Graph Positional Autoencoders as Self-supervised Learners

Yang Liu*
Beijing University of Posts and
Telecommunications
Beijing, China
liuyangjanet@bupt.edu.cn

Deyu Bo*
National University of Singapore
Singapore
bodeyu1996@gmail.com

Wenxuan Cao
Beijing University of Posts and
Telecommunications
Beijing, China
wenxuanc@bupt.edu.cn

Yuan Fang
Singapore Management University
Singapore
yfang@smu.edu.sg

Yawen Li
Beijing University of Posts and
Telecommunications
Beijing, China
warmly0716@126.com

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

Abstract

Graph self-supervised learning seeks to learn effective graph representations without relying on labeled data. Among various approaches, graph autoencoders (GAEs) have gained significant attention for their efficiency and scalability. Typically, GAEs take incomplete graphs as input and predict missing elements, such as masked node features or edges. Although effective, our experimental investigation reveals that traditional feature or edge masking paradigms primarily capture low-frequency signals in the graph and fail to learn expressive structural information. To address these issues, we propose Graph Positional Autoencoders (GraphPAE), which employ a dual-path architecture to reconstruct both node features and positions. Specifically, the feature path uses positional encoding to enhance the message-passing processing, improving the GAEs' ability to predict the corrupted information. The position path, on the other hand, leverages node representations to refine positions and approximate eigenvectors, thereby enabling the encoder to learn diverse frequency information. We conduct extensive experiments to verify the effectiveness of GraphPAE, including heterophilic node classification, graph property prediction, and transfer learning. The results demonstrate that GraphPAE achieves state-of-the-art performance and consistently outperforms the baselines by a large margin.

CCS Concepts

• **Information systems → Data mining**.

Keywords

Graph Neural Networks, Self-supervised Learning, Graph Autoencoders, Positional Encoding

---

*Both authors contributed equally to this research.
†Corresponding author.

---

1 Introduction

Graph neural networks (GNNs) have achieved significant success across various fields, including social network analysis [7, 47, 72], recommendation systems [13, 44], and drug discovery [5, 24, 67]. However, training effective GNNs in real-world applications remains challenging due to the limited availability of labeled data in many domains [73, 79]. To address this problem, graph self-supervised learning is proposed to learn graph representations without labeled data [33, 66, 74].

Among existing approaches, the generative [56, 62] and contrastive [25] learning paradigms have dominated recent advances. In particular, graph autoencoders (GAEs) [18, 26] have gained attention due to their simplicity, efficiency, and scalability. The GAEs follow a corruption-reconstruction framework, which learns graph representations by recovering the missing information of the incomplete input graphs, as Table 1 illustrates. For example, Graph-MAE [18] replaces node features with a learnable token, while Bandana [77] proposes a non-discrete edge masking strategy. Moreover, some GAEs even go beyond reconstructing node features and edges by targeting structural features like degree [28]. Despite their success, the performance of GAEs highly depends on the choice of corruption and reconstruction objectives. Therefore, a natural question arises: *Do existing feature or edge masking mechanisms fully exploit the graph data? If not, what alternative objectives could be designed to further improve the performance of GAEs?* A well-informed answer can help us identify the weaknesses of existing masking strategies and deepen our understanding of GAEs.

To answer this question, we first revisit the feature and edge masking strategies from a spectral perspective [2, 3]. Specifically, we transform the node features $\mathbf{X}$ into the spectral domain by using the eigenvectors of graph Laplacian $\mathbf{U}^\top$ and examine its frequency

(a) Feature Masking
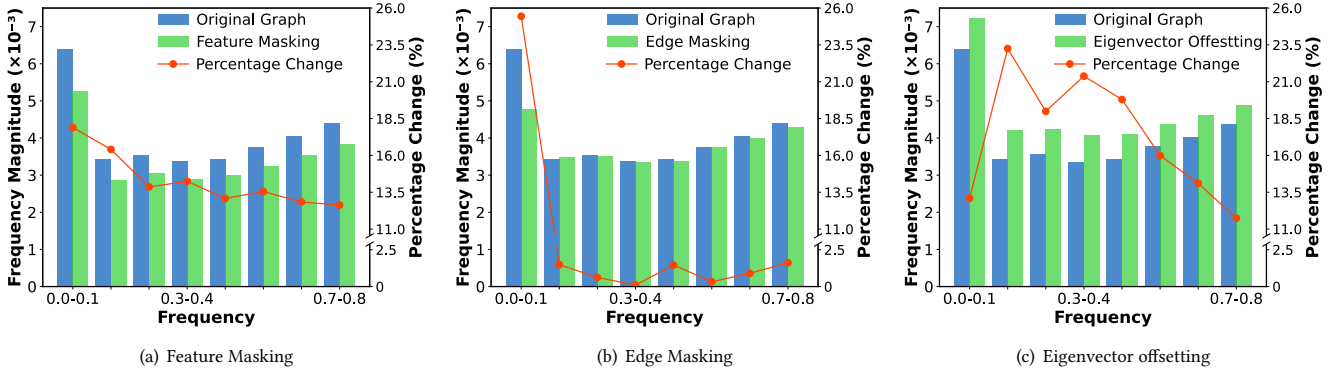
(b) Edge Masking

(c) Eigenvector offsetting

Figure 1: Frequency magnitudes of the original and corrupted graphs in the Squirrel dataset.

Table 1: Comparison of different graph autoencoders.

Model	Corruption			Reconstruction		
	Feature	Edge	Position	Feature	Edge	Other
GraphMAE [18]	✓	-	-	✓	-	-
StructMAE [32]	✓	-	-	✓	-	-
AUG-MAE [59]	✓	-	-	✓	-	-
S2GAE [50]	-	✓	-	-	✓	-
SeeGera [30]	✓	✓	-	✓	✓	-
Bandana [77]	-	✓	-	-	✓	-
MaskGAE [28]	-	✓	-	✓	-	Degree
GiGaMAE [45]	✓	✓	-	-	-	Latent
Our work	✓	-	✓	✓	-	Position

magnitude $\mathbf{U}^\top \mathbf{X}$. In case of feature masking, we randomly mask 20% node features and denote it as $\widetilde{\mathbf{X}}$, while for edge masking, we randomly remove 20% edges and construct the corrupted graph eigenvectors $\widetilde{\mathbf{U}}^\top$. The frequency magnitude changes of $\mathbf{U}^\top \widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{U}}^\top \mathbf{X}$ are shown in Figures 1(a) and 1(b), respectively. It can be observe that the magnitude differences between original and corrupted graphs are more pronounced in the low-frequency band, *i.e.*, [0.0, 0.1], while the differences decrease at higher frequencies. Notably, to effectively minimize the reconstruction loss, GAEs will primarily focus on the frequency bands with larger discrepancies. As a result, existing GAEs mainly reconstruct low-frequency information and overlook the high-frequency information, which has been shown to be valuable in real-world tasks [4, 60]. More experimental details are provided in Appendix A.

Given the above weakness of existing GAEs, it is natural to ask: *How can we design the corruption and reconstruction objectives to exploit the diverse frequency information?* Essentially, the eigenvectors of graph Laplacian represent different frequencies. Directly perturbing and reconstructing the eigenvectors can help GAEs learn different frequency information in graphs. Figure 1(c) illustrates the frequency magnitudes of the eigenvectors with random offsets, *i.e.*, $\widetilde{\mathbf{U}}^\top X = (\mathbf{U} + \delta)^\top X$. We observe that this strategy perturbs a broader range of frequency components, particularly in higher-frequency, *i.e.* [0.2, 0.5], enabling the model to capture more diverse

frequency information. However, reconstructing the eigenvectors is not a trivial task and presents two main challenges: (1) Eigenvectors represent the global structural patterns of a graph, which cannot be easily approximated by basic GNNs. Existing GAEs typically adopt message-passing neural networks (MPNNs) as the encoder, whose expressiveness is bounded by the 1-WL test [68]. The intrinsic weakness of MPNNs restricts GAEs' ability to capture long-range dependencies between nodes and higher-order graph patterns [12, 63]. (2) Eigenvectors suffer from sign- and basis-ambiguity issues [31]. Directly reconstructing the eigenvectors leads to non-unique solutions, affecting the robustness of GAEs [1, 58].

To overcome these challenges, we propose Graph Positional Autoencoders (GraphPAE) for graph self-supervised learning. Graph-PAE adopts a dual-path architecture to address both the expressivity and ambiguity issues. Specifically, in the feature path, GraphPAE integrates positional encoding (PE) into the message-passing process to enhance the expressiveness of MPNNs, thus improving the model's ability to reconstruct corrupted information. Moreover, in the position path, node representations are used to refine the PE, which approximates the eigenvectors, thereby transferring diverse frequency information into the encoder. Finally, in the reconstruction stage, instead of directly recovering the raw eigenvectors, GraphPAE uses the relative node distance as a surrogate objective, avoiding potential ambiguity issues.

