i-th node of the group is denoted as x_i. Then the hidden representation $h^v_{T_j}$ of type T_j after the Bi-LSTM layer can be represented as:

$$h^v_{T_j} = \frac{\sum_{i \in S^v_{T_j}} \text{Bi-LSTM}\{x_i\}}{|S^v_{T_j}|}, \tag{6}$$

where $S^v_{T_j}$ denotes the node group having type T_j; the Bi-LSTM is composed of a forward LSTM and a backward LSTM. The forward LSTM is specified as follows:

$$\mathbf{j}_i = \delta(\mathbf{W}_{xj} x_i + \mathbf{W}_{hj} h_{i-1} + \mathbf{W}_{cj} c_{i-1} + \mathbf{b}_j), \tag{7}$$

$$\mathbf{f}_i = \delta(\mathbf{W}_{xf} x_i + \mathbf{W}_{hi} h_{i-1} + \mathbf{W}_{cf} c_{i-1} + \mathbf{b}_f), \tag{8}$$

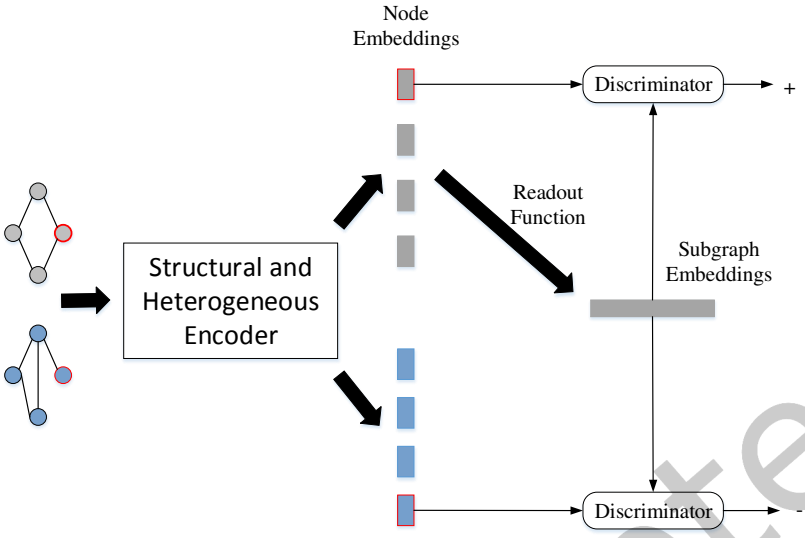


Fig. 4. Framework of the contrastive module.

$$\mathbf{z}_i = \tanh(\mathbf{W}_{xc} x_i + \mathbf{W}_{hc} h_{i-1} + \mathbf{b}_c), \tag{9}$$

$$c_i = \mathbf{f}_i \odot c_{i-1} + \mathbf{j}_i \odot \mathbf{z}_i, \tag{10}$$

$$\mathbf{o}_i = \delta(\mathbf{W}_{xo} x_i + \mathbf{W}_{ho} h_{i-1} + \mathbf{W}_{co} c_i + \mathbf{b}_o), \tag{11}$$

$$h_i = \mathbf{o}_i \tanh(c_i), \tag{12}$$

where $h_i \in \mathbb{R}^{d/2}$ denotes the output hidden representation, $\mathbf{W} \in \mathbb{R}^{(d/2)\times(d/2)}$ and $b \in \mathbb{R}^{d/2}$ are learnable parameters, denoting weight and bias, respectively; δ denotes the activation function; $\mathbf{j}_i$, $\mathbf{f}_i$, $\mathbf{o}_i$ are the input gate vector, forget gate vector and output gate vector, respectively. The output of the forward and backward LSTMs are concatenated, along with a mean-pooling layer to generate the hidden embedding of a specific type.

We combine the type-specific embeddings via an attention mechanism to generate the final representation of the given node. The reason for using an attention mechanism is that different types of node may have a different impact on the given node. Mathematically,

$$h_v = \sum_{T_j \in \{T\}} \alpha^{v,j} h^v_{T_j}, \tag{13}$$

$$\alpha^{v,j} = \frac{\exp\{\delta(u^{\mathrm{T}} h^v_{T_j})\}}{\sum_{T_l \in \{T\}} \exp\{\delta(u^{\mathrm{T}} h^v_{T_l})\}}, \tag{14}$$

where δ denotes the activation function, for which we use a *LeakyReLU*; $u \in \mathbb{R}^d$ is the attention parameter.

3.4.2 Contrastive module. Fig. 4 presents the framework of contrastive learning. Many previously proposed meta-learning models ignore unsupervised information in a few-shot setting. However, only employing supervised signals may limit the performance because only a handful of labeled nodes may be available. Therefore, we introduce a contrastive module to make full use of unlabeled nodes in the graph.

After the structural module and heterogeneous GNN module, we obtain the node embeddings of a subgraph denoted as $H = \{h_1, h_2, \ldots, h_n\}$. We summarize the node embeddings of a subgraph using a *readout function* to generate the subgraph embedding g. That is, $g = \mathrm{READOUT}(H)$, where READOUT may be any permutation

invariant function; we simply use mean pooling here. Our goal is to maximize the mutual information between the node representation and the subgraph representation. In this way, the subgraph representation is able to represent aspects of the data that are shared across all substructures.

For contrastive learning, for a given subgraph, the positive sample is the node that belongs to the subgraph, and the negative sample is a node from other subgraphs. Note that there will not be duplicate nodes in a single task. The negative node embeddings of another subgraph are denoted as $H' = \{h'_1, h'_2, \ldots, h'_n\}$.

Next, we present a *discriminator* $\mathcal{D}(h_i, g)$ that calculates the probability score between this node and a subgraph pair. The score will be higher if the node belongs to the subgraph.

To define the loss function, we adopt a noise-contrastive type objective with a standard binary cross-entropy loss between positive and negative samples [20]. Mathematically, the combination can be represented as:

$$\mathcal{L}_{contrastive} = \sum_{i=1}^{n} \mathbb{E}_H \left[\log \mathcal{D}(h_i, g)\right] + \sum_{j=1}^{n} \mathbb{E}_{H'} \left[\log(1 - \mathcal{D}(h'_j, g))\right]. \tag{15}$$

By doing so, the mutual information between node embedding h_i and subgraph embedding g is maximized via Jensen-Shannon divergence between positive and negative samples [39]. Note that each node embedding obtained in the given subgraph is required to compute the mutual information with the representation of this subgraph.

To further boost the model performance, we introduce a GAN-based mechanism to generate negative samples that are hard to be distinguished. In addition to the discriminator discussed above, a generator is introduced. They are trained in an alternating fashion. Aside from distinguishing original negative and positive samples, the discriminator is also first trained to identity the negative samples generated from the generator. Then the generator is trained to generate high-quality negative samples to fool the discriminator.

