CENTRALITY-GUIDED PRE-TRAINING FOR GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised learning (SSL) has shown great potential in learning generalizable representations for graph-structured data. However, existing SSL-based graph pre-training methods largely focus on improving graph representations by learning the structure information based on disturbing or reconstructing graphs, which ignores an important issue: the importance of different nodes in the graph structure may vary. To fill this gap, we propose a Centrality-guided Graph Pre-training (CenPre) framework to integrate the distinct importance of nodes in graph structure into the corresponding representations of nodes based on the centrality in graph theory. In this way, the different roles played by different nodes can be effectively leveraged when learning graph structure. The proposed CenPre contains three modules for node representation pre-training and alignment. The node-level structure learning module fuses the fine-grained node importance into node representation based on degree centrality, allowing the aggregation of node representations with equal/similar importance. The graph-level structure learning module characterizes the importance between all nodes in the graph based on eigenvector centrality, enabling the exploitation of graph-level structure similarities/differences when learning node representation. Finally, a representation alignment module aligns the pre-trained node representation using the original one, essentially allowing graph representations to learn structural information without losing their original semantic information, thereby leading to better graph representations. Extensive experiments on a series of real-world datasets demonstrate that the proposed CenPre outperforms the state-of-the-art baselines in node classification and achieves better performance in link prediction and graph classification than the baseline models.

1 INTRODUCTION

Graph neural networks (GNNs) aim to model the structural information of the graph by neighborhood aggregation schemes, becoming increasingly popular in the field of graph representation for learning graph-structured data, such as knowledge graphs (Baek et al., 2020), social networks (Fan et al., 2019), point clouds (Shi & Rajkumar, 2020), and chemical analysis (De Cao & Kipf, 2018). Graph representation learning can produce low-dimensional vector representations for graph-structured data in many applications, including node/graph classification, link prediction, and graph generation (You et al., 2020; Wang et al., 2020; Zhang & Chen, 2018; Hou et al., 2019).

Recently, self-supervised learning on graph-structured data has shown great potential in learning generalizable, transferable, and robust representations from graph-structured data due to the advantage of not requiring annotated data (Hu* et al., 2020; You et al., 2020). Among many, graph predictive learning and graph contrastive learning (GCL) have become two mainstream paradigms in learning graph representation, where the mutual information between graph representations can be learned by leveraging the similarity/difference between augmented views (Hu* et al., 2020; Kim et al., 2022; You et al., 2020; 2021; Xia et al., 2022b). Despite promising progress made by existing graph representation learning methods, they are largely based on randomly perturbing graph structures or reconstructing graph structures from a graph level to learn graph structure information for graph pre-training, ignoring an important issue that the importance of different nodes in the graph structure may vary. This deficiency may result in the inability to exploit important characteristic information in the graph structure when learning graph representations.

To alleviate this issue, in this paper, we propose a novel graph pre-training framework to produce better graph representation that leverages graph structure in learning graph representations by exploiting the distinct importance of different nodes in graph structure based on the notion of centrality, dubbed Centrality-guided Graph Pre-training (CenPre). There are three modules in the proposed CenPre: 1) **node-level structure learning**, which aims to enhance the node representations with the node importance in graph structure based on the property of degree centrality; 2) **graph-level structure learning** module, which is designed to leverage the global importance based on the property of eigenvector centrality when learning node representation; and 3) **graph representation alignment** module aligns the structure-fused representation using the original one.

Node-level structure learning. Degree centrality defines the importance of a node based on the degree of that node. The higher the degree, the more crucial a node becomes in the graph (Zaki & Meira, 2014). Based on this property, we argue that the degree of a node is important for its representation, so its representation can be refined by degree. Therefore, we explore predictive learning to map each node representation into the corresponding degree. This essentially enables the model to learn statistical regularity between representation and graph structure, so as to perceive the importance of different nodes in the graph, thereby assigning similar representations to nodes with the same degree and distinguishing node representations with differences in importance.

Graph-level structure learning. Eigenvector centrality, or eigencentrality, measures a node's influence within a connected network by accounting for both direct and indirect connections (Zaki & Meira, 2014; Bonacich, 2007). This provides a comprehensive understanding of a node's role in the global structure. Building on this, we propose identifying the most influential neighbors of each node to obtain a global structural perspective, which can guide the refinement of node representations. To achieve this, we introduce a Contrastive Representation-Structure Pre-Training (CReSP) strategy that aligns node representations with the graph's structural pattern. Matrix decomposition is used to extract significant eigenvalues and eigenvectors, facilitating the identification of key neighbors in large, sparse graphs. Using cross-attention, the graph representation guides the structure matrix in determining node importance relative to the current node. Inspired by CLIP (Radford et al., 2021), CReSP refines this alignment by matching node representations with their structural patterns, maximizing the similarity between nodes with shared important neighbors, and enhancing graph-level representation by incorporating global structural information.

Graph representation alignment. After the graph structure pre-training, a graph representation alignment module is devised to align the pre-trained structure-fused graph representation using the original representation, allowing the graph representation to retain the original semantic information while learning the graph structure information.

Our main contributions are as follows:

- We are the first to explore node importance in learning graph structure and align the graph representation with graph structure in the pre-training process, aiming to produce better graph representations for downstream tasks.
- Based on the notion of centrality, a novel Centrality-guided Graph Pre-training (CenPre) framework is proposed to learn structure-fused graph representation from both local and global perspectives.
- We conduct a series of experiments on real-world graph-structured benchmark datasets to evaluate the effectiveness of our CenPre in learning graph representation. Experimental results show that our CenPre significantly outperforms baselines in n the tasks of node classification, link prediction, and graph classification.

2 Related Work

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) capture node dependencies through graph topology. The Graph Convolutional Network (GCN) (Kipf & Welling, 2017) aggregates local neighborhood features, excelling in node classification but struggling with global structures. Graph Attention Network (GAT) (Velickovic et al., 2018) improves feature aggregation via attention mechanisms, enhancing performance in heterogeneous graphs but increasing computational demands. Graph Isomorphism Network (GIN) (Xu et al., 2019) improves structure distinction but risks overfitting on small datasets, while GraphSAGE (Hamilton et al., 2017) scales well for large graphs using neighborhood sampling, though it may lose information in dense networks. Graph Transformers (Dwivedi & Bresson, 2020) capture long-range dependencies with self-attention, but are computationally intensive for large-scale graphs. Some other previous works extend GNNs by leveraging centrality to enhance structural understanding. For instance, Maurya et al., 2019 proposes a GNN framework for approximating betweenness centrality by leveraging constrained message passing and ranking loss to learn structural node importance efficiently. (Avelar et al., 2018) introduces a multitask GNN-based learning framework to approximate multiple centrality measures, enabling shared representations and accurate structural feature predictions.

2.2 Self-Supervised Learning on Graphs

Self-supervised learning has achieved promising performance in graph pre-training. Early works like DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) use random walks to capture local structures, while contrastive learning methods such as DGI (Veličković et al., 2019), InfoGraph (Sun et al., 2020), and GraphCL (You et al., 2020) maximize agreement between augmented views. MVGRL (Hassani & Khasahmadi, 2020) contrasts different graph views using diffusion, but these methods face challenges in selecting augmentations and handling negative samples. GCA (Zhu et al., 2021) adaptively retains critical structural features using centrality-guided augmentations to enhance representation learning. GGD (Zheng et al., 2022) eliminates contrastive similarity computation by discriminating augmented positive and negative node groups through a binary classification task. Further, avoid view generation and focus on intrinsic graph properties. BGRL (Thakoor et al., 2021) and CCA-SSG (Zhang et al., 2021) align representations without negative samples, offering more efficient alternatives. (Jin et al., 2020) combines tasks like clustering and node distance prediction to learn robust structural embeddings. (Hu et al., 2019) leverages centrality score ranking as a task to guide GNNs in capturing multi-perspective structural features from synthetic graphs for downstream applications. GBT (Bielak et al., 2021) proposes to incorporate multi-task objectives, including node proximity and subgraph features, to capture intrinsic graph properties without relying on augmentation. Graph autoencoders (GAE) (García-Durán & Niepert, 2017), VGAE (Kipf & Welling, 2016), and other variants reconstruct graph structure but often underperform in classification. GraphMAE (Hou et al., 2022) improves performance using masked feature reconstruction, bypassing augmentations, while GPT-GNN (Hu et al., 2020) introduces autoregressive graph generation for pre-training. Unlike the above approaches, we first explore the role of different nodes in graph structure learning by learning the importance of nodes in the graph based on degree centrality. Further, we learn the important neighborhood information based on Eigenvector centrality, aiming to aggregate the important neighboring information at the graph level when learning graph representation. In this way, it is possible to effectively leverage the different roles played by different nodes when learning graph structure. Our work leverages centrality metrics to guide graph pre-training, offering an efficient framework that integrates both local and global graph properties into the graph representation.