Contributions. The contributions of our paper are as follows:

(1) We are the first to explore the masking strategy in GAEs from a spectral perspective. By comparing the frequency magnitudes between original and corrupted graphs, we identify that existing GAEs focus on reconstructing the low-frequency information of graphs and neglecting other frequencies.

(2) We propose GraphPAE, a novel graph positional autoencoder that learns graph representations by reconstructing both node features and positions, thereby enabling GAEs to capture a broader range of frequency information.

(3) We benchmark GraphPAE against state-of-the-art baselines across various tasks, including node classification, graph property prediction, and transfer learning. The results on 14 graphs demonstrate that GraphPAE consistently outperforms the baselines and shows impressive performance on heterophilic graphs.

2 Related Work

We provide an overview of two main areas related to our work, namely generative approaches in graph self-supervised learning and positional encoding techniques to enhance structural expressiveness in GNNs.

2.1 Generative Graph Self-supervised Learning

Generative graph learning encompasses a set of graph self-supervised learning techniques aimed at reconstructing missing information in incomplete input graphs. It can be broadly categorized into autoregressive and autoencoding approaches.

Graph Autoregressive models (GARs). GARs treat sequential graph generation as the pre-training task, where the node or edge is predicted based on its prior context. GPT-GNN [21] factorizes each node generation into attribute and edge generation and replenishes the omitted parts by an adaptive queue. MGSSL [76] introduces motif generation into pre-training. GraphsGPT [14] transforms non-Euclidean graphs into learnable Euclidean words and reconstructs the original graph from these words.

Graph Autoencoders (GAEs). GAEs reconstruct all desired content once from the latent representation output by the encoder. Early GAEs (*e.g.*, VGAE [26] and ARGVA [38]) learn representations through link reconstruction and spark a series of work, including feature reconstruction (*e.g.*, GALA [39] and WGDN [8]) and combination reconstruction of structure and feature (*e.g.*, GATE [43]). However, these traditional GAEs often perform poorly in downstream tasks, except link prediction, which is attributed to their overemphasis on proximity information at the expense of structural information [15]. Recently, masked GAEs [18, 20] have become highly successful models for representation learning. GraphMAE [18] and GraphMAE2 [17] successfully bridge the performance gap between graph contrastive learning and generative learning by reconstructing masked features for training. AUG-MAE [59] introduces an adversarial masking strategy to enhance feature alignment and add a uniformity regularizer to promote high-quality graph representations. Additionally, some works are focusing on masking and reconstructing graph structures (*e.g.*, edges [28, 50] and paths [28]). Bandana [77] uses continuous and dispersive edge masks and bandwidth prediction instead of discrete edge masks and reconstruction. Besides, some works mask features and edges simultaneously [30, 45] and propose novel reconstruction objectives such as latent embeddings [45].

2.2 Graph Positional Encoding

Positional encodings (PEs), originally designed to enhance transformers by encoding positional information in sequential data, have been introduced in graph learning to provide explicit position-aware features. By assigning unique identifiers to nodes based on their structural positions, PEs enable graph neural networks (GNNs) to distinguish non-isomorphic structures beyond the limitations of the 1-WL test. Existing graph PEs can be divided into two main categories: Laplacian-based PEs and Distance-based PEs.

Laplacian-based PEs. These methods leverage the eigenvectors of the graph Laplacian matrix as initial node positional features [9–11].

Eigenvectors form a natural basis for encoding graph structure, similar to how sinusoidal functions represent positional information in sequential models [53]. Serving as PEs, eigenvectors have two traditional constraints, *i.e.*, sign- and basis ambiguity [31]. To keep the models robust to sign ambiguity, [9, 10] randomly flip the sign of the eigenvectors during training. SAN [27] introduces the information of eigenvalues into PEs and leverages a transformer-based positional encoder, enabling models to be more informative and expressive. PEG [58] solves the sign- and basis-ambiguity by treating the distance of eigenvectors between node pairs as PEs. [31] propose SignNet and BasisNet to learn sign- and basis-invariant PEs, respectively. [1, 22] study stable PEs that are robust to disturbance of the Laplacian matrix.

Distance-based PEs. These approaches assign positional features to nodes based on their relative distances within the graph structure, which are typically derived from spatial relationships [29, 41]. One common approach is to use the random walk matrix capturing structural relationships in a probabilistic manner [10, 11]. Another popular method is to encode the shortest distance between node pairs [29, 41, 69]. For example, Graphormer [69] incorporates shortest-path distances into its attention mechanism. Additionally, GraphiT [37] encodes PEs with a diffusion kernel, enabling more flexible and adaptive representations of node relationships. [46] introduces a novel positional encoding scheme that extends transformers to tree-structured data, enabling efficient and parallelizable sequence-to-tree, tree-to-sequence, and tree-to-tree mappings. [61] first introduces PEs in graph self-supervised learning and designs a GNN framework that integrates a k-hop message-passing mechanism to enhance its expressiveness.

3 Preliminaries

Before describing our method, we first define some notations and introduce some important concepts used in this paper.

Problem Definition. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V}$ is the node set with $\mathcal{V} = \{v_i\}_{i=1}^N$, $\mathcal{E}$ is the set of edges, and $\mathbf{X} \in \mathbb{R}^{N \times d}$ represents the d-dimensional node feature matrix. We use the adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ to describe the graph structure, where $A_{ij} = 1$ if there is an edge between v_i and v_j, and $A_{ij} = 0$ otherwise. The goal of our framework is to learn an effective graph encoder $f_{enc}(\cdot)$ without relying on labels from downstream tasks. Once trained, the encoder $f_{enc}(\cdot)$ generates node representations as $\mathbf{H} = f_{enc}(\mathcal{G}) \in \mathbb{R}^{N \times d_h}$ or graph representations as $\mathbf{H}_{\mathcal{G}} = \text{READOUT}\left(\mathbf{H}_{v_i} \mid v_i \in \mathcal{V}\right) \in \mathbb{R}^{d_h}$, where READOUT is a permutation-invariant function such as mean, max or sum pooling. The representations can then be used to train a simple linear classifier on labeled data from downstream tasks.

Laplacian Graph Eigenvectors. The normalized graph Laplacian matrix $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, where $\mathbf{I}_N$ is an identity matrix and $\mathbf{D}$ is the degree matrix with $D_{ii} = \sum_j A_{ij}$ for $v_i \in \mathcal{V}$, and $D_{ij} = 0$ for $i \neq j$. The Laplacian matrix $\mathbf{L}$ can be decomposed as $\mathbf{L} = \mathbf{U} \Lambda \mathbf{U}^\top$, where $\Lambda = \text{diag}(\{\lambda_i\}_{i=1}^N)$ is the diagonal matrix of eigenvalues, and $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N] \in \mathbb{R}^{N \times N}$ consists of a set of eigenvectors. Each eigenvector $\mathbf{u}_i \in \mathbb{R}^N$ corresponds to eigenvalue λ_i, where the eigenvalues are ordered as $0 \leq \lambda_1 \leq \cdots \leq \lambda_N \leq 2$. Given eigen-decomposition $\mathbf{L} \mathbf{u}_i = \lambda_i \mathbf{u}_i$ with $\mathbf{u}_i^\top \mathbf{u}_i = 1$, we have

(a) Overall pipeline of GraphPAE.
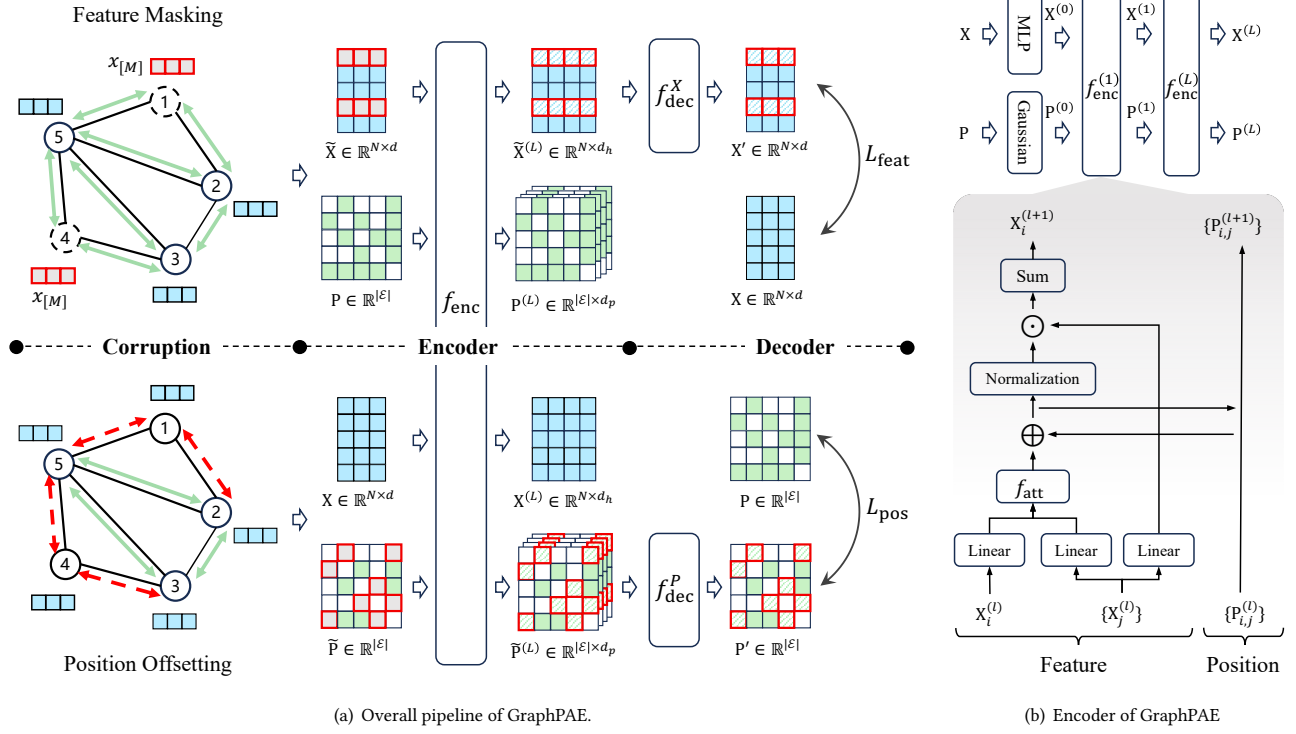
(b) Encoder of GraphPAE