3.4.3 Supervised loss. Here we use the supervised signal to calculate the training loss. For the node classification task, we first apply a *multi-layer* perception (MLP) on top of the subgraph embedding s, denoted as: $\hat{y} = \text{MLP}(s)$. Next, we leverage the cross entropy loss between the predicted labels and ground-truth labels:

$$\mathcal{L}_{supervised}^{classification} = -\frac{1}{m} \sum_{i=1}^{m} \text{CrossEntropy}(y_i, \hat{y}_i), \tag{16}$$

where m denotes the number of samples.

The link prediction and anomaly detection could be regarded as binary classification tasks. However, instead of simply adopting binary entropy loss, we borrow the idea of contrastive learning and employ the contrastive loss. Specifically, for link prediction, an existing link is the positive signal, and for anomaly detection, an abnormal node is regarded as positive. Note that link prediction could be realized by a pair of nodes in the subgraph. Then we introduce a Noise-Contrastive Estimation (NCE) Loss [48] as follows:

$$\mathcal{L}_{supervised}^{contrastive} = -\log \frac{\exp(q^{\mathrm{T}} k^{+}/\gamma)}{\sum_{i=1}^{m} \exp(q^{\mathrm{T}} k_i/\gamma)}, \tag{17}$$

where k is the node embedding of the support set, q is the node embedding of the query set, and γ is the temperature hyperparameter; we set $\gamma = 0.07$.

3.4.4 Joint loss. For a given task $\mathcal{T}_i$, the joint loss of the meta-training process is a combination of the contrastive loss and supervised loss. Mathematically,

$$\mathcal{L}_{\mathcal{T}_\tau}(\theta) = \lambda \mathcal{L}_{constrative} + (1 - \lambda)\mathcal{L}_{supervised}, \tag{18}$$

where λ is the trade-off parameter; we set $\lambda = 0.2$.

3.4.5 Optimization-based meta-learning. Given the support set $\mathcal{G}_\tau$ and query set $\mathcal{G}'_\tau$, the optimization approach first adapts the initial model parameters θ to θ'_τ for each learning task in support $\mathcal{T}_\tau$ independently. A batch of training samples will be leveraged to calculate the updated parameter θ'_τ. This process can be represented as:

$$\theta'_\tau = \theta - \epsilon \nabla_\theta \mathcal{L}_{\mathcal{T}_\tau}(\theta), \tag{19}$$

where ϵ is the step size. Suppose we have a task distribution $\mathcal{T}_\tau \sim p(\mathcal{T})$, then we employ stochastic gradient descent (SGD) to update the model parameters across all tasks in the query set. Mathematically,

$$\theta = \theta - \mu \nabla_\theta \sum_{\mathcal{T}_\tau \sim p(\mathcal{T})} \eta(\mathcal{T}_\tau) \mathcal{L}'_{\mathcal{T}_\tau}(\theta'_\tau), \tag{20}$$

where μ is the meta-learning rate and $\mathcal{L}'_{\mathcal{T}_\tau}$ is the joint loss over query set of $\mathcal{T}_\tau$. Note that different tasks may contribute differently to the meta-learner. We add a self-attention layer to measure the task weight $\eta(\mathcal{T}_\tau)$. We first calculate the task embedding. Given the task $\mathcal{T}_\tau$, its representation $H_{\mathcal{T}_\tau}$ is calculated as the average of all node embeddings of $\mathcal{T}_\tau$. Mathematically,

$$H_{\mathcal{T}_\tau} = \text{AVERAGE}(h_{\mathcal{T}_\tau}{}_{i=1}^{n}). \tag{21}$$

Then $\eta(\mathcal{T}_\tau)$ is computed as follows:

$$\eta(\mathcal{T}_\tau) = \frac{\exp(\delta(\omega^{\mathrm{T}} H_{\mathcal{T}_\tau}))}{\sum_{\mathcal{T}_{\tau'} \in \mathcal{T}} \exp(\delta(\omega^{\mathrm{T}} H_{\mathcal{T}_{\tau'}}))}, \tag{22}$$

where δ is the activation function and we leverage *LeakyReLU*; $\omega \in \mathbb{R}^d$ is the attentive parameter.

During the meta-testing phase, we repeat the above process using the final updated parameter θ^*. We learn θ^* from knowledge across all meta-training tasks and it is the optimal parameter to adapt to unseen tasks quickly.

Summary. Algorithm 1 presents the overall learning procedure of META-HIN. Given heterogeneous graphs and randomly initialized parameters θ as input, the algorithm constructs a support set and a query set via its heterogeneous subgraph sampling strategy (line 1). Then we sample a batch of tasks (line 3) and for all tasks, we sample k support graph instances (line 5). After that we iterate over the sampled graphs (line 6–10), where in each graph we first generate structural node embeddings and then feed them into the heterogeneous GNN module for an update. We obtain the task embedding based on the node embeddings (line 11). We calculate the loss function (line 12) and update the parameter based on the loss (line 13). After this, we sample n query graph instances (line 14) and follow a similar procedure as with the support set to learn the node embeddings and loss function (line 15–20). Then we calculate the task weight (line 22) to update the final adapted parameters θ (line 23).

Complexity analysis. We conduct a complexity analysis of our training procedure, which is calculated as $O(e \cdot |\mathcal{T}| \cdot |\mathcal{S}| \cdot d)$, where e denotes the number of epochs, $|\mathcal{T}|$ denotes the number of meta-training tasks, $|\mathcal{S}|$ denoted the number of subgraphs, and d denoted the dimension of node embeddings. e and d are usually small, so the complexity of the proposed model is linear with $O(|\mathcal{T}| \cdot |\mathcal{S}|)$.

4 EXPERIMENTAL SETUP

We detail our datasets, baseline models, and parameter settings.

Algorithm 1: The learning algorithm of META-HIN.

