Heterogeneous Graph Masked Autoencoders

Yijun Tian[1,2], Kaiwen Dong[1,2], Chunhui Zhang[3], Chuxu Zhang[3], Nitesh V. Chawla[1,2]

[1] Department of Computer Science and Engineering, University of Notre Dame, USA
[2] Lucy Family Institute for Data and Society, University of Notre Dame, USA
[3] Department of Computer Science, Brandeis University, USA
{yijun.tian, kdong2}@nd.edu, {chunhuizhang, chuxuzhang}@brandeis.edu, nchawla@nd.edu

Abstract

Generative self-supervised learning (SSL), especially masked autoencoders, has become one of the most exciting learning paradigms and has shown great potential in handling graph data. However, real-world graphs are always heterogeneous, which poses three critical challenges that existing methods ignore: 1) *how to capture complex graph structure?* 2) *how to incorporate various node attributes?* and 3) *how to encode different node positions?* In light of this, we study the problem of generative SSL on heterogeneous graphs and propose HGMAE, a novel heterogeneous graph masked autoencoder model to address these challenges. HGMAE captures comprehensive graph information via two innovative masking techniques and three unique training strategies. In particular, we first develop metapath masking and adaptive attribute masking with dynamic mask rate to enable effective and stable learning on heterogeneous graphs. We then design several training strategies including metapath-based edge reconstruction to adopt complex structural information, target attribute restoration to incorporate various node attributes, and positional feature prediction to encode node positional information. Extensive experiments demonstrate that HGMAE outperforms both contrastive and generative state-of-the-art baselines on several tasks across multiple datasets. Codes are available at https://github.com/meettyj/HGMAE.

Introduction

Heterogeneous graphs are ubiquitous in the real world with their ability to model heterogeneous relationships among different types of nodes, such as academic graphs (Wang et al. 2019), social graphs (Cao et al. 2021), biomedical graphs (Bai et al. 2021), and food graphs (Tian et al. 2022b). Accordingly, many heterogeneous graph neural networks (HGNNs) are proposed (Hu et al. 2020; Tian et al. 2021; Zhang et al. 2019) to capture the complex structure and rich semantics in heterogeneous graphs. Generally, HGNNs achieve great success in handling graph heterogeneity and have been applied in various real-world applications such as recommendation systems (Tian et al. 2022a) and healthcare systems (Wang et al. 2021b). However, most HGNNs adhere to the supervised or semi-supervised learning paradigms (Wang et al. 2021a), in which the learning process is guided by labeled data. Nevertheless, the strict requirement for labels

is impractical because acquiring labels in real applications is always challenging and expensive (Wang et al. 2020). Therefore, self-supervised learning (SSL), which attempts to extract information from the data itself, becomes a promising solution when no or few explicit labels are provided.

In the context of SSL on graphs, contrastive SSL methods are the dominant approaches (Zhao et al. 2023, 2021). The success of contrastive methods is largely built upon manually selected high-quality data augmentations and complex optimization algorithms (Hou et al. 2022). However, most of the augmentations on graphs are based on heuristics, with performance varying significantly across different graphs and tasks (You et al. 2021; Ding et al. 2022; You et al. 2020). On the other hand, complicated optimization algorithms such as momentum and exponential moving average for parameter update are always required due to the computational constraint and the need for stable training (Qiu et al. 2020; Thakoor et al. 2022). In addition, negative sampling, as a necessity for the majority of contrastive objectives, often involves laborious designs and arduous constructions from graphs. Therefore, there is no assurance that contrastive methods will achieve satisfactory results.

Generative SSL naturally avoids the aforementioned issues of contrastive methods, by focusing on directly reconstructing the input graph data without considering complicated augmentations and negative sampling (Liu et al. 2020; Wu et al. 2021). Recently, masked autoencoders demonstrate strong learning capability in computer vision (He et al. 2022) and natural language processing (Devlin et al. 2019) by removing a large proportion of the input data and using the removed content to guide the training. Inspired by this, GraphMAE (Hou et al. 2022) proposes to reconstruct node features with masking on graphs. Although GraphMAE shows excellent performance in certain graph learning tasks, it ignores the following challenges faced for real-world graphs and limits its applicability to real applications.

- **C1**: Real-world graphs are usually heterogeneous, with multiple types of nodes and edges, which naturally implies their complex structure. To learn effective node embeddings that fully consider the semantics involved in this complex structure, simply reconstructing the masked features is far from sufficient. Thus the challenge 1 is: how to capture the complex graph structure that contains informative semantics, as indicated by C1 in Figure 1.
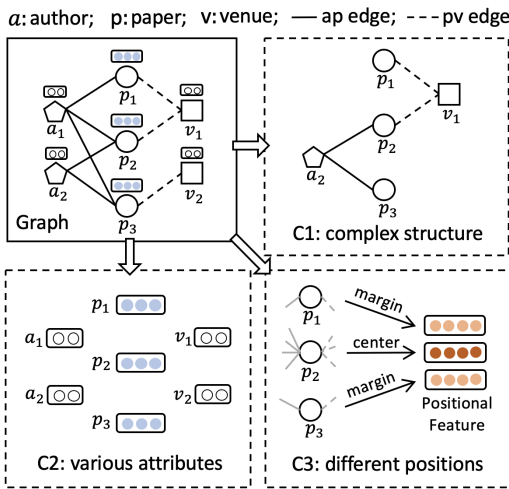
Figure 1: The illustration of three challenges in real-world graphs: C1 - complex structure with heterogeneous types of edges; C2 - various attributes that each node type associates; C3 - different positions that every node holds in the graph.

- **C2**: Different types of nodes are associated with various types of attributes. It is inappropriate to fixate the attributes of every node type and apply the same masking strategy on top of them. Thus the challenge 2 is: how to determine which node type to mask and how to design a suitable masking strategy, as indicated by C2 in Figure 1.

- **C3**: Different nodes can carry different structural roles and be influenced by other nodes with respect to their positions in the graph. In addition to graph structure and attributes, it is important to encode the node positional information from the graph. Thus the challenge 3 is: how to incorporate different node positions and design learning objectives on top of it, as indicated by C3 in Figure 1.

