Teacher-guided Edge Discriminator for Personalized Graph Masked Autoencoder

Qiqi Zhang, Chao Li, Zhongying Zhao*

College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
zqqstu0818@163.com, lichao@sdust.edu.cn, zzysuin@163.com or zyzhao@sdust.edu.cn

Abstract

Graph Masked AutoEncoder (GMAE) has recently attracted vast interest in handling graph-related tasks by adopting the 'masking-reconstruction' learning paradigm. Most existing GMAE-based methods adhere to the homophily assumption, i.e., connected nodes share the same attributes or labels. However, this assumption is not always right because most graphs from real-world applications are mixed by both homophilic and heterophilic edges. Therefore, it is necessary to distinguish them to improve the representative ability of GMAE. In this paper, we propose a teacher-guided edge discriminator for the personalized graph masked autoencoder (TEDMAE). Specifically, we design a teacher-guided edge discriminator that distinguishes homophilic and heterophilic edges by leveraging the embeddings from teacher models with structure and attribute knowledge. Then, we present a personalized graph masked autoencoder that individually tailors the masking, encoding, and reconstruction processes for each graph. Finally, we optimize the model by minimizing two types of loss functions, i.e., the scaled cosine error (SCE) loss and the InfoNCE loss. Experimental results on 10 datasets demonstrate the superior performance of TEDMAE on the tasks of node classification and node clustering.

Code —
https://github.com/ZZY-GraphMiningLab/TEDMAE.

Introduction

Self-Supervised Learning (SSL) on graphs empowers machine learning models to extract informative node representations from unlabeled data (Ju et al. 2024; Zhang, Wang, and Wang 2022; Liu et al. 2023a). SSL methods have garnered significant attention and achieved remarkable success across various domains, such as drug discovery, traffic prediction, and social network modeling (Thakoor et al. 2021; Zhao et al. 2024; Liu et al. 2023b; Shi et al. 2023).

Existing SSL methods are mainly classified into two categories: contrastive and generative methods (Liu et al. 2021; Hou et al. 2022). The former utilizes the 'augmenting-contrasting' learning paradigm to extract the underlying semantics and mine the rich information within graph data.

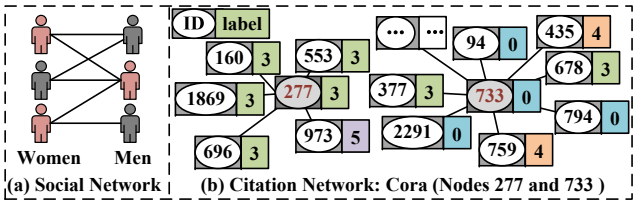

Figure 1: Interaction between nodes in social network and academic citation network, such as Cora.

However, these methods require carefully designed augmentation, high-quality negative samples, and an efficient optimization algorithm to achieve superior performance (Hou et al. 2022, 2023). Generative methods within SSL are proposed to address the problem of complex algorithm design and non-general experimental results. These methods learn effective node representations by reconstructing the graph structure or feature information of nodes. Thus, they are widely applied in various domains, such as BERT (Devlin et al. 2019) and ChatGPT (Radford et al. 2019) in NLP, and MAE (He et al. 2022) in CV. Nevertheless, they are still facing the following challenges when extending to graph representation learning:

(1) How to distinguish homophilic and heterophilic edges? In real-world networks, homophilic nodes (i.e., connected by inner-class edges) and heterophilic nodes (i.e., connected by inter-class edges) are often intertwined (Zhu et al. 2020; Sun et al. 2022). Taking a social network as an example, interactions between individuals of different genders are common, as illustrated in Figure 1(a). In the academic citation network Cora (Figure 1(b)), node 277 is connected to nodes with the same label, such as nodes 160 and 553, as well as to a node with a different label, such as node 973. Similarly, node 733 is connected to both nodes representing similar and different types of papers. Most existing GMAE methods primarily study the selection of masked targets and reconstruction objectives (Hou et al. 2022; Shi et al. 2023; Zhao et al. 2024). For example, MaskGAE (Li et al. 2023a) attempts to reconstruct masked edges and node degrees simultaneously. GiGaMAE (Shi et al. 2023) enhances node representations by reconstructing target embeddings containing diverse information. However, these methods fail

*Corresponding author.

to clearly differentiate homophilic and heterophilic edges, which diminishes the representation ability of the GMAE.

(2) How to design a personalized graph masked autoencoder? The main idea of GMAE methods is to mask part of the features and structure of the input original nodes (Hou et al. 2022, 2023; Tan et al. 2023). Meanwhile, they utilize the learned representations and the relationships with neighboring nodes for reconstruction (Li et al. 2023b; Tu et al. 2024; Tian et al. 2024). For example, GraphMAE (Hou et al. 2022) focuses on feature reconstruction for generative self-supervised graph pre-training. S2GAE (Tan et al. 2023) reconstructs masked edges with a specific masking strategy and a sophisticated decoder network. The above methods mix homophilic and heterophilic nodes together when performing random masking. They fail to discriminatively learn the representations of these two types of nodes during the masking and reconstruction process. Overall, it leads to the following problems. (i) The non-discriminatory masking strategy impairs reconstruction performance. (ii) A single learning strategy cannot effectively capture the distinctive attributes of the two types of nodes. (iii) The unified masking method results in an imbalance in resource allocation due to the difference in the number of homophilic and heterophilic edges. Therefore, it is crucial to design a personalized mask autoencoder for each graph.

In this paper, we propose a generative graph self-supervised learning method, called **TEDMAE** (Teacher-guided Edge Discriminator for personalized graph Masked AutoEncoder). Specifically, we design a teacher-guided edge discriminator that differentiates homophilic and heterophilic edges by leveraging the embeddings from teacher models with structure and attribute knowledge. Then, we present a personalized graph masked autoencoder that individually tailors the masking, encoding, and reconstruction processes for each graph. Moreover, we calculate the scaled cosine error (SCE) loss and InfoNCE loss by reconstructing node features and aligning the teacher embeddings to optimize the model. Finally, we conduct comprehensive experiments to evaluate the proposed framework on the tasks of node classification and node clustering. The contributions of this paper are as follows.

