GraphMSE: Efficient Meta-path Selection in Semantically Aligned Feature Space for Graph Neural Networks

Yi Li[1], **Yilun Jin**[2], **Guojie Song**[3*], **Zihao Zhu**[4], **Chuan Shi**[4], **Yiming Wang**[1]
[1]Peking University, Beijing, China
[2]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[3]Key Laboratory of Machine Perception (Ministry of Education), Peking University, Beijing, China
[4]Beijing University of Posts and Telecommunications, Beijing, China
{liyi2015, gjsong, wangyiming17}@pku.edu.cn, yilun.jin@connect.ust.hk, {zhuzihao,shichuan}@bupt.edu.cn

Abstract

Heterogeneous information networks (HINs) are ideal for describing real-world data with different types of entities and relationships. To carry out machine learning on HINs, meta-paths are widely utilized to extract semantics with pre-defined patterns, and models such as graph convolutional networks (GCNs) are thus enabled. However, previous works generally assume a fixed set of meta-paths, which is unrealistic as real-world data are overwhelmingly diverse. Therefore, it is appealing if meta-paths can be automatically selected given an HIN, yet existing works aiming at such problem possess drawbacks, such as poor efficiency and ignoring feature heterogeneity. To address these drawbacks, we propose GraphMSE, an efficient heterogeneous GCN combined with automatic meta-path selection. Specifically, we design highly efficient meta-path sampling techniques, and then injectively project sampled meta-path instances to vectors. We then design a novel semantic feature space alignment, aiming to align the meta-path instance vectors and hence facilitate meta-path selection. Extensive experiments on real-world datasets demonstrate that GraphMSE outperforms state-of-the-art counterparts, figures out important meta-paths, and is dramatically (e.g. 200 times) more efficient.

Introduction

Heterogeneous information networks (HINs) (Sun and Han 2012) effectively extend the notion of networks by allowing their nodes and edges to be of different types. Such heterogeneity makes HINs ideal for describing real-world data with complex entities and relations, such as academic data (with authors, papers, conferences, and journals) and e-commerce data (with customers, items, and businesses).

However, along with the versatility of HINs comes with intractability, where the heterogeneity of nodes and edges bear intricate semantics that cannot be extracted in an ordinary manner. Fortunately, researchers propose meta-paths (Sun et al. 2011), paths with pre-defined patterns to extract corresponding semantics from HINs. Based upon meta-paths, numerous HIN mining models have been enabled, including the popular Graph Convolutional Networks (GCNs) (Wang et al. 2019; Zhang et al. 2019).

Yet, the majority of existing works leveraging meta-path based GCNs assume that an indicative set of meta-paths have been given *a priori* upon which the GCNs are built, which is clearly not the case in reality. Specifically, real-world data are notoriously diverse, and it is not even remotely possible to manually select informative meta-paths for every type of HIN. What is more, even on a fixed HIN, selecting optimal meta-paths may require trials and errors, which takes significant effort and also domain expertise. Consequently, it would be highly appreciated if algorithms are developed that can automatically detect important meta-paths from HINs, such that the manual "meta-path engineering" can be alleviated.

Up till now, to the best of our knowledge, GTN (Yun et al. 2019) is the only work that combines automatic meta-path selection with semi-supervised heterogeneous GCNs. However, we identify several drawbacks which cast doubts on GTN. On one hand, GTN leverages matrix multiplications for the selection of meta-paths, which is equivalent to enumerating *all* meta-path instances starting from *all* nodes, which may incur high computational cost. On the other hand, GTN assumes homogeneous features even for heterogeneous nodes, i.e. different types of nodes possess identical feature semantics, which is clearly unlikely in reality.

In this paper, we propose GraphMSE, combining semi-supervised graph neural networks with automatic meta-path selection, and simultaneously address both aforementioned drawbacks. Specifically, first, instead of taking $|V| \times |V|$ matrix multiplications to enumerate *all* meta-paths from *all* nodes, we resort to sampling, both on meta-paths and nodes, which drastically reduces the effort of meta-path enumeration. In addition, we tackle the problem of feature heterogeneity (i.e. different types of nodes possess heterogeneous features) by injectively projecting all meta-path features into the same length, and apply a novel *semantic feature space alignment* technique to mitigate their heterogeneity, which can be viewed as an analogue of disentangled representation (Ma et al. 2019) and further facilitates stable selection. Finally, we utilize an attention layer for meta-path selection. We carry out extensive experiments on real-world HINs, where GraphMSE outperforms state-of-the-art counterparts, while being dramatically more efficient.

Our contributions can be summarized as follows;

- We propose GraphMSE, an efficient and effective frame-

*Corresponding Author

work unifying automatic meta-path selection and semi-supervised graph convolution network on HINs.

- We design efficient node and meta-path sampling techniques that eliminate the need to carry out full matrix multiplication and significantly boosts efficiency.
- We address the problem of feature heterogeneity, which is ignored by GTN, by projecting all meta-paths to one unified feature length and conducting a novel semantic feature space alignment, which further facilitates selection.
- Extensive experiments demonstrate that GraphMSE is competitive against strong baselines while being dramatically more efficient. [1]

Related Work

HINs and Heterogeneous GCNs

HINs (Sun and Han 2012) are able to represent a wider range of data due to node and edge heterogeneity. While homogeneous network mining methods (Kipf and Welling 2017; Veličković et al. 2018) generally fall ineffective due to the heterogeneity, researchers resort to meta-paths (Sun et al. 2011) to extract hierarchical semantics from HINs and develop effective HIN mining algorithms (Shang et al. 2016; Dong, Chawla, and Swami 2017). (Yang et al. 2020) provides an extensive survey on recent advances in mining HINs.