To address these challenges, we propose HGMAE, a novel heterogeneous graph masked autoencoder for generative SSL on graphs. HGMAE contains two innovative masking techniques and three unique training strategies. Specifically, we first develop metapath masking and adaptive attribute masking to enable the encoding of complex information in heterogeneous graphs. We also employ a dynamic mask rate in adaptive attribute masking to ensure more effective and stable learning. In addition, we validate the effectiveness of leaving unchanged and replacing for masking in the graph domain, and reach a conclusion that differs from GraphMAE. Then, we design a metapath-based edge reconstruction strategy to capture the high-order relations through metapaths and encode complex graph structural information. Next, we introduce a target attribute restoration strategy to reconstruct the attributes of target nodes. The model is encouraged to fully grasp the knowledge in attributes and maintain the learning capability across different masking settings. After that, we develop a positional feature prediction strategy to fully capture the node positional information. Finally, we design a novel combined objective function of different strategies to optimize the model. To summarize, our major contributions in this paper are as follows:

- To our best knowledge, this is the first attempt to study generative self-supervised learning on heterogeneous graphs. We identify and address three critical challenges pertaining to this problem.
- We develop two novel masking techniques including metapath masking and adaptive attribute masking. In addition, we introduce a dynamic mask rate and verify the effectiveness of leaving unchanged and replacing in the graph domain, which contradicts to previous findings.
- We propose HGMAE, a novel heterogeneous graph masked autoencoder model to address the challenges. HGMAE can capture comprehensive graph information including complex structure, various attributes and different node positions via several training strategies.
- Extensive experiments demonstrate the superiority of HGMAE compared to contrastive and generative state-of-the-art baselines on several tasks across multiple datasets.

Related Work

This work is closely related to heterogeneous graph neural networks and self-supervised learning on graphs.

Heterogeneous Graph Neural Networks. In the past few years, many heterogeneous graph neural networks have been developed to learn node representations in heterogeneous graphs (Yang et al. 2020; Wang et al. 2022; Fan et al. 2022). For example, HAN (Wang et al. 2019) proposes hierarchical attention to encode node-level and semantic-level structures. MAGNN (Fu et al. 2020) leverages the node attributes and intermediate nodes of metapaths to encapsulate rich semantic information of heterogeneous graphs. HGT (Hu et al. 2020) introduces a transformer-based architecture to handle web-scale heterogeneous graphs. HetGNN (Zhang et al. 2019) proposes to capture both structure and content information in heterogeneous graphs. However, the aforementioned methods can only learn when node labels are provided and are unable to extract supervised signals from the data itself to learn general node embeddings. Therefore, how to exploit the complex information involved in heterogeneous graphs as self-supervised guidance remains a question.

Self-supervised Learning on Graphs. Self-supervised methods on graphs can be naturally divided into contrastive and generative approaches (Liu et al. 2022; Yu et al. 2022). Contrastive graph learning encourages alignment between different augmentations or distributions. For example, GCC (Qiu et al. 2020) focuses on aligning different local structures of two sampled subgraphs. GraphCL (You et al. 2020) introduces the alignment between different graph augmentations. HeCo (Wang et al. 2021a) applies two network structures to align both local schema and global metapath information. However, the performance of these contrastive graph learning models highly depends on the manually selected data augmentations, which vary from graph to graph. On the other hand, generative graph learning aims to recover missing parts of the input data. In previous research, the performance of generative methods falls behind the contrastive methods (Kipf and Welling 2016; Park et al. 2019; Garcia Duran and Niepert 2017). Recently,

GraphMAE (Hou et al. 2022) leverages masked autoencoder to reconstruct features, achieving outstanding performance. However, GraphMAE disregards several challenges in real-world data such as the complex structure, various attributes and different node positions, which we address in this paper.

Problem Definition

In this section, we describe the concept of heterogeneous graph and formally define the problem of generative self-supervised learning on heterogeneous graphs.

Definition 1. Heterogeneous Graph. *A heterogeneous graph is defined as a graph $G = (\mathcal{V}, \mathcal{A}, T_\mathcal{V}, T_\mathcal{E}, X, \Phi)$ with multiple types of nodes $\mathcal{V}$ and adjacency matrix $\mathcal{A}$, where $T_\mathcal{V}$ represent the set of node types and $T_\mathcal{E}$ is the set of edge types in $\mathcal{A}$. X denotes the attributes and Φ represents the metapaths. A metapath $\phi \in \Phi$ is a path that connects different types of nodes with distinct types of edges, i.e., $T_{\mathcal{V}_1} \xrightarrow{T_{\mathcal{E}_1}} T_{\mathcal{V}_2} \xrightarrow{T_{\mathcal{E}_2}} \dots \xrightarrow{T_{\mathcal{E}_l}} T_{\mathcal{V}_{(l+1)}}$, where l is path length.*

Problem 1. Generative Self-supervised Learning on Heterogeneous Graphs. *Given a heterogeneous graph $G = (\mathcal{V}, \mathcal{A}, T_\mathcal{V}, T_\mathcal{E}, X, \Phi)$, the task is to first design an encoder f_E that learns node embeddings $H = f_E(G)$, which encodes complex heterogeneous graph information such as structure, attributes, and node positions. Then, a decoder f_D is designed to reconstruct the input, indicated as $G' = f_D(H)$, where G' can be either reconstructed adjacency matrix, attributes or node positions. The learned embeddings H can be utilized in various downstream graph mining tasks, such as node classification and node clustering.*

Heterogeneous Graph Masked Autoencoders

In this section, we formally present HGMAE to resolve the challenges described in Introduction. In particular, HGMAE introduces two masking techniques and contains three training strategies: (1) metapath-based edge reconstruction; (2) target attribute restoration; (3) positional feature prediction. Figure 2 illustrates the framework of HGMAE.

Metapath-based Edge Reconstruction

In order to capture the semantics involved in the complex graph structure, we design the metapath-based edge reconstruction strategy to explore the high-order relations through metapaths and encode the complex graph structural information. Specifically, masking the metapath-based edges breaks the short-range semantic connections between nodes, forcing the model to look elsewhere to predict the relations that are masked out. As a result, it can exploit the structure-dependency patterns more effectively and capture the high-order proximity naturally. In particular, given a heterogeneous graph $G = (\mathcal{V}, \mathcal{A}, T_\mathcal{V}, T_\mathcal{E}, X, \Phi)$, we create the metapath-based adjacency matrix $\mathcal{A}^\phi$ for each metapath $\phi \in \Phi$ via metapath sampling (Dong, Chawla, and Swami 2017). Since different metapaths contain different semantic information, we take them into account separately by masking and reconstructing each metapath-based adjacency matrix individually. Concretely, for each $\mathcal{A}^\phi$, we create a binary mask following the Bernoulli distribution $M_\mathcal{A}^\phi \sim$