3 Preliminaries

Notations Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ represent an undirected graph, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes and $\mathcal{E}$ the set of edges. $X_v \in \mathbb{R}^{N \times d_v}$ and $X_e \in \mathbb{R}^{|\mathcal{E}| \times d_e}$ denote the node and edge feature matrices, respectively. The representation of a node v_i is h_i, and the graph-level representation is $H_{\mathcal{G}} = \{h_1, h_2, ..., h_n\}$.

Graph Neural Networks GNNs update the graph representation $H_{\mathcal{G}}$ by leveraging the graph topology. The representation h_i of node v_i at the k-th GNN layer is computed as:

$$h_i^{(k)} = f(h_i^{(k-1)}; \theta) = \mathrm{UPDATE}^{(k)}\big(h_i^{(k-1)}, \mathrm{AGGREGATE}^{(k)}(h_j^{(k-1)} : \forall v_j \in \mathcal{N}(v_i))\big) \quad (1)$$

where $f(\cdot; \theta)$ is the GNN encoder with trainable parameters θ, and $\mathrm{AGGREGATE}(\cdot)$ gathers information from neighbors $\mathcal{N}(v_i)$. $\mathrm{UPDATE}(\cdot)$ updates the node representation based on the aggregated information. After K iterations, the graph representation is obtained by pooling the node representations:

$$H_{\mathcal{G}} = \mathrm{POOL}\big(\{h_i^{(K)} : \forall v_i \in \mathcal{V}\}\big) \quad (2)$$

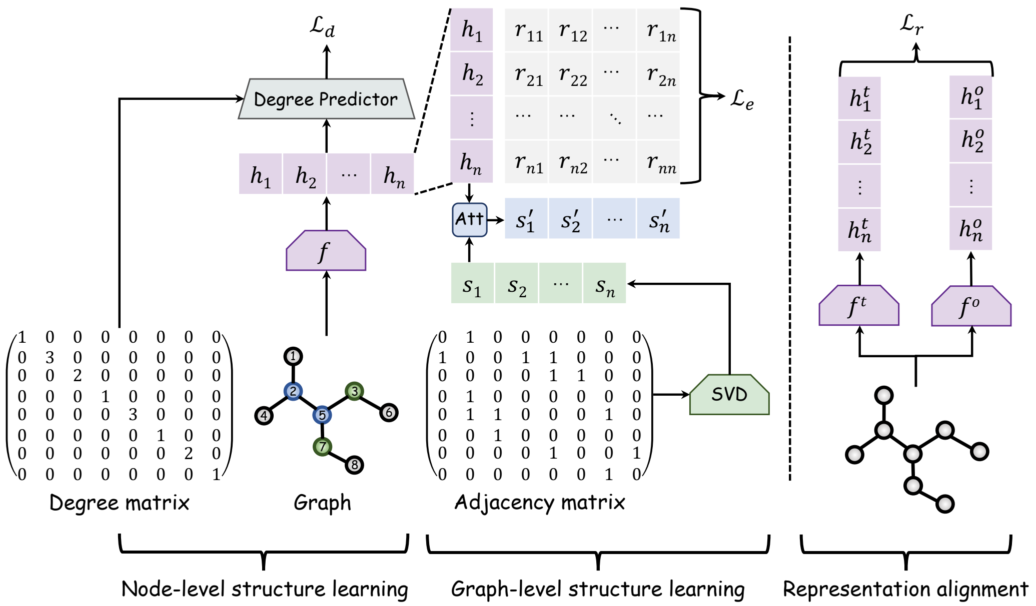


Figure 1: Illustration of our CenPre framework. f is the GNN encoder that is used to generate node representation, f^t is the pre-trained GNN encoder produced by our CenPre framework, and f^o is the original GNN encoder without pre-training. "Att" represents cross-attention. h, s, and r are embeddings, where $r_{ii} = h_i \cdot s_i^{'}$.

Pooling operations can be simple (e.g., mean or sum) or more complex methods like clustering (Ying et al., 2018) or node dropping (Gao & Ji, 2019). A widely-used GNN model is GCN (Kipf & Welling, 2016), which updates node representations by aggregating neighborhood information based on the adjacency matrix A. The degree matrix D, with D_{ii} representing the degree of node v_i, normalizes the influence of neighboring nodes. The propagation rule for GCN is:

$$H^{(K)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(K-1)}W^{(K)}\right) \tag{3}$$

where $\tilde{A} = A + I$ includes self-loops, $\tilde{D}$ is its degree matrix, $W^{(K)}$ is the trainable weight matrix, and $\sigma(\cdot)$ is a non-linear activation function.

4 Method

The motivation of the proposed CenPre is from node2vec (Grover & Leskovec, 2016) that nodes sharing similar roles have similar embeddings. Further, as described in (McNulty, 2022), in many visual graph layouts, more important or influential vertices that have stronger roles in overall connectivity will usually be positioned toward the center of a group of other vertices. Therefore, based on the relation between node importance and centrality, we propose a novel Centrality-guided Graph Pre-training (CenPre) framework for learning representations of nodes. The architecture of our CenPre is illustrated in Figure 1. Note that, different from (Grover & Leskovec, 2016), our goal is not to generate node representations based on similar roles between nodes, but rather to aggregate or differentiate node representations based on the importance of nodes in light of the notion of Centrality, thereby improving node representations.

To reach this goal, we propose to integrate the node importance into the learning of node representation from two perspectives: 1) **node-level importance learning**, which aims to improve the node representation based on the importance of each node itself in a graph structure; 2) **graph-level importance learning**, which aims to improve the representation of a node based on its related node importance in the entire graph. Based on this, we devise a **graph representation alignment** module to align the centrality-guided node representation using the original one. This essentially allows node representations to retain the original semantic information while learning the node's importance. In this study, we employ degree centrality and eigenvector centrality as simple yet effective local and global measures of node importance, respectively, to guide the pre-training process in CenPre, enabling more accurate and structure-aware node representations. It is worth noting that there are several widely used centrality measures; however, due to space limitations, we refer readers to Appendix A for a brief overview of these measures.

4.1 Node-level Importance Learning

In this section, we introduce the node-level importance learning module of our CenPre framework in detail. Based on the Definition 1 from Degree Centrality, we propose node-level importance learning, aiming to identify the degrees of nodes to distinguish the importance of nodes, so as to guide the refinement of the node representation according to the degree of the node.