- We present a generative SSL method, TEDMAE, to learn generalizable node representations by reconstructing node features and embeddings.
- We design a teacher-guided edge discriminator with structure and attribute knowledge to distinguish homophilic and heterophilic edges.
- We propose a personalized graph masked autoencoder that is individually tailored for masking, encoding, and reconstructing each graph.
- We conduct extensive experiments on 10 benchmark datasets to evaluate the performance and generalization ability of TEDMAE. The experimental results demonstrate the superiority of TEDMAE.

Related Work

Graph autoencoders. Graph autoencoders are designed to reconstruct certain inputs given the contexts (Hou et al. 2022). GAE and VGAE (Kipf and Welling 2016) employ GNNs as encoders and utilize dot products for link prediction decoding. For example, VGAE targets to predict missing edges. EP (Garcia Duran and Niepert 2017) is designed to recover vertex features utilizing mean squared error without input corruption. GAEs are widely used for fine-grained graph analysis tasks and have achieved superior performance, such as link prediction (Tan et al. 2023; Kipf and Welling 2016). However, their performance on coarse-grained tasks, such as node classification, is usually less satisfactory than that of contrastive methods (Tan et al. 2023).

Masked autoencoders. Masked autoencoders, employing the 'masking-reconstruction' paradigm, have proven to be highly successful in graph representation learning (Wang et al. 2024a; Zheng and Jia 2024; Liu et al. 2024; Wang et al. 2024b). For example, GraphMAE (Hou et al. 2022) addresses common issues faced by GAEs by refining the reconstruction objective, learning process, loss function, and model architecture. To mitigate the impact of input feature discriminability, GraphMAE2 (Hou et al. 2023) introduces regularization on feature reconstruction for graph SSL. To extend the utility of GAE beyond link prediction, S2GAE (Tan et al. 2023) innovates with direction-aware graph masking strategies and a tailored cross-correlation decoder. Meanwhile, MaskGAE (Li et al. 2023a) attempts to reconstruct masked edges and node degrees. Moreover, GiGaMAE (Shi et al. 2023) enhances node representation by reconstructing target embeddings with diverse information. To learn representations with topological information, Bandana (Zhao et al. 2024) explores non-discrete masking using bandwidth masking and a Boltzmann-Gibbs scheme.

Methodology

Preliminaries

Notations. Let $\mathcal{G} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ be a graph, where $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{E}}$ represent the sets of nodes and edges, respectively. $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ represents the adjacency matrix of $\mathcal{G}$, where N is the number of nodes, and $\boldsymbol{A}_{ij} = 1$ indicates that there exists an edge between nodes i and j, otherwise, $\boldsymbol{A}_{ij} = 0$. Node i is associated with a feature vector $\boldsymbol{x_i}$. $\mathbf{X} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_i}, \ldots, \boldsymbol{x_N}\} \in \mathbb{R}^{N \times D}$ that represents the feature matrix of $\mathcal{G}$, where D represents the feature dimension of a node.

Problem. Learning Generalizable Node Representations. Given an unlabeled graph $\mathcal{G}$, we take the graph structure and node feature as input. Our goal is to train a graph encoder, denoted as $f_e(\boldsymbol{X}, \boldsymbol{A})$ that learns generalizable node embeddings. These embeddings can be utilized directly for downstream tasks, including node classification, node clustering, and link prediction.

The Proposed Method

The framework of the proposed TEDMAE is illustrated in Figure 2. It consists of two modules: the teacher-guided edge discriminator module and the personalized graph masked autoencoder module. The details are given below.

Teacher-guided Edge Discriminator Module. To distinguish the homophilic and heterophilic edges, we design a

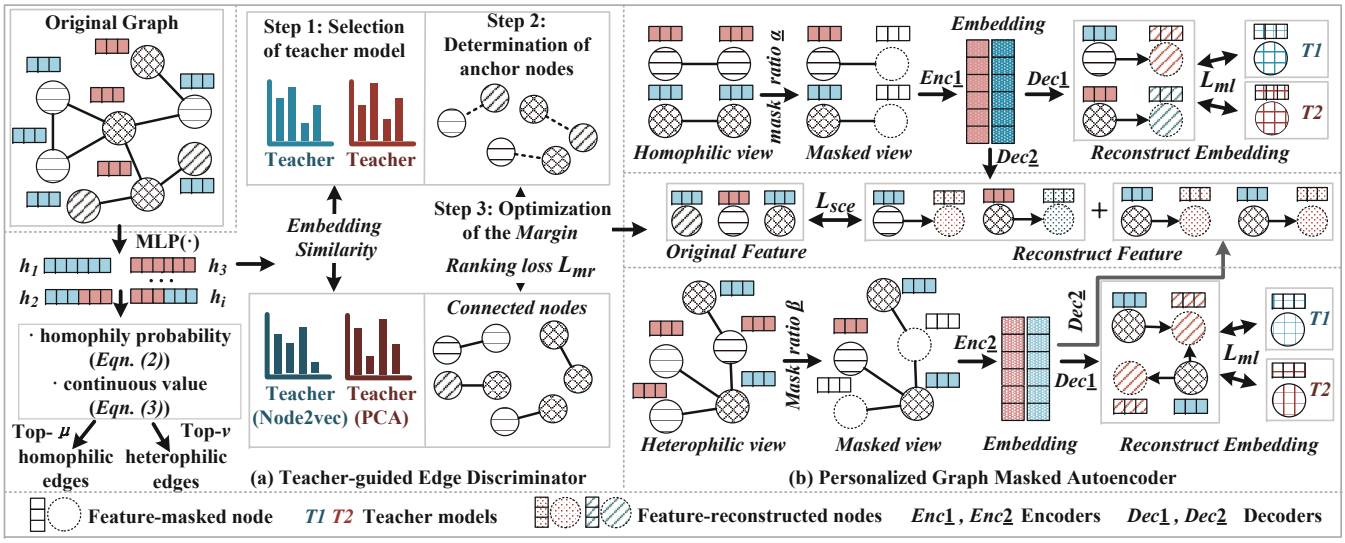

Figure 2: The framework of the proposed TEDMAE.

teacher-guided edge discriminator (as shown in Figure 2(a)) that predicts the homophily probability of each edge.