Input : Heterogeneous Graphs $\mathcal{G} = \{G_1, G_2, \ldots, G_N\}$, Randomly initialized θ;

Construct support set $\mathcal{G}_\tau$ and query set $\mathcal{G}'_\tau$ via Heterogeneous subgraph sampling strategy;
while *not done* **do**
  Sample batch of tasks $\mathcal{T}_\tau \sim p(\mathcal{T})$;
  for *all* $\mathcal{T}_\tau$ **do**
    Sample k instances $\{\mathcal{G}_{\tau 1}, \mathcal{G}_{\tau 2}, \ldots, \mathcal{G}_{\tau k}\} \in \mathcal{G}_\tau$;
    for *i=1 to k* **do**
      Generate structural node embedding via structure module;
      Feed structural embedding into heterogeneous GNN module;
      Calculate node representation via Eq. (13);
    end
    Calculate task embedding via Eq. (21);
    Calculate $\mathcal{L}_{\mathcal{T}_\tau}$ via Eq. (18);
    $\theta'_\tau = \theta - \epsilon \nabla \mathcal{L}_{\mathcal{T}_\tau}$;
    Sample n instances $\{\mathcal{G}'_{\tau 1}, \mathcal{G}'_{\tau 2}, \ldots, \mathcal{G}'_{\tau k}\} \in \mathcal{G}'_\tau$;
    for *j=1 to n* **do**
      Generate structural node embedding via structure module;
      Feed structural embedding into heterogeneous GNN module;
      Calculate node representation via Eq. (13);
    end
    Calculate $\mathcal{L}'_{\mathcal{T}_\tau}$ via Eq. (18);
  end
  Calculate task weight $\eta(\mathcal{T}_\tau)$ via Eq. (22) ;
  $\theta = \theta - \mu \nabla_\theta \sum_{\mathcal{T}_\tau \sim p(\mathcal{T})} \eta(\mathcal{T}_\tau) \mathcal{L}'_{\mathcal{T}_\tau}(\theta'_\tau)$.
end

4.1 Datasets

We first adopt three bibliographic graph networks, i.e., (i) OAG,[3] (ii) DBLP,[4] and (iii) AMiner,[5] which are all heterogeneous information networks with four types of nodes (i) authors, (ii) papers, (iii) venues, and (iv) topics. As for the label information, the authors in these networks are split into five areas: (i) information retrieval, (ii) database, (iii) natural language processing, (iv) data mining, and (v) machine learning. Unlike most previous research focusing on a single graph, here we mix the OAG, DBLP, and AMiner datasets to construct a new dataset named ODA.

The OAG, DBLP, and AMiner datasets are all from the same domain. We include two other social graphs from different domains. One is YELP,[6] containing restaurant reviews and four types of node: (i) customers, (ii) restaurants, (iii) reviews, and (iv) food-related keywords. The restaurants are labeled as (i) Chinese food,

[3] https://www.openacademic.ai/oag/
[4] http://dblp.uni-trier.de
[5] https://www.aminer.cn/data
[6] https://www.yelp.com/dataset_challenge

Table 1. Dataset statistics.

Dataset	#nodes	#edges	#node types	# label
OAG	432,362	1,837,362	5	5
DBLP	357,362	1,547,364	5	5
AMiner	263,473	1,022,362	5	5
ODA	1,053,197	4,407,088	5	5
YELP	213,476	1,622,327	4	3
OYE	645,838	3,459,689	9	8
YAGO	273,276	2,452,371	5	5
OYA	705,638	4,289,733	10	10

(ii) fast food, and (iii) sushi bar. The other social graph that we add is YAGO,[7] which contains movie information with nodes having five types: (i) movie, (ii) director, (iii) actor, (iv) producer, and (v) composer. The movies are labeled as (i) action, (ii) adventure, (iii) sci-fi, (iv) crime, and (v) horror. Next, we create two mixed datasets. One combines OAG and YELP, denoted as OYE; the other combines OAG and YAGO, denoted as OYA.

To conduct the tasks on a single large graph, we only choose OAG to be analyzed for simplicity.[8]

The dataset statistics are shown in Table 1.

We have three tasks to be analyzed, i.e., node classification, link prediction and anomaly detection. To simplify our presentation, for our single-graph experiments on those tasks, we opt to zoom in on a single large graph, namely OAG.

4.2 Algorithms used for comparison

We adopt two GNN models that are directly trained on the supervised information, i.e., GCN [24], which is a spectral graph model, and GraphSage [16], which is a spatial model.

Three baselines specifically designed for heterogeneous information network representation are also compared, i.e., HetGNN [57], HAN [52], and HGT [21].

We also include baselines designed for few-shot problem on graphs, i.e., Meta-Graph [2], Meta-GNN [61], G-META [22], Meta-MGNN [13], and Meta-GDN [8]. Detailed information about these models can be found in Section 2.

To increase the number of models used for comparison in a multi-graph scenario, we also include meta-learning baselines that are not specifically designed for graph data. KNN [47] first embeds the meta-training set using a GNN, and each query example is represented via the label of the voted K-closest example in the support set. Finetune [47] first learns the embeddings of the meta-training set and then the models are fine-tuned on the meta-testing set. ProtoNet [43] employs prototype learning on each subgraph embedding and then follows the standard few-shot learning procedure. MAML [10] chooses a MAML framework as the meta-learner.

4.3 Parameters

As our sampling strategy, we set the length of our rank-guided heterogeneous walk to 20, in other words, the number of nodes in a subgraph is 20. The update step in training tasks is set to 10 and the update step in testing tasks to 20. We use 10 shots for each task.

In node classification, nodes of two labels will be used for meta-testing datasets and nodes of other labels will

[7] https://old.datahub.io/dataset/yago

[8] For the user with particular interest, single-graph experiment on DBLP, AMiner, YELP, YAGO is also welcomed. In our implementation, we use OAG to illustrate the model effectiveness on a single graph.

Table 2. Results on the multi-label node classification task.

	OAG		ODA		OYE		OYA	
Model	ACC	F1	ACC	F1	ACC	F1	ACC	F1
GCN	0.332	0.438	0.278	0.321	0.217	0.236	0.207	0.226
GraphSage	0.387	0.452	0.284	0.302	0.221	0.272	0.218	0.263
HetGNN	0.395	0.463	0.292	0.307	0.228	0.279	0.224	0.278
HAN	0.398	0.468	0.289	0.305	0.232	0.283	0.228	0.283
HGT	0.407	0.472	0.298	0.313	0.239	0.288	0.235	0.290
Meta-GNN	0.457	0.523	N/A	N/A	N/A	N/A	N/A	N/A
Meta-MGNN	0.559	0.637	N/A	N/A	N/A	N/A	N/A	N/A
G-META	0.538	0.621	0.468	0.521	0.372	0.381	0.362	0.379
KNN	0.497	0.589	0.427	0.463	0.304	0.323	0.293	0.321
Finetune	0.413	0.538	0.389	0.413	0.291	0.312	0.272	0.295
ProtoNet	0.472	0.548	0.411	0.421	0.297	0.327	0.283	0.301
MAML	0.485	0.611	0.419	0.409	0.299	0.328	0.295	0.312
META-HIN	**0.581**$^{\blacktriangle}$	**0.655**$^{\blacktriangle}$	**0.516**$^{\blacktriangle}$	**0.551**$^{\blacktriangle}$	**0.397**$^{\blacktriangle}$	**0.414**$^{\blacktriangle}$	**0.395**$^{\blacktriangle}$	**0.411**$^{\blacktriangle}$

be involved in meta-training tasks. In link prediction, we separate 30% of the edges for the support set and 70% of the edges for the query set; the negative edges are randomly sampled having the same number of positive edges. The detailed settings for link prediction can be found in [59]. In both link prediction and anomaly detection, there are no overlapping edges to be predicted and anomalies to be detected between meta-training and meta-testing datasets.