Definition 1. ***Degree Centrality*** *defines the importance of a node based on the degree of that node. The higher the degree, the more crucial it becomes in the graph.*

From Eq.3, it is evident that GNNs like GCNs use the adjacency matrix to aggregate information from neighboring nodes in an equal and uniform manner, assuming that all sources of information contribute equally. However, GCNs focus solely on the connections between nodes when aggregating neighbor information and do not account for the specific roles or characteristics of nodes. In this study, we address this limitation by incorporating node importance into the learning process, allowing for more accurate representations. To be specific, for each graph, we obtain the degree d_i of each node v_i by summing the values in its corresponding row of the adjacency matrix $\mathcal{A}$, which represents the connections between nodes:

$$\mathcal{Y}_i^d = d_i = \sum_j \mathcal{A}_{ij} \tag{4}$$

where $\mathcal{A}_{ij}$ represents the entry in the adjacency matrix $\mathcal{A}$ indicating the presence (1) or absence (0) of an edge between nodes v_i and v_j. Then we train a degree predictor $\mathcal{P}_d$ to predict the degree of each node based on the node representations $\{h_1, h_2, ..., h_n\}$. The loss of $\mathcal{P}_d$ is defined as:

$$\mathcal{L}_d = -\sum_{i=1}^{n} \mathcal{Y}_i^d \log\big(\mathcal{P}_d(h_i; \theta_d)\big) \tag{5}$$

where $\mathcal{P}_d(h_i; \theta_d)$ is the predicted distribution of the degree of node v_i and $\mathcal{Y}_i^d$ is the ground-truth distribution. θ_d represents the trainable parameters of the degree predictor $\mathcal{P}_d$. In this way, the degree information can be integrated to refine the node representation.

4.2 Graph-level Importance Learning

Node-level importance learning reflects a local perspective, focusing on the importance derived from the node itself. Solely utilizing node degree as a metric for node-level learning might be misleading, as nodes with similar degrees are not necessarily similar in other respects (e.g., two users in a social network can have the same number of followers). A broader, more global perspective that considers attributes like "the degree distribution of a node's neighbors" is necessary. Such an intuition is captured by the essence of Eigenvector Centrality (Definition 2), which reflects how a node's importance is shaped by the importance of its neighbors. Therefore, beyond learning local node-level importance, we propose to integrate graph-level importance using Eigenvector Centrality.

Definition 2. ***Eigenvector Centrality*** *defines relationships with high-scoring nodes have more contribution to the score of a node than connections to nodes with low eigenvector centrality scores.*

As described previously and by definition, degree centrality provides insight into a node's immediate environment, while eigenvector centrality reflects its importance relative to the entire network. Together, they offer a more comprehensive picture of a node's role within the graph, which is naturally divided into four categories shown in Table 7 in Appendix A.

To compute Eigenvector Centrality, we first obtain $\lambda_{\max}$, which is the maximum absolute eigenvalue of the adjacency matrix $\mathcal{A}$, and solve for the eigenvector $\vec{v}$:

$$\lambda_{\max}\vec{v} = \mathcal{A}\vec{v} \tag{6}$$

This is an example of the Eigen-Decomposition, where we decompose $\mathcal{A}$ into a set of eigenvectors and eigenvalues. This classic technique is powerful but may not always be numerically stable for ill-conditioned matrices[1], which can arise in large, sparse, or noisy networks. Moreover, it is also computationally expensive for large matrices. To address this, we turn to (Truncated) Singular Value Decomposition (ED and SVD are equal if the $\mathcal{A}^\top = \mathcal{A} \wedge \mathcal{A} \succeq 0$, which is usually the case for $\mathcal{A}$ from undirected graph, see Appendix B for a proof). It retains only the top k singular values and vectors[2], reducing computational cost while preserving the most important structural information:

$$\mathcal{A} = U\Sigma V^\top \approx U_k \Sigma_k V_k^\top \tag{7}$$

where U and V are orthogonal matrices containing the left and right singular vectors, and Σ is a diagonal matrix storing the singular values. The left singular vectors in U_k can be used as structural representations for each node. Specifically, each row of the matrix U_k provides a k-dimensional embedding that captures the most important structural properties of the graph with respect to its overall connectivity. Denoting this structural representation retrieval process as a function $f_{\text{truncated\_svd}}$ that returns the truncated left singular matrix U_k for a given adjacency matrix $\mathcal{A}$, we can obtain the structural representation s_i for each node v_i as follows:

$$S_{\mathcal{G}} = \{s_1, s_2, \ldots, s_n\} = \{U^k_{0,*}, U^k_{1,*}, \ldots, U^k_{n,*}\} = U_k = f_{\text{truncated\_svd}}(\mathcal{A}) \tag{8}$$

Based on this, the issue of producing importance-fused structure representation for a specific node v_i evolved into how to determine the importance of each node in a graph for v_i. In our case, we treat the structure representation $S_{\mathcal{G}}$ and graph representation $H_{\mathcal{G}}$ as two modalities–one encoding the structural properties and the other capturing the feature-based characteristics of the graph–and use graph representation $H_{\mathcal{G}}$ to cross-attend over structural representation. By aligning the two modalities, we produce the importance-fused graph-level structural representation $S'_{\mathcal{G}}$:

$$S'_{\mathcal{G}} = \{s'_1, s'_2 ..., s'_n\} = \mathrm{CrossAtt}(H_{\mathcal{G}}; S_{\mathcal{G}}) = \mathrm{Softmax}\big((W_q H_{\mathcal{G}})(W_k S_{\mathcal{G}})\big)(W_v S_{\mathcal{G}}) \tag{9}$$

where W_q, W_k, and W_v are the weight matrices of query, key, and value in the cross-attention mechanism, respectively. After obtaining the importance-fused structure representation $S'_{\mathcal{G}}$, inspired by CLIP (Radford et al., 2021) aligning text and image modalities, we propose a Contrastive Representation-Structure Pre-Training (CReSP) model $\mathcal{P}_e$ to train the node representation based on the graph-level importance-fused structure representation. This contrastive learning objective is designed to capture both local node features and global structural roles by aligning these two complementary modalities in the same embedding space. The loss of $\mathcal{P}_e$ is defined as:

$$\mathcal{L}_e = -\sum_{i=1}^{n} \frac{1}{2}\Big(\mathcal{Y}^e_i \log\big(\mathcal{P}_e(r_i;\theta_e)\big) + \mathcal{Y}^e_i \log\big(\mathcal{P}_e(r_i^\top;\theta_e)\big)\Big), \quad r_i = [r_{i1}, r_{i2}, ..., r_{in}] \tag{10}$$

where $\mathcal{Y}^e$ represents the index label towards the importance-fused structure matrix. $r_{ij} = h_i s'_j$ represents the computation of similarity between h_i and s'_j. θ_e represents the trainable parameters of the CReSP model $\mathcal{P}_e$.

4.3 Graph Representation Alignment

Based on the pre-trained GNN encoder f^t learned by our CenPre, we use the L_2-norm to align the structure-fused node representation with the original one, aiming to prevent the loss of the original semantic information while learning the graph structure information[3]. The loss of graph representation alignment module is defined as:

$$\mathcal{L}_r = \big|\big|\big(f^t(v_i), f^o(v_i)\big)\big|\big|_2 = \sqrt{\sum_{i=1}^{n}\big(f^t(v_i) - f^o(v_i)\big)^2} \tag{11}$$

where $f^t(v_i)$ and $f^o(v_i)$ represent the representation of node v_i produced by the pre-trained GNN encoder f^t and the original GNN encoder f^o.

[1]An ill-conditioned matrix typically has a high condition number, which is the ratio of the largest to the smallest singular value. This makes numerical operations such as Inversion or Eigen-Decomposition highly sensitive to small perturbations in the data.

[2]Conventionally, we set k to explain ≥ 0.95 variance.

[3]In the preliminary experiments, we also tried other loss functions, such as KL-divergence, L_1-norm, Cosine Distance, etc. We found that the performance of L_2-norm was slightly more stable, so we use L_2-norm in our method.

4.4 Overall Learning Objective

The overall learning objective of our CenPre is to train the framework by jointly minimizing the three losses derived from Node-level Importance Learning, Graph-level Importance Learning, and Graph Representation Alignment. The overall loss $\mathcal{L}$ is defined as:

$$\mathcal{L} = \lambda_1 \mathcal{L}_d + \lambda_2 \mathcal{L}_e + \lambda_3 \mathcal{L}_r \tag{12}$$

where hyperparameters λ_1, λ_2 and λ_3 are scaling weights to balance the losses. In this way, our proposed CenPre can effectively integrate structural information into node representations to produce structure-fused node representation by learning important information from graph structures. Meanwhile, the graph representation alignment that aligns the structure-fused representation using the original node representation can also prevent the loss of original semantic information due to excessive learning of structural information.