For any two connected nodes, the consistency of the label is influenced by the key factors: node attributes. Therefore, we put the node attributes into a multi-layer perception (MLP) to derive latent embeddings.

$$\boldsymbol{h}_i = MLP_1(\boldsymbol{x}_i), \boldsymbol{h}_j = MLP_1(\boldsymbol{x}_j). \tag{1}$$

To ensure that the estimation results are not affected by the direction of edges, we employ a two-layer MLP to encode the various orders of embedding. Then, the homophily probability $\sigma_{i,j}$ of each edge e_{ij} is estimated by Eqn. (2).

$$\sigma_{i,j} = \left(MLP_2([\boldsymbol{h}_i \| \boldsymbol{h}_j]) + MLP_2([\boldsymbol{h}_j \| \boldsymbol{h}_i])\right)/2, \tag{2}$$

where $[\cdot\|\cdot]$ represents the concatenation operation, and σ is the estimated discrete probability value for each edge.

We draw a sample from the Bernoulli distribution with probability $\sigma_{i,j}$, resulting in a binary indicator $\partial_{i,j} \sim B(\sigma_{i,j})$. In this context, $\partial_{i,j} = 1$ indicates homophily, while $\partial_{i,j} = 0$ indicates the presence of heterophily. Subsequently, concrete-sampling reparametrization is employed to convert this discrete binary indicator into a continuous approximation. The continuous weight $\check{\partial}$ is evaluated by:

$$\check{\partial}_{i,j} = \frac{\exp\left(\left(\log(\sigma_{i,j}) + \delta_i\right)/\tau\right)}{\sum_j \exp\left(\left(\log(\sigma_{i,j}) + \delta_j\right)/\tau\right)}, \quad \delta_i = -\log\left(-\log\left(\mu_i\right)\right), \tag{3}$$

where $\mu_i \sim U(0,1)$ is random variable, U represents the uniform distribution, and τ is the temperature hyperparameter guided by (Liu et al. 2023b).

All edges are sorted based on the parameter $\check{\partial}$, and then the top $\mu\%$ and $\nu\%$ of the edges are selected from homophilic and heterophilic indexes, respectively. The training process of the edge discriminator consists of the following three steps:

Step 1: Selection of teacher model. We introduce knowledge from structure and feature perspectives to guide the edge discriminator. The design of the teacher model should consider the following factors: 1) no need for labels, 2) lower time and space complexity, and 3) high performance for graph representation learning. Specifically, we designate the **Node2vec** (Grover and Leskovec 2016) model as *Teacher 1* to capture the structure information. It excels at capturing the topological relationships of nodes and effectively learning neighborhood information. Meanwhile, we employ the **PCA** model as *Teacher 2* to extract feature information. It is adept at extracting the principal components of the data with dimensionality reduction, thereby reducing noises while retaining essential information (Candès et al. 2011).

Step 2: Determination of anchor nodes. We select pairs of nodes at random to serve as 'anchor nodes'. Note that, the 'anchor nodes' are used to differentiate whether the connected nodes are similar or not. We aim to ensure that node pairs linked by homophilic edges are more similar than anchor node pairs. In contrast, node pairs linked by heterophilic edges are more dissimilar than anchor node pairs (Liu et al. 2023b).

To evaluate the embedding similarity of node pairs, we employ the cosine similarity function as the criterion. The similarity values for the two types of node pairs, connected node pairs (CNP) and randomly selected node pairs (RNP), are calculated by Eqn. (4).

$$sim(i,j) = \frac{\boldsymbol{h}_i \cdot \boldsymbol{h}_j}{\|\boldsymbol{h}_i\| \, \|\boldsymbol{h}_j\|}, \tag{4}$$

where $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ denote the embeddings of nodes i and j, respectively. They are obtained from the teacher models Node2vec and PCA, denoted as $\boldsymbol{h}^{T1}$ and $\boldsymbol{h}^{T2}$, respectively.

Step 3: Optimization of the loss. The margin ranking loss is utilized to optimize the edge discriminator. We separately calculate the loss in the predicted homophilic and

heterophilic edge indexes, as depicted in Eqn. (5). $\boldsymbol{h}_i, \boldsymbol{h}_j$ represent the embeddings of connected node pairs, $\boldsymbol{h}_s, \boldsymbol{h}_t$ represent the embeddings of anchor node pairs. ρ_{ho}, ρ_{he} are the margin parameters for the pivot-anchored ranking loss (Liu et al. 2023b).

$$\begin{aligned} l^{homo}_{e_{(i,j)}} &= \max\left(sim(\boldsymbol{h}_s, \boldsymbol{h}_t) - sim(\boldsymbol{h}_i, \boldsymbol{h}_j) + \rho_{\text{ho}}, 0\right), \\ L^{homo} &= l^{homo}_{e_{(i,j)} \in E} \cdot \max\left(1/\check{\partial}), 0\right), \\ l^{heter}_{e_{(i,j)}} &= \max\left(sim(\boldsymbol{h}_i, \boldsymbol{h}_j) - sim(\boldsymbol{h}_s, \boldsymbol{h}_t) + \rho_{\text{he}}, 0\right), \\ L^{heter} &= l^{heter}_{e_{(i,j)} \in E} \cdot \max\left(1/(1-\check{\partial})), 0\right), \end{aligned} \tag{5}$$

Guided by the teacher model (embedding from structure and attribute), the edge discriminator is optimized via Eqn. (6) to obtain the homophily matrix $\boldsymbol{A}^{homo}$ and the heterophily matrix $\boldsymbol{A}^{heter}$.

$$L_{mr} = L^{homo}_{T1} + L^{heter}_{T1} + L^{homo}_{T2} + L^{heter}_{T2}, \tag{6}$$

where L^{homo}_{T1}, L^{heter}_{T1} represent the homophily and heterophily losses calculated based on Node2Vec, and L^{homo}_{T2}, L^{heter}_{T2} represent the homophily and heterophily losses calculated based on PCA.