Figure 2: (a): GraphPAE integrates a positional corruption-reconstruction mechanism to encourage the GAE to capture diverse frequency information. For feature reconstruction, masked features are encoded with P and decoded to recover original features. For positional reconstruction, noise is added to $\widetilde{U}$ to corrupt relative distances, which are then encoded with X and decoded to recover the original distances. Figure (b): The encoder employs a dual-path architecture to update node $X^{(l)}$ and positional representations $P^{(l)}$ at each layer. The feature path integrates positional encodings to enhance message passing, improving the GAE's ability to reconstruct corrupted features. The position path utilizes node representations to refine positional embeddings to approximate original pairwise distances.

$\mathbf{u}_i^\top \mathbf{L} \mathbf{u}_i = \lambda_i$. Since $\mathbf{u}_i^\top \mathbf{L} \mathbf{u}_i = \sum_{(j,k) \in \mathcal{E}} (\mathbf{u}_{i,j} - \mathbf{u}_{i,k})^2$, we finally derive that $\sum_{(j,k) \in \mathcal{E}} (\mathbf{u}_{i,j} - \mathbf{u}_{i,k})^2 = \lambda_i$, which indicates that λ_i reflects the frequency magnitude of eigenvector $\mathbf{u}_i$ over the graph. In this paper, we follow the popular positional encoding methods and adopt top-k eigenvectors as initial positions to reduce complexity. Therefore, we redefine $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_K] \in \mathbb{R}^{N \times K}$, where $K \leq N$.

4 Proposed Framework: GraphPAE

In this section, we introduce GraphPAE, a novel GAE designed to reconstruct both node features and positions. There are three key components in GraphPAE: corruption, encoder, and decoder, as illustrated in Figure 2(a).

4.1 Data Corruption

The corruption-reconstruction paradigm has become a basic component of autoencoders. A well-designed masking strategy can prevent information leakage and enhance model efficiency [16]. Existing GAEs commonly apply the random masking strategy to the discrete nodes and edges. However, node positions, such as eigenvectors, are continuous by nature and are not suitable for the random masking approach. Therefore, we adopt different corruption strategies for these two modalities.

Feature Masking. We follow GraphMAE [18] and replace the masked node features with a learnable vector. Specifically, we randomly sample a subset of nodes $\widetilde{\mathcal{V}} \subset \mathcal{V}$ and reset their features. The corrupted feature matrix $\widetilde{X}$ is defined as:

$$\widetilde{X}_i = \begin{cases} \mathbf{x}_{[M]}, & \text{if } v_i \in \widetilde{\mathcal{V}} \\ X_i, & \text{if } v_i \notin \widetilde{\mathcal{V}} \end{cases} \tag{1}$$

where $X_i \in \mathbb{R}^d$ is the original feature of node v_i, and $\mathbf{x}_{[M]} \in \mathbb{R}^d$ is a learnable vectors.

Position Offsetting. Inspired by the recent advances in molecular representation learning [78], we propose to add random offsets to node positions. Specifically, we use the top-k eigenvectors of graph Laplacian $\mathbf{U} \in \mathbb{R}^{N \times K}$ as the initialization of node positions. Similar to the feature corruption, we define the corruption of node positions as:

$$\widetilde{U}_i = \begin{cases} \mathbf{U}_i + \delta, & \text{if } v_i \in \widetilde{\mathcal{V}} \\ \mathbf{U}_i, & \text{if } v_i \notin \widetilde{\mathcal{V}} \end{cases} \tag{2}$$

where $\mathbf{U}_i \in \mathbb{R}^K$ denotes the position for node v_i and $\delta \in \mathbb{R}^K$ is a noise vector sampled from a uniform distribution $\mathcal{U}\left(-\mu_p, \mu_p\right)$. In practice, we set μ_p to 0.001 or 0.01 for different datasets.

Relative Positional Encoding. Reconstructing eigenvectors enables GAEs to learn different frequency information. However, it is challenging to recover the corrupted eigenvectors as they suffer from the sign- and basis-ambiguity [1, 31]. To address this, we compute the Euclidean distance between each pair of nodes to obtain their relative node distances

$$\mathbf{P}_{i,j} = \begin{cases} \|\mathbf{U}_i - \mathbf{U}_j\|_2, & \text{if } \mathbf{A}_{i,j} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{P} \in \mathbb{R}^{N \times N}$ is the relative distance matrix and we use $\widetilde{\mathbf{P}}$ to indicate its corrupted version. The relative distance matrix is used as a surrogate of node positions to eliminate the ambiguity of eigenvectors.

4.2 GraphPAE Encoder

After corruption, the node features and pair-wise distances are then fed into the encoder to learn node and position representations through message-passing, which can be formulated as

$$\mathbf{X}_i^{(l+1)}, \mathbf{P}_i^{(l+1)} = f_{\text{enc}}^{(l+1)}\left(\mathbf{X}_i^{(l)}, \left\{\mathbf{X}_j^{(l)}\right\}_{j \in \mathcal{N}_i}, \mathbf{P}_i^{(l)}\right), \tag{4}$$

where $l \in \{0, 1, 2, ..., L\}$ indicates the layer of encoder, and $\mathcal{N}_i$ is the neighbors of node v_i. The first layer of the encoder, $e.g., f_{\text{enc}}^{(0)}$, is designed to align the dimensions of node features and position representations. Specifically, it first lifts the scalar relative distance $\mathbf{P}_{i,j} \in \mathbb{R}$ into a vector representation $\mathbf{P}_{i,j}^{(0)} \in \mathbb{R}^{d_h}$ through a series of Gaussian RBF kernels

$$\mathbf{P}_{i,j}^{(0)} = \text{MLP}\left(\left[G(\mathbf{P}_{i,j}; \mu_1, \sigma), \cdots, G(\mathbf{P}_{i,j}; \mu_d, \sigma)\right]\right), \tag{5}$$

where MLP stands for Multi-layer Perceptron and $G(\mathbf{P}_{i,j}; \mu_k, \sigma) = \exp\left(-\left(\mathbf{P}_{i,j} - \mu_k\right)^2 / 2\sigma^2\right)$ is the k-th Gaussian basis functions with mean μ_k and standard deviation σ. As for the node features, it uses another MLP to transform them into d_h-dimension representations

$$\mathbf{X}_i^{(0)} = \text{MLP}\left(\mathbf{X}_i\right). \tag{6}$$

After transformation, the encoder needs to aggregate information from both sides to update the node and position representations layer by layer, as shown in Figure 2(b).