5 Experiments

In this section, we evaluate the performance of our CenPre compared with existing state-of-the-art (SOTA) competitors in the tasks of node classification, link prediction, and graph classification.

5.1 Datasets and Experimental Settings

Dataset We evaluate the effectiveness of our proposed CenPre framework through node classification, link prediction, and graph classification experiments on 13 widely used benchmark datasets. These include the Citation Networks triplet (Kipf & Welling, 2017) (Cora, Citeseer, Pubmed), Amazon-Co-Purchase networks (Shchur et al., 2018) (Computer, Photo), TUD Benchmark datasets (Morris et al., 2020) (MUTAG, NCI1, PROTEINS, DD, IMDB-B, RDT-B), and two large-scale graphs from the Open Graph Benchmark (Hu et al., 2020) (ogbn-arXiv, ogbl-Collab)[4]. As summarized in Table 1, these datasets span various domains, with the number of graphs ranging from 1 to 4,110, the average number of nodes ranging from 17.93 to 235,868, and the average number of edges ranging from 19.79 to 2,315,598, demonstrating the diversity and comprehensiveness of our dataset selection.

Table 1: Statistics of datasets.

Datasets	Task	#Graphs	#Nodes	#Edges	#Features	#Classes
Cora	Node&Link	1	2,708	10,556	1,433	7
CiteSeer	Node&Link	1	3,327	9,104	3,703	6
PubMed	Node&Link	1	19,717	88,648	500	3
Computer	Node	1	13,752	491,722	767	10
Photo	Node	1	7,650	238,162	745	8
arXiv	Node	1	169,343	2,315,598	128	40
Collab	Link	1	235,868	1,285,465	128	-
MUTAG	Graph	188	17.93	19.79	-	2
NCI1	Graph	4,110	29.87	32.30	-	2
PROTEINS	Graph	1,113	39.06	72.82	-	2
DD	Graph	1,178	284.32	715.66	-	2
IMDB-B	Graph	1,000	19.77	96.53	-	2
RDT-B	Graph	2,000	429.63	497.75	-	2

Baselines & Implementation Details We compare our CenPre with a series of SOTA baseline models, including 1) supervised learning methods: GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), GIN (Xu et al., 2019), and SAGE (Hamilton et al., 2017); 2) graph kernels methods: WL (Shervashidze et al., 2011) and DGK (Yanardag & Vishwanathan, 2015); 3) self-supervised learning methods: node2vec (Grover & Leskovec, 2016), graph2vec (Narayanan et al., 2017), InfoGraph (Sun et al., 2020), GAE (Kipf & Welling, 2016), VGAE (Kipf & Welling, 2016), ARGA (Pan et al., 2018), GraphMAE (Hou et al., 2022), DGI (Veličković et al., 2019), GRACE (Zhu et al., 2020), GCA (Zhu et al., 2021), BGRL (Thakoor et al., 2021), CCA-SSG (Zhang et al., 2021), GraphCL (You et al., 2020), JOAO (You et al., 2021), InfoGCL (Xu et al., 2021), SimGRACE (Xia et al., 2022a), AutoGCL (Yin et al., 2022), MaskGAE$_e$ (Li et al., 2023), MaskGAE$_p$ (Li et al., 2023), TopoGCL (Chen et al., 2024), Patcher (Ju et al., 2023) and GPA (Zhang et al., 2024). Due to length limit, Readers are referred to Appendix C for further information of these baselines and Appendix D for implementation details, which follows the evaluation protocol established by previous works (Li et al., 2023; Hou et al., 2022).

[4]For undirected graphs in OGB, the number of edges is doubled due to the automatic addition of bidirectional edges.

Table 2: Experimental results of **node classification**. Averaged accuracy±std. (%) over 10 runs are reported. The best and second-best results are highlighted in **red** and **blue**, respectively. A.R. is short for the average rank. The smaller the value of A.R., the higher the ranking of model performance. Results with ⋆ denote the significance tests of our CenPre over the baseline models at $p-$value < 0.05.

	Methods	Cora	CiteSeer	PubMed	Computer	Photo	arXiv	A.R.↓
Supervised	GCN	81.50±0.20	70.30±0.40	79.00±0.50	86.51±0.54	92.42±0.22	70.40±0.30	10.4
	GAT	83.00±0.70	72.50±0.70	79.00±0.30	86.93±0.29	92.56±0.35	70.60±0.30	7.9
Self-supervised	GAE	74.90±0.40	65.60±0.50	74.20±0.30	85.10±0.40	91.00±0.10	63.60±0.50	14.5
	VGAE	76.30±0.20	66.80±0.20	75.80±0.40	85.80±0.30	91.50±0.20	64.80±0.20	13.5
	ARGA	77.95±0.70	64.44±1.19	80.44±0.74	85.86±0.11	91.82±0.08	67.34±0.09	12.2
	DGI	82.30±0.60	71.80±0.70	76.80±0.60	83.95±0.47	91.61±0.22	65.10±0.40	11.7
	GRACE	81.90±0.40	71.20±0.50	80.60±0.40	86.25±0.25	92.15±0.24	68.70±0.40	10.2
	GCA	81.80±0.20	71.90±0.40	81.00±0.30	87.85±0.31	92.53±0.16	68.20±0.20	8.7
	BGRL	82.86±0.49	71.41±0.92	82.05±0.85	**90.34**±0.19	93.17±0.30	71.64±0.12	5.5
	CCA-SSG	83.59±0.73	73.36±0.72	80.81±0.38	88.74±0.28	93.14±0.14	69.22±0.22	6.7
	GraphMAE	84.20±0.40	73.40±0.40	81.10±0.40	89.51±0.08	93.23±0.13	71.75±0.17	3.8
	Patcher	84.17±0.54	71.65±0.05	81.13±0.68	89.44±0.79	81.23±0.32	**72.31**±0.22	6.8
	MaskGAE$_e$	83.77±0.33	72.94±0.20	82.69±0.31	89.44±0.11	93.30±0.04	70.97±0.29	4.6
	MaskGAE$_p$	**84.30**±0.39	**73.80**±0.81	**83.58**±0.45	89.54±0.06	**93.31**±0.13	71.16±0.33	**2.7**
	CenPre (ours)	**85.15**±0.49⋆	**76.94**±2.12⋆	**83.91**±0.12	**91.22**±0.05⋆	**93.96**±0.14	**72.47**±0.15	**1.0**

Table 3: Experimental results of **link prediction**. Average AUC, Average Precision (AP), and Hit@50±std. (%) over 10 runs are reported. Hit@50 measures the proportion of correct links among the top 50 predictions. The best and second-best results are highlighted in **red** and **blue**, respectively.