Personalized Graph Masked Autoencoder Module. Unlike traditional GMAE methods that encode the masked graph, TEDMAE designs a personalized GMAE for the generated homophilic and heterophilic views (Figure 2(b)).

We introduce a masking strategy for graph data inspired by the successful explorations of BERT (Devlin et al. 2019) and MAE (He et al. 2022). It ensures the unbiasedness of node selection by adopting the same masking probability for each node. Specifically, it randomly samples subsets of nodes in the homophilic and heterophilic views based on the ratio α and β to mask, denoted as $\boldsymbol{V}^{homo}_{mask}$ and $\boldsymbol{V}^{heter}_{mask}$, respectively. Taking the homophilic graph as an example, the masked feature $\widetilde{\boldsymbol{x}}_i$ of node i in $\boldsymbol{V}^{homo}_{mask}$ is defined as follows:

$$\widetilde{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_{[M]}, & \text{if } i \in \boldsymbol{V}^{homo}_{mask} \\ \boldsymbol{x}_i. & \text{if } i \notin \boldsymbol{V}^{homo}_{mask} \end{cases} \tag{7}$$

The GNNs, serving as the core backbone of the encoders, map node features into a unified shallow space. The process of learning node representations with message aggregation can be expressed as follows:

$$\begin{aligned} \boldsymbol{H}^{homo} &= f_{homo}(\boldsymbol{A}^{homo}, \widetilde{\boldsymbol{X}}^{homo}), \\ \boldsymbol{H}^{heter} &= f_{heter}(\boldsymbol{A}^{heter}, \widetilde{\boldsymbol{X}}^{heter}), \end{aligned} \tag{8}$$

where $f_{homo}(\cdot)$, $f_{heter}(\cdot)$ represent the GNN encoders used for the homophily and heterophily graphs, respectively, and $\boldsymbol{A}^{homo}$, $\boldsymbol{A}^{heter}$ are the adjacency matrices generated during the pre-trained edge discriminator. It is worth noting that $f_{homo}(\cdot)$ and $f_{heter}(\cdot)$ are two independent encoders, and their parameters are not shared between them.

There are two reconstruction objectives: **embeddings** and **features**. The former focuses on aligning different representations. The latter aims to capture informative features.

Reconstruct objective 1: node embeddings. The decoder $f_{d1}(\cdot)$ is introduced to separately reconstruct embeddings from two perspectives. To improve computational efficiency and enhance the model flexibility, the MLP is selected as the decoder. The process can be represented as:

$$\begin{aligned} \boldsymbol{RH}^{homo} &= f_{d1}(\boldsymbol{A}^{homo}, \boldsymbol{H}^{homo}), \\ \boldsymbol{RH}^{heter} &= f_{d1}(\boldsymbol{A}^{heter}, \boldsymbol{H}^{heter}), \end{aligned} \tag{9}$$

where $\boldsymbol{RH}^{homo}$ and $\boldsymbol{RH}^{heter}$ represent the embeddings of masked nodes.

To train the encoder and extract as much useful information as possible from the teacher models, we maximize the MI between the predicted embeddings $\boldsymbol{RH}^{homo}$/$\boldsymbol{RH}^{heter}$ and the teacher embeddings $\boldsymbol{T}^{homo}$/$\boldsymbol{T}^{heter}$. The objective is optimized by maximizing the InfoNCE loss:

$$\begin{aligned} l(\boldsymbol{RH}(v), \boldsymbol{T}(v)) &= f(sim(\boldsymbol{RH}(v), \boldsymbol{T}(v))) \\ &= \exp\left(\frac{\boldsymbol{RH}(v) \cdot \boldsymbol{T}(v)}{\|\boldsymbol{RH}(v)\| \, \|\boldsymbol{T}(v)\|} \cdot \frac{1}{\tau}\right). \end{aligned} \tag{10}$$

where τ represents a temperature coefficient.

The positive pairs are regarded as different latent representations of a single node within the graph, while the negative samples consist of representations from all other nodes within the same graph. Overall, the losses in homophilic and heterophilic graphs are defined as:

$$\begin{aligned} L_{ml} &= L^{homo}(\boldsymbol{RH}, \boldsymbol{T}) + L^{heter}(\boldsymbol{RH}, \boldsymbol{T}) \\ &= \frac{1}{N} \sum_{v=1}^{N} [l^{homo}(\boldsymbol{RH}(v), \boldsymbol{T}(v)) + l^{heter}(\boldsymbol{RH}(v), \boldsymbol{T}(v))]. \end{aligned} \tag{11}$$

Reconstruct objective 2: node features. The decoder $f_{d2}(\cdot)$ is designed to separately map the $\boldsymbol{H}^{homo}$ and $\boldsymbol{H}^{heter}$ to the feature space. The process can be represented as:

$$\begin{aligned} \boldsymbol{RX}^{homo} &= f_{d2}(\boldsymbol{A}^{homo}, \boldsymbol{H}^{homo}), \\ \boldsymbol{RX}^{heter} &= f_{d2}(\boldsymbol{A}^{heter}, \boldsymbol{H}^{heter}), \end{aligned} \tag{12}$$

where $\boldsymbol{RX}^{homo}$ and $\boldsymbol{RX}^{homo}$ represent the reconstructed node features.