More specifically, GCNs on HINs have become an attractive topic to researchers. For example, HetGNN (Zhang et al. 2019) adopted different RNNs for different types of nodes. GraphInception (Zhang et al. 2018) proposed hierarchical aggregation for feature learning in HINs. HAN (Wang et al. 2019) designed hierarchical attention mechanisms for aggregation from different semantic levels. MAGNN (Fu et al. 2020) proposed to utilize intermediate nodes within multiple meta-paths so as to minimize the information loss. However, these works all assume a given set of meta-paths, which is unrealistic given the diverse nature of real-world HINs.

Automatic Meta-path Selection

The problem of automatic meta-path selection in HINs has been considered from multiple domains. (Wang et al. 2018) proposed unsupervised meta-path selection by building a minimum spanning tree to sort out the importance of meta-paths. (Wei et al. 2018) proposed to select important meta-paths by maintaining meta-path based proximity between nodes. However, both of them consider the problem for information retrieval, but not for representation learning. While (Yang et al. 2018) combines meta-path discovery and representation learning with auto-encoders, semi-supervised GCNs with meta-path selection remain to be explored.

GTN (Yun et al. 2019) is a representative model combining meta-path selection with GCNs. GTN depends on stacking l graph transformer (GT) layers to softly select meta-paths of length l. A GT layer is defined as

$$Q_i = \phi(\mathbb{A}; \text{softmax}(W_\phi)), i = 1, 2 \quad (1)$$

$$A = D^{-1}Q_1Q_2. \quad (2)$$

[1] Visit https://github.com/pkuliyi2015/GraphMSE for code.

After generating C such soft adjacency matrices $\{A_j\}_{j=1}^{C}$, GTN aggregates them by learning a channel-wise GCN and concatenating their outputs (denoted by $\|$).

$$Z = \Big\|_{j=1}^{C} \sigma(A_j X W). \quad (3)$$

We identify two drawbacks of GTN. First, GTN leverages matrix multiplication to extract meta-paths, which is equivalent to enumerating *all* meta-path instances starting from *all* nodes, which is clearly unnecessary. In addition, the feature matrix X implicitly assumes that all nodes share a common feature space, which is unrealistic since different types of nodes generally have varying features. As a simple example, GTN will fall ineffective in the simple case where feature dimensionality varies among node types.

Preliminaries

In this section, we present related backgrounds related to our work. We first present the concepts of HINs and meta-paths.

Definition 1 (HIN). *An HIN is denoted as $G = (V, E, \phi, \psi, \mathcal{A}, \mathcal{R}, \mathcal{X})$, with V and E denoting the node set and the feature set correspondingly. $\mathcal{A}, \mathcal{R}$ denote sets of node and edge types where $|\mathcal{A}| + |\mathcal{R}| > 2$, and $\phi : V \to \mathcal{A}, \psi : E \to \mathcal{R}$ are functions mapping nodes and edges to their types. $\mathcal{X} = \{X_v \in \mathbb{R}^{d_{\phi(v)}}, v \in V\}$ denotes the set of node features, where we allow different types of nodes to have heterogeneous features.*

Definition 2 (Meta-path). *A meta-path P of length l is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} ... \xrightarrow{R_l} A_{l+1}$ (abbreviated as $A_1A_2...A_{l+1}$), where $A_i \in \mathcal{A}, R_j \in \mathcal{R}$, which describes a relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between object types A_1 and A_{l+1}. We denote the set of all meta-paths of length l as $\mathcal{P}_l$, and $\mathcal{P}_{\leq l}$ for those no longer than l.*

We then formulate the concepts of meta-path instances and instance sets.

Definition 3 (Meta-path instances and instance sets). *Given a meta-path P and an HIN G, a meta-path instance p of P is defined as a node sequence in G following the schema defined by P. All meta-path instances p of P starting from node v consists of a meta-path instance set $S_P(v)$.*

Finally, we formalize the task of learning GCNs on HINs.

Definition 4 (GCN learning on HINs). *The task of learning a GCN on an HIN can be formulated as, given an HIN G, a meta-path candidate set $\mathcal{P}'$, find a mapping function $f : V \to \mathbb{R}^{d_G}$ that captures node-level semantic information.*

In this work, we hope to automatically determine important meta-paths from a general candidate set $\mathcal{P}_{\leq l}$, instead of a pre-defined one, upon which the GCN is learned.

Model: GraphMSE

In this section, we introduce our model *Graph Meta-path Selection and Embedding Network*,i.e. *GraphMSE*. Fig. 1 gives a brief overview of GraphMSE.

Our model consists of four major components.