	Methods	Cora		CiteSeer		PubMed		COLLAB	A.R.↓
		AUC	AP	AUC	AP	AUC	AP	Hit@50	
Supervised	GCN	86.70±0.20	87.55±0.05	91.10±0.50	91.72±0.43	84.66±0.10	86.20±0.61	47.14±0.01	9.4
	GAT	86.84±0.27	88.66±0.08	91.20±0.10	92.02±0.44	84.23±0.10	86.62±0.22	-	10.1
	GIN	86.66±0.59	87.62±0.58	92.62±0.24	92.54±0.11	84.05±0.32	86.17±0.25	-	10.4
	SAGE	86.33±1.06	88.81±1.36	92.54±0.87	92.70±1.02	84.98±2.65	87.12±2.95	54.63±1.12	7.3
Latent	node2vec	78.32±0.74	78.91±0.77	75.36±1.22	76.03±0.11	79.98±0.35	81.55±0.83	57.03±0.52	10
	MatrixFactor	62.25±2.21	64.20±1.17	61.65±3.80	61.99±2.50	68.56±12.13	68.23±3.13	48.96±0.29	11.5
Self-supervised	GAE	91.09±0.01	92.83±0.03	96.40±0.01	96.50±0.02	90.52±0.04	91.68±0.05	47.14±1.45	7.1
	VGAE	91.40±0.01	92.60±0.01	94.40±0.02	94.70±0.02	90.80±0.02	92.00±0.02	45.53±1.87	7.8
	ARGA	92.40±0.00	93.23±0.00	96.81±0.00	97.11±0.00	91.94±0.00	93.03±0.00	28.39±2.51	6.3
	GraphMAE	94.88±0.23	93.52±0.51	96.24±0.36	95.47±0.41	94.32±0.40	93.54±0.22	53.97±0.64	5.3
	MaskGAE$_e$	96.42±0.17	95.91±0.25	**98.02**±0.22	**98.18**±0.21	98.75±0.04	98.66±0.06	65.84±0.47	2.8
	MaskGAE$_p$	**96.45**±0.18	**95.95**±0.21	97.87±0.22	98.09±0.17	**98.84**±0.04	**98.78**±0.05	**65.98**±0.39	**2.3**
	CenPre (ours)	**97.44**±0.85	**96.05**±0.65	**99.02**±0.59⋆	**99.04**±0.37	**99.82**±0.10⋆	**99.62**±0.10⋆	**66.03**±0.27	**1.0**

Node Classification Table 2 shows the results of node classification. Our CenPre overall outperforms the SOTA baseline models, in which our CenPre achieves the best performance on all datasets. This demonstrates the effectiveness of the proposed CenPre in node classification. Further, our CenPre performs best on both the Cora dataset with 2,708 nodes and the arXiv dataset with 169,343 nodes, indicating that our CenPre can be effective on datasets of different types and sizes.

Link Prediction We further evaluate the performance of our CenPre in link prediction and report the comparison results with SOTA baseline models in Table 3. Our CenPre achieves overall better performance than the baseline models, reaching optimal performance in all datasets. This indicates that leveraging the node importance based on the notion of Centrality can improve the representations of nodes in a graph, thereby leading to better link prediction performance.

Graph Classification Table 4 shows the experimental results of graph classification. We can see that our CenPre consistently outperforms the baseline models on all datasets. This verifies the effectiveness of CenPre in graph classification, indicating that the proposed Centrality-guided method can improve the learning of the entire graph based on the improvement of node representations, therefore achieving optimal performance in graph classification. Further, our CenPre performs consistently better than TopoGCL (Chen et al., 2024) which considers the topological information of the graph. This denotes that leveraging node-level and graph-level importance can improve

Table 4: Experimental results of **graph classification**. Averaged accuracy±std. (%) over 10 runs are reported. The best and second-best results are highlighted in **red** and **blue**, respectively.

	Methods	NCI1	PROTEINS	DD	MUTAG	IMDB-B	RDT-B	A.R.↓
Supervised	GCN	76.29±1.79	75.17±3.63	73.26±4.46	79.81±1.58	57.35±4.04	81.30±6.93	12
	GAT	74.90±1.72	74.72±4.01	77.30±3.68	78.89±2.05	54.60±7.45	72.70±2.30	12.7
	SAGE	74.73±1.34	74.01±4.27	75.78±3.91	78.75±1.18	58.95±6.74	83.10±5.40	12.7
Kernel	WL	80.31±0.46	72.92±0.56	76.44±2.35	80.72±3.00	72.30±3.44	68.82±0.41	11.2
	DGK	81.01±1.06	73.30±0.82	74.85±0.74	87.44±2.72	66.96±0.56	78.04±0.39	10.3
Self-supervised	node2vec	54.89±1.61	57.49±3.57	-	72.63±10.20	56.40±2.80	69.70±4.10	16.2
	Graph2Vec	73.22±1.81	73.30±2.05	70.32±2.32	83.15±9.25	71.10±0.54	75.78±1.03	13.1
	InfoGraph	76.20±1.06	74.44±0.31	74.24±0.86	89.01±1.13	73.03±0.87	82.50±1.42	9.3
	GraphCL	77.87±0.41	74.39±0.45	78.62±0.40	86.80±1.34	71.14±0.44	89.53±0.84	8.2
	JOAO	78.07±0.47	74.55±0.41	77.32±0.54	87.35±1.02	70.21±3.08	85.29±1.35	9.2
	InfoGCL	80.20±0.60	-	-	**91.20**±1.30	75.10±0.90	-	10.5
	GraphMAE	80.40±0.30	75.30±0.39	-	88.19±1.26	**75.52**±0.66	88.01±0.19	7.5
	SimGRACE	79.12±0.44	75.35±0.09	77.44±1.11	89.01±1.31	71.30±0.77	89.51±0.89	6.3
	AutoGCL	**82.00**±0.29	75.80±0.36	77.57±0.60	88.64±1.08	72.32±0.93	88.58±1.49	5
	TopoGCL	81.30±0.27	**77.30**±0.89	79.15±0.35	90.09±0.93	74.67±0.32	**90.40**±0.53	**2.8**
	GPA	80.42±0.41	75.94±0.25	**79.90**±0.35	89.68±0.80	74.64±0.35	89.32±0.38	5.3
	CenPre (ours)	**88.13**±0.91*	**80.25**±0.67*	**85.18**±1.37*	**94.74**±0.48*	**78.05**±1.21*	**91.20**±0.50	**1.0**

the learning of graph structure when modeling a graph, thus improving the performance of graph classification.

5.2 Comparisons with Baselines

5.3 Analysis of Our CenPre

Ablation Study We conduct an ablation study to analyze the impact of each module of the CenPre on performance. The results are reported in Table 5. It can be seen that the removal of graph representation alignment ($\mathcal{L}_r$) seriously degrades the performance of our CenPre. This indicates that it is necessary to align the node representation when learning structural information, since it can prevent the loss of the graph semantic information, especially for node classification. Further, the removal of node-level importance learning ($\mathcal{L}_d$) and graph-level importance learning ($\mathcal{L}_e$) can lead to considerable performance degradation, which demonstrates that learning node importance from both node and graph levels can make full use of the important structural information of node in a graph, thus improving the model's performance. In addition, the performance of "w/o SVD" shows that the removal of SVD noticeably degrades the performance of our CenPre. This implies that exploiting SVD to produce the structure representation based on important singular values and eigenvectors can help the model better learn the graph structural information.

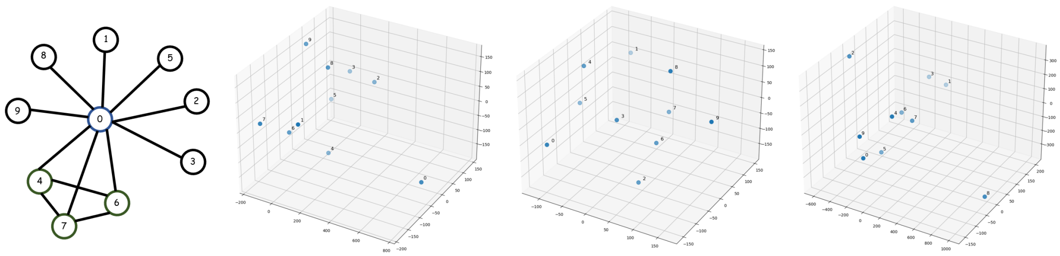
Table 5: Experimental results of the ablation study. Δ denotes the performance drop relative to the full CenPre model. "w/o SVD" means the adjacency matrix $\mathcal{A}$ is used directly as the structural representation, and AUC is used as the evaluation metric for Cora-Link.