The objective is optimized by minimizing the SCE loss:

$$\begin{aligned} L_{sce} &= L^{homo}_{sce} + L^{heter}_{sce} \\ &= \frac{1}{\left|\boldsymbol{V}^{homo}_{mask}\right|} \sum_{\boldsymbol{v}_i \in \boldsymbol{V}^{homo}_{mask}} \left(1 - \frac{\boldsymbol{x}_i^T \boldsymbol{RX}_i^{homo}}{\|\boldsymbol{x}_i\| \cdot \left\|\boldsymbol{RX}_i^{homo}\right\|}\right)^{\gamma} + \\ &\quad \frac{1}{\left|\boldsymbol{V}^{heter}_{mask}\right|} \sum_{\boldsymbol{v}_i \in \boldsymbol{V}^{heter}_{mask}} \left(1 - \frac{\boldsymbol{x}_i^T \boldsymbol{RX}_i^{heter}}{\|\boldsymbol{x}_i\| \cdot \left\|\boldsymbol{RX}_i^{heter}\right\|}\right)^{\gamma}, \gamma \geq 1 \end{aligned} \tag{13}$$

where γ is used to optimize the sample weights.

The loss L is calculated by Eqn. (14). As stated above, the loss consists of two components: L_{ml} (as per Eqn. (11)) and L_{sce} (as per Eqn. (13)).

$$L = L_{ml} + L_{sce}. \tag{14}$$

Algorithm 1: Algorithm of TEDMAE

Input: adjacency matrix $\boldsymbol{A}$, homophily / heterophily metrics $\boldsymbol{A}^{homo}$/$\boldsymbol{A}^{heter}$, node attributes $\boldsymbol{X}$, masked feature metrics $\boldsymbol{X}^{homo}$/$\boldsymbol{X}^{heter}$, latent representations $\boldsymbol{H}^{homo}$/$\boldsymbol{H}^{heter}$, teacher model T1/T2, parameters α, β, γ, training rounds t;

Output: node representations Z;

 # Stage 1. teacher-guided edge discriminator module.

for i=1 to $max - epoch$ **do**
  Calculate the edge weight σ of $\mathcal{G}$ (Eqn. (1-2));
  Learn the continuous weight of edge $\check{\partial}$ (Eqn. (3));
  Generate the teacher embedding $\boldsymbol{h}^{T1}$ and $\boldsymbol{h}^{T2}$;
  Calculate the margin ranking loss L_{mr} (Eqn. (4-6));
  Generate matrices $\boldsymbol{A}^{homo/heter}$;
end for

 # Stage 2. personalized graph masked autoencoder module.

for j=1 to t **do**
  Mask $\boldsymbol{A}^{homo}$ and $\boldsymbol{A}^{heter}$ based on α and β (Eqn. (7));
  Generate latent embeddings $\boldsymbol{H}^{homo/heter}$ (Eqn. (8));
  // Objective 1. reconstruct embedding.
  Decode the latent embeddings $\boldsymbol{H}^{homo}$ and $\boldsymbol{H}^{heter}$ to the embedding space (Eqn. (9));
  Calculate the InfoNCE loss L_{ml} (Eqn. (10-11));
  // Objective 2. reconstruct feature.
  Decode the latent embeddings $\boldsymbol{H}^{homo}$ and $\boldsymbol{H}^{heter}$ to the feature space (Eqn. (12));
  Calculate the SCE loss L_{sce} (Eqn. (13));
end for

Return **Z**.

The Algorithm and Complexity Analysis

The learning process of TEDMAE is presented in Algorithm 1. The teacher-guided edge discriminator serves as a pre-training module, with its time complexity primarily dependent on the teacher models and the training loss L_{mr}. Since Node2vec and PCA are selected as the teacher models, the time complexity is $\mathcal{O}(rl|\mathcal{V}|d_e + |\mathcal{V}|D^2 + D^3 + |\mathcal{E}|d_e)$, where r is the number of random walks, l is the length of random walks, d_e represents the dimension of node embeddings, $|\mathcal{V}|$ and $|\mathcal{E}|$ is the number of nodes and edges, respectively. The complexity of the graph masked autoencoder module primarily depends on the choice of the encoder and decoder. As the GAT and MLP are selected as encoder and decoder, the time complexity of this module is $\mathcal{O}(K|\mathcal{E}|D)$+$\mathcal{O}(l_m d_m^2)$, where K represents the number of attention heads, l_m is the layer number, and d_m is the dimension of each layer output.

Experiment

We conduct extensive experiments to assess the effectiveness of TEDMAE on various node-level tasks, aiming to address the following three key research questions.

RQ1: Does the proposed TEDMAE generalize well to multiple downstream tasks? **RQ2:** To what extent do the various modules in TEDMAE contribute to the classification results? **RQ3:** How do hyper-parameters affect performance?

Experiments Setup

Datasets and Baselines. The experiments are conducted on publicly available benchmark datasets, in line with previous studies, including homophily datasets Cora, WikiCS, Amazon-Computers, Amazon-Photo, CoAuthor-CS, CoAuthor-Physics and heterophily datasets Cornell, Texas, Wisconsin, and Actor.

We compare the performance of TEDMAE with that of 10 state-of-the-art models. These competitive baselines are classified into three categories. (i) a random-walk based model, i.e., Deepwalk (Perozzi, Al-Rfou, and Skiena 2014). (ii) contrastive models, including MVGRL (Hassani and Khasahmadi 2020), GCA (Zhu et al. 2021), BGRL (Thakoor et al. 2021), and GREET (Liu et al. 2023b). (iii) generative models, including VGAE (Kipf and Welling 2016), GraphMAE (Hou et al. 2022), GraphMAE2(Hou et al. 2023), S2GAE (Tan et al. 2023), GiGaMAE (Shi et al. 2023), Bandana (Zhao et al. 2024), and MaskGAE (Li et al. 2023a).

Parameter Settings. For the baselines, we closely follow the hyper-parameter settings of Bandana and GiGaMAE in (Zhao et al. 2024) and (Shi et al. 2023). After initializing the model using the parameters recommended by (Shi et al. 2023) and (Zhao et al. 2024), we adjust them further to achieve the optimal performance. As for the proposed TEDMAE, we utilize the Adam optimizer, set the dropout rate to 0.5, and initialize the learning rate to 0.01. Additionally, We also perform an hyper-parameter search for α, β, μ, and ν, with values ranging from 0.1 to 1.

Overall Performance (RQ1)

The node representations for downstream tasks are obtained by fusing representations from homophily and heterophily views through an adaptive attention mechanism.