Bernoulli(p_e), where $p_e < 1$ is the edge masking rate. Then, we leverage the $M_\mathcal{A}^\phi$ to obtain the masked metapath-based adjacency matrix $\widetilde{\mathcal{A}}^\phi = M_\mathcal{A}^\phi \cdot \mathcal{A}^\phi$. After that, we feed $\widetilde{\mathcal{A}}^\phi$ and node attributes X into encoder f_E to obtain the latent node embeddings H_1^ϕ. The process is formulated as follows:

$$H_1^\phi = f_E(\widetilde{\mathcal{A}}^\phi, X). \qquad (1)$$

After that, we send H_1^ϕ with $\widetilde{\mathcal{A}}^\phi$ into the decoder f_D to generate the decoded node embeddings H_2^ϕ. Later, we leverage H_2^ϕ to reconstruct the adjacency matrix:

$$H_2^\phi = f_D(\widetilde{\mathcal{A}}^\phi, H_1^\phi), \quad \mathcal{A}^{\phi\prime} = \sigma((H_2^\phi)^T \cdot H_2^\phi), \quad (2)$$

where $\mathcal{A}^{\phi\prime}$ is the reconstructed metapath-based adjacency matrix and σ is the sigmoid activation function. Next, we compare the target $\mathcal{A}^\phi$ and the reconstruction $\mathcal{A}^{\phi\prime}$ by each node using a scaled cosine error:

$$\mathcal{L}^\phi = \frac{1}{|\mathcal{A}^\phi|} \sum_{v \in \mathcal{V}} (1 - \frac{\mathcal{A}_v^\phi \cdot \mathcal{A}_v^{\phi\prime}}{\|\mathcal{A}_v^\phi\| \times \|\mathcal{A}_v^{\phi\prime}\|})^{\gamma_1}, \qquad (3)$$

where $\mathcal{L}^\phi$ is the loss for metapath ϕ and γ_1 is the scaling factor. To combine $\mathcal{L}^\phi$ for each metapath $\phi \in \Phi$, we first introduce a semantic-level attention vector q to automatically learn the important score s^ϕ of each metapath. Then, we normalize the scores of all metapaths using softmax function to obtain the weight of metapath ϕ, denoted as α^ϕ. The process is shown as follows:

$$s^\phi = q^T \cdot \tanh(W \cdot H_1^\phi + b), \quad \alpha^\phi = \frac{\exp(s^\phi)}{\sum_{\phi \in \Phi} \exp(s^\phi)}, \quad (4)$$

where W is the weight matrix and b is the bias vector. After that, we fuse the loss for each metapath to obtain the combined loss $\mathcal{L}_{\text{MER}}$ for metapath-based edge reconstruction:

$$\mathcal{L}_{\text{MER}} = \sum_{\phi \in \Phi} \alpha^\phi \cdot \mathcal{L}^\phi. \qquad (5)$$

Target Attribute Restoration

In order to explore the content information involved in node attributes and facilitate the model to focus on the target node type, we design the target attribute restoration strategy. In particular, we mask the attributes for nodes of target type, and let the model reconstruct the masked attributes. Considering the node attributes play a vital role in deriving the node embeddings, the model can benefit from restoring the attributes and preserving the meaningful content information in the training loop. In addition, we develop an adaptive attribute masking technique that includes dynamic mask rate, leaving unchanged and replacing to ensure more effective and stable learning.

Dynamic Mask Rate. Masking a variable percentage of attributes instead of using a fixed mask rate enables the model to maintain stable performance despite the changeable amount of input information (Gupta et al. 2022). Concretely, we gradually increase the attribute mask rate according to a linear mask scheduling function so that the model can
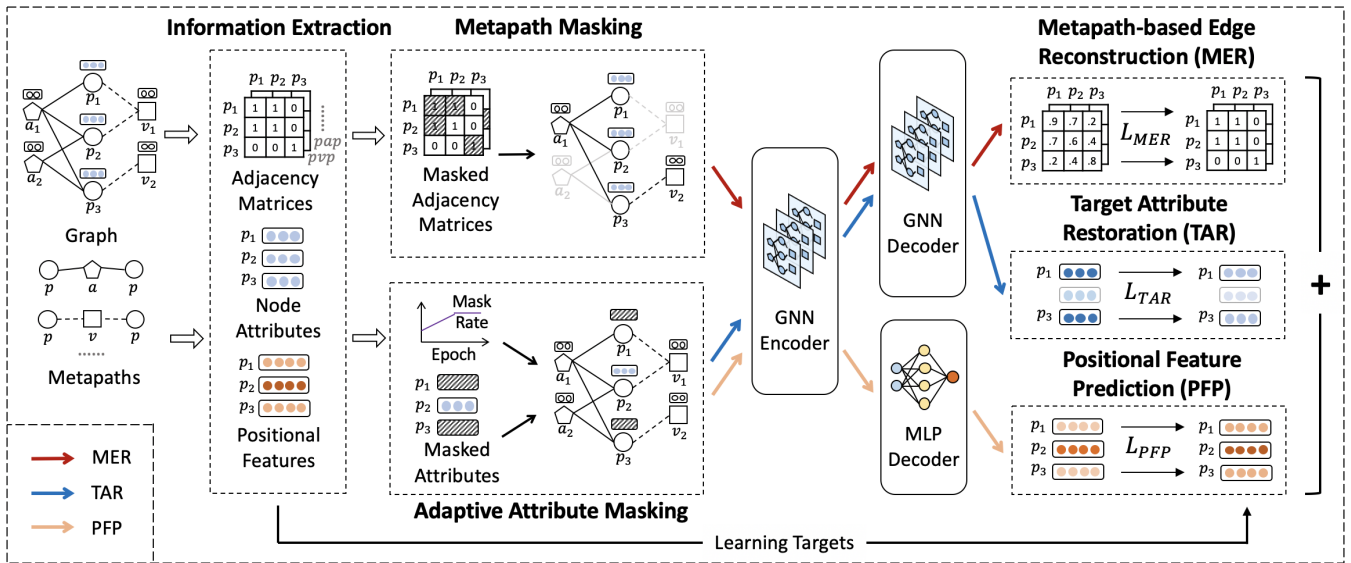
Figure 2: The overall framework of HGMAE: we first extract the metapath-based adjacency matrices, node attributes and positional features from the graph. We then design two masking techniques (i.e., metapath masking and adaptive attribute masking) to mask the inputs. Later, we feed the masked inputs into the encoder and decoders sequentially, and optimize them via three training strategies (i.e., metapath-based edge reconstruction, target attribute restoration, and positional feature prediction). The proposed strategies enable the model to capture comprehensive graph information and address identified challenges.