The dimensions of all embeddings in META-HIN are set to 128. We use grid search to find the best parameter configuration. The task number is chosen from $\{4, 8, 16, 32, 64\}$; the inner update learning rate ϵ is chosen from $\{0.01, 0.005, 0.001, 0.0005\}$, and the meta-level learning rate μ is also chosen from $\{0.01, 0.005, 0.001, 0.0005\}$. The optimal parameters are task-specific and dataset-specific. For the other models, we adopt the best configurations reported in the source publications.

The model parameters are trained on training datasets, and they are not tuned on testing datasets.

We report on statistical significance with a paired two-tailed t-test and we mark a significant improvement of META-HIN over the best baseline for $p < 0.05$ with $^{\blacktriangle}$.

5 RESULTS AND ANALYSIS

We present the results of META-HIN on three tasks: (i) node classification, (ii) link prediction, and (iii) anomaly detection. We also conduct an ablation analysis and study the parameter sensitivity.

5.1 Results of tasks

5.1.1 Node classification. Here we report on the results for the multi-label node classification task. We adopt multi-class classification accuracy, ACC, and F1 value as evaluation metrics (five-fold average).

Table 2 presents the experimental results on the node classification task; the highest scores are set in bold. N/A means the model does not work in the graph meta-learning problem.

META-HIN consistently and significantly outperforms the baselines on every dataset, which verifies the model's effectiveness. Concretely, GCN, and GraphSage, which lack specific few-shot learning facilities, achieve the worst performance; they are trained in an end-to-end supervised manner but there are only handful of labeled nodes

Table 3. Results on the link prediction task.

	OAG		ODA		OYE		OYA	
Model	AUC	F1	AUC	F1	AUC	F1	AUC	F1
GCN	0.632	0.536	0.472	0.417	0.324	0.275	0.259	0.236
GraphSage	0.648	0.547	0.483	0.421	0.327	0.281	0.264	0.243
HetGNN	0.657	0.553	0.489	0.434	0.335	0.289	0.272	0.255
HAN	0.661	0.558	0.493	0.438	0.341	0.293	0.278	0.260
HGT	0.669	0.567	0.503	0.447	0.354	0.304	0.285	0.271
Meta-Graph	0.723	0.645	N/A	N/A	N/A	N/A	N/A	N/A
G-META	0.771	0.673	0.621	0.523	0.421	0.396	0.393	0.385
KNN	0.711	0.623	0.534	0.473	0.397	0.357	0.348	0.333
Finetune	0.682	0.601	0.523	0.444	0.356	0.326	0.311	0.298
ProtoNet	0.726	0.647	0.598	0.492	0.407	0.364	0.365	0.358
MAML	0.731	0.658	0.592	0.485	0.412	0.371	0.371	0.363
META-HIN	**0.808**▲	**0.696**▲	**0.657**▲	**0.558**▲	**0.449**▲	**0.426**▲	**0.414**▲	**0.408**▲

that can be leveraged. HetGNN, HAN, and HGT perform slightly better for leveraging heterogeneous features, but are still incomparable to models equipped with few-shot techniques. Meta-GNN trains on the entire graph, which limits its performance and speaks to the effectiveness of our subgraph sampling strategy; a subgraph is sufficient to capture local features while an entire graph may introduce noise and may be too sparse to be trained.

On a single graph (OAG), Meta-MGNN is the best performing baseline as it also employs unlabeled information using generative learning. However, META-HIN still outperforms it and we attribute this to the fact that in addition to the unsupervised information, META-HIN harnesses structural information and heterogeneous features.

Additionally, for models designed for graph meta-learning, only G-META and META-HIN can be applied across different graphs. META-HIN outperforms G-META consistently on every dataset; this reflects the advantage of our structural module, contrastive module, and heterogeneous GNN module. How these modules contribute to META-HIN's performance is examined in detail in Section 5.2. As to other few-shot learning methods – KNN, Finetune, ProtoNet, and MAML –, META-HIN consistently outperforms them across different graphs, which confirms that META-HIN's meta-learning framework is more effective on graph data.

5.1.2 Link prediction. Here we report on the results for the link prediction task. We adopt the AUC value and the F1 value as evaluation metrics (five-fold average).

We present the link prediction results in Table 3, with the highest results set in bold. N/A means that a model does not work in the graph meta-learning problem. As in the node classification task, GCN and GraphSage perform the worst because of the limited supervised information. HetGNN, HAN, and HGT perform poorly with no few-shot tricks applied. Meta-Graph, like Meta-GNN in the node classification task, trains on the entire graph causing its worse performance compared to G-META and META-HIN. G-META still performs worse than META-HIN, consistently across different graphs, which, again, verifies the effectiveness of META-HIN by leveraging structural information, unsupervised information, and heterogeneous features.

5.1.3 Anomaly detection. In this task, since there are no ground-truth anomalies available in the original datasets, we adopt an anomaly injection model [7] to insert sets of anomalies. We inject structural anomalies, which are actually a set of small cliques. A small clique is regarded as a typical abnormal sub-structure since nodes in the clique are more closely linked to each other than average; 5% of the datasets are created as anomalies that need to be detected. We adopt AUC-ROC and AUC-PR as evaluation metrics. AUC-ROC evaluates the probability

Table 4. Results on the anomaly detection task.

	OAG		ODA		OYE		OYA	
Model	ROC	PR	ROC	PR	ROC	PR	ROC	PR
GCN	0.472	0.278	0.387	0.212	0.253	0.115	0.215	0.089
GraphSage	0.488	0.283	0.389	0.218	0.262	0.121	0.217	0.095
HetGNN	0.499	0.295	0.402	0.231	0.275	0.133	0.232	0.103
HAN	0.487	0.289	0.397	0.226	0.271	0.128	0.227	0.099
HAT	0.506	0.302	0.406	0.233	0.279	0.138	0.241	0.108
Meta-GDN	0.654	0.445	N/A	N/A	N/A	N/A	N/A	N/A
KNN	0.606	0.378	0.444	0.278	0.278	0.196	0.248	0.152
Finetune	0.572	0.353	0.402	0.248	0.257	0.165	0.236	0.132
ProtoNet	0.622	0.378	0.478	0.391	0.289	0.211	0.259	0.205
MAML	0.616	0.389	0.481	0.398	0.296	0.217	0.264	0.211
META-HIN	**0.671**▲	**0.459**▲	**0.532**▲	**0.432**▲	**0.334**▲	**0.255**▲	**0.293**▲	**0.242**▲

that a randomly chosen anomaly has a higher score than a normal node. AUC-PR is the area under the curve of precision against recall at different thresholds; it only assesses a model's performance on the positive class (abnormal nodes); it is calculated as the average precision defined in [33].