Feature Path: PE-enhanced MPNNs. It is well-established that the expressive power of traditional MPNNs is bounded by the 1-WL test [6, 10]. Fortunately, existing methods prove that adding position information to the message-passing process can significantly improve the expressive power of MPNNs [11, 22]. Inspired by the recent progress in graph PEs [57], we propose to incorporate the position representation into MPNNs as follows:

$$\begin{aligned} \alpha_{i,j}^{(l)} &= f_{\text{att}}\left(\mathbf{X}_i^{(l)}, \mathbf{X}_j^{(l)}\right), \quad \alpha_{i,j}^{(l)} \in \mathbb{R}^d, \\ \mathbf{X}_i^{(l+1)} &= \sum_{j \in \mathcal{N}_i}\left(\alpha_{i,j}^{(l)} + \mathbf{P}_{i,j}^{(l)}\right) \odot \text{MLP}\left(\mathbf{X}_j^{(l)}\right), \end{aligned} \tag{7}$$

where f_{att} is the attention function to calculate the weights of neighbors and $\odot$ indicates the element-wise multiplication. In general, there are many MPNNs to implement the attention function. For

example, if the encoder is GAT [54], then d_h corresponds to the number of attention heads, and the attention function is defined as

$$\alpha_{i,j}^{(l)} = \text{LeakyReLU}\left(\mathbf{w}^T\left[\mathbf{W}^{(l)}\mathbf{X}_i^{(l)} \,\|\, \mathbf{W}^{(l)}\mathbf{X}_j^{(l)}\right]\right), \quad \mathbf{w} \in \mathbb{R}^{2d_h}. \tag{8}$$

For GatedGCN, d_h is equal to the dimension of the node representation, and the attention function is defined as

$$\alpha_{i,j}^{(l)} = \text{Sigmoid}\left(\mathbf{W}_1^{(l)}\mathbf{X}_i^{(l)} + \mathbf{W}_2^{(l)}\mathbf{X}_j^{(l)}\right). \tag{9}$$

Without loss of generality, we omit edge features in the formulation. Regarding the normalization function in Figure 2(b), GAT uses the Softmax function, and GatedGCN applies degree normalization.

Position Path: Refine Node Positions. The feature path improves the expressiveness of GraphPAE, but still lacks diverse frequency information. To solve this issue, the position path uses the learned attention weights to update the position representations

$$\mathbf{P}_{i,j}^{(l+1)} = \alpha_{i,j}^{(l)} + \mathbf{P}_{i,j}^{(l)}. \tag{10}$$

Intuitively, the attention weights gradually refine position representations to approach the ground-truth node positions. Notably, the relative distance $\mathbf{P}$ is calculated based on the eigenvectors of graph Laplacian, which contains various frequency information, from low-frequency $\mathbf{u}_1$ to high-frequency $\mathbf{u}_K$. Therefore, leveraging the refined position representations to reconstruct the relative distance can force the encoder to learn diverse frequency information. We find that this design is quite useful in the heterophilic node classification task, where high-frequency information dominates the classification performance. Experiments can be seen in Section 5.1.

4.3 GraphPAE Decoder

So far, we have described the encoder of GraphPAE, which effectively learns both node and position representations. The decoder then utilizes these representations to recover the corrupted information. In practice, we find that corrupting both node features and positions simultaneously can negatively impact performance, as the recovery process in one path depends on the information from the other. To resolve this, we corrupt only one path's data during training, leaving the other path's data intact.

Feature Reconstruction. The node representations are learned by masking node features and preserving the node positions

$$\widetilde{\mathbf{X}}_i^{(L)}, \mathbf{P}_i^{(L)} = f_{\text{enc}}\left(\widetilde{\mathbf{X}}_i^{(0)}, \left\{\widetilde{\mathbf{X}}_j^{(0)}\right\}_{j \in \mathcal{N}_i}, \mathbf{P}_i^{(0)}\right), \tag{11}$$

To reconstruct the original node features, we apply a feature decoder f_{dec}^X to map the node representations back to the feature space. The reconstruction process is defined as

$$\mathbf{X}_i' = f_{\text{dec}}^X\left(\widetilde{\mathbf{X}}_i^{(L)}\right), \tag{12}$$

where $\mathbf{X}_i'$ is the is the reconstructed feature of node v_i. We follow GraphMAE and use the scaled cosine error (SCE) as the loss function of feature reconstruction

$$\mathcal{L}_{\text{feat}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v_i \in \widetilde{\mathcal{V}}}\left(1 - \frac{\mathbf{X}_i^T\mathbf{X}_i'}{\|\mathbf{X}_i\| \cdot \|\mathbf{X}_i'\|}\right)^\gamma, \tag{13}$$

where $\gamma \geq 1$ is a hyperparameter.

Algorithm 1 GraphPAE.

1: **Input:** Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, masking ratio r, epochs T, and noise scale μ_p.
2: **Preprocess:** Compute top-K eigenvectors $\mathbf{U}$ and node distance matrix $\mathbf{P}$ with $\mathbf{U}$.
3: **Init:** Encoder f_{enc}, feature decoder f_{dec}^X, position decoder f_{dec}^P, and learnable token $\mathbf{x}_{[M]}$.
4: **for** $t = 1$ to T **do**
5:     Randomly select $r|\mathcal{V}|$ nodes from $\mathcal{V}$ to form $\widetilde{\mathcal{V}}$.
6:     Replace features in $\widetilde{\mathcal{V}}$ with $\mathbf{x}_{[M]}$ to obtain $\widetilde{\mathbf{X}}$.
7:     Add noise $\delta \sim \mathcal{U}(-\mu_p, \mu_p)$ to eigenvectors of nodes in $\widetilde{\mathcal{V}}$ for $\widetilde{\mathbf{U}}$; compute corrupted distances $\widetilde{\mathbf{P}}$ with $\widetilde{\mathbf{U}}$.
                                                  ▷ Data Corruption
8:     Encode $\widetilde{\mathbf{X}}$ with $\mathbf{P}$ via f_{enc} for node representations $\widetilde{\mathbf{X}}^{(L)}$.
9:     Encode $\widetilde{\mathbf{P}}$ with $\mathbf{X}$ via f_{enc} for positional encodings $\widetilde{\mathbf{P}}^{(L)}$.
                                                     ▷ Encoder
10:    Decode $\widetilde{\mathbf{X}}^{(L)}$ and $\widetilde{\mathbf{P}}^{(L)}$ via f_{dec}^X and f_{dec}^P to reconstruct features $\mathbf{X}'$ and distances $\mathbf{P}'$.
11:    Compute $\mathcal{L}_{\text{feat}}$ with $\mathbf{X}$ and $\mathbf{X}'$, and $\mathcal{L}_{\text{pos}}$ with $\mathbf{P}$ and $\mathbf{P}'$.
12:    $\mathcal{L} = \mathcal{L}_{\text{feat}} + \alpha \mathcal{L}_{\text{pos}}$.
                                                    ▷ Decoder
13:    Update $f_{\text{enc}}, f_{\text{dec}}^X$, and f_{dec}^P by minimizing $\mathcal{L}$.
14: **end for**
15: **return** Trained encoder f_{enc}.

Position Reconstruction. The position representations are learned by offsetting eigenvectors and preserving the original node features

$$\mathbf{X}_i^{(L)}, \widetilde{\mathbf{P}}_i^{(L)} = f_{\text{enc}}\left(\mathbf{X}_i^{(0)}, \left\{\mathbf{X}_j^{(0)}\right\}_{j \in \mathcal{N}_i}, \widetilde{\mathbf{P}}_i^{(0)}\right), \tag{14}$$

Similarly, a position decoder f_{dec}^P is used to recover the original pair-wise node distances

$$\mathbf{P}_{i,j}' = f_{\text{dec}}^P\left(\widetilde{\mathbf{P}}_{i,j}^{(L)}\right) \tag{15}$$

For position reconstruction loss, we adopt Huber loss [23], which can make smooth gradients for better convergence

$$\mathcal{L}_{\text{pos}}^{i,j} = \begin{cases} \frac{\left(\mathbf{P}_{i,j}' - \mathbf{P}_{i,j}\right)^2}{2}, & \text{if } |\mathbf{P}_{i,j}' - \mathbf{P}_{i,j}| < 1 \\ |\mathbf{P}_{i,j}' - \mathbf{P}_{i,j}| - \frac{1}{2}, & \text{otherwise} \end{cases}$$
$$\mathcal{L}_{\text{pos}} = \frac{1}{\sum_{v_i \in \widetilde{\mathcal{V}}} |\mathcal{N}_i|} \sum_{v_i \in \widetilde{\mathcal{V}}, j \in \mathcal{N}_i} \mathcal{L}_{\text{pos}}^{i,j} \tag{16}$$

The overall loss function is formulated as a weighted combination of the feature and position reconstruction losses

$$\mathcal{L} = \mathcal{L}_{\text{feat}} + \alpha \mathcal{L}_{\text{pos}} \tag{17}$$

where α is the hyperparameter. The pseudocode of GraphPAE is presented in Algorithm 1.

5 Experiments

In this section, we conduct extensive experiments, including node classification, graph prediction, and transfer learning on large-scale molecule graphs to verify the effectiveness of GraphPAE. Moreover, we perform ablation studies on position reconstruction and dual-path design. Finally, we analyze the influence of the loss of weight and the number of eigenvectors.

5.1 Node Classification

Dataset. We evaluate the performance of GraphPAE on 6 representative heterophilic graphs: BlogCatalog [35], Chameleon, Squirrel, Actor [40], arXiv-year [19], and Penn94 [52]. Specifically, arXiv-year and Penn94 are large-scale graphs (> 40,000) to evaluate the scalability of the methods. As these datasets place greater emphasis on high-frequency information, they are well suited to evaluate GraphPAE's ability to capture diverse frequency components.