adaptively learn from easy to difficult. Formally, let $\delta(m)$ represent the mask scheduling function that computes the attribute mask rate p_a with respect to training epoch m, where $m \in \{0, 1, ..., \mathcal{M}\}$ and $\mathcal{M}$ is the total number of training epochs. We choose $\delta(m)$ such that it is linearly increasing by step Δ for each epoch m, namely, $\delta(m+1) = \delta(m) + \Delta$. We further set $\delta(0) = \text{MIN}_{p_a}$, $\delta(\mathcal{M}) = \text{MAX}_{p_a}$, and $\delta(m) \leq \delta(\mathcal{M})$ to ensure our model converges, where Δ, MIN_{p_a}, MAX_{p_a} are hyper-parameters adjustable over different datasets. After we get the attribute mask rate $p_a = \delta(m)$ given epoch m, we first sample a subset of nodes $\widetilde{\mathcal{V}} \in \mathcal{V}_t$ with rate p_a for target node type t. We then mask each of their attributes using a learnable mask token $[M]$. Correspondingly, for each $v \in \mathcal{V}_t$, the node attribute $\widetilde{\boldsymbol{x}}_v$ in the masked attribute matrix $\widetilde{X}$ can be defined as:

$$\widetilde{\boldsymbol{x}}_v = \begin{cases} x_{[M]} & \text{if } v \in \widetilde{\mathcal{V}} \\ x_v & \text{if } v \notin \widetilde{\mathcal{V}}. \end{cases} \quad (6)$$

Leaving Unchanged and Replacing. Due to the fact that the mask token $[M]$ does not appear during inference, a mismatch between training and inference can occur, jeopardizing the model's ability to learn (Yang et al. 2019). Therefore, we introduce the Leaving Unchanged and Replacing methods. Specifically, we first replace a percentage of mask tokens by random tokens, with the replace rate p_r. In addition, we select another percentage of nodes with rate p_u and leave them unchanged by utilizing the origin attribute x_v, while p_u indicates the leave unchanged rate. The idea of leaving unchanged and replacing is simple, yet as we will see, effective and practical, which is also the opposite of what GraphMAE claims. After that, we send node attributes $\widetilde{X}$

and graph adjacency matrix $\mathcal{A}$ into encoder f_E to obtain the latent node embeddings H_3. To enforce the encoder learns informative embeddings without relying on the decoder's capability for restoration, we apply another mask token $[DM]$ to H_3 before sending it into the decoder. The process is formulated as follows:

$$H_3 = f_E(\mathcal{A}, \widetilde{X}), \quad \widetilde{H}_3 = \begin{cases} h_{[DM]} & \text{if } v_i \in \widetilde{\mathcal{V}} \\ h_i & \text{if } v_i \notin \widetilde{\mathcal{V}}, \end{cases} \quad (7)$$

where $\widetilde{H}_3$ denotes the masked latent node embeddings. Next, we send $\mathcal{A}$ and $\widetilde{H}_3$ into the decoder f_D to obtain the restored node attributes Z:

$$Z = f_D(\mathcal{A}, \widetilde{H}_3). \quad (8)$$

Subsequently, we define the loss of target attribute restoration $\mathcal{L}_{\text{TAR}}$ by comparing masked attribute matrix $\widetilde{X}$ and Z with scaling factor γ_2. The loss function is described as follows:

$$\mathcal{L}_{\text{TAR}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v \in \widetilde{\mathcal{V}}} (1 - \frac{\widetilde{X}_v \cdot Z_v}{\|\widetilde{X}_v\| \times \|Z_v\|})^{\gamma_2}. \quad (9)$$

Positional Feature Prediction

In order to incorporate the positional information for each node, we design the positional feature prediction strategy. In particular, we first extract metapath aware positional features P using Mp2vec (Dong, Chawla, and Swami 2017). Then, we let the model predict the positional features while only using the latent node embeddings H_3 from the encoder f_E. To encourage the encoder to take positional information into