Node Classification. We conduct node classification experiments on both homophilic and heterophilic datasets. Table 1 presents the results on homophily datasets, whereas Table 2 shows the results on heterophily datasets. We evaluate the quality of node representations with a logistic-regression classifier, fine-tuned with grid search. Overall, it is observed that the proposed TEDMAE outperforms all baselines on 5 out of 10 datasets, which demonstrates the outstanding performance of TEDMAE in the task of node classification.

We have several observations from Table 1. (i) Contrastive methods (MVGRL, GCA, and BGRL) outperform the traditional VGAE on almost all classification scenarios. (ii) The GMAE methods (GraphMAE, GraphMAE2, G2GAE, GiGaMAE, and Bandana) outperform the SOTA contrastive methods. (iii) TEDMAE outperforms GCL and GMAE methods. The primary reason is that it leverages a pre-trained model to refine the edge discriminator and mitigates the impact of the homophily assumption on GMAE. We have several observations from Table 2. (i) GAE methods do not perform as well as traditional GCN methods (Kipf and Welling 2017; Velickovic et al. 2018; Perozzi, Al-Rfou, and Skiena 2014) since dot-product probing is less than satisfactory. (ii) Contrastive methods outperform the traditional GCN methods. This is due to the additional su-

Model Type	Year	Dataset	Cora	WikiCS	Computers	Photo	CS	Physics
Randwalk	2014	Deepwalk	73.32±0.61	76.37±0.19	86.59±0.10	90.08±0.29	85.40±0.13	92.37±0.08
Contrastive	2020	MVGRL	84.39±0.34	80.15±0.27	88.28±0.13	92.29±0.19	92.91±0.07	95.36±0.06
	2021	GCA	83.86±0.45	78.86±0.26	88.06±0.12	92.44±0.29	92.66±0.09	95.50±0.18
	2021	BGRL	82.35±0.26	79.47±0.26	89.80±0.17	92.77±0.15	92.60±0.05	95.60±0.04
	2023	GREET	83.92±0.00	80.83±0.00	87.91±0.00	92.89±0.00	**94.59±0.00**	**95.98±0.00**
Generative	2016	VGAE	80.62±0.32	78.21±0.43	81.60±0.28	90.77±0.44	92.41±0.13	95.24±0.03
	2022	GraphMAE	84.23±0.42	80.57±0.17	89.39±0.84	92.83±0.43	92.68±0.30	95.52±0.15
	2023	GraphMAE2	84.41±0.30	81.01±0.34	89.50±0.76	92.87±0.14	92.74±0.14	95.38±0.08
	2023	S2GAE	84.14±0.65	79.14±0.23	89.64±0.12	92.09±0.28	89.92±0.17	94.37±0.14
	2023	GiGaMAE	84.42±0.47	81.14±0.16	**90.45±0.16**	93.01±0.41	92.72±0.32	95.66±0.14
	2023	MaskGAE	84.30±0.39	-	88.54±0.06	92.75±0.13	91.40±0.30	94.33±0.07
	2024	Bandana	84.45±0.65	80.12±0.37	88.71±0.16	92.98±0.07	92.97±0.08	95.44±0.04
	Our	**TEDMAE**	**84.64±0.30**	**81.31±0.14**	89.41±0.41	**93.26±0.15**	92.54±0.35	95.50±0.27

Table 1: Node classification performance comparison on homophilic benchmarks (Accuracy). The best results in each column are in **bold**. The runner-up results are highlighted with underline.

Methods	Cornell	Texas	Wisconsin	Actor
GCN	59.57	60.00	56.47	**30.83**
GAT	56.38	61.62	54.71	28.06
Deepwalk	47.74	46.49	33.53	22.78
node2vec	41.93	41.92	37.45	28.28
GAE	33.84	58.64	52.55	28.03
VGAE	35.22	59.20	56.67	26.99
MVGRL	51.07	62.38	62.37	30.02
GCA	49.80	59.46	50.78	29.65
BGRL	47.46	59.19	52.35	29.86
GraphMAE	56.76	**64.86**	57.53	29.81
GiGaMAE	56.76	62.16	64.69	29.80
TEDMAE	**59.64**	64.67	**64.71**	29.61

Table 2: Node classification performance on heterophilic benchmarks (Accuracy).

pervisory information providing a richer graph structure information and higher-order relationships. (iii) The performance of TEDMAE surpasses that of GiGaMAE, which validates the significance of simultaneously reconstructing embeddings and features. The proposed method performs the best on Cora, WikiCS, and Amazon-Photo, but not so well on Computers and CS/Physics datasets. The main reason is that Computers dataset has high edge homogeneity, while CoAuthor dataset has a lower density. Therefore, our method can effectively handle the graphs that are not too sparse and contain balanced homophily/heterophily edges.

Node Clustering. For the node clustering task, we employ the K-Means algorithm and set the number of clusters to match the number of label classes. Table 3 reports the clustering results, including the values of normalized mutual information (NMI) and average rand index (ARI). It can be observed that the proposed TEDMAE achieves the best/runner-up performance on 5 of 6 datasets. The following observations can be drawn from Table 3. (i) The generative model outperforms the contrastive and randwalk models on most datasets, particularly on Cora and WikiCS datasets. It suggests that the generative model has an advantage in generating structured data. (ii) The proposed TEDMAE, as a masked graph autoencoder method, achieves superior performance, particularly on Amazon-Photo and CoAuthor-CS datasets. Thus, it fully demonstrates the importance of simultaneously reconstructing features and embeddings.

Ablation Study (RQ2)

Evaluation on variants. To verify the effectiveness of each module, we conduct the ablation study which includes four variants. '-w/o-HomoMAE' means to remove the homophilic GMAE module. '-w/o-HeterMAE' means to remove the heterophilic GMAE module. '-w/o-Recon_Emb' means that it does not reconstruct embeddings. '-w/o-Recon_Fea' means that it does not reconstruct features.