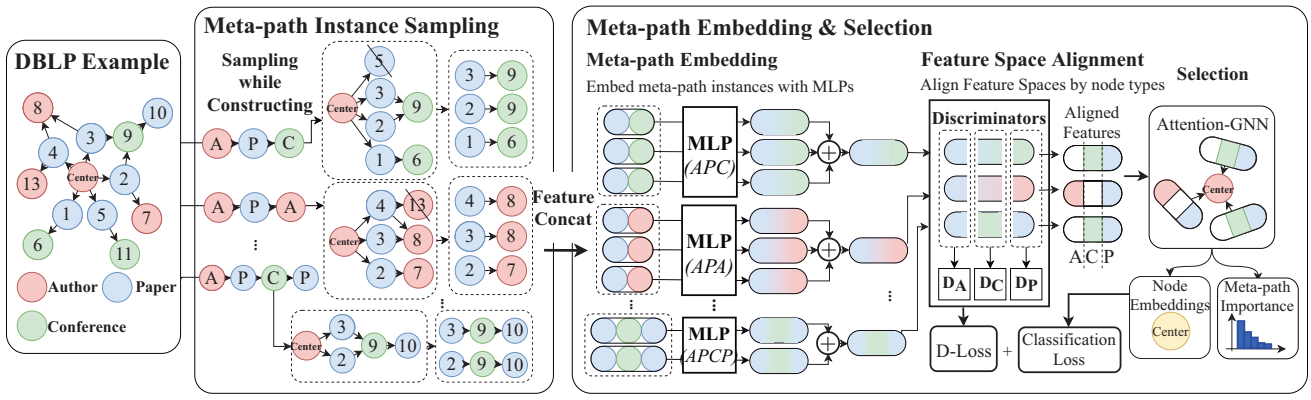


Figure 1: An overview of GraphMSE. GraphMSE consists of four major components: (a) Meta-path Instance Sampling, (b) Meta-path Embedding, (c) Feature Space Alignment. (d) Selection through Attention.

- First, we sample meta-path instances from a subset of nodes $V_C \subset V$ and construct meta-path instance sets, alleviating the need for full matrix multiplications.
- Next, the node features along each meta-path instance will be concatenated and fed into the SUM-MLP encoder corresponding to the meta-path type, injectively projecting them to vectors of equal length.
- In addition, we carry out semantic feature space alignment on the feature space, which disentangles and aligns the feature space to alleviate their heterogeneity, and hence facilitates stable selection.
- Finally, a graph attention layer is used, from which we get node embeddings as outputs and attention weights representing meta-path importance.

Meta-path Extraction

As stated above, the GT layer extracting meta-paths in GTN suffers from low efficiency. In this section, we introduce how we address it via sampling techniques.

Meta-path Sampling In GTN, $|V| \times |V|$ matrix multiplications are carried out, which is equivalent to extracting *all* meta-path instances from *all* nodes and train the GT layers on them. However, we hardly consider it necessary and propose two sampling techniques to accelerate training:

- First, we only sample a candidate node set V_C upon which meta-paths are extracted and GraphMSE is trained. We empirically found that $|V_C| \approx 0.2|V|$ is sufficient to train a well generalizable model.
- Second, we also sample, instead of enumerate, meta-path instances starting from each node. Specifically, we utilize breadth-first search to sample instances under the constraint of each meta-path, and set a limit λ determining at most how many neighbors will be searched in the next step, controlling the scale of neighborhood expansion.

Essentially, the first technique can be seen as a counterpart of inductive learning in homogeneous GCNs, such as GraphSAGE (Hamilton, Ying, and Leskovec 2017), while the second technique, also discussed in (Yang, Zhang, and Han 2019), intentionally discard a proportion of meta-path instances, which can be viewed as a heterogeneous analogue of DropEdge (Rong et al. 2020), which alleviates over-fitting and over-smoothing caused by full neighborhood expansion (Li, Han, and Wu 2018). We demonstrate that sampling is not only sufficient, but may also perform better than enumerating in our ablation studies. We then represent the sampled instances of each meta-path P starting from v as a meta-path instance set $S_P(v)$.

Meta-path Embedding

In this section we introduce how we embed meta-path instance sets $S_P(v)$ into vectors of equal length injectively.

Meta-path Instance Embedding We embed a meta-path instance $p \in S_P(v)$ with features of nodes along it. Considering heterogeneity of node features, taking sum or mean along a path p may not be possible. Therefore, we concatenate all node features, excluding the center node v along each meta-path instance p as its embedding,

$$X_p = \text{CONCAT}(X_{p_1}, X_{p_2}, ..., X_{p_n}). \tag{4}$$

By concatenating, we efficiently preserve the intermediate nodes and their orders in a meta-path instance (Fu et al. 2020), which is helpful towards better representation.

Instance Aggregation We aggregate meta-path instances through an encoder E_P, and then sum them up to obtain the instance set embedding $h_P(v)$.

$$h_P(v) = \sum_{p \in S_P(v)} E_P(X_p). \tag{5}$$

Considering that $S_P(v)$ may be a multi-set (i.e. it may contain identical instances), inspired by GIN (Xu et al. 2019), we prove that above aggregator can injectively map different $S_P(v)$ to different real numbers, and similarly vectors.

Lemma 1. *Assume that the multi-set S_P is countable. There exists an encoder $E_p(X_p) : \mathbb{R}^n \to \mathbb{R}$ so that $h(S_P(v)) = \sum E_p(X_p), p \in S_P(v)$ is unique for each set $S_P(v) \in S_P$ with bounded size.*

We put detailed proofs into appendices. We also show that common aggregators such as MEAN fail to injectively map $S_P(v)$ into vectors:

Lemma 2. *Assume that the multi-set S_P is countable. There exist two multi-sets $S_P(v_1)$, $S_P(v_2)$ with bounded size, so that for $\forall E_p(X_p) : \mathbb{R}^n \to \mathbb{R}$, $h(S_P(v)) = \frac{1}{|S_P(v)|}\sum E_p(X_p), p \in S_P(v)$, we have $h(S_P(v_1)) \equiv h(S_P(v_2))$.*

Essentially, by taking mean over a multi-set, information about cardinality will be lost. In HINs, meta-path instance sets contain structural information, and their cardinalities show relative frequency of different meta-paths. Therefore, using MEAN instead of SUM will lead to loss of structural information, as we will demonstrate in the ablation studies.