Methods	Cora-Node	Δ	Cora-Link	Δ	MUTAG-Graph	Δ
CenPre	85.15±0.49	0.00	95.05±0.18	0.00	94.74±0.48	0.00
w/o $\mathcal{L}_d$	80.40±0.10	4.75	94.08±0.29	0.97	86.84±0.13	7.9
w/o $\mathcal{L}_e$	84.21±0.62	0.94	94.07±0.54	0.98	92.11±0.26	2.63
w/o $\mathcal{L}_r$	78.63±0.50	7.12	91.29±1.57	3.76	89.47±2.55	5.27
w/o SVD	83.45±0.61	1.70	93.72±0.22	1.33	93.52±0.42	1.22

Analysis of Generalizability To analyze the generalizability of our CenPre to different graph autoencoders, we conduct comparative experiments based on different graph autoencoders and report the results in Table 6. We can see that the proposed CenPre can directly adapt to different graph autoencoders and achieve varying degrees of performance improvement, which validates the generalizability of our CenPre in graph pre-training. In addition, it can be noted that our CenPre, which uses the original node representation, achieves better performance than

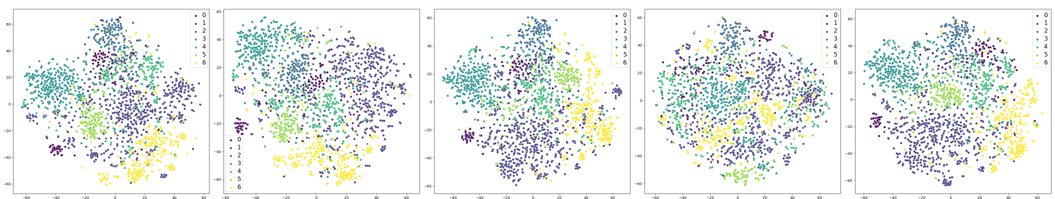
Table 6: Experimental results of using different graph autoencoders on the Cora dataset.

Methods	Original	w/ CenPre	Improvement
GAE	74.90±0.40	81.56±0.60	6.66
VGAE	76.30±0.20	78.30±1.15	2.00
ARGA	77.95±0.70	80.73±0.27	2.78
DGI	82.30±0.60	83.11±0.54	0.81
CenPre (ours)	85.15±0.49		0.00



(a) graph (b) original (c) w/ node (d) CenPre

Figure 2: The T-SNE 3D plots of the extracted nodes during different pre-training stages.



(a) original (b) w/o node (c) w/o graph (d) w/o regularization (e) CenPre

Figure 3: The T-SNE 2D plots of all the test samples from the CORA dataset.

those that use graph autoencoders. This implies that using the original representation can better assist our CenPre in learning the structural information of the graph, thereby obtaining better node representations. Furthermore, this also indicates that exploring better methods for graph autoencoders may further enhance the learning of graphs.

Analysis of Node Representation To analyze how our CenPre improves the representation of nodes, we select a node from Cora data and retain its first-order neighbors to analyze the changes in their representation at different learning stages. The results are shown in Figure 2. We can see that, the original representations of nodes are divergent and the correlation between them cannot be observed. After node-level importance learning, some important nodes are gathered together. Furthermore, through our complete CenPre, important nodes (4, 6, 7) are further aggregated (Figure 2 (d)). This indicates that our CenPre can learn the similarity importance between nodes based on Centrality, thereby improving the representation of nodes.

Visualizations To qualitatively demonstrate how our CenPre improves the node representations, Figure 3 shows the t-SNE (van der Maaten & Hinton, 2008) visualization of node representations from original representations (a), the variants of our CenPre (b), (c) and (d), and our CenPre (e). We can observe that the original node representations largely diffuse and overlap between different labels. The variants of our CenPre can show differences between different labels, which denotes that exploring preferable methods to learn better node representations is key to improving node classification. Further, the node representations derived by our CenPre can be better separated from different labels. This indicates that our CenPre can make full use of the node importance based on Centrality, and further improve the representations of nodes based on the representation alignment, therefore leading to improved node representations.

6 Conclusion

In this paper, we propose **CenPre**, a **Cen**trality-guided **Pre**-training framework for node representation learning. Unlike methods focused on augmentation, CenPre leverages Degree and Eigenvector Centrality to integrate structural information into node representations. Additionally, a graph alignment module retains semantic information while learning graph structure. Experiments on real-world datasets show that CenPre outperforms state-of-the-art models in node classification, link prediction, and graph classification, demonstrating the effectiveness of centrality-based structure integration.

References

Pedro H. C. Avelar, Henrique Lemos, Marcelo O. R. Prates, and L. Lamb. Multitask learning on graph neural networks - learning multiple graph centrality measures with a unified network. In *International Conference on Artificial Neural Networks*, 2018. URL https://api.semanticscholar.org/CorpusID:52311433.

Jinheon Baek, Dong Bok Lee, and Sung Ju Hwang. Learning to extrapolate knowledge: Transductive few-shot out-of-graph link prediction. *Advances in Neural Information Processing Systems*, 33: 546–560, 2020.

Alex Bavelas. Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, 22:725–730, 1950. URL https://api.semanticscholar.org/CorpusID:121849458.

Piotr Bielak, Tomasz Kajdanowicz, and N. Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *ArXiv*, abs/2106.02466, 2021. URL https://api.semanticscholar.org/CorpusID:235352934.

Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2:113–120, 1972. URL https://api.semanticscholar.org/CorpusID:120637488.

Phillip Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

Yuzhou Chen, Jose Frias, and Yulia R Gel. Topogcl: Topological graph contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 11453–11461, 2024.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699, 2020. URL https://arxiv.org/abs/2012.09699.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv*, abs/1903.02428, 2019. URL https://api.semanticscholar.org/CorpusID:70349949.

Leonard M. Freeman. A set of measures of centrality based upon betweenness. In *Sociometry*, 1977. URL https://api.semanticscholar.org/CorpusID:62597900.

Linton C. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1: 215–239, 1978. URL https://api.semanticscholar.org/CorpusID:751590.

Hongyang Gao and Shuiwang Ji. Graph u-nets. In *international conference on machine learning*, pp. 2083–2092. PMLR, 2019.

Alberto García-Durán and Mathias Niepert. Learning graph representations with embedding propagation. *ArXiv*, abs/1710.03059, 2017. URL https://api.semanticscholar.org/CorpusID:39741524.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

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

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*, 2019.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, C. Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. URL https://api.semanticscholar.org/CorpusID:248987361.

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

Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH.

Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. Pre-training graph neural networks for generic structural feature extraction. *ArXiv*, abs/1905.13728, 2019. URL https://api.semanticscholar.org/CorpusID:173188920.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1857–1867, 2020.

Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. Self-supervised learning on graphs: Deep insights and new direction. *ArXiv*, abs/2006.10141, 2020. URL https://api.semanticscholar.org/CorpusID:219792818.

Mingxuan Ju, Tong Zhao, Wenhao Yu, Neil Shah, and Yanfang Ye. Graphpatcher: Mitigating degree bias for graph neural networks via test-time augmentation. *Advances in Neural Information Processing Systems*, 36, 2023.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953. URL https://api.semanticscholar.org/CorpusID:121768822.

Dongki Kim, Jinheon Baek, and Sung Ju Hwang. Graph self-supervised learning with accurate discrepancy learning. In *36th Conference on Neural Information Processing Systems, NeurIPS 2022*. Neural Information Processing Systems, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. 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*, pp. 1268–1279, 2023.

Massimo Marchiori and Vito Latora. Harmony in the small-world. *Physica A-statistical Mechanics and Its Applications*, 285:539–546, 2000. URL https://api.semanticscholar.org/CorpusID:10523345.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Fast approximations of betweenness centrality with graph neural networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019. URL https://api.semanticscholar.org/CorpusID:207758212.

Keith McNulty. *Handbook of graphs and networks in people analytics: with examples in R and python*. Chapman and Hall/CRC, 2022.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder. *ArXiv*, abs/1802.04407, 2018. URL https://api.semanticscholar.org/CorpusID:51608273.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *ArXiv*, abs/1912.01703, 2019. URL https://api.semanticscholar.org/CorpusID:202786778.