The experimental results of ablation study are shown in Table 4. It can be observed that TEDMAE achieves the best performance compared to the other four variants. It suggests that all components of TEDMAE are effective in boosting performance. Specifically, '-w/o-Recon_Emb' performs better than '-w/o-Recon_Fea', which demonstrates that reconstructing embeddings plays an essential role. '-w/o-HomoMAE' performs better than '-w/o-HeterMAE', indicating that the heterophilic view provides additional information for homophilic datasets. The results prove that the two modules of the proposed TEDMAE are beneficial to improve the performance and generalization of classification.

Evaluation on teacher models. It is crucial to select the teacher model that plays an essential role in guiding the edge discriminator. We compare the performance of TEDMAE under the guidance of different teacher models, with the experimental results presented in Table 5. It is observed

Model Type	Dataset	Cora	WikiCS	Computers	Photo	CS	Physics
Randwalk	Deepwalk	0.4161 / 0.3416	0.4660 / 0.3587	0.4202 / 0.2637	0.6482 / 0.5129	0.6445 / 0.4863	0.6995 / 0.7985
Contrastive	MVGRL	0.5481 / 0.5167	0.2135 / 0.1101	0.2657 / 0.1806	0.1776 / 0.1127	0.6436 / 0.4737	0.4948 / 0.4799
	GCA	0.4645 / 0.3268	0.1463 / 0.0176	0.4062 / 0.1512	0.4480 / 0.2518	0.6975 / 0.5578	0.6638 / 0.7468
	BGRL	0.2851 / 0.0920	0.2767 / 0.0937	0.4396 / 0.2096	0.6189 / 0.4754	0.7740 / 0.6422	0.7249 / 0.8130
Generative	VGAE	0.4930 / 0.4392	0.3453 / 0.1478	0.3073 / 0.2054	0.4847 / 0.3539	0.7736 / 0.6646	0.4925 / 0.2628
	GraphMAE	0.5781 / 0.5082	0.4038 / 0.2951	0.5015 / 0.3298	0.6676 / 0.5703	0.7297 / 0.5691	0.6348 / 0.6734
	GraphMAE2	0.5821 / 0.5310	0.3674 / 0.2541	0.5053 / 0.3418	0.6496 / 0.5613	0.4423 / 0.2449	0.2820 / 0.1564
	S2GAE	0.5127 / 0.4481	0.3143 / 0.1110	0.4397 / 0.2297	0.5624 / 0.3427	0.6251 / 0.4289	0.6152 / 0.7059
	GiGaMAE	**0.5836** / 0.5453	0.4910 / **0.4239**	**0.5228** / 0.3579	0.7066 / 0.5859	0.7622 / 0.6417	**0.7373 / 0.8271**
	Bandana	0.5793 / **0.5504**	0.3795 / 0.2540	0.5013 / 0.3550	0.6172 / 0.4823	0.7214 / 0.5954	0.6374 / 0.5131
	TEDMAE	0.5823 / 0.5460	**0.4966** / 0.4192	0.5146 / **0.4049**	**0.7157 / 0.6191**	**0.7815 / 0.7638**	0.7106 / 0.7851

Table 3: Node Clustering performance comparison on homophilic benchmarks (NMI/ARI).

Dataset	Cora	WikiCS	Amazon-Photo
-w/o-HomoMAE	83.88	81.06	92.94
-w/o-HeterMAE	84.07	81.31	92.91
-w/o-Recon_Emb	83.59	79.83	92.29
-w/o-Recon_Fea	84.33	80.54	92.95
TEDMAE	**84.96**	**81.35**	**93.26**

Table 4: The experimental results of ablation study.

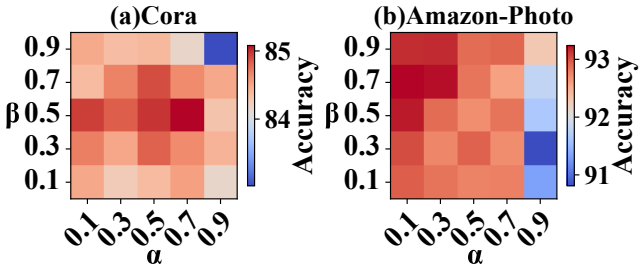

Figure 3: Parameter sensitivity analysis of α and β.

that TEDMAE_*node2vec* and TEDMAE_*PCA* outperform Node2vec, PCA, and GAE. It demonstrates the superiority of TEDMAE. Moreover, TEDMAE also outperforms TEDMAE_*node2vec* and TEDMAE_*PCA*. It indicates that a teacher model that considers both topology and features is more important than training the discriminator from either the topology or feature perspective alone. Although TEDMAE_*GCN* and _*GAT* also achieve outstanding results, they often require adequate label information.

Dataset	Cora		WikiCS	
Metrics	**ACC**	**NMI**	**ACC**	**NMI**
Node2vec	71.76	0.3944	71.57	0.4081
PCA	42.22	0.0212	68.52	0.3181
GAE	81.29	0.4907	70.33	0.1091
TEDMAE_$Node2vec$	83.74	0.5581	81.21	0.4889
TEDMAE_PCA	84.17	0.5591	81.33	0.4845
TEDMAE_GCN	84.59	0.5884	81.61	0.5077
TEDMAE_GAT	84.74	0.6156	81.51	0.5109
TEDMAE	**84.94**	**0.5823**	**81.36**	**0.4966**

Table 5: Different teacher-guided edge discriminator performance comparison.

Parameter Sensitivity Analysis (RQ3)

Analysis of α and β. α and β represent the ratio of feature masking in homophily and heterophily views, respectively. Figure 3 depicts the performance with different masking ratios on Cora and Amazon-Photo datasets. On Cora dataset, it is observed that the classification performance is optimal when the masking ratios α and β are set to 0.5 and 0.7, respectively. However, the classification performance declines when the masking ratios fall outside this range. On Amazon-Photo dataset, the best classification performance is achieved when parameters α and β are set to 0.2 and 0.2, respectively. If α is set to a higher value of 0.9, the classification performance is less desirable. The main reason is that a higher mask ratio results in more features being discarded. It leads to a reduction in the amount of information available to the model, which in turn affects its performance.