Given the existence of an injective E_P, we resort to multilayer perceptrons (MLPs) to approximate them (Hornik, Stinchcombe, and White 1989). As X_p may be of different length due to feature heterogeneity, we use different MLPs for different meta-paths. We obtain the embedding for meta-path instance set $S_P(v)$ as:

$$h_P(v) = \sum_{p \in S_P(v)} \mathrm{MLP}_P\left(\mathrm{CONCAT}\left(X_{p_1}, X_{p_2}, ..., X_{p_n}\right)\right) \tag{6}$$

where $p = (p_1, p_2, ...p_n) \in S_P(v)$.

Though the number of MLPs needed grows exponentially with l, we empirically found that $l \leq 5$ and 1 or 2 layer MLPs are generally sufficient. More importantly, once the model is trained, top-k meta-paths with highest weights and their MLPs would be well sufficient to generate node embeddings. Therefore, the number of MLPs would pose little challenge to efficiency.

Semantic Feature Space Alignment

In the previous section, we injectively project meta-path instance sets to vectors with equal length. However, despite the unified length, the entangled nature of neural networks may lead to arbitrary change in the semantics of the meta-path feature space, i.e. different SUM-MLP encoders may project the same information into arbitrary position of the feature space. Such great discrepancy of feature space will confuse the selector, leading to weights with great variance and thus compromising selection.

To address this problem, we propose to semantically align the feature space, i.e. identical dimensions always correspond to identical semantics, similar to disentangled representations (Ma et al. 2019), so that the selector can stably evaluate different meta-paths. In this work, we propose to manually align the feature space by grouping them according to *node types*, while it is also possible to use other alignment standards, e.g edge types.

Specifically, we let SUM-MLP encoders generate $h_P(v) \in \mathbb{R}^{T \times d}$, where $T = |\mathcal{A}|$ is the number of node types, and d is the dimension for each block. We denote each length d block in $h_P(v)$ as $h_P^i(v), i = 1, ...T$. Correspondingly, we design loss functions L_i to ensure the information of node type A_i resides in and *only in* block $h_P^i(v)$. Intuitively, for node type A_i, when they are correctly projected into $h_P^i(v)$, relatively low loss is expected; on the contrary, when they are projected into wrong blocks, high loss is anticipated.

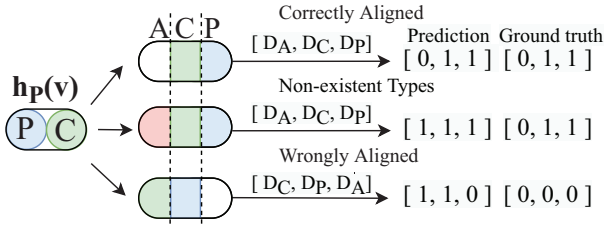


Figure 2: The intuition of the loss. Loss occurs when the semantics are wrongly aligned.

To achieve our purpose, we introduce T discriminators $D_i : \mathbb{R}^d \to \mathbb{R}$ to discriminate the existence of A_i. Ideally, if information of A_i exists in D_i's input, it outputs 1, and 0 otherwise. Thus, a loss exemplifying the previous intuition can be designed as:

$$\begin{aligned} L_{D_i}(v) = \mathbb{E}_{P \in \mathcal{P}_{\leq l}} \Big[& \mathrm{CE}\left(D_i\left(h_P^i(v)\right), y\right) \\ & + \mathbb{E}_{j \neq i}\left[\mathrm{CE}(D_i(h_P^j(v)), 0)\right]\Big] \end{aligned} \tag{7}$$

where $y = 1$ iff $A_i \in P$, CE denotes cross-entropy loss.

We employ $D_i(x) = \sigma(w_i^T x + b_i), \forall i$, where $\sigma(x)$ is the sigmoid function. We illustrate the intuition of Eq. 7 in Fig 2. In implementation, we build $j \neq i$ by shuffling meta-path instance set embeddings $h_P(v)$ in blocks, and train D to detect such incorrect alignments. By joint training of SUM-MLPs and the discriminators, they become strong together, and the feature space would be automatically aligned.

Attention GNN

We finally leverage a graph attention (Veličković et al. 2018) that aggregates all meta-path embeddings around a node v.

$$h_v = W_1 X_v + \sum_{P_i \in \mathcal{P}_{\leq l}} \alpha_i h_{P_i}(v) \tag{8}$$

$$\alpha_i = \frac{1}{|V|}\sum_{v \in V} \alpha_{vi} \tag{9}$$

$$\alpha_{vi} = \frac{\exp(-e_{vi})}{\sum_i \exp(-e_{vi})} \tag{10}$$

$$e_{vi} = \mathrm{LReLU}(a^T W_2 \tanh([h_v || h_{P_i}(v)])), P_i \in \mathcal{P}_{\leq l} \tag{11}$$

The model is trained by minimizing the cross-entropy with respect to the training set labels, which is also the set where we extract meta-paths V_C, along with the discriminator losses (Eq. 7). W_C is the weight matrix for classification.

$$\begin{aligned} \min L = \frac{1}{|V_C|}\sum_{v \in V_C} \Big(& \mathrm{CE}\left(\mathrm{softmax}\left(W_C h_v\right), y_v\right) \\ & + \frac{1}{|\mathcal{A}|}\sum_{A_j \in \mathcal{A}} L_{D_j}(v)\Big). \end{aligned} \tag{12}$$

Upon convergence, we obtain node representations $\{h_v\}, v \in V_C$ along with attention weights of meta-paths

$\alpha_i, P_i \in \mathcal{P}_{\leq l}$. For $v \notin V_C$, their node representations can be generated inductively, either using the full meta-path set, or using meta-paths with the top-k attention weights, as we will show in node classification.