Datasets	Metric	Split	GraphSAGE	GAE	Mp2vec	HERec	HetGNN	HAN	DGI	DMGI	HeCo	GraphMAE	HGMAE
DBLP	Mi-F1	20	71.44±8.7	91.55±0.1	89.67±0.1	90.24±0.4	90.11±1.0	90.16±0.9	88.72±2.6	90.78±0.3	91.97±0.2	89.31±0.7	**92.71±0.5**
		40	73.61±8.6	90.00±0.3	89.14±0.2	90.15±0.4	89.03±0.7	89.47±0.9	89.22±0.5	89.92±0.4	90.76±0.3	87.80±0.5	**92.43±0.3**
		60	74.05±8.3	90.95±0.2	91.17±0.1	91.01±0.3	90.43±0.6	90.34±0.8	90.35±0.8	90.66±0.5	91.59±0.2	89.82±0.4	**93.05±0.3**
	Ma-F1	20	71.97±8.4	90.90±0.1	88.98±0.2	89.57±0.4	89.51±1.1	89.31±0.9	87.93±2.4	89.94±0.4	91.28±0.2	87.94±0.7	**92.28±0.5**
		40	73.69±8.4	89.60±0.3	88.68±0.2	89.73±0.4	88.61±0.8	88.87±1.0	88.62±0.6	89.25±0.4	90.34±0.3	86.85±0.7	**92.12±0.3**
		60	73.86±8.1	90.08±0.2	90.25±0.1	90.18±0.3	89.56±0.5	89.20±0.8	89.19±0.9	89.46±0.6	90.64±0.3	88.07±0.6	**92.33±0.3**
	AUC	20	90.59±4.3	98.15±0.1	97.69±0.0	98.21±0.2	97.96±0.4	98.07±0.6	96.99±1.4	97.75±0.3	98.32±0.1	92.23±3.0	**98.90±0.1**
		40	91.42±4.0	97.85±0.1	97.08±0.0	97.93±0.1	97.70±0.3	97.48±0.6	97.12±0.4	97.23±0.2	98.06±0.1	91.76±2.5	**98.55±0.1**
		60	91.73±3.8	98.37±0.1	98.00±0.0	98.49±0.1	97.97±0.2	97.96±0.5	97.76±0.5	97.72±0.4	98.59±0.1	91.63±2.5	**98.89±0.1**
Freebase	Mi-F1	20	54.83±3.0	55.20±0.7	56.23±0.8	57.92±0.5	56.85±0.9	57.24±3.2	58.16±0.9	58.26±0.9	61.72±0.6	64.88±1.8	**65.15±1.3**
		40	57.08±3.2	56.05±2.0	61.01±1.3	62.71±0.7	53.96±1.1	63.74±2.7	57.82±0.8	54.28±1.6	64.03±0.7	62.34±1.0	**67.23±0.8**
		60	55.92±3.2	53.85±0.4	58.74±0.8	58.57±0.5	56.84±0.7	61.06±2.0	57.96±0.7	56.69±1.2	63.61±1.6	59.48±6.2	**67.44±1.2**
	Ma-F1	20	45.14±4.5	53.81±0.6	53.96±0.7	55.78±0.5	52.72±1.0	53.16±2.8	54.90±0.7	55.79±0.9	59.23±0.7	59.04±1.0	**62.06±1.0**
		40	44.88±4.1	52.44±2.3	57.80±1.1	59.28±0.6	48.57±0.5	59.63±2.3	53.40±1.4	49.88±1.9	61.19±0.6	56.40±1.1	**64.64±0.9**
		60	45.16±3.1	50.65±0.4	55.94±0.7	56.50±0.4	52.37±0.8	56.77±1.7	53.81±1.1	52.10±0.7	60.13±1.3	51.73±2.3	**63.84±1.0**
	AUC	20	67.63±5.0	73.03±0.7	71.78±0.7	73.89±0.4	70.84±0.7	73.26±2.1	72.80±0.6	73.19±1.2	76.22±0.8	72.60±0.2	**78.36±1.1**
		40	66.42±4.7	74.05±0.9	75.51±0.8	76.08±0.4	69.48±0.2	77.74±1.2	72.97±1.1	70.77±1.6	78.44±0.5	72.44±1.6	**79.69±0.7**
		60	66.78±3.5	71.75±0.4	74.78±0.4	74.89±0.4	71.01±0.5	75.69±1.5	73.32±0.9	73.17±1.4	78.04±0.4	70.66±1.6	**79.11±1.3**
ACM	Mi-F1	20	49.72±5.5	68.02±1.9	53.13±0.9	57.47±1.5	71.89±1.1	85.11±2.2	79.63±3.5	87.60±0.8	88.13±0.8	82.48±1.9	**90.24±0.5**
		40	60.98±3.5	66.38±1.9	64.43±0.6	62.62±0.9	74.46±0.8	87.21±1.2	80.41±3.0	86.02±0.9	87.45±0.5	82.93±1.1	**90.18±0.6**
		60	60.72±4.3	65.71±2.2	62.72±0.3	65.15±0.9	76.08±0.7	88.10±1.2	80.15±3.2	87.82±0.5	88.71±0.5	80.77±1.1	**91.34±0.4**
	Ma-F1	20	47.13±4.7	62.72±3.1	51.91±0.9	55.13±1.5	72.11±0.9	85.66±2.1	79.27±3.8	87.86±0.2	88.56±0.8	82.26±1.5	**90.66±0.4**
		40	55.96±6.8	61.61±3.2	62.41±0.6	61.21±0.8	72.02±0.4	87.47±1.1	80.23±3.3	86.23±0.8	87.61±0.5	82.00±1.1	**90.15±0.6**
		60	56.59±5.7	61.67±2.9	61.13±0.4	64.35±0.8	74.33±0.6	88.41±1.1	80.03±3.3	87.97±0.4	89.04±0.5	80.29±1.0	**91.59±0.4**
	AUC	20	65.88±3.7	79.50±2.4	71.66±0.7	75.44±1.3	84.36±1.0	93.47±1.5	91.47±2.3	96.72±0.3	96.49±0.3	92.09±0.5	**97.69±0.1**
		40	71.06±5.2	79.14±2.5	80.48±0.4	79.84±0.5	85.01±0.6	94.84±0.9	91.52±2.3	96.35±0.3	96.40±0.4	92.65±0.5	**97.52±0.1**
		60	70.45±6.2	77.90±2.8	79.33±0.4	81.64±0.7	87.64±0.7	94.68±1.4	91.41±1.9	96.79±0.2	96.55±0.3	91.49±0.6	**97.87±0.1**
AMiner	Mi-F1	20	49.68±3.1	65.78±2.9	60.82±0.4	63.64±1.1	61.49±2.5	68.86±4.6	62.39±3.9	63.93±3.3	78.81±1.3	68.21±0.3	**80.30±0.7**
		40	52.10±2.2	71.34±1.8	69.66±0.6	71.57±0.7	68.47±2.2	76.89±1.6	63.87±2.9	63.60±2.5	80.53±0.7	74.23±0.2	**82.35±1.0**
		60	51.36±2.2	67.70±1.9	63.92±0.5	69.76±0.8	65.61±2.2	74.73±1.4	63.10±3.0	62.51±2.6	**82.46±1.4**	72.28±0.2	81.69±0.6
	Ma-F1	20	42.46±2.5	60.22±2.0	54.78±0.5	58.32±1.1	50.06±0.9	56.07±3.2	51.61±3.2	59.50±2.1	71.38±1.1	62.64±0.2	**72.28±0.6**
		40	45.77±1.5	65.66±1.5	64.77±0.5	64.50±0.7	58.97±0.9	63.85±1.5	54.72±2.6	61.92±2.1	73.75±0.5	68.17±0.2	**75.27±1.0**
		60	44.91±2.0	63.74±1.6	60.65±0.3	65.53±0.7	57.34±1.4	62.02±1.2	55.45±2.4	61.15±2.5	**75.80±1.8**	68.21±0.2	74.67±0.6
	AUC	20	70.86±2.5	85.39±1.0	81.22±0.3	83.35±0.5	77.96±1.4	78.92±2.3	75.89±2.2	85.34±0.9	90.82±0.6	86.29±4.1	**93.22±0.6**
		40	74.44±1.3	88.29±1.0	88.82±0.2	88.70±0.4	83.14±1.6	80.72±2.1	77.86±2.1	88.02±1.3	92.11±0.6	89.98±0.0	**94.68±0.4**
		60	74.16±1.3	86.92±0.8	85.57±0.2	87.74±0.5	84.77±0.9	80.39±1.5	77.21±1.4	86.20±1.7	92.40±0.7	88.32±0.0	**94.59±0.3**

Table 1: Node classification performance comparison. The best results are highlighted in bold.