Table 4 shows the performance on the anomaly detection task. The highest scores are set in bold. N/A means that the model is not applicable on these datasets. Note that G-META, which is able to deal with node classification and link prediction, cannot be applied to the anomaly detection task. This confirms META-HIN's broad applicability; it can deal with three meta-learning tasks. GCN, GraphSage, HetGNN, HAN, and HGT perform the worst, and other general few-shot learning methods, KNN, Finetune, ProtoNet, and MAML, are not comparable to META-HIN. The best performing baseline on a single graph is Meta-GDN, which is designed for few-shot graph anomaly detection, but it cannot be applied in a multiple-graph scenario. META-HIN performs the best across different graphs, which further confirms META-HIN's effectiveness and broad applicability.

5.2 Ablation analysis

We analyze the contribution of different components of META-HIN via an ablation analysis.

5.2.1 Evaluation setup. To evaluate the contribution of the structure module, we add a variant, denoted as META-HIN\structure, from which we remove the structure module.

To evaluate the effect of the contrastive module, we introduce a variant, denoted as META-HIN\contrastive, in which we remove the contrastive module. And to evaluate the effect of the GAN mechanism, we introduce a variant denoted as META-HIN\GAN, in which we only use the original negative samples from the graph.

As to the heterogeneous GNN module, since we still need an encoder to encode the subgraph features, this module cannot simply be removed. Instead, we introduce two variants for comparison. That is, our heterogeneous GNN module is replaced by a GCN and GraphSage, respectively, denoted as META-HIN(GCN) and META-HIN(GraphSage), respectively.

To evaluate the effect of our choice of contrastive loss in the link prediction and anomaly detection task, we introduce a variant using the traditional binary cross entropy loss, which is denoted as META-HIN(BCE).

To evaluate the effect of our self-attention mechanism to calculate the task weights, we introduce a variant assigning equal weights for each task, denoted as META-HIN(Equal).

We also assess the effect of our sampling strategy. We introduce three variants for comparison: (i) a breadth-first

Table 5. Results of the ablation analysis on the OYE dataset. Abbreviations used: MC – multi-label node classification, LP – link prediction, AD – anomaly detection.

	MC		LP		AD	
Model	ACC	F1	AUC	F1	AUC-ROC	AUC-PR
META-HIN\structure	0.359	0.383	0.398	0.378	0.279	0.195
META-HIN\contrastive	0.353	0.364	0.386	0.371	0.255	0.191
META-HIN\GAN	0.382	0.401	0.442	0.420	0.319	0.247
META-HIN(GCN)	0.368	0.383	0.402	0.394	0.298	0.208
META-HIN(GraphSage)	0.371	0.390	0.408	0.401	0.305	0.218
META-HIN(BCE)	N/A	N/A	0.427	0.406	0.317	0.236
META-HIN(Equal)	0.375	0.395	0.428	0.411	0.318	0.239
META-HIN(BFS)	0.372	0.384	0.411	0.393	0.296	0.227
META-HIN(DFS)	0.364	0.380	0.402	0.384	0.285	0.225
META-HIN(random)	0.353	0.375	0.397	0.366	0.272	0.216
META-HIN\betweenness	0.382	0.402	0.432	0.408	0.306	0.242
META-HIN\eigen	0.377	0.397	0.436	0.413	0.312	0.233
META-HIN\closeness	0.385	0.404	0.426	0.402	0.316	0.238
META-HIN	**0.397**	**0.414**	**0.449**	**0.426**	**0.334**	**0.255**

search (BFS) strategy, META-HIN(BFS); (ii) a depth-first search (DFS) strategy, META-HIN(DFS); and (iii) randomly sampling the neighboring nodes, META-HIN(random). Recall that we adopt three centrality metrics to measure the importance of a node before applying our sampling strategy. Here, we introduce three variants to assess the contribution of these centrality metrics. The first excludes the betweenness centrality denoted as META-HIN\betweenness; the second excludes the eigencentrality denoted as META-HIN\eigen; the third excludes closeness centrality denoted as META-HIN\closeness. Each variant assigns equal weights to the two remaining centrality metrics.

5.2.2 Results. We report the experimental outcomes on the two combined datasets with different domains, i.e., OYE and OYA datasets; the findings on the OAG and ODA are qualitatively similar.

Table 5 presents the experimental results of our ablation analysis on the OYE dataset. Both the structure module and the contrastive module play a vital role for the meta-learning tasks, as one captures the graph structural information and the other describes the unsupervised information. More concretely, the contrastive module has a greater impact than the structure module as removing it causes a bigger drop in performance. Additionally, incorporating the GAN mechanism with a contrastive module also contributes to the performance improvement. It also verifies that the compatibility of each module is good since combining them all will generate the best model performance.

As for the replacements of our heterogeneous GNN module, both of the variants, i.e., META-HIN(GCN) and META-HIN(GraphSage), perform worse than META-HIN, which we attribute to the fact that META-HIN leverages heterogeneous information.

The variant of META-HIN that uses BCE loss instead of a contrastive loss, performs worse than META-HIN, which is due to the fact that a contrastive loss function is better able to distinguish the binary labels by maximally separating the positive samples from the negative samples.

Table 6. Results of the ablation analysis on the OYA dataset. Abbreviations used: MC: multi-label node classification, LP: link prediction, AD: anomaly detection.

	MC		LP		AD	
Model	ACC	F1	AUC	F1	AUC-ROC	AUC-PR
META-HIN\structure	0.352	0.367	0.376	0.362	0.236	0.183
META-HIN\contrastive	0.347	0.356	0.359	0.352	0.219	0.177
META-HIN\GAN	0.377	0.393	0.403	0.388	0.266	0.203
META-HIN(GCN)	0.359	0.375	0.382	0.372	0.252	0.195
META-HIN(GraphSage)	0.364	0.381	0.387	0.379	0.257	0.201
META-HIN(BCE)	N/A	N/A	0.402	0.386	0.269	0.227
META-HIN(Equal)	0.368	0.381	0.395	0.392	0.276	0.224
META-HIN(BFS)	0.365	0.369	0.386	0.371	0.257	0.204
META-HIN(DFS)	0.357	0.362	0.374	0.362	0.245	0.192
META-HIN(random)	0.345	0.357	0.365	0.348	0.235	0.178
META-HIN\betweenness	0.371	0.384	0.399	0.391	0.273	0.225
META-HIN\eigen	0.365	0.380	0.404	0.394	0.374	0.214
META-HIN\closeness	0.375	0.386	0.392	0.384	0.278	0.229
META-HIN	**0.395**	**0.411**	**0.414**	**0.408**	**0.293**	**0.242**

Assigning equal weights to the node classification, link prediction, and anomaly detection tasks reduces the performance of META-HIN, which supports our intuition that different tasks contribute differently to the meta-learner.