For stable selection, we pre-train the discriminators without attention (i.e. $\alpha_i = \frac{1}{|\mathcal{P}_{\leq l}|}$) to first align the feature space, and then introduce attention to tune the selection layer.

Experiments

In this section, we show the experimental evaluations of GraphMSE. We carry out node classification under various settings and ablation studies to analyze individual components of our model. We also evaluate the efficiency of our model. Finally, we provide qualitative results that help understanding of our model.

Experimental Settings

Datasets We employ three datasets from GTN (Yun et al. 2019) for our experiments: DBLP, IMDB, ACM. The statistics of these datasets are shown in Table 1. We also list the node types below.

- ACM: Paper (P), Author (A), Subject (S).
- DBLP: Author (A), Paper (P), Conference (C);
- IMDB: Movie (M), Actor (A), Director (D);

For ACM and DBLP, target nodes are labeled with research fields. For IMDB, movies are labeled with genres. It is important to note that features are processed to be homogeneous for all three datasets, so as to enable homogeneous GCNs and GTN. We will demonstrate GraphMSE's ability to handle heterogeneous features later.

Baselines We compare GraphMSE with the following competitive baselines.

- **Homogeneous GCNs**: We take GCN (Kipf and Welling 2017) and GAT (Veličković et al. 2018). On HINs, these models will ignore the heterogeneity of nodes.
- **Heterogeneous Network Embedding Models**: We take Metapath2vec (Dong, Chawla, and Swami 2017).
- **Heterogeneous GCNs**: We take HAN (Wang et al. 2019), GTN (Yun et al. 2019) and MAGNN (Fu et al. 2020).

Hyperparameter Settings For all methods, we use 120 dimensional embeddings and split identical train, validation and test sets. For GraphMSE, we set learning rate to 0.01. On ACM and DBLP, we take linear perceptrons. On IMDB, we take 2-layer MLPs. The sampling limit λ for ACM, IMDB, and DBLP is set to 8, 5, and 10, respectively. We put settings for baselines into appendices.

Dataset	Nodes	Edges	Edge Type	Features
ACM	8994	25922	PA,AP,PS,SP	1902
DBLP	18405	67946	PA,AP,PC,CP	334
IMDB	12772	37288	MD,DM,MA,AM	1256

Table 1: Dataset Statistics

Regarding meta-path settings, Metapath2vec, HAN, and MAGNN take fixed meta-paths for training, which we omit for space limitations. For GTN and GraphMSE, we search $l \leq 5$ in $\mathcal{P}_{\leq l}$ and select the optimal l. The optimal l for ACM, DBLP, and IMDB are 2, 4, 3 respectively.

Node Classification

Homogeneous Node Features We carry out node classification on all three datasets with different proportions of training nodes. For the unsupervised method Metapath2vec, classification is carried out via a separate logistic regression. We independently run each experiment 10 times and report means and standard deviations of performances. Table 2 shows all results on node classification. It can be shown that GraphMSE outperforms all strong baselines consistently. What is more, by comparing with GTN, it can be shown that GraphMSE achieves significant improvements across all settings (e.g. by 3% on ACM and 5% on IMDB).

Heterogeneous Node Features While all three datasets are processed to have homogeneous features, we would like to investigate how they perform in feature heterogeneity, which is common in reality. To do so, we randomly shuffle feature indices for each type of nodes, and re-run the node classification experiments with 20% training nodes. The results are shown in Table 3. Compared with Table 2, it can be shown that while GCN and GTN both suffered from drops in performance (e.g. for GCN, 5% and for GTN, 10% on IMDB), GraphMSE is able to maintain performance regardless of the shuffle. Such performance endorses the ability of GraphMSE to handle heterogeneous features.

Meta-path Selection Since the attention weights represent meta-path importance, it should be possible to keep only a small proportion of important meta-paths without compromising performance. Therefore, we carry out node classification with 20% training nodes and only Top-k meta-paths, to validate our selection results.

The meta-paths chosen and results are shown in Table 4, which clearly demonstrate that GraphMSE successfully figures out important meta-paths and hence maintains similar, or even better performance with only 2 or 3 meta-paths.

Ablation Studies

We carry out ablation studies to verify contributions of individual model components. We run node classifications with 20% training node on IMDB to reflect performance changes.

MEAN instead of SUM We replace the sum aggregation by mean in Eq. 6 to empirically verify Lemma 1 and 2. We refer to this variant as GraphMSE-MEAN, and show the results in Table 5. A significant drop is observed with SUM replaced by MEAN, which verifies Lemma 1 and 2.

Feature Space Alignment We run GraphMSE, and GraphMSE without feature space alignment (denoted as GraphMSE-xAlign), and test the classification performance, as well as the meta-path attention values, to demonstrate the effect of Semantic Feature Space Alignment.