account during training, we abandon the graph structure input for decoding and instead employ the traditional multilayer perceptron (MLP) f_{MLP} as the decoder. Correspondingly, the encoder is able to capture node positional information within a broader context of the graph structure. The process is formulated as follows:

$$P' = f_{\mathrm{MLP}}(H_3), \qquad (10)$$

where P' denotes the predicted node positional features. Then, the loss of positional feature prediction $\mathcal{L}_{\mathrm{PFP}}$ is determined by comparing the target positional features P and the predicted positional features P':

$$\mathcal{L}_{\mathrm{PFP}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} (1 - \frac{P_v \cdot P'_v}{\|P_v\| \times \|P'_v\|})^{\gamma_3}, \qquad (11)$$

where γ_3 is the scaling factor. The final objective function $\mathcal{L}$ is defined as the weighted combination of the metapath-based edge reconstruction loss $\mathcal{L}_{\mathrm{MER}}$, the target attribute restoration loss $\mathcal{L}_{\mathrm{TAR}}$ and the positional feature prediction loss $\mathcal{L}_{\mathrm{PFP}}$:

$$\mathcal{L} = \lambda \mathcal{L}_{\mathrm{MER}} + \mu \mathcal{L}_{\mathrm{TAR}} + \eta \mathcal{L}_{\mathrm{PFP}}, \qquad (12)$$

where λ, μ and η are trade-off weights for balancing $\mathcal{L}_{\mathrm{MER}}$, $\mathcal{L}_{\mathrm{TAR}}$ and $\mathcal{L}_{\mathrm{PFP}}$, respectively.

Experiments

In this section, we conduct extensive experiments to compare the performances of different models. We also show ablation studies, parameter sensitivity, and embedding visualization to demonstrate the superiority of HGMAE.

Experimental Setup

Datasets and Baselines. We employ four real datasets to evaluate the proposed model, including DBLP (Fu et al. 2020), Freebase (Li et al. 2020), ACM (Zhao et al. 2020), and AMiner (Hu, Fang, and Shi 2019). We compare with 10 baselines including unsupervised homogeneous methods GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAE (Kipf and Welling 2016), DGI (Velickovic et al. 2019), GraphMAE (Hou et al. 2022), unsupervised/semi-supervised heterogeneous methods Mp2vec (Dong, Chawla, and Swami 2017), HERec (Shi et al. 2019), HetGNN (Zhang et al. 2019), DMGI (Park et al. 2020), HeCo (Wang et al. 2021a), and HAN (Wang et al. 2019).

Implementation Details. For baselines, we adhere to the settings described in their original papers and follow the setup in HeCo (Wang et al. 2021a). For the proposed HGMAE, we

Datasets	DBLP		Freebase		ACM		AMiner	
Metrics	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
GraphSage	51.50	36.40	9.05	10.49	29.20	27.72	15.74	10.10
GAE	72.59	77.31	19.03	14.10	27.42	24.49	28.58	20.90
Mp2vec	73.55	77.70	16.47	17.32	48.43	34.65	30.80	25.26
HERec	70.21	73.99	19.76	19.36	47.54	35.67	27.82	20.16
HetGNN	69.79	75.34	12.25	15.01	41.53	34.81	21.46	26.60
DGI	59.23	61.85	18.34	11.29	51.73	41.16	22.06	15.93
DMGI	70.06	75.46	16.98	16.91	51.66	46.64	19.24	20.09
HeCo	74.51	80.17	20.38	20.98	56.87	56.94	32.26	28.64
GraphMAE	65.86	69.75	19.43	20.05	47.03	46.48	17.98	21.52
HGMAE	**76.92**	**82.34**	**22.05**	**22.84**	**66.68**	**71.51**	**41.10**	**38.27**

Table 2: Node clustering performance comparison.

Datasets	Metric	w/o MER	w/o TAR	w/o PFP	HGMAE
DBLP	Mi-F1	91.99±0.5	92.06±0.5	92.21±0.6	**92.73±0.4**
	Ma-F1	91.39±0.5	91.53±0.6	91.68±0.5	**92.24±0.4**
	AUC	98.60±0.1	98.66±0.1	98.66±0.1	**98.78±0.1**
Freebase	Mi-F1	66.01±0.8	64.76±1.6	66.20±1.6	**66.61±1.1**
	Ma-F1	63.16±0.7	61.12±1.1	62.10±1.3	**63.51±1.0**
	AUC	78.04±0.6	78.71±1.0	77.92±0.9	**79.05±1.0**
ACM	Mi-F1	76.85±0.2	88.54±0.4	89.81±0.5	**90.59±0.5**
	Ma-F1	71.93±0.4	88.82±0.4	89.94±0.4	**90.80±0.5**
	AUC	84.84±1.5	96.47±0.1	97.22±0.1	**97.69±0.1**
AMiner	Mi-F1	72.12±1.6	81.22±0.7	80.88±1.0	**81.45±0.8**
	Ma-F1	65.01±1.1	73.66±0.7	73.66±0.9	**74.07±0.7**
	AUC	85.59±1.6	93.70±0.3	93.65±0.3	**94.16±0.4**

Table 3: Results of different model variants.

use HAN (Wang et al. 2019) as the default encoder and decoder. We search the learning rate from 1e-4 to 5e-3, tune the patience for early stopping from 5 to 20, and test the leave unchanged and replaced rates from 0 to 0.5 with step 0.1. For dynamic mask rate, we set MIN_{p_a} to 0.5, MAX_{p_a} to 0.8 and Δ equals 0.005. For all methods, we report the mean and standard deviation of 10 runs with different random seeds.

Node Classification

We first evaluate different models for the node classification task and report their performances in Table 1. Specifically, we use 20, 40, 60 labeled nodes per class as training set and 1000 nodes each for validation and test sets. We use Micro-F1, Macro-F1 and AUC as evaluation metrics. According to the table, we find that our model HGMAE outperforms all the baselines across various datasets except for few cases. In AMiner, HGMAE has a small difference compared to the best baseline HeCo on few F1 values, but still remains the second best among all models. Considering AMiner has fewer edges and metapaths compared to other datasets, it is understandable that HGMAE could be biased with simple generative losses, while HeCo performs better by employing the complex contrastive loss on intricate graph views. However, HGMAE is more stable and performs better across various datasets and experimental settings. In addition, GraphMAE performs poorly on all datasets. This further demonstrates the significance of addressing the identified challenges and proves the effectiveness of our model.