As to alternative sampling strategies, META-HIN(BFS) performs better than both META-HIN(DFS) and META-HIN (random), which illustrates that aggregating a node's closest neighbors is more representative than sampling far-away nodes or randomly chosen nodes. META-HIN, in turn, outperforms META-HIN(BFS); generating the subgraph by selecting nodes with a higher importance results in better depiction of node's neighborhood information.

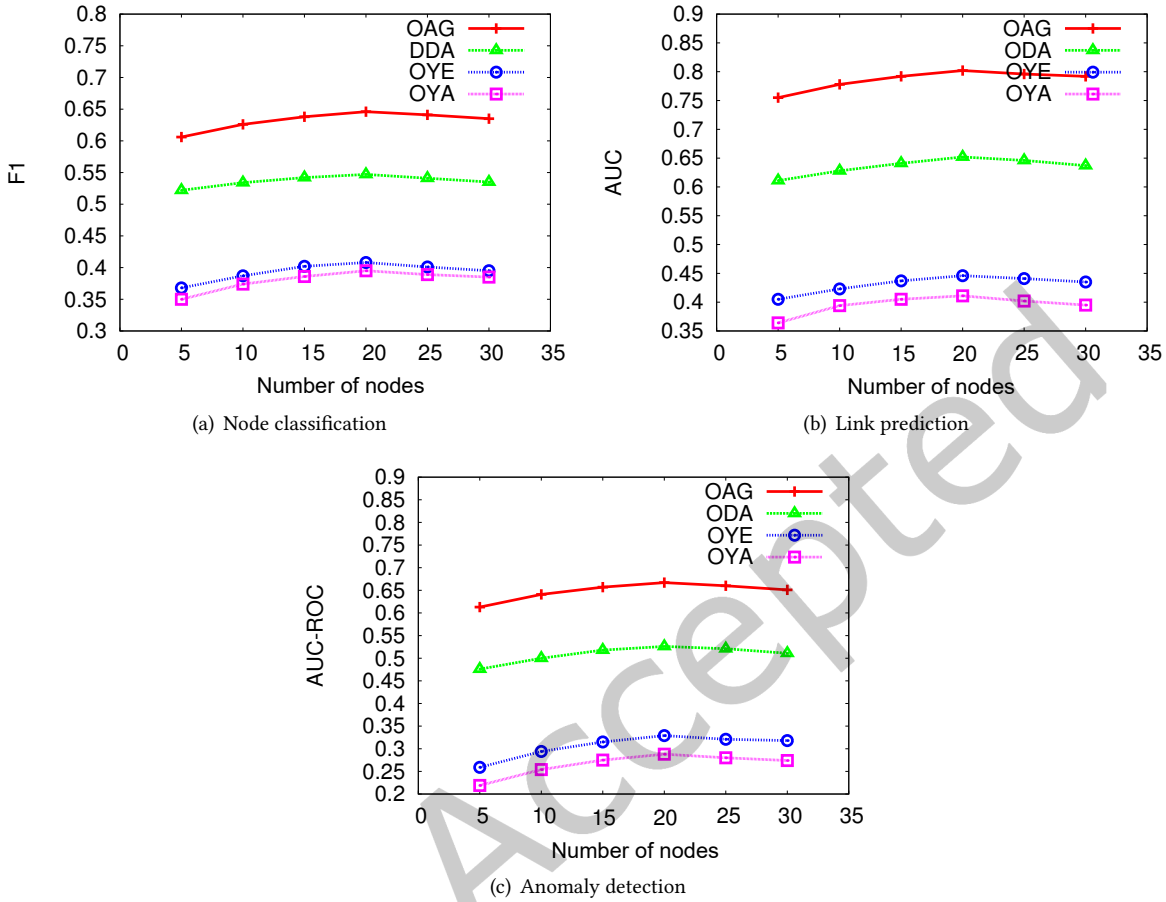
In summary, the individual design choices we made for the components of META-HIN are justified in the sense that obvious alternatives consistently lead to a performance drop.

5.3 Parameter sensitivity analyses

Next, we conduct a parameter sensitivity analysis of META-HIN. Two parameters are chosen for our analysis: (i) the maximum number of nodes of a subgraph, and (ii) the number of shots used for meta-learning. For each task, we only choose one metric for assessment: F1 value for multi-label node classification, AUC for link prediction, and AUC-ROC for anomaly detection. Fig. 5 and 6 present the results of our parameter analyses.

Fig. 5 shows the sensitivity to the maximum number of nodes of the input subgraph. As the number of nodes increases from 0 to 20 (on the X-axis), the model performance correspondingly improves rapidly. A subgraph with only a few nodes may not be able to fully capture the features of a node's neighborhood. However, when the number of nodes exceeds 20, it negatively impacts the model performance. Including too many far-away nodes may introduce noise: the immediate neighborhood already provides sufficient neighborhood information. Based on this analysis, we select the maximum number of nodes of the subgraph to be 20 in our experiments.

As to the number of shots used for meta-learning, unsurprisingly, including more shots leads to better