We show the performances and attention weights α_i of different meta-paths in Table 5 and 6. It can be shown that

Baseline	20%		40%		60%		80%	
	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
				ACM				
Metapath2vec	76.74±0.95	76.46±9.75	78.76±1.02	78.49±1.01	79.06±0.90	78.79±0.92	80.98±1.27	80.74±1.29
GCN	90.00±0.73	89.96±0.73	90.18±0.80	90.11±0.81	89.98±1.15	90.00±1.10	90.73±1.47	90.68±1.48
GAT	92.29±0.63	92.27±0.60	93.28±0.27	93.29±0.26	92.85±0.35	92.87±0.34	**94.20±0.52**	**94.16±0.51**
HAN	91.03±0.98	90.96±0.97	91.58±0.12	91.55±0.13	92.45±0.35	92.46±0.34	92.83±0.33	92.75±0.32
GTN	91.06±0.73	91.06±0.64	90.63±1.46	90.64±1.48	91.24±0.97	91.13±1.02	91.09±0.84	91.19±0.84
MAGNN	88.22±0.77	88.17±0.78	90.06±0.63	90.04±0.63	90.74±0.50	90.76±0.50	91.04±0.57	91.05±0.57
GraphMSE	**92.97±0.38**	**93.01±0.38**	**93.65±0.16**	**93.62±0.15**	**93.36±0.31**	**93.35±0.31**	93.92±0.33	93.84±0.33
				DBLP				
Metapath2vec	92.14±0.27	92.64±0.25	92.42±0.26	92.94±0.25	92.44±0.31	92.92±0.30	92.74±0.57	93.24±0.54
GCN	90.30±0.93	90.99±0.85	91.26±0.49	91.84±0.41	91.60±0.79	92.06±0.74	91.26±0.49	91.82±0.40
GAT	78.61±0.27	80.30±0.21	79.62±0.84	81.56±0.51	82.16±0.59	83.60±0.41	85.15±0.83	86.08±0.76
HAN	89.55±1.14	90.43±0.99	89.12±2.26	90.03±1.77	90.30±1.52	91.07±1.33	90.97±1.00	91.49±0.90
GTN	91.46±1.86	92.26±1.50	93.41±0.19	93.83±0.17	93.60±0.26	94.02±0.25	93.56±0.19	94.01±0.19
MAGNN	91.97±0.29	92.50±0.26	92.61±0.22	93.13±0.20	92.89±0.19	93.42±0.17	93.06±0.15	93.61±0.14
GraphMSE	**94.03±0.22**	**94.41±0.22**	**94.31±0.16**	**94.64±0.16**	**94.55±0.33**	**94.86±0.30**	**94.60±0.21**	**94.99±0.21**
				IMDB				
Metapath2vec	54.16±0.86	58.15±0.94	58.33±1.53	63.06±1.30	58.37±0.96	63.36±0.96	58.64±1.34	64.04±0.99
GCN	56.92±1.25	60.13±1.84	62.14±1.46	65.41±1.39	63.89±1.45	66.79±1.07	64.71±0.99	67.92±0.71
GAT	44.99±2.05	58.49±0.89	50.40±1.61	62.60±0.76	53.62±1.19	64.38±0.79	54.28±1.67	64.33±0.95
HAN	47.58±2.35	59.14±0.93	54.24±2.53	63.30±0.99	60.89±0.72	66.60±0.38	59.41±1.05	65.38±0.73
GTN	57.59±0.75	**64.75±0.36**	47.91±5.40	63.42±1.56	46.42±0.90	63.73±0.80	47.40±0.74	65.01±0.75
MAGNN	55.74±1.24	60.19±0.95	57.54±1.13	61.85±0.85	58.30±1.19	62.57±0.85	58.93±1.19	63.22±0.91
GraphMSE	**57.78±1.90**	62.63±0.64	**63.47±0.89**	**66.97±0.82**	**65.39±1.13**	**68.24±1.22**	**67.48±1.67**	**70.58±1.16**

Table 2: Experimental results of node classification on three datasets.

Baseline	ACM		DBLP		IMDB	
	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
GCN	90.35±0.95	90.22±0.98	89.30±0.53	90.51±0.45	50.28±1.31	51.76±1.69
GTN	88.95±0.63	88.96±0.62	89.20±1.73	90.06±1.53	43.64±0.52	55.17±0.60
GraphMSE	**92.58±0.50**	**92.54±0.49**	**94.08±0.14**	**94.44±0.13**	**57.60±2.13**	**62.37±1.03**

Table 3: Experimental results of node classification with heterogeneous (shuffled) features.

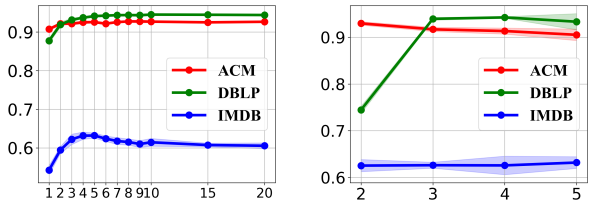
the introduction of feature alignment significantly improved the overall performance, which can be attributed to the lower variance in meta-path selection in Table 6.

Intermediate Nodes in Meta-paths We also carry out ablation studies on our preservation of intermediate nodes within meta-paths. We run experiments with all intermediate nodes discarded, leaving only start and end nodes in Eq. 4. We denote this variant as GraphMSE-xIN and show its results in Table 5. It can be shown that GraphMSE outperforms GraphMSE-xIN, indicating that the preservation of intermediate nodes contributes to better modeling of meta-paths.

Parameter Analysis

We carry out analysis on hyperparameters of our model, specifically sampling parameters λ and l.

First, we analyze λ, namely the maximum number of neighbors to expand during sampling. We vary $\lambda = 1, 2...10, 15, 20$, and plot the results in Fig. 3a. It can be shown that, in general it only takes a few neighbors to expand to achieve satisfactory performance, which contributes to the efficiency of GraphMSE. Moreover, when λ goes excessively large (> 10), we can observe a drop in performance on IMDB, which indicates that discarding meta-path instances actually contributed to better generalization, similar to (Rong et al. 2020).



(a) Sampling limit λ (b) Max meta-path length l

Figure 3: Performances with varying parameters λ and l. Shades denote standard deviation.