Conclusion

In this paper, we propose TEDMAE, a teacher-guided edge discriminator for the personalized graph masked autoencoder method. Specifically, we design an edge discriminator to differentiate between nodes connected by homophilic and heterophilic edges. In addition, we present a personalized graph masked autoencoder that individually masks, encodes, and reconstructs each graph. Furthermore, we employ SCE loss and InfoNCE loss to optimize TEDMAE by reconstructing embeddings and features. Finally, the experimental results demonstrate the effectiveness of TEDMAE.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (Grant No. 62472263, 62072288), the Taishan Scholar Program of Shandong Province, Shandong Youth Innovation Team, the Natural Science Foundation of Shandong Province (Grant No. ZR2024MF034, ZR2022MF268).

References

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *JACM*, 58(3): 1–37.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 4171–4186.

Garcia Duran, A.; and Niepert, M. 2017. Learning graph representations with embedding propagation. *NeurIPS*, 30: 1–12.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*, 855–864.

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *ICML*, 4116–4126.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *CVPR*, 16000–16009.

Hou, Z.; He, Y.; Cen, Y.; Liu, X.; Dong, Y.; Kharlamov, E.; and Tang, J. 2023. GraphMAE2: A decoding-enhanced masked self-supervised graph learner. In *WWW*, 737–746.

Hou, Z.; Liu, X.; Cen, Y.; Dong, Y.; Yang, H.; Wang, C.; and Tang, J. 2022. GraphMAE: Self-supervised masked graph autoencoders. In *KDD*, 594–604.

Ju, W.; Wang, Y.; Qin, Y.; Mao, Z.; Xiao, Z.; Luo, J.; Yang, J.; Gu, Y.; Wang, D.; Long, Q.; et al. 2024. Towards Graph Contrastive Learning: A Survey and Beyond. *arXiv preprint arXiv:2405.11868*, 1–35.

Kipf, T. N.; and Welling, M. 2016. Variational graph auto-encoders. *NeurIPS*, 1–23.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*, 1–14.

Li, J.; Wu, R.; Sun, W.; Chen, L.; Tian, S.; Zhu, L.; Meng, C.; Zheng, Z.; and Wang, W. 2023a. What's Behind the Mask: Understanding Masked Graph Modeling for Graph Autoencoders. In *KDD*, 1268–1279.

Li, X.; Ye, T.; Shan, C.; Li, D.; and Gao, M. 2023b. SeeGera: Self-supervised Semi-implicit Graph Variational Auto-encoders with Masking. In *WWW*, 143–153.

Liu, C.; Wang, Y.; Zhan, Y.; Ma, X.; Tao, D.; Wu, J.; and Hu, W. 2024. Where to Mask: Structure-Guided Masking for Graph Masked Autoencoders. In *IJCAI*, 2180–2188.

Liu, C.; Zhan, Y.; Ma, X.; Ding, L.; Tao, D.; Wu, J.; and Hu, W. 2023a. Gapformer: Graph Transformer with Graph Pooling for Node Classification. In *IJCAI*, 2196–2205.

Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; and Tang, J. 2021. Self-supervised learning: Generative or contrastive. *TKDE*, 35(1): 857–876.

Liu, Y.; Zheng, Y.; Zhang, D.; Lee, V. C.; and Pan, S. 2023b. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *AAAI*, volume 37, 4516–4524.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Shi, Y.; Dong, Y.; Tan, Q.; Li, J.; and Liu, N. 2023. GiGaMAE: Generalizable graph masked autoencoder via collaborative latent space reconstruction. In *CIKM*, 2259–2269.

Sun, Y.; Deng, H.; Yang, Y.; Wang, C.; Xu, J.; Huang, R.; Cao, L.; Wang, Y.; and Chen, L. 2022. Beyond Homophily: Structure-aware Path Aggregation Graph Neural Network. In *IJCAI*, 2233–2240.

Tan, Q.; Liu, N.; Huang, X.; Choi, S.-H.; Li, L.; Chen, R.; and Hu, X. 2023. S2GAE: Self-supervised graph autoencoders are generalizable learners with graph masking. In *WSDM*, 787–795.

Thakoor, S.; Tallec, C.; Azar, M. G.; Munos, R.; Veličković, P.; and Valko, M. 2021. Bootstrapped representation learning on graphs. In *ICLR*, 1–14.

Tian, Y.; Zhang, C.; Kou, Z.; Liu, Z.; Zhang, X.; and Chawla, N. V. 2024. UGMAE: A Unified Framework for Graph Masked Autoencoders. *arXiv preprint*, 1–10.

Tu, W.; Liao, Q.; Zhou, S.; Peng, X.; Ma, C.; Liu, Z.; Liu, X.; Cai, Z.; and He, K. 2024. RARE: Robust Masked Graph Autoencoder. *TKDE*, 36(10): 5340–5353.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*.

Wang, L.; Tao, X.; Liu, Q.; Wu, S.; and Wang, L. 2024a. Rethinking Graph Masked Autoencoders through Alignment and Uniformity. In *AAAI*, 15528–15536.

Wang, Y.; Yan, X.; Hu, C.; Xu, Q.; Yang, C.; Fu, F.; Zhang, W.; Wang, H.; Du, B.; and Jiang, J. 2024b. Generative and Contrastive Paradigms Are Complementary for Graph Self-Supervised Learning. In *ICDE*, 3364–3378.

Zhang, Q.; Wang, Y.; and Wang, Y. 2022. How Mask Matters: Towards Theoretical Understandings of Masked Autoencoders. In *NeurIPS*, 27127–27139.

Zhao, Z.; Li, Y.; Zou, Y.; Tang, J.; and Li, R. 2024. Masked Graph Autoencoder with Non-discrete Bandwidths. In *WWW*, 377–388.

Zheng, Y.; and Jia, C. 2024. ProtoMGAE: Prototype-Aware Masked Graph Auto-Encoder for Graph Representation Learning. *TKDD*, 18(6): 137:1–137:22.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *NeurIPS*, 33: 7793–7804.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*, 2069–2080.