Node Clustering

We further evaluate different models for node clustering task and report their performances in Table 2. In particular, we apply K-means as the learning algorithm and utilize normalized mutual information (NMI) and adjusted rand index (ARI) as the evaluation metrics. From the table, we can find that HGMAE consistently achieves the best results on all datasets, which validates the effectiveness of HGMAE from a different perspective. Concretely, HGMAE maintains superior learning capability and outperforms existing baselines by a large margin in most cases. For example, compared with the best baseline HeCo, HGMAE improves NMI by +17% and ARI by +25% on ACM dataset, as well as 27% and 33% on AMiner dataset. This further demonstrates the superiority of our model.
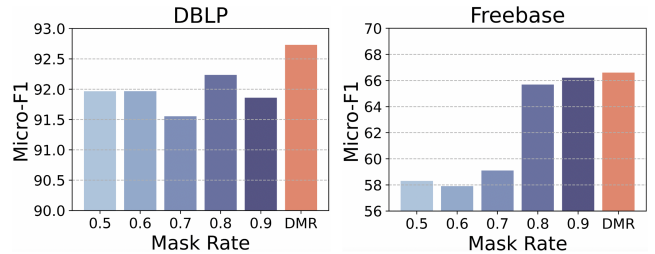

Figure 3: Performance of HGMAE *w.r.t.* different mask rates.

Ablation Study

Since HGMAE contains various training strategies (i.e., metapath-based edge reconstruction (MER), target attribute restoration (TAR), and positional feature prediction (PFP)), we conduct ablation studies on the node classification task to analyze the contributions of different strategies by removing each of them independently (see Table 3). Specifically, removing MER significantly affects the performance, showing that MER has large contribution to HGMAE. In addition, the decreasing performances of removing TAR and PFP demonstrate the effectiveness of TAR and PFP in enhancing the model, respectively. Finally, HGMAE achieves the best results in all cases, indicating the strong capability of different strategies in our model.

Impact of Dynamic Mask Rate

A salient property of HGMAE is the incorporation of the dynamic mask rate (DMR), which masks a variable percentage of attributes to encourage the stable learning across various masking settings. To better understand the effectiveness of DMR, we provide a detailed analysis by comparing the performance of DMR with fixed mask rates, as shown in Figure 3. We find that DMR can always achieve the best result, demonstrating its superiority compared to other mask rates. In general, the performance increases when increasing the mask rate, which indicates that a suitable mask rate will lead to better model performances. However, a mask rate that is suitable for one dataset might not be the optimal choice for other datasets. For example, without DMR, fixed mask rate 0.9 is the optimal rate for Freebase, but it performs poorly on DBLP. Consequently, it is essential that the model
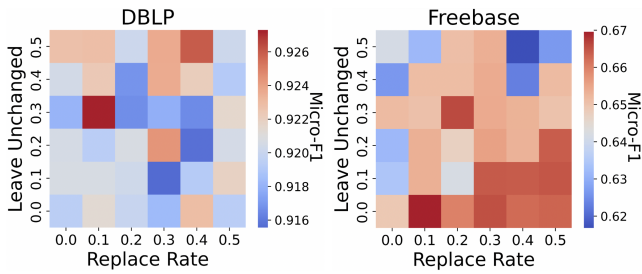
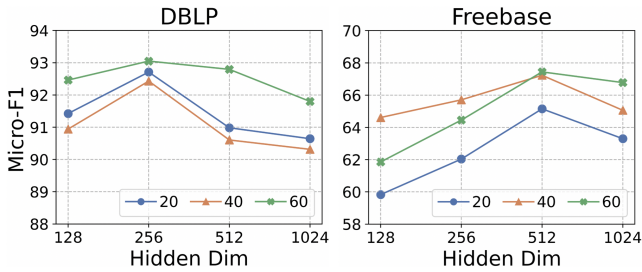Figure 4: Performance of HGMAE *w.r.t.* leave unchanged and replace rates.

Figure 5: Performance of HGMAE *w.r.t.* hidden dimensions.

Figure 6: Embedding visualization of DBLP dataset. Different colors indicate different node category labels.

learns adaptively across various mask rates. In addition, DMR enables the model to learn effectively without the need for manual mask rate determination.

Impact of Leaving Unchanged and Replacing

We conduct experiments to show the effectiveness of leaving unchanged and replacing in Figure 4. We surprisedly discover that the use of these two techniques always improves the model's ability to learn better node embeddings, which is opposite to the statement in GraphMAE (Hou et al. 2022). Specifically, HGMAE achieves the optimal performance in DBLP when leave unchanged and replace rates are set to 0.3 and 0.1, and performs the best in Freebase when they are set to 0.0 and 0.1, respectively. In general, we observe that for DBLP, a high leave unchanged rate and a low replace rate are usually beneficial, whereas for Freebase, the opposite is true. This demonstrates the effectiveness of leaving unchanged and replacing, and indicates that different rates may have different impacts across datasets. Correspondingly, determining appropriate rates can result in a better model performance.

Parameter Sensitivity

We further perform parameters sensitivity analysis to show the impact of hidden dimensions in Figure 5. In particular, we search the number of hidden dimensions from {128, 256, 512, 1024}. By analyzing the figure, we find that the optimal hidden dimension can be different across datasets. In addition, increasing the hidden dimension generally enhances the performance. We ascribe this improvement to the comprehensive modeling of the data itself, while using a small hidden dimension could prevent the model from fully capturing the knowledge. However, further increasing the
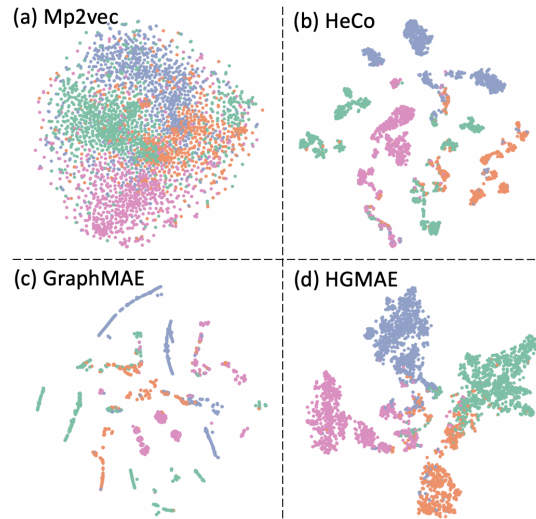
hidden dimension (e.g., 1024) degrades the performance. This is because applying a too wide hidden layer could decentralize the model's focus on meaningful information. Therefore, using the standard hidden dimensions (e.g., 256 or 512) is sufficient for the proposed model to capture complex information and achieve superior performance.