(a) Node classification

(b) Link prediction

(c) Anomaly detection

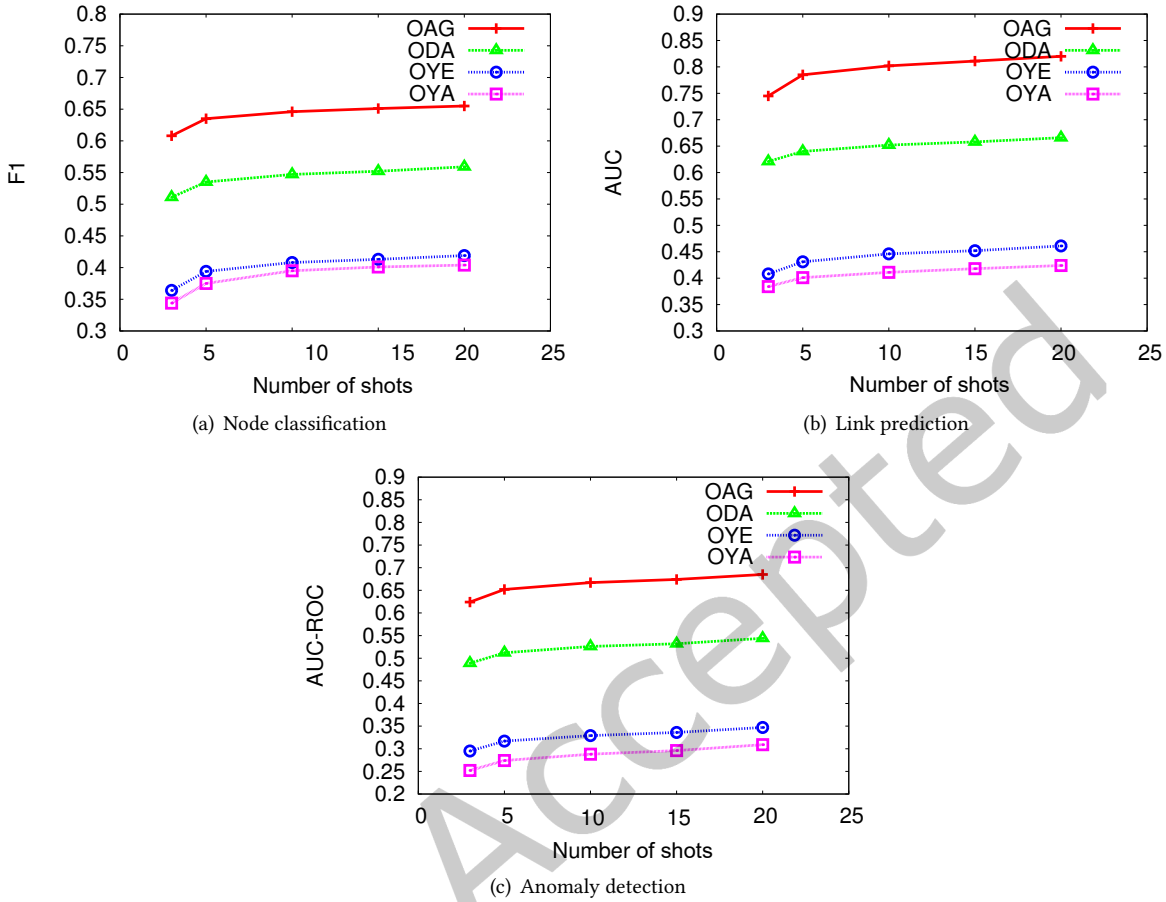
Fig. 5. Sensitivity w.r.t. the maximum number of nodes of sequences.

performance (see Fig. 6). The use of more training and testing samples consistently helps to improve the performance. However, in a few-shot setting, the shots that could be chosen are limited (by definition); choosing the number of shots to be 10 is a reasonable balance between effectiveness and practical limitations.

6 CONCLUSION

Heterogeneous information networks (HINs) are an essential resource in many information retrieval scenarios. We have proposed a novel meta-learning model, META-HIN, to address the few-shot learning problem for HINs. META-HIN is the first model that is applicable to three tasks (i.e., node classification, link prediction, and anomaly detection) across different HINs. As part of META-HIN, we have introduced a structure module, a heterogeneous GNN module, and a GAN-based contrastive module to make effective use of structural, heterogeneous and unsupervised information in a network, respectively. For the link prediction and anomaly detection tasks, we choose a contrastive loss to train the meta-learner. A self-attention mechanism is applied on the training tasks as different tasks have different influences on the meta-learner.

In our experiments, META-HIN consistently and significantly outperforms state-of-the-art methods on every



(a) Node classification

(b) Link prediction

(c) Anomaly detection

Fig. 6. Sensitivity w.r.t. the number of shots for meta-learning.

task across datasets, which demonstrates the advantage of META-HIN. We conduct an ablation analysis to evaluate the effect of different modules or strategies we adopt. The results verify the effectiveness of META-HIN's design. META-HIN can deal with both single-graph scenario and multiple-graph scenarios, while in a multi-graph scenario, aside from graphs from the same distribution, graphs from other domains can also be processed. In addition, META-HIN is able to handle three tasks in a general framework, with each having specific training loss. We also conduct parameter sensitivity analyses to demonstrate the stability of our model.

This work fills the gap on addressing the few-shot learning issues on HINs. It can be used in broader real-life applications such as recommendation. Many recommender systems face cold-start issues with limited items or users, which is a typical few-shot scenario. Our technique is promising to alleviate the above issue if one views the recommender system as a HIN. With only a handful of labels, when coming a new item, META-HIN could help predict which user might be interested like the link prediction task. In addition, anomaly detection could also be conducted by META-HIN in real-life financial or social networks to find the abnormal elements which are rare and limited labeled.

META-HIN can be improved in several ways. First of all, attribute information exists in many HINs, which could

be further incorporated into our framework in future work. Secondly, we aim to realize automatic hyperparameter optimization and reduce the number of parameters to further improve META-HIN's accuracy and efficiency. Furthermore, In contrastive module, we could design more sophisticated negative samples for the model to contrast and differentiate, so as to enhance the model to mine the unsupervised information. Additionally, real-life networks are always evolving, so it is also of interest to design an algorithm to deal with the few-shot learning on dynamic HINs.

ACKNOWLEDGMENTS

This work was partially supported by National Key R&D Program of China No. 2022YFB3102600, NSFC under grants Nos. U23A20296, 62306322 and 62272469, The Science and Technology Innovation Program of Hunan Province No. 2023RC1007, and the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, project LESSEN with project number NWA.1389.20.183 of the research program NWA ORC 2020/21, which is (partly) financed by the Dutch Research Council (NWO), and the FINDHR (Fairness and Intersectional Non-Discrimination in Human Recommendation) project that received funding from the European Union's Horizon Europe research and innovation program under grant agreement No 101070212.

All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Phillip Bonacich. 1987. Power and Centrality: A Family of Measures. *Amer. J. Sociology* 92, 5 (1987), 1170–1182.

[2] Avishek Joey Bose, Ankit Jain, Piero Molino, and William L. Hamilton. 2019. Meta-Graph: Few shot Link Prediction via Meta Learning. *CoRR* abs/1912.09867 (2019).

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.

[4] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Trans. Knowl. Data Eng.* 30, 9 (2018), 1616–1637.

[5] Yue Cui, Hao Sun, Yan Zhao, Hongzhi Yin, and Kai Zheng. 2022. Sequential-Knowledge-Aware Next POI Recommendation: A Meta-Learning Approach. *ACM Trans. Inf. Syst.* 40, 2 (2022), 23:1–23:22. https://doi.org/10.1145/3460198

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*. 3837–3845.

[7] Kaize Ding, Jundong Li, and Huan Liu. 2019. Interactive Anomaly Detection on Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*, J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman (Eds.). ACM, 357–365. https://doi.org/10.1145/3289600.3290964

[8] Kaize Ding, Qinghai Zhou, Hanghang Tong, and Huan Liu. 2021. Few-shot Network Anomaly Detection via Cross-network Meta-learning. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 2448–2456. https://doi.org/10.1145/3442381.3449922

[9] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Trans. Inf. Syst.* 39, 1 (2020), 10:1–10:42. https://doi.org/10.1145/3426723

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1126–1135. http://proceedings.mlr.press/v70/finn17a.html

[11] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-Scale Learnable Graph Convolutional Networks. In *KDD*. 1416–1424.