Bryan Perozzi, Rami Al-Rfou, and Steven S. Skiena. Deepwalk: online learning of social representations. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014. URL https://api.semanticscholar.org/CorpusID:3051291.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31:581–603, 1966. URL https://api.semanticscholar.org/CorpusID:119981743.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1711–1719, 2020.

Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1lfF2NYvH.

Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Velivckovi'c, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*, 2021. URL https://api.semanticscholar.org/CorpusID:243756979.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rklz9iAcKQ`.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 207–217, 2020.

Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*, pp. 1070–1079, 2022a.

Jun Xia, Yanqiao Zhu, Yuanqi Du, and Stan Z Li. A survey of pretraining on graphs: Taxonomy, methods, and applications. *arXiv preprint arXiv:2202.07893*, 2022b.

Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 30414–30425, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8892–8900, 2022.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

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

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

Mohammed J Zaki and Wagner Meira. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

Hengrui Zhang, Qitian Wu, Junchi Yan, David Paul Wipf, and Philip S. Yu. From canonical correlation analysis to self-supervised graph neural networks. In *Neural Information Processing Systems*, 2021. URL `https://api.semanticscholar.org/CorpusID:235606326`.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Xin Zhang, Qiaoyu Tan, Xiao Huang, and Bo Li. Graph contrastive learning with personalized augmentation. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Yizhen Zheng, Shirui Pan, Vincent Cs Lee, Yu Zheng, and Philip S Yu. Rethinking and scaling up graph contrastive learning: An extremely efficient approach with group discrimination. In *Advances in Neural Information Processing Systems*, 2022.

Yanqiao Zhu, Yichen Xu, Feng Yu, Q. Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *ArXiv*, abs/2006.04131, 2020. URL `https://api.semanticscholar.org/CorpusID:219531264`.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

A Appendix

A Centrality Measures

Centrality is a key concept in network analysis used to identify the most important or influential nodes within a graph. Different centrality measures capture various aspects of a node's importance, ranging from its immediate connections to its role in facilitating communication between other nodes. These measures help in understanding the structure of the graph, the flow of information, and the relative significance of individual nodes in maintaining the overall connectivity. Below, we introduce several commonly used centrality measures:

- **Degree Centrality** (Freeman, 1978) counts the number of direct connections a node has. Nodes with higher degree centrality are considered more locally important due to their numerous direct interactions within the network. For a node v_i, the degree centrality is given by:
$$C_D(v_i) = \sum_j \mathcal{A}_{ij} \tag{13}$$
where $\mathcal{A}_{ij}$ is the entry in the adjacency matrix $\mathcal{A}$, indicating the presence or absence of an edge between nodes i and j.
- **Eigenvector Centrality** (Bonacich, 1972) assigns relative scores to nodes based on the principle that connections to high-scoring nodes contribute more to the score of the node itself. It captures both direct and indirect influences in the network. For a node v_i, the eigenvector centrality is given by:
$$C_E(v_i) = \frac{1}{\lambda} \sum_j \mathcal{A}_{ij} C_E(v_j) \tag{14}$$
where λ is the largest eigenvalue of the adjacency matrix $\mathcal{A}$.
- **Betweenness Centrality** (Freeman, 1977) quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. For a node v_i, betweenness centrality is given by:
$$C_B(v_i) = \sum_{s \neq v_i \neq t} \frac{\sigma_{st}(v_i)}{\sigma_{st}} \tag{15}$$
where σ_{st} is the total number of shortest paths from node s to node t, and $\sigma_{st}(v_i)$ is the number of those paths that pass through node v_i.
- **Closeness Centrality** (Bavelas, 1950; Sabidussi, 1966) measures how close a node is to all other nodes in the network. For a node v_i, closeness centrality is defined as:
$$C_C(v_i) = \frac{1}{\sum_j d(v_i, v_j)} \tag{16}$$
where $d(v_i, v_j)$ is the shortest path distance between nodes v_i and v_j.
- **PageRank Centrality** is a variant of eigenvector centrality that evaluates the importance of a node based on the quality and quantity of incoming connections, where connections from more important nodes weigh more heavily. The PageRank score $PR(v_i)$ for a node v_i is given by:
$$PR(v_i) = \frac{1-d}{N} + d \sum_{j \in \mathcal{N}(v_i)} \frac{PR(v_j)}{|\mathcal{N}(v_j)|} \tag{17}$$
where d is the damping factor (typically set to 0.85), N is the total number of nodes, and $\mathcal{N}(v_j)$ is the set of neighbors of node v_j.

- **Katz Centrality** (Katz, 1953) extends degree centrality by considering the total number of walks between nodes. It is defined as:
$$C_K(v_i) = \alpha \sum_j \mathcal{A}_{ij} C_K(v_j) + \beta \tag{18}$$
where α is a constant (decay factor) and β is an additional weight applied to each node, allowing for paths of different lengths to contribute to the centrality.
- **Harmonic Centrality** (Marchiori & Latora, 2000) is a variation of closeness centrality that computes the sum of the reciprocals of the shortest path distances from one node to all other nodes. It is defined as:
$$C_H(v_i) = \sum_{j \neq i} \frac{1}{d(v_i, v_j)} \tag{19}$$
where $d(v_i, v_j)$ is the shortest path distance between nodes v_i and v_j.

Table 7 provides a detailed comparison of node types based on Degree Centrality and Eigenvector Centrality, highlighting their roles and characteristics in network structures. The table categorizes nodes into four distinct types depending on their local and global importance, as measured by high or low values of these centrality metrics. Each category is accompanied by a description of the node's structural characteristics within the network and an illustrative example from social networks, offering a practical understanding of how centrality measures reflect different types of influence and connectivity in real-world scenarios.

Degree	**Eigen**	**Node Characteristics**	**Examples in Social Network**
High	Low	A node connected to many other nodes that are not highly central. Important locally but not globally.	A celebrity with many followers but most of those followers are inactive or non-influential accounts.
Low	High	A node with few direct connections but connected to highly central nodes. Important globally but not locally.	A journalist or analyst who is followed by a few highly influential public figures or news organizations.
High	High	A node that is both locally and globally important, with many connections and highly central neighbors.	A popular politician who has many followers who are themselves influential, such as other public figures.
Low	Low	A node with few connections and low global importance. Neither locally nor globally important.	A regular person with few followers, none of whom are influential or central.

Table 7: Comparison of Degree Centrality and Eigenvector Centrality for Different Node Types and their examples in social networks.

B Eigen-Decomposition (ED) and Singular Vector Decomposition (SVD)

Theorem 1. *ED and SVD of a matrix $\mathcal{A}$ are equivalent if and only if $\mathcal{A}$ is symmetric positive semi-definite, i.e., $\mathcal{A}^\intercal = \mathcal{A} \wedge \mathcal{A} \succeq 0$.*

Proof. Given a matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$, the SVD of $\mathcal{A}$ is written as:
$$\mathcal{A} = U \Sigma V^\intercal \tag{20}$$
where:

- $U \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (i.e., $U^\intercal U = I$),
- $V \in \mathbb{R}^{n \times n}$ is an orthogonal matrix (i.e., $V^\intercal V = I$),

- $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix containing the singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$, where $\sigma_i \geq 0$.

The ED of a matrix $\mathcal{A} \in \mathbb{R}^{n\times n}$ is given by:

$$\mathcal{A} = Q\Lambda Q^{-1} \tag{21}$$

where:

- $Q \in \mathbb{R}^{n\times n}$ is an invertible matrix whose columns are the eigenvectors of $\mathcal{A}$,
- $\Lambda \in \mathbb{R}^{n\times n}$ is a diagonal matrix whose entries are the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.