Baselines and Settings. We benchmark GraphPAE against a wide range of graph self-supervised baselines, which can be roughly divided into: **(1) contrastive learning,** *i.e.*, DGI [55], BGRL [51], MV-GRL [15], CCA-SSG [75], and Sp^{2}GCL [1]. **(2) graph autoencoders,** *i.e.*, VGAE [26], GraphMAE [18], GraphMAE2 [17], MaskGAE [28], S2GAE [50], and AUG-MAE [59]. We use GAT with 4 heads and 1024 hidden units as the encoder for all methods, and the number of layers is searched in the range of {2, 3}. Moreover, we adopt two-layer MLPs with ReLU activation as both feature and position decoders for efficiency and scalability. In the evaluation protocol, we freeze the encoder and generate node representations. The node representations are then input into a linear classifier for training with labeled data and inference for node classification. For all methods, we use the Adam optimizer and run 10 times on each graph.

Results. As shown in Table 2, GraphPAE consistently outperforms all baseline methods across all datasets, demonstrating the effectiveness of our framework. We highlight two important observations as follows. (1) GraphPAE surpasses both spatial- and spectral-based contrastive learning methods on most datasets, indicating its ability to effectively encode both spatial and spectral patterns through the joint reconstruction of features and positions. (2) GraphPAE also consistently outperforms existing GAEs, including those focused on feature reconstruction, structure reconstruction, and hybrid strategies. This superior performance is mainly attributed to two key factors. First, the incorporation of positional encodings enhances the expressivity of node representations. Second, reconstructing positions encourages the GAE to capture diverse frequency information, leading to high-quality graph representations.

5.2 Graph Prediction

Datasets. For graph-level tasks, we evaluate GraphPAE on 7 OGB datasets [19], including 3 graph regression tasks and 4 graph classification tasks. For all datasets, we use public splits for a fair comparison. We use MSE and ROC-AUC as the evaluation metrics for regression and classification, respectively.

Baselines and Settings. We select 5 **contrastive learning** methods, *i.e.*, InfoGraph [49], GraphCL [71], MVGRL [15], JOAO [70], and Sp^{2}GCL [1], and 4 **graph autocoders**, *i.e.*, GraphMAE [18], GraphMAE2 [17], StructMAE [32], and AUG-MAE [59]. We use a two-layer GatedGCN as the encoder and set the hidden dimension $d_h = 300$. We adopt two-layer MLPs as feature and position decoders. For evaluation, we freeze the encoder to output node representations and input them into pooling functions for graph representations. Similarly, we input the graph representations into

Table 2: Node classification results of different graph self-supervised learning, mean accuracy (%) ± standard deviation. Bold indicates the best performance and underline means the runner-up.

Dataset	Small Graphs				Large Graphs	
	BlogCatalog	Chameleon	Squirrel	Actor	arXiv-year	Penn94
Supervised	80.52±2.10	80.02±0.87	71.91±1.03	33.93±2.47	46.02±0.26	81.53±0.55
DGI	72.07±0.16	43.83±0.14	34.56±0.10	27.98±0.09	-	-
BGRL	79.74±0.46	61.24±1.07	43.24±0.52	26.61±0.57	<u>41.43±0.04</u>	63.31±0.49
MVGRL	63.24±0.94	73.19±0.42	60.09±0.44	34.64±0.20	-	-
CCA-SSG	74.00±0.28	75.00±0.75	61.58±1.98	27.79±0.58	40.78±0.01	62.63±0.20
Sp^2GCL	72.73±0.46	78.88±1.04	62.61±0.87	<u>34.70±0.92</u>	39.09±0.02	68.80±0.45
VGAE	60.47±1.84	62.32±1.90	42.50±1.35	31.57±0.75	36.39±0.21	55.31±0.28
GraphMAE	79.90±1.13	<u>79.50±0.57</u>	61.13±0.60	32.15±1.33	40.30±0.04	67.97±0.21
GraphMAE2	77.34±0.12	79.13±0.19	<u>70.27±0.88</u>	34.48±0.26	38.97±0.03	67.86±0.42
MaskGAE	73.10±0.08	74.50±0.87	68.53±0.44	33.44±0.34	40.59±0.04	63.84±0.03
S2GAE	75.76±0.43	60.24±0.37	68.60±0.56	26.17±0.38	40.32±0.12	<u>70.24±0.09</u>
AUG-MAE	<u>82.03±0.69</u>	70.10±1.88	62.57±0.67	33.42±0.38	37.10±0.13	69.90±0.43
GraphPAE	**85.76±1.22**	**80.51±1.25**	**72.05±1.40**	**38.55±1.35**	**41.85±0.04**	**71.79±0.37**

Table 3: Graph regression and classification results of different graph self-supervised learning on OGB datasets. Bold indicates the best performance and underline means the runner-up. ↓ means lower the better and ↑ means higher the better.

Task	Regression (Metric: RMSE ↓)			Classification (Metric: ROC-AUC% ↑)			
Dataset	molesol	molipo	molfreesolv	molbace	molbbbp	molclintox	moltocx21
Supervised	1.173±0.057	0.757±0.018	2.755±0.349	80.42±0.96	68.17±1.48	88.14±2.51	74.91±0.51
InfoGraph	1.344±0.178	1.005±0.023	10.005±8.147	73.64±3.64	66.33±2.79	64.50±5.32	69.74±0.57
GraphCL	1.272±0.089	0.910±0.016	7.679±2.748	73.32±2.70	68.22±2.19	74.92±4.42	72.40±1.07
MVGRL	1.433±0.145	0.962±0.036	9.024±1.982	74.88±1.43	67.24±3.19	73.84±2.75	70.48±0.83
JOAO	1.285±0.121	0.865±0.032	5.131±0.782	74.43±1.94	67.62±1.29	71.28±4.12	71.38±0.92
Sp^2GCL	1.235±0.119	<u>0.835±0.026</u>	4.144±0.573	78.76±1.43	**68.72±1.53**	80.88±3.86	73.06±0.75
GraphMAE	<u>1.050±0.034</u>	0.850±0.022	2.740±0.233	79.14±1.31	66.55±1.78	80.56±5.55	73.84±0.58
GraphMAE2	1.225±0.081	0.885±0.019	2.913±0.293	<u>80.74±1.53</u>	67.67±1.44	75.75±3.65	72.93±0.69
StructMAE	1.499±0.043	1.089±0.002	2.568±0.262	77.75±0.42	65.66±1.16	79.42±4.56	71.13±0.61
AUG-MAE	1.248±0.026	0.917±0.013	<u>2.395±0.158</u>	78.54±2.49	67.05±0.63	<u>82.66±1.98</u>	<u>74.33±0.07</u>
GraphPAE	**1.015±0.045**	**0.810±0.018**	**2.058±0.188**	**81.11±1.24**	<u>68.56±0.71</u>	**82.69±3.39**	**74.46±0.54**

a linear classifier to evaluate the performance for downstream tasks. We use Adam optimizer and report the metrics with mean results and standard deviation of 10 seeds.

Results. Table 3 summarizes the results of graph-level tasks. Graph-PAE consistently achieves superior performance across both regression and classification benchmarks, highlighting its ability to learn high-quality graph representations. Notably, GraphPAE shows competitive performance against contrastive learning methods and even exhibits notable improvements compared against the spectral-based method Sp^2GCL on molfreesolv, molbace, and molclintox. In addition, GraphPAE also outperforms other GAEs across most datasets. We attribute these gains to the proposed position corruption-reconstruction strategy, which improves the encoder's ability to identify crucial substructures for downstream tasks.

5.3 Transfer Learning

Settings. We conduct transfer learning experiments on molecule property prediction to evaluate the generalization ability of Graph-PAE. Specifically, we follow the setting of [34], which first pre-trains the encoder on 2 million molecules sampled from ZINC15 [48], and then fine-tunes on QM9 [64] to predict the quantum chemistry properties. We compare GraphPAE with the state-of-the-art molecule graph pre-training model, *i.e.*, GraphCL [71], GraphMAE [18], Mole-BERT [65], and SimSGT [34]. We use a five-layer GatedGCN with 300 hidden units. Upon finishing pre-training, a two-layer MLP is attached after the graph representations for property prediction. During the fine-tuning protocol, the encoder and attached MLP are trained together on labeled data of downstream tasks. Consistent with [34], we divided QM9 into train/validation/test sets with

Table 4: Quantum chemistry property results of transfer learning on QM9. The best and runner-up results are highlighted with bold and underline, respectively.