[12] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A Survey on Knowledge Graph-Based Recommender Systems. *arXiv preprint arXiv:2003.00911* (2020).

[13] Zhichun Guo, Chuxu Zhang, Wenhao Yu, John Herr, Olaf Wiest, Meng Jiang, and Nitesh V. Chawla. 2021. Few-Shot Graph Learning for Molecular Property Prediction. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2559–2567.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*. IEEE Computer Society, 1735–1742.

[15] Hakim Hafidi, Mounir Ghogho, Philippe Ciblat, and Ananthram Swami. 2020. GraphCL: Contrastive Self-Supervised Learning of Graph Representations. *CoRR* abs/2007.08025 (2020).
[16] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html
[17] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *ICML (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 4116–4126.
[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*. IEEE, 9726–9735.
[19] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. *CoRR* abs/1506.05163 (2015). arXiv:1506.05163
[20] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning Deep Representations by Mutual Information Estimation and Maximization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=Bklr3j0cKX
[21] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 2704–2710. https://doi.org/10.1145/3366423.3380027
[22] Kexin Huang and Marinka Zitnik. 2020. Graph Meta Learning via Local Subgraphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/412604be30f701b1b1e3124c252065e6-Abstract.html
[23] Zhiyi Jiang, Jianliang Gao, and Xinqi Lv. 2021. MetaP: Meta Pattern Learning for One-Shot Knowledge Graph Completion. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2232–2236. https://doi.org/10.1145/3404835.3463086
[24] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl
[25] John Boaz Lee, Ryan A. Rossi, and Xiangnan Kong. 2018. Graph Classification using Structural Attention. In *KDD*. 1666–1674. https://doi.org/10.1145/3219819.3219980
[26] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. 2019. CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters. *IEEE Trans. Signal Process.* 67, 1 (2019), 97–109. https://doi.org/10.1109/TSP.2018.2879624
[27] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive Graph Convolutional Neural Networks. In *AAAI*. 3546–3553.
[28] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. 2021. Leveraging Meta-path Contexts for Classification in Heterogeneous Information Networks. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 912–923. https://doi.org/10.1109/ICDE51399.2021.00084
[29] Zijian Li, Wenhao Zheng, Xueling Lin, Ziyuan Zhao, Zhe Wang, Yue Wang, Xun Jian, Lei Chen, Qiang Yan, and Tiezheng Mao. 2020. TransN: Heterogeneous Network Representation Learning by Translating Node Embeddings. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 589–600. https://doi.org/10.1109/ICDE48307.2020.00057
[30] Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Shuai Li, Ruiming Tang, Xiuqiang He, Jianye Hao, and Yong Yu. 2021. A Graph-Enhanced Click Model for Web Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1259–1268.
[31] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven C. H. Hoi. 2021. Relative and Absolute Location Embedding for Few-Shot Node Classification on Graph. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 4267–4275. https://ojs.aaai.org/index.php/AAAI/article/view/16551
[32] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on Heterogeneous Information Networks for Cold-start Recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 1563–1573. https://doi.org/10.1145/3394486.3403207
[33] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
[34] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 3111–3119.
[35] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric Deep

Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*. 5425–5434. https://doi.org/10.1109/CVPR.2017.576

[36] Shanlei Mu, Yaliang Li, Wayne Xin Zhao, Siqing Li, and Ji-Rong Wen. 2022. Knowledge-Guided Disentangled Representation Learning for Recommender Systems. *ACM Trans. Inf. Syst.* 40, 1 (2022), 6:1–6:26. https://doi.org/10.1145/3464304

[37] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In *ICML*. 2014–2023.

[38] Guanglin Niu, Yang Li, Chengguang Tang, Ruiying Geng, Jian Dai, Qiao Liu, Hao Wang, Jian Sun, Fei Huang, and Luo Si. 2021. Relational Learning with Gated and Attentive Neighbor Aggregator for Few-Shot Knowledge Graph Completion. In *SIGIR '21*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 213–222.

[39] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 271–279. https://proceedings.neurips.cc/paper/2016/hash/cedebb6e872f539bef8c3f919874e9d7-Abstract.html

[40] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 259–270.

[41] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. ACM, 1150–1160.

[42] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2020. Knowledge Graphs: An Information Retrieval Perspective. *Foundations and Trends in Information Retrieval* 14, 4 (October 2020), 289–444.

[43] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 4077–4087. https://proceedings.neurips.cc/paper/2017/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html

[44] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.

[45] Yizhou Sun and Jiawei Han. 2012. Mining Heterogeneous Information Networks: A Structural Analysis Approach. *SIGKDD Explorations* 14, 2 (2012), 20–28.

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive Multiview Coding. In *ECCV (Lecture Notes in Computer Science)*, Vol. 12356. Springer, 776–794.

[47] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. 2020. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *ICLR*.

[48] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR* abs/1807.03748 (2018).

[49] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[50] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.

[51] Ning Wang, Minnan Luo, Kaize Ding, Lingling Zhang, Jundong Li, and Qinghua Zheng. 2020. Graph Few-shot Learning with Attribute Matching. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 1545–1554. https://doi.org/10.1145/3340531.3411923

[52] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 2022–2032. https://doi.org/10.1145/3308558.3313562

[53] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. 2018. Unsupervised Feature Learning via Non-Parametric Instance Discrimination. In *CVPR*. IEEE Computer Society, 3733–3742.

[54] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[55] Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. 2020. Automated Relational Meta-learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

[56] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh V. Chawla, and Zhenhui Li. 2020. Graph Few-Shot Learning via Knowledge Transfer. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 6656–6663. https://aaai.org/ojs/index.php/AAAI/article/view/6142

[57] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In

Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019, Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 793–803. https://doi.org/10.1145/3292500.3330961

[58] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *UAI*. 339–349.

[59] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181. https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html

[60] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. 2020. AutoSF: Searching Scoring Functions for Knowledge Graph Embedding. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 433–444. https://doi.org/10.1109/ICDE48307.2020.00044

[61] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 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, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2357–2360.

[62] Zifeng Zhuang, Xintao Xiang, Siteng Huang, and Donglin Wang. 2021. HINFShot: A Challenge Dataset for Few-Shot Node Classification in Heterogeneous Information Network. In *ICMR '21: International Conference on Multimedia Retrieval, Taipei, Taiwan, August 21-24, 2021*. ACM, 429–436. https://doi.org/10.1145/3460426.3463614