Embedding Visualization

For a more intuitive understanding and comparison, we visualize the learned node embeddings of different models using t-SNE. As shown in Figure 6, Mp2vec does not perform well. Nodes from different categories are mixed together. HeCo can successfully distinguish different categories, but fail to cluster nodes that share the same category. GraphMAE can separate each category well, but the nodes tend to stay close to each other and form small chunks, while the difference between chunks is large, even when chunks belong to the same category. However, our model HGMAE can clearly identify each category and maintain a clear boundary between them. In addition, nodes with the same category form an exclusive dense cluster, instead of splitting into numerous chunks as shown in other methods. This again demonstrates the effectiveness of HGMAE and the capability of learning discriminative node embeddings.

Conclusion

In this paper, we propose and formalize the problem of generative self-supervised learning on heterogeneous graphs. To solve this problem, we propose HGMAE, a novel heterogeneous graph masked autoencoder model. HGMAE jointly considers the complex graph structure, various node attributes, and different node positions via two innovative masking techniques and three unique training strategies. Extensive experiments on multiple datasets and several tasks demonstrate the superiority of HGMAE compared to state-of-the-art methods.

References

Bai, Y.; Ying, Z.; Ren, H.; and Leskovec, J. 2021. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. *NeurIPS*.

Cao, Y.; Peng, H.; Wu, J.; Dou, Y.; Li, J.; and Yu, P. S. 2021. Knowledge-preserving incremental social event detection via heterogeneous gnns. In *WWW*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Ding, K.; Xu, Z.; Tong, H.; and Liu, H. 2022. Data augmentation for deep graph learning: A survey. *arXiv preprint arXiv:2202.08235*.

Dong, Y.; Chawla, N. V.; and Swami, A. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *KDD*.

Fan, Y.; Ju, M.; Zhang, C.; and Ye, Y. 2022. Heterogeneous temporal graph neural network. In *SDM*.

Fu, X.; Zhang, J.; Meng, Z.; and King, I. 2020. MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. In *WWW*.

Garcia Duran, A.; and Niepert, M. 2017. Learning graph representations with embedding propagation. *NeurIPS*.

Gupta, A.; Tian, S.; Zhang, Y.; Wu, J.; Martin-Martin, R.; and Fei-Fei, L. 2022. MaskViT: Masked Visual Pre-Training for Video Prediction. *arXiv preprint arXiv:2206.11894*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollar, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *CVPR*.

Hou, Z.; Liu, X.; Dong, Y.; Wang, C.; Tang, J.; et al. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. *KDD*.

Hu, B.; Fang, Y.; and Shi, C. 2019. Adversarial Learning on Heterogeneous Information Networks. In *KDD*.

Hu, Z.; Dong, Y.; Wang, K.; and Sun, Y. 2020. Heterogeneous Graph Transformer. In *WWW*.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Li, X.; Ding, D.; Kao, B.; Sun, Y.; and Mamoulis, N. 2020. Leveraging Meta-path Contexts for Classification in Heterogeneous Information Networks. *arXiv preprint arXiv:2012.10024*.

Liu, X.; Zhang, F.; Hou, Z.; Wang, Z.; Mian, L.; Zhang, J.; and Tang, J. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218*.

Liu, Y.; Jin, M.; Pan, S.; Zhou, C.; Zheng, Y.; Xia, F.; and Yu, P. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Park, C.; Kim, D.; Han, J.; and Yu, H. 2020. Unsupervised Attributed Multiplex Network Embedding. In *AAAI*.

Park, J.; Lee, M.; Chang, H. J.; Lee, K.; and Choi, J. Y. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *ICCV*.

Qiu, J.; Chen, Q.; Dong, Y.; Zhang, J.; Yang, H.; Ding, M.; Wang, K.; and Tang, J. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*.

Shi, C.; Hu, B.; Zhao, W. X.; and Yu, P. S. 2019. Heterogeneous Information Network Embedding for Recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Thakoor, S.; Tallec, C.; Azar, M. G.; Munos, R.; Velickovic, P.; and Valko, M. 2022. Large-Scale Representation Learning on Graphs via Bootstrapping. In *ICLR*.

Tian, Y.; Zhang, C.; Guo, Z.; Huang, C.; Metoyer, R.; and Chawla, N. V. 2022a. RecipeRec: A Heterogeneous Graph Learning Model for Recipe Recommendation. In *IJCAI*.

Tian, Y.; Zhang, C.; Guo, Z.; Ma, Y.; Metoyer, R.; and Chawla, N. V. 2022b. Recipe2Vec: Multi-modal Recipe Representation Learning with Graph Neural Networks. In *IJCAI*.

Tian, Y.; Zhang, C.; Metoyer, R.; and Chawla, N. V. 2021. Recipe representation learning with networks. In *CIKM*.

Velickovic, P.; Fedus, W.; Hamilton, W. L.; Lio, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. In *ICLR*.

Wang, B.; Li, A.; Li, H.; and Chen, Y. 2020. GraphFL: A Federated Learning Framework for Semi-Supervised Node Classification on Graphs. *arXiv preprint arXiv:2012.04187*.

Wang, X.; Bo, D.; Shi, C.; Fan, S.; Ye, Y.; and Philip, S. Y. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous Graph Attention Network. In *WWW*.

Wang, X.; Liu, N.; Han, H.; and Shi, C. 2021a. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *KDD*.

Wang, Z.; Wen, R.; Chen, X.; Cao, S.; Huang, S.-L.; Qian, B.; and Zheng, Y. 2021b. Online Disease Diagnosis with Inductive Heterogeneous Graph Convolutional Networks. In *WWW*.

Wu, L.; Lin, H.; Tan, C.; Gao, Z.; and Li, S. Z. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*.

Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; and Han, J. 2020. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *IEEE Transactions on Knowledge and Data Engineering*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *NeurIPS*.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph contrastive learning automated. In *ICML*.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *NeurIPS*.

Yu, L.; Pei, S.; Ding, L.; Zhou, J.; Li, L.; Zhang, C.; and Zhang, X. 2022. Sail: Self-augmented graph contrastive learning. In *AAAI*.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous Graph Neural Network. In *KDD*.

Zhao, J.; Wang, X.; Shi, C.; Liu, Z.; and Ye, Y. 2020. Network Schema Preserving Heterogeneous Information Network Embedding. In *IJCAI*.

Zhao, J.; Wen, Q.; Ju, M.; Zhang, C.; and Ye, Y. 2023. Self-Supervised Graph Structure Refinement for Graph Neural Networks. In *WSDM*.

Zhao, J.; Wen, Q.; Sun, S.; Ye, Y.; and Zhang, C. 2021. Multi-view Self-supervised Heterogeneous Graph Embedding. In *ECML PKDD*.