Target	μ	α	ϵ_{homo}	ϵ_{lumo}	Δ_ϵ	R^2	ZPVE	U_0	U	H	G	C_v
Unit	D	a_0^3	10^{-2}meV	10^{-2}meV	10^{-2}meV	a_0^2	10^{-2}meV	meV	meV	meV	meV	cal/mol/K
GraphCL	1.035	2.321	2.030	3.667	4.523	40.725	2.063	2.461	1.745	1.734	1.751	1.747
GraphMAE	1.030	2.924	2.407	6.373	4.813	41.955	4.623	1.411	2.207	2.208	2.207	2.200
Mole-BERT	1.031	1.918	1.477	4.127	4.240	44.374	2.190	2.532	2.509	2.511	2.516	2.508
SimSGT	1.064	2.413	2.837	4.227	4.107	40.504	2.127	1.948	2.420	2.416	2.416	2.410
GraphPAE	**0.703**	**0.879**	**1.199**	**2.141**	**2.289**	**36.480**	**0.502**	**0.510**	**0.639**	**0.639**	**0.641**	**0.643**

Table 5: Ablation studies of position reconstruction and framework design on node- and graph-level tasks. *Exp No.*: the number of different experimental settings. *Corrupt Info.*: corrupted information during training. *Recon Info.*: information to be reconstructed during training. Bold indicates the best performance.

Exp	Corrupt Info.		Recon Info.		Dual-Path	Node-level		Graph-level		
No.	Feature	Position	Feature	Position		Blog (↑)	Squirrel (↑)	Bace (↑)	Bbbp (↑)	Freesolv (↓)
a	✓	✓	✓			82.8±1.7	66.4±1.6	78.4±1.2	66.4±1.7	2.79±0.40
b	✓	✓	✓	✓		83.5±1.0	68.5±0.9	78.9±2.1	66.8±0.6	2.44±0.36
c	✓		✓			84.6±1.6	71.3±0.9	79.4±3.4	67.7±0.9	2.20±0.14
d	✓	✓	✓	✓	✓	**85.8±1.2**	**72.1±1.4**	**81.1±1.2**	**68.6±0.7**	**2.06±0.19**

80%/10%/10% by scaffold splits. We use Adam optimizer and run it 5 times to report average MAE and standard deviation.

Results. From Table 4, we observe that GraphPAE demonstrates robust generalization across all prediction targets compared to state-of-the-art models. Among the baselines, Mole-BERT and SimSGT are specially designed for molecular graph pretraining, incorporating customized reconstruction objectives tailored to the characteristics of molecular graphs. Specifically, Mole-BERT pretrains a discrete codebook within the subgraph of nodes, inherently capturing both features and local structural patterns. During reconstruction, the model replaces the raw features with codebook entries as predictive targets. SimSGT employs a tokenizer to encode feature and local structure information, using tokenized output as reconstruction targets. Despite these tailored strategies, GraphPAE consistently outperforms these models. The superiority is largely attributed to the effective positional encodings obtained by position corruption-reconstruction, enabling the graph representations to capture crucial substructures for prediction targets such as cycles.

5.4 Ablation Studies

To verify the effectiveness of important designs of GraphPAE, we conduct extensive ablation experiments. We select 2 datasets for node-level tasks and 3 datasets for graph-level tasks.

Effectiveness of Position Reconstruction. To achieve position corruption-reconstruction, GraphPAE inevitably introduces positional encodings into the encoder. Firstly, we verify that the superior performance is attributed not only to the introduction of positional encodings but also to the position reconstruction. We conducted two sets of comparative experiments in Table 5: one comparing Exp a with Exp b, and another comparing Exp c with Exp d. We

have the following summaries: (1) In Exp a, both features and positions are corrupted, but only the features are reconstructed. In Exp b, both features and positions are reconstructed. We observe that Exp b performs better than Exp a, demonstrating the effectiveness of position reconstruction. (2) In Exp c, we remove both position corruption and reconstruction from GraphPAE. Comparing Exp d against Exp c, we find that the encoder incorporating position reconstruction consistently improves performance compared to feature reconstruction alone.

Effectiveness of Dual-Path Reconstruction. Comparing Exp b and Exp c in Table 5, we observe that position reconstruction without the dual-path architecture fails to improve performance and may even lead to degradation. Specifically, in Exp b, both masked features and noisy positions are reconstructed directly from node representations. However, recovering relative distances from node representations generated with corrupted positions is more difficult than recovering them from refined distance encoding directly. Rather than enhancing node representations or capturing diverse frequency information, the corrupted positional inputs introduce additional noise during training. As a result, applying position corruption-reconstruction without appropriate design hinders the model's ability to learn meaningful structural patterns. Based on these observations, we propose a dual-path architecture that disentangles feature and relative distance encodings for reconstruction. Experimental results from Exp d demonstrate that this strategy consistently achieves the best performance across datasets, validating the effectiveness of our dual-path encoder design.

Improvement Attributed to $\mathcal{L}_{pos}$ and PEs. Since integrating positional encodings also enhances representation learning, it remains unclear how much of the performance gain in GraphPAE stems

Table 6: Ablation studies of $\mathcal{L}_{pos}$ and positional encodings.

Methods	Bace (↑)	Bbbp (↑)	Freesolv (↓)
GraphPAE	**81.11±1.24**	**68.56±0.71**	**2.058±0.19**
w/o $\mathcal{L}_{pos}$	79.40±3.45	67.74±0.92	2.204±0.14
w/o $\mathcal{L}_{pos}$ & PEs	79.14±1.31	66.55±1.78	2.740±0.23

(a) Molbbbp

(b) BlogCatalog

(c) Squirrel

(d) Chameleon
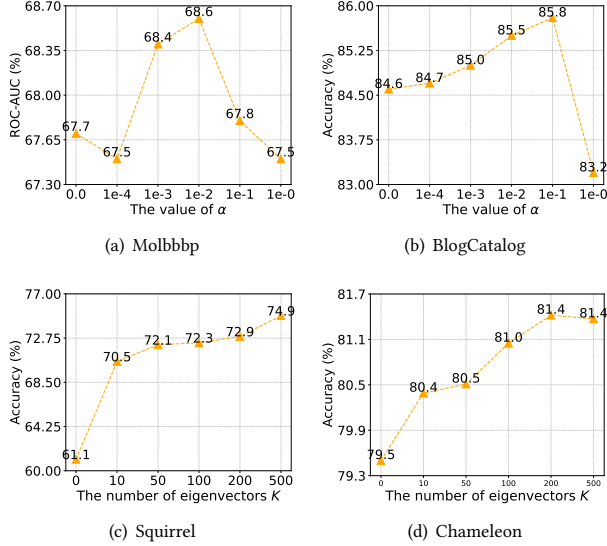
Figure 3: Influence of the loss weight α and the number of eigenvectors K.

from the position corruption-reconstruction mechanism, and how much from the positional information itself. Therefore, we conduct ablation studies on both $\mathcal{L}_{pos}$ and positional encodings. Specifically, we remove $\mathcal{L}_{pos}$ from Equation 17 while keeping PEs, to evaluate the effect of position reconstruction. Then, we remove both $\mathcal{L}_{pos}$ and PEs to isolate the impact of positional information. The results are reported in Table 6. Notably, even with positional encodings retained, removing $\mathcal{L}_{pos}$ consistently leads to a noticeable performance decline across all datasets, highlighting the importance of the proposed position corruption-reconstruction mechanism in enhancing GAEs.

5.5 Parameters Analysis

We conduct additional parameter analysis of the loss weight α and the number of eigenvectors K.

Influence of the loss weight α in GraphPAE. Figure 3(a) and 3(b) present the hyperparameter analysis of α to further examine the influence of the positional reconstruction loss $\mathcal{L}_{pos}$. We summarize the key observations as follows: (1) As α increases within a certain threshold, the performance improves progressively, indicating that position reconstruction contributes positively to the quality of learned representations. (2) However, excessively large values of α lead to performance degeneration. This suggests assigning a high weight to position reconstruction may cause the encoder to

overemphasize positional information at the expense of overall representation learning. (3) While the optimal value of α varies slightly across datasets, we observe that the best-performing range typically lies within {1e-3, 1e-2, 1e-1}, making it easy to search in practice. Additional hyperparameter details are provided in Appendix B.2.

Influence of the number of eigenvectors K in GraphPAE. We further analyze the effect of the number of eigenvectors K on performance using the Squirrel and Chameleon datasets, as shown in Figure 3(c) and 3(d). As K increases, performance generally improves due to the model's ability to capture a wider range of frequency information. However, once K surpasses a certain threshold, the performance gains become marginal. Therefore, to balance effectiveness and computational efficiency, we adopt a moderate value of K, typically in the range of [50, 100], which provides a good trade-off in practice.

6 Conclusion

In this paper, we propose GraphPAE, a graph autoencoder that reconstructs both node features and relative distance. To enhance models' expressivity to distinguish intricate patterns and the ability to integrate various frequency information, GraphPAE introduces position corruption and recovery into GAEs and designs a dual-path reconstruction strategy. Extensive experiments, including node classification, graph prediction, and transfer learning, demonstrate the superiority of our GraphPAE.

Acknowledgments

References

[1] Deyu Bo, Yuan Fang, Yang Liu, and Chuan Shi. 2024. Graph contrastive learning with stable and scalable spectral encoding. *Advances in Neural Information Processing Systems* 36 (2024).

[2] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. 2023. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint arXiv:2303.01028* (2023).