Baseline	ACM Top 2: PA,PS		DBLP Top3: AP,APA,APC		IMDB Top3: MD,MAMA,MDM	
	Macro F1	Micro F1	Macro F1	Micro F1	Macro F1	Micro F1
GraphMSE (Full)	**92.97±0.38**	**93.01±0.38**	94.03±0.22	94.41±0.22	57.78±1.90	62.63±0.64
GraphMSE (Top-k)	91.49±1.38	91.43±1.40	94.06±0.32	94.44±0.29	**59.31±2.88**	**63.48±1.84**

Table 4: Experimental results of GraphMSE with top-k meta-paths.

	Macro F1	Micro F1
GraphMSE	**58.46±2.56**	**62.59±1.98**
GraphMSE-MEAN	48.46±2.69	53.70±0.74
GraphMSE-xAlign	56.93±2.11	60.84±1.76
GraphMSE-xIN	52.30±2.69	58.75±1.20

Table 5: Classification Performance of GraphMSE, GraphMSE-MEAN, GraphMSE-xAlign and GraphMSE-xIN on IMDB, 20%

Attention Weights	GraphMSE	GraphMSE-xAlign
MD	19.33±**4.72**	5.86±14.47
MAMA	17.51±**4.59**	22.94±24.17
MDM	13.26±1.83	0.68±1.83
MDMA	13.19±**1.55**	33.46±23.55
MA	11.93±**1.63**	6.63±15.58
MAMD	10.1±**1.11**	6.32±15.55
MAM	9.69±**0.85**	1.06±2.48
MDMD	4.98±**9.79**	23.04±24.26

Table 6: Attention weights of different meta-paths on GraphMSE and GraphMSE-xAlign. Lower variance is bolded.

In addition, we analyze the length of meta-paths to search l. We vary $l = 2, 3, 4, 5$ and show the results in Fig. 3b. It can be shown that generally $l = 3$ or 4 is sufficient for a good performance, while excessively large l will pose challenges towards selection and compromise the performance (lower accuracy and higher variance).

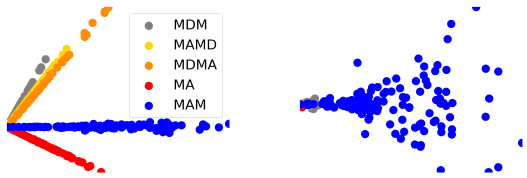
Efficiency

We evaluate the efficiency of baseline methods along with GraphMSE. We run experiments on DBLP, which contains 18405 nodes and 67946 edges. We run all baseline methods for 30 epochs, and for GraphMSE, we run 30 epochs for pre-training, and 30 epochs for tuning the selection layer. We record the average time elapsed for all methods to reach the lowest validation error.

We show the results in Table 7 with two settings: CPU and GPU. For CPU, we use 2 Intel Xeon E5-2697A V4 with 128GB RAM. For GPU, we use one NVIDIA Tesla P100 with 16GB RAM. It can be shown that GraphMSE is astonishingly (200 times) more efficient than its counterpart GTN, and is also slightly (2 or 3 times) more efficient than HAN and MAGNN. Such dramatic improvement in efficiency underscores its competence in addition to accuracy.

Baseline	Meta-paths	CPU	GPU
GCN	N/A	25.50	20.26
HAN	2	88.90	28.67
MAGNN	2	68.83	57.68
GTN	$l \leq 4$	5527.91	OOM
GraphMSE	$l \leq 4$	**22.94**	**11.08**

Table 7: Average time (s) elapsed to achieve lowest validation error. OOM denotes Out Of Memory.



(a) With Alignment (b) Without Alignment

Figure 4: Visualization of meta-path embeddings on IMDB.

Visualization

Meta-path Embeddings We visualize the meta-path embeddings $h_P(v)$ on IMDB to show how our model organizes the feature space. Specifically, we sample 1000 nodes from IMDB, obtain their $h_P(v)$ for 5 meta-paths, and project them into a 2-dimensional space via PCA.

The visualization results are shown in Fig. 4. It can be shown that with feature alignment, GraphMSE is able to project semantics of different meta-paths in a clear and organized manner. On the contrary, the feature space would become cluttered in the absence of the alignment.

Conclusion

In this paper, we propose GraphMSE combining semi-supervised heterogeneous GCNs and automatic meta-path selection. Specifically, we sample, instead of enumerate, meta-path instances starting from nodes, boosting efficiency. We then project all meta-path instances into a unified feature space, which is semantically aligned and disentangled via our elaborate discriminator losses. Extensive experiments demonstrate that our model is highly competitive in both predictive performances and efficiency compared to existing state-of-the-art heterogeneous GCNs.

For future work, we plan to dig further into meta-path sampling techniques, especially how they contribute theoretically to model capability and model learning.

Acknowledgments

We are grateful to Ziyao Li for his insightful advice towards this work. This work was supported by the National Natural Science Foundation of China (Grant No. 61876006).

Appendices

Proof for Lemma 1

The proof is inspired by GIN (Xu et al. 2019).

Proof. We first prove that there exists an encoder $E_p(X_p) : \mathbb{R}^n \to \mathbb{R}$ so that $h(S_P(v)) = \sum E_p(X_p), p \in S_P(v)$ is unique for each multi-set $S_P(v) \in S_P$. Because set S_P is countable, there exists a mapping $Z_P : \mathbb{R}^n \to \mathbb{N}$ from $X_p \in \mathbb{R}^n$ to natural numbers. Because the cardinality of multisets $S_P(v)$ is bounded, there exists a number $N \in \mathbb{N}$ so that $|S_P(v)| < N$ for all $|S_P(v)|$.

(Xu et al. 2019) provides an example encoder:

$$h_P = \sum E_P(X_p) = \sum \frac{1}{N^{Z(X_p)}}, p \in S_P(v)$$

Since $\frac{1}{N^{Z(X_p)}}$ can be regarded as a unique base N representation of X_p, $h_P = \sum E_P(X_p)$ is injective. In other words, $h(S_P(v)) = \sum E_p(X_p), p \in S_P(v)$ is unique for each multi-set $S_P(v) \in S_P$.

Proof for Lemma 2

Proof. Suppose multisets $S_p(v_1)$ and $S_p(v_2)$ have the same underlying set S. When each meta-path instance in $S_p(v_1)$ is k times of that in $S_p(v_1)$, for $\forall E_p(X_p) : \mathbb{R}^n \to \mathbb{R}$, $h(S_P(v)) = \frac{1}{|S_P(v)|} \sum E_p(X_p), p \in S_P(v)$, we have

$$\begin{aligned} h(S_P(v_1)) &= \frac{1}{|S_P(v_1)|} \sum E_p(X_p), p \in S_P(v_1) \\ &= \frac{1}{k|S_P(v_2)|} \sum kE_p(X_p), p \in S_P(v_2) \\ &= \frac{1}{|S_P(v_2)|} \sum E_p(X_p), p \in S_P(v_2) \\ &= h(S_P(v_2)) \end{aligned}$$

Hence, for $\forall E_p(X_p) : \mathbb{R}^n \to \mathbb{R}$, we have $h(S_P(v_1)) \equiv h(S_P(v_2))$.

Hyperparameters

For some baselines, we apply their hyperparameter settings as corresponding papers. For other baselines, we empirically optimize their hyperparameters. The hyperparameter details are as follow:

- For metapath2vec (MP2vec for short), we set window size to 11, walk length to 100, walks per node to 40, the number of negative samples to 5, learning rate to 0.01, epochs to 50. We do not evaluate its efficiency since the official version of metapath2vec is implemented in C++ instead of Python.
- For GCN and GAT, we optimize the learning rate and the weight-decay parameter with the validation set, and set epochs to 200.

Baseline	ACM	DBLP	IMDB
MP2vec	SPAPS	CPAPC	DMAMD
HAN	PAP PSP	APA APCPA	MAM MDM
MAGNN	PAP PSP APA APSPA SPS SPAPS	APA APCPA	MAM MDM DMD DMAMD AMA AMDMA
GTN	Length≤ 3	Length≤ 4	Length≤ 4
GraphMSE	Length≤ 2	Length≤ 4	Length≤ 4

Table 8: Meta-path settings

- For HAN and GTN, since ACM, DBLP, and IMDB datasets are also used in their experiments, we directly apply the corresponding settings in their papers.
- For MAGNN, since settings for DBLP and IMDB are already provided in its paper, we directly apply the settings. We transfer the hyper-parameters for DBLP to ACM, then optimize them with the validation set.
- For proposed GraphMSE, we set learning rate to 0.01, pre-training epochs to 50, selection epochs to 50, neighbor sampling limit λ for ACM, IMDB, DBLP to 8, 5, 10, respectively. On ACM and DBLP, we take linear perceptrons. On IMDB, we take 2-layer MLPs.

Meta-path settings vary with datasets and tasks. Table 8 shows related settings. For node classification tasks, we optimize max-length limits for GTN and GraphMSE with validation sets.

Since the optimal meta-path lengths on ACM and IMDB in node classification tasks are relatively short, we demonstrate the ability of GraphMSE with longer meta-paths in ablation studies.

References

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 135–144.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*, 2331–2341.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, 1024–1034.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2(5): 359–366.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR 2017 : International Conference on Learning Representations 2017*.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learn-

ing. In *AAAI-18 AAAI Conference on Artificial Intelligence*, 3538–3545.

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019. Disentangled graph convolutional networks. In *International Conference on Machine Learning*, 4212–4221.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

Shang, J.; Qu, M.; Liu, J.; Kaplan, L. M.; Han, J.; and Peng, J. 2016. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769* .

Sun, Y.; and Han, J. 2012. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery* 3(2): 1–159.

Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4(11): 992–1003.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR 2018 : International Conference on Learning Representations 2018*.

Wang, C.; Song, Y.; Li, H.; Zhang, M.; and Han, J. 2018. Unsupervised meta-path selection for text similarity measure based on heterogeneous information networks. *Data Mining and Knowledge Discovery* 32(6): 1735–1767.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*, 2022–2032.

Wei, X.; Liu, Z.; Sun, L.; and Yu, P. S. 2018. Unsupervised meta-path reduction on heterogeneous information networks. *arXiv preprint arXiv:1810.12503* .

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks. In *ICLR 2019 : 7th International Conference on Learning Representations*.

Yang, C.; Liu, M.; He, F.; Zhang, X.; Peng, J.; and Han, J. 2018. Similarity Modeling on Heterogeneous Networks via Automatic Path Discovery. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML-PKDD 2018*, 37–54.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *arXiv preprint arXiv:2004.00216* .

Yang, C.; Zhang, J.; and Han, J. 2019. Neural Embedding Propagation on Heterogeneous Networks. In *2019 IEEE International Conference on Data Mining (ICDM)*, 698–707.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. In *Advances in Neural Information Processing Systems*, 11983–11993.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 793–803.

Zhang, Y.; Xiong, Y.; Kong, X.; Li, S.; Mi, J.; and Zhu, Y. 2018. Deep collective classification in heterogeneous information networks. In *Proceedings of the 2018 World Wide Web Conference*, 399–408.