[3] Deyu Bo, Xiao Wang, Yang Liu, Yuan Fang, Yawen Li, and Chuan Shi. 2023. A survey on spectral graph neural networks. *arXiv preprint arXiv:2302.05631* (2023).

[4] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 3950–3957.

[5] Pietro Bongini, Monica Bianchini, and Franco Scarselli. 2021. Molecular generative graph neural networks for drug discovery. *Neurocomputing* 450 (2021), 242–252.

[6] Semih Cantürk, Renming Liu, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. [n. d.]. Graph Positional and Structural Encoder. In *Forty-first International Conference on Machine Learning*.

[7] Peter J Carrington, John Scott, and Stanley Wasserman. 2005. *Models and methods in social network analysis*. Vol. 28. Cambridge university press.

[8] Jiashun Cheng, Man Li, Jia Li, and Fugee Tsung. 2023. Wiener graph deconvolutional network improves graph self-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 7131–7139.

[9] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020).

[10] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *Journal of Machine Learning Research* 24, 43 (2023), 1–48.

[11] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875* (2021).

[12] Vijay Prakash Dwivedi, Ladislav Rampášek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. 2022. Long range graph benchmark. *Advances in Neural Information Processing Systems* 35 (2022), 22326–22340.

[13] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph

neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[14] Zhangyang Gao, Daize Dong, Cheng Tan, Jun Xia, Bozhen Hu, and Stan Z Li. [n. d.]. A Graph is Worth *K* Words: Euclideanizing Graph using Pure Transformer. In *Forty-first International Conference on Machine Learning*.

[15] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*. PMLR, 4116–4126.

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. 2022. Masked Autoencoders Are Scalable Vision Learners. In *CVPR*. IEEE, 15979–15988.

[17] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*. 737–746.

[18] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 594–604.

[19] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. [n. d.]. STRATEGIES FOR PRE-TRAINING GRAPH NEURAL NETWORKS. ([n. d.]).

[21] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1857–1867.

[22] Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. 2023. On the stability of expressive positional encodings for graph neural networks. *arXiv preprint arXiv:2310.02579* (2023).

[23] Peter J Huber. [n. d.]. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 492–518.

[24] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of cheminformatics* 13 (2021), 1–23.

[25] Wei Ju, Yifan Wang, Yifang Qin, Zhengyang Mao, Zhiping Xiao, Junyu Luo, Junwei Yang, Yiyang Gu, Dongjie Wang, Qingqing Long, et al. 2024. Towards Graph Contrastive Learning: A Survey and Beyond. *arXiv preprint arXiv:2405.11868* (2024).

[26] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[27] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems* 34 (2021), 21618–21629.

[28] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. 2023. What's Behind the Mask: Understanding Masked Graph Modeling for Graph Autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1268–1279.

[29] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems* 33 (2020), 4465–4478.

[30] Xiang Li, Tiandi Ye, Caihua Shan, Dongsheng Li, and Ming Gao. 2023. Seegera: Self-supervised semi-implicit graph variational auto-encoders with masking. In *Proceedings of the ACM web conference 2023*. 143–153.

[31] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. 2022. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013* (2022).

[32] Chuang Liu, Yuyao Wang, Yibing Zhan, Xueqi Ma, Dapeng Tao, Jia Wu, and Wenbin Hu. 2024. Where to mask: structure-guided masking for graph masked autoencoders. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence* (Jeju, Korea) *(IJCAI '24)*. Article 241, 9 pages. https://doi.org/10.24963/ijcai.2024/241

[33] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE transactions on knowledge and data engineering* 35, 6 (2022), 5879–5900.

[34] Zhiyuan Liu, Yaorui Shi, An Zhang, Enzhi Zhang, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. 2024. Rethinking tokenizer and decoder in masked graph modeling for molecules. *Advances in Neural Information Processing Systems* 36 (2024).

[35] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-embedding attributed networks. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 393–401.

[36] Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901* (2020).

[37] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667* (2021).

[38] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407* (2018).

[39] Jiwoong Park, Minsik Lee, Hyung Jin Chang, Kyuewang Lee, and Jin Young Choi. 2019. Symmetric graph convolutional autoencoder for unsupervised graph representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6519–6528.

[40] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[41] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems* 35 (2022), 14501–14515.

[42] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[43] Amin Salehi and Hasan Davulcu. 2019. Graph attention auto-encoders. *arXiv preprint arXiv:1905.10715* (2019).

[44] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. 2024. A survey of graph neural networks for social recommender systems. *Comput. Surveys* 56, 10 (2024), 1–34.

[45] Yucheng Shi, Yushun Dong, Qiaoyu Tan, Jundong Li, and Ninghao Liu. 2023. Gigamae: Generalizable graph masked autoencoder via collaborative latent space reconstruction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2259–2269.

[46] Vighnesh Shiv and Chris Quirk. 2019. Novel positional encodings to enable tree-based transformers. *Advances in neural information processing systems* 32 (2019).

[47] Shashank Sheshar Singh, Samya Muhuri, Shivansh Mishra, Divya Srivastava, Harish Kumar Shakya, and Neeraj Kumar. 2024. Social Network Analysis: A Survey on Process, Tools, and Application. *Comput. Surveys* 56, 8 (2024), 1–39.

[48] Teague Sterling and John J Irwin. 2015. ZINC 15–ligand discovery for everyone. *Journal of chemical information and modeling* 55, 11 (2015), 2324–2337.

[49] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000* (2019).

[50] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. 2023. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the sixteenth ACM international conference on web search and data mining*. 787–795.

[51] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. 2021. Bootstrapped representation learning on graphs. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*.

[52] Amanda L Traud, Peter J Mucha, and Mason A Porter. 2012. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180.

[53] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[55] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341* (2018).

[56] Lilapati Waikhom and Ripon Patgiri. 2023. A survey of graph neural networks in various learning paradigms: methods, applications, and challenges. *Artificial Intelligence Review* 56, 7 (2023), 6295–6364.

[57] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. [n. d.]. Equivariant and Stable Positional Encoding for More Powerful Graph Neural Networks. In *International Conference on Learning Representations*.

[58] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. 2022. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199* (2022).

[59] Liang Wang, Xiang Tao, Qiang Liu, and Shu Wu. 2024. Rethinking Graph Masked Autoencoders through Alignment and Uniformity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15528–15536.

[60] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*. 1243–1253.

[61] Asiri Wijesinghe, Hao Zhu, and Piotr Koniusz. [n. d.]. Graph Self-Supervised Learning with Learnable Structural and Positional Encodings. In *THE WEB CONFERENCE 2025*.

[62] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 4216–4235.

[63] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. 2021. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems* 34 (2021), 13266–13279.

[64] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.

[65] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z Li. 2023. Mole-bert: Rethinking pre-training graph neural networks for molecules. (2023).

[66] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence* 45, 2 (2022), 2412–2429.

[67] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. 2019. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry* 63, 16 (2019), 8749–8760.

[68] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[69] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in neural information processing systems* 34 (2021), 28877–28888.

[70] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *International Conference on Machine Learning*. PMLR, 12121–12132.

[71] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[72] Xingtong Yu, Zhou Chang, Kuai Zhongwei, Zhang Xinming, and Fang Yuan. 2025. GCoT: Chain-of-Thought Prompt Learning for Graphs. In *the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*.

[73] Xingtong Yu, Yuan Fang, Zemin Liu, Yuxia Wu, Zhihao Wen, Jianyuan Bo, Xinming Zhang, and Steven CH Hoi. 2024. Few-Shot Learning on Graphs: from Meta-learning to Pre-training and Prompting. *arXiv preprint arXiv:2402.01440* (2024).

[74] Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. 2024. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs. *IEEE TKDE* (2024).

[75] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 76–89.

[76] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. 2021. Motif-based graph self-supervised learning for molecular property prediction. *Advances in Neural Information Processing Systems* 34 (2021), 15870–15882.

[77] Ziwen Zhao, Yuhua Li, Yixiong Zou, Jiliang Tang, and Ruixuan Li. 2024. Masked Graph Autoencoder with Non-discrete Bandwidths. In *Proceedings of the ACM on Web Conference 2024*. 377–388.

[78] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. 2023. Uni-Mol: A Universal 3D Molecular Representation Learning Framework. In *ICLR*. OpenReview.net.

[79] Yun Zhu, Haizhou Shi, Xiaotang Wang, Yongchao Liu, Yaoke Wang, Boci Peng, Chuntao Hong, and Siliang Tang. 2025. Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs. In *Proceedings of the ACM on Web Conference 2025*. 2183–2197.

Appendices

A Spectral Analysis of Masking Strategies

Masking Strategies for Node Features and Edges. We conduct the experiments in the Squirrel datasets with a 20% masking ratio. For node feature masking, given the node feature matrix $\mathbf{X}$, we randomly mask 20% of the node in the dataset and set the feature vector to zero. The remaining 80% of nodes retain their original feature. The corrupted feature matrix is defined as $\widetilde{\mathbf{X}}$. For edge masking, given the original graph structure $\mathbf{A}$, we randomly remove 20% of the edges from the graph while preserving the node features. This process results in a structurally corrupted graph with a new adjacency matrix $\widetilde{\mathbf{A}}$.

Table 7: Statistics of node classification datasets.

Datasets	Nodes	Edges	Features	Classes	Train/Valid/Test
BlogCatalog	5,196	343,486	8,189	6	120/1,000/1,000
Chameleon	2,277	62,792	2,325	5	1,092/729/456
Squirrel	5,201	396,846	2,089	5	2,496/1,664/1,041
Actor	7,600	53,411	932	5	3,648/2,432/1,520
arXiv-year	169,343	2,315,598	128	5	84671/42335/42337
Penn94	41,554	2,724,458	4,814	2	19,407/9,703/9,705

Table 8: Statistics of graph prediction datasets.

Dataset	Graphs	Avg. Nodes	Avg. Edges	Classes	Task	Metric
molesol	1,128	13.3	13.7	1	Regression	RMSE
mollipo	4,200	27.0	29.5	1	Regression	RMSE
molfreesolv	642	8.7	8.4	1	Regression	RMSE
molbace	1,513	34.1	36.9	1	Binary Class.	ROC-AUC
molbbbp	2,039	24.1	26.0	1	Binary Class.	ROC-AUC
molcintox	1,477	26.2	27.9	2	Binary Class.	ROC-AUC
moltox21	7,831	18.6	19.3	12	Binary Class.	ROC-AUC
ZINC15	2,000,000	26.62	57.72	-	Pre-Training	-
QM9	1,177,631	8.80	18.81	12 Targets	Finetuning	MAE

Computation of Frequency Magnitudes. First, we calculate Laplacian matrix by:

$$\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{18}$$

The eigenvectors of $\mathbf{L}$ are decomposed as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$, where $\mathbf{\Lambda} = \text{diag}(\{\lambda_i\}_{i=1}^N)$, $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N] \in \mathbb{R}^{N \times N}$, and the eigenvalues are ordered as $0 \le \lambda_1 \le \cdots \le \lambda_N \le 2$. λ_i reflects the frequency magnitude of corresponding eigenvector $\mathbf{u}_i$ over the graph. We transform node features into the spectral domain by

$$\mathbf{X}^s = \mathbf{U}^{\top}\mathbf{X}, \quad \mathbf{X}^s \in \mathbb{R}^{N \times d}. \tag{19}$$

In particular, the i-th row of $\mathbf{X}^s$ corresponds to the frequency magnitude at eigenvalue λ_i. With d-dimensional $\mathbf{X}^s$, we first calculate the mean magnitude by:

$$\bar{\mathbf{X}}^s = \frac{1}{d}\sum_{i=1}^{d}\mathbf{X}^s_{:,i}, \quad \bar{\mathbf{X}}^s \in \mathbb{R}^N \tag{20}$$

For a given frequency band $[f_1, f_2]$, we extract the set of indices I corresponding to eigenvalues within this range:

$$I = \{i \mid f_1 \le \lambda_i \le f_2\} \tag{21}$$

The average frequency magnitude within this band is computed as:

$$\bar{\mathbf{X}}^s_{[f_1,f_2]} = \frac{1}{|I|}\sum_{i \in I}\bar{\mathbf{X}}^s[i], \quad \bar{\mathbf{X}}^s_{[f_1,f_2]} \in \mathbb{R} \tag{22}$$

For feature masking case, we calculate $\mathbf{L}$ by substituting $\mathbf{A}$ into Equation 18. Then we decompose $\mathbf{L}$ and get $\mathbf{U}$. By substituting $\mathbf{U}$, $\mathbf{X}$, and $\widetilde{\mathbf{X}}$ in Equation (19)-(22), we can get the frequency magnitude of original and corrupted graphs. The comparison of differences is shown in Figure 1(a). For edge masking case, we calculate $\mathbf{L}$ and $\widetilde{\mathbf{L}}$ by substituting $\mathbf{A}$ and $\widetilde{\mathbf{A}}$ into Equation 18. Then we decompose $\mathbf{L}$ and $\widetilde{\mathbf{L}}$, and get $\mathbf{U}$ and $\widetilde{\mathbf{U}}$. By substituting $\mathbf{U}$, $\widetilde{\mathbf{U}}$, and $\mathbf{X}$ in Equation (19) - (22), we can also get the frequency magnitude of original and corrupted graphs as shown in Figure 1(b).

Table 9: Hyperparameters of node classification.

Dataset	lr	wd	r	α	dp	dp_{edge}	K
BlogCatalog	0.001	0.0	0.25	0.1	0.6	0.0	50
Chameleon	0.001	0.0	0.25	0.01	0.0	0.0	50
Squirrel	0.001	0.0	0.5	0.001	0.6	0.0	50
Actor	0.0005	0.0	0.25	0.01	0.0	0.0	50
arXiv-year	0.001	0.0	0.5	0.01	0.0	0.0	100
Penn94	0.001	0.0	0.25	0.001	0.0	0.0	200

Table 10: Hyperparameters of graph prediction.

Dataset	pooling	epoch	lr	wd	r	α	dp	dp_{edge}	K
molesol	sum	20	0.0005	0.0	0.75	0.1	0.6	0.5	8
molipo	sum	20	0.0005	0.0001	0.25	0.001	0.6	0.0	30
molfreesolv	sum	100	0.0001	0.0	0.5	0.1	0.5	0.5	15
molbace	mean	100	0.001	0.0	0.75	0.1	0.5	0.5	30
molbbbp	mean	20	0.001	0.0	0.5	0.01	0.6	0.6	30
molclintox	mean	20	0.0001	0.0	0.25	0.01	0.6	0.0	30
moltocx21	mean	20	0.0001	0.0	0.25	0.1	0.0	0.6	8

Table 11: Additional experiment results on homophilic and large-scale graphs.

Methods	Facebook	Wiki	ogbg-products
BGRL	89.71±0.35	79.02±0.13	78.59±0.02
CCA-SSG	89.45±0.60	78.85±0.32	75.27±0.05
GraphMAE	89.54±0.36	78.94±0.48	78.89±0.01
GraphMAE2	88.49±0.43	78.84±0.44	**81.59±0.02**
GraphPAE	**91.46±0.23**	**79.32±0.29**	79.10±0.02

B Experimental Details

B.1 Statistics of Datasets

We benchmark GraphPAE across various tasks, including node classification, graph prediction, and transfer learning. Specifically, we conduct experiments on six node classification datasets and seven graph-level prediction datasets and select ZINC15 and QM9 for transfer learning. For datasets in node classification and graph prediction, we utilize public data splits. In transfer learning, we follow [34] and divide QM9 into train/valid/test sets with 80%/10%/10%

by scaffold split. Detailed statistics of the node- and graph-level datasets are presented in Table 7 and Table 8, respectively.

B.2 Hyperparameters

Hyperparameter details of node classification and graph prediction are reported in Table 9 and Table 10. r denotes the mask ratio and α is the loss weight of $\mathcal{L}_{pos}$. dp and dp_{edge} represent the dropout of node features and edge features, respectively. For transfer learning, we pre-train GraphPAE on 2 million molecules sampled from ZINC15 with 100 epochs. The mask ratio is set to 0.35, and the loss weight for $\mathcal{L}_{pos}$ is 0.01. Both node and edge dropout are set to 0.0, and the number of eigenvectors for positional encoding is 6. During fine-tuning, the encoder and the attached MLP are jointly trained on the QM9 dataset using an initial learning rate of 0.001 and no weight decay. We apply a dropout rate of 0.1 to both node features and edge features within GNN layers.

B.3 Additional Experiment Results.

We further evaluate GraphPAE on homophilic graphs, including Facebook [42] and Wiki [36], as well as a large-scale graph ogbn-products [19]. Notably, ogbn-products contains 2,449,029 nodes and 61,859,140 edges, making it a suitable benchmark to demonstrate the scalability of GraphPAE. The experimental results are summarized in Table 11, and we make the following observations: (1) GraphPAE consistently achieves competitive performance across these different benchmarks. (2) The performance gains on homophilic graphs are less pronounced compared to those on heterophilic graphs. We conjecture this is because, in homophilic graphs, a small portion of very low-frequency information is sufficient.

C Complexity Analysis

The eigen-decomposition complexity is $O(N^3)$. However, we only need to decompose the smallest K eigenvalues, which reduces complexity to $O(N^2 K)$. Additionally, the decomposition is performed once per graph, so its cost is amortized over the entire experiment.

During training, GraphPAE introduces an additional $O(\mathcal{E})$ complexity per layer for PE computation and an extra cost for relative node distance reconstruction. Since we only reconstruct corrupted node pairs, the complexity remains much lower than $O(\mathcal{E})$. However, this slight overhead enables the encoder to capture more diverse frequency information.