We want to establish the conditions under which the SVD and ED of a matrix $\mathcal{A}$ are equivalent. Specifically, we need to check under what conditions:

$$U\Sigma V^{\mathsf{T}} \equiv Q\Lambda Q^{-1} \tag{22}$$

For the two decompositions to be equivalent, the matrix $\mathcal{A}$ must be symmetric, which implies that the left and right singular vectors are the same:

$$\mathcal{A} = \mathcal{A}^{\mathsf{T}} \rightarrow \mathcal{A} = U\Sigma U^{\mathsf{T}} \tag{23}$$

In SVD, the singular values σ_i in Σ are always non-negative. For the eigenvalues λ_i in the ED to match the singular values, all eigenvalues must be non-negative as well. This requires that $\mathcal{A}$ be **positive semi-definite**. For SVD and ED to coincide, the matrix Q in the ED must be orthogonal, i.e., $Q^{\mathsf{T}}Q = I$. This occurs when the eigenvectors of A are orthonormal. In this case, $Q^{-1} = Q^{\mathsf{T}}$, and the ED becomes:

$$\mathcal{A} = Q\Lambda Q^{\mathsf{T}} \tag{24}$$

Thus, for $\mathcal{A}$ to have orthonormal eigenvectors, it must be symmetric.
In conclusion, the SVD and ED of a matrix $\mathcal{A}$ are equivalent if and only if:

- $\mathcal{A}$ is symmetric, i.e., $\mathcal{A} = \mathcal{A}^{\mathsf{T}}$,
- $\mathcal{A}$ is positive semi-definite,
- The eigenvectors of $\mathcal{A}$ are orthonormal.

In this case, the SVD and ED both yield the same decomposition:

$$U\Sigma U^{\mathsf{T}} \equiv Q\Lambda Q^{\mathsf{T}} \tag{25}$$

where $\Sigma = \Lambda$, and the columns of U (in SVD) and Q (in ED) are the same orthonormal eigenvectors of $\mathcal{A}$.Thus, **SVD and ED are equivalent when $\mathcal{A}$ is symmetric and positive semi-definite**. ☐

C BASELINES

We compare and evaluate our CenPre framework with a series of baseline models, which are grouped by graph kernel methods, graph supervised learning methods, and graph self-supervised learning methods. The more detailed introduction of the baseline models is as follows:

Graph Kernels Methods:

- **WL** (Shervashidze et al., 2011) measures graph similarity by iteratively refining node labels through aggregation of neighboring node labels, capturing graph structure and preserving node attribute information.
- **DGK** (Yanardag & Vishwanathan, 2015) uses neural networks to learn representations of subgraphs and combine them to compute a similarity score, enabling more expressive and flexible graph comparisons for various graph analysis tasks.

Graph Supervised Learning Methods:

- **GCN** (Kipf & Welling, 2017) uses spectral graph theory to aggregate neighboring node features, learning new node representations for tasks like classification and link prediction.
- **GAT** (Velickovic et al., 2018) uses an attention mechanism to assign weights to neighboring nodes, computing weighted sums of their features to capture complex relationships for tasks where neighbor importance varies.
- **GIN** (Xu et al., 2019) aggregates node features by summing a node's features with its neighbors' and applying an MLP, capturing local structure and complex feature interactions for effective node and graph classification.
- **SAGE** (Hamilton et al., 2017) generates node embeddings by sampling and aggregating neighborhood features, enabling it to generalize to unseen nodes and graphs for scalable node classification and link prediction.

Graph Self-supervised Learning Methods

- **GAE** (Kipf & Welling, 2016) encodes nodes into latent embeddings and reconstructs the graph structure to learn meaningful representations for tasks such as node classification and link prediction.
- **VGAE** (Kipf & Welling, 2016) extends the GAEs with variational inference to learn probabilistic latent variable models, providing robust and expressive node embeddings for graph-based tasks such as link prediction and anomaly detection.
- **ARGA** (Pan et al., 2018) enhances GAEs by incorporating adversarial training, ensuring more robust and discriminative node embeddings for graph tasks like node classification and link prediction through adversarial regularization.
- **GraphMAE** (Hou et al., 2022) leverages masked node feature reconstruction to learn rich and informative node embeddings, enhancing performance on downstream graph tasks such as node classification and graph classification.
- **MaskGAE** (Li et al., 2023) applies masking and reconstruction strategies to learn meaningful graph representations, enhancing performance on tasks such as node classification and link prediction.
- **DGI** (Veličković et al., 2019) maximizes mutual information between local and global graph representations, producing highly expressive node embeddings that excel in tasks such as node classification and graph classification.
- **GRACE** (Zhu et al., 2020) leverages contrastive learning to maximize agreement between different views of the same graph, resulting in robust and informative node embeddings for various graph-based tasks.
- **GCA** (Zhu et al., 2021) enhances contrastive learning by incorporating adaptive augmentation strategies, yielding discriminative and robust node embeddings for diverse graph-related tasks.
- **BGRL** (Thakoor et al., 2021) learns graph representations by leveraging a bootstrapping mechanism to predict target network outputs, producing effective node embeddings for tasks such as node classification and link prediction without the need for negative sampling.
- **CCA-SSG** (Zhang et al., 2021) utilizes canonical correlation analysis to maximize agreement between different graph views, generating high-quality node embeddings for downstream tasks such as node classification and clustering.
- **node2vec** (Grover & Leskovec, 2016) uses biased random walks to generate node sequences, which are then used with the Skip-gram model to produce continuous node embeddings for downstream tasks like node classification and link prediction.
- **graph2vec** (Narayanan et al., 2017) uses the Skip-gram model to generate continuous embeddings, capturing structural and global graph characteristics for downstream tasks.
- **InfoGraph** (Sun et al., 2020) maximizes mutual information between graph-level and substructure-level representations, resulting in informative embeddings for downstream graph classification and clustering tasks.

- **GraphCL** (You et al., 2020) designs four contrastive learning with graph augmentations to capture rich and robust graph representations, improving performance on tasks such as graph classification and clustering.
- **JOAO** (You et al., 2021) dynamically optimizes data augmentations to enhance contrastive learning, producing robust and generalizable node embeddings for various graph tasks.
- **InfoGCL** (Xu et al., 2021) combines contrastive learning with mutual information maximization to enhance node and graph-level embeddings, yielding improved performance on various graph-based tasks.
- **Patcher** (Ju et al., 2023) mitigates degree bias in Graph Neural Networks through test-time augmentation, enhancing the robustness and generalization of graph representations for various graph-based tasks.
- **SimGRACE** (Xia et al., 2022a) enhances graph contrastive learning by leveraging similarity-based augmentations, producing robust node and graph embeddings for improved performance on downstream tasks.
- **AutoGCL** (Yin et al., 2022) employs learnable view generators to create optimized augmentations, leading to robust and effective graph embeddings for various downstream tasks.
- **TopoGCL** (Chen et al., 2024) leverages topological invariance and extended persistence to capture higher-order substructures, enhancing graph representations and delivering significant performance gains in unsupervised graph classification, particularly in biological, chemical, and social interaction graphs.
- **GPA** (Zhang et al., 2024) customizes augmentation strategies for each graph based on its topology and node attributes, enhancing representation learning and outperforming state-of-the-art models across diverse benchmark datasets.

D More Details of Experimental Implementation

For node classification and link prediction tasks, we use a 2-layer GCN (Kipf & Welling, 2017) as the encoder for self-supervised pretraining under the CenPre framework, increasing to 4 layers for the arXiv (Hu et al., 2020) dataset. After pretraining, we freeze the encoder to extract node embeddings, which are then input into a 2-layer MLP for classification and prediction. Results are reported as mean and standard deviation over 5 runs. For graph classification, we use a GIN (Xu et al., 2019) encoder instead, which is commonly used in previous graph classification works. we use the Adam (Kingma & Ba, 2014) optimizer with an initial learning rate of $lr = 0.01$. The balancing hyperparameters for loss components are set to $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 5$, determined through pilot studies. For model parameters, we use grid search to find the optimal parameter combination. For the three hyperparameters λ_1, λ_2, and λ_3, in preliminary experiments, we found that the performance of the model is stable within a certain range of values for these three hyperparameters. Therefore, for the scalability of our method, we set $\lambda_1 = 1$, $\lambda_2 = 1$, and $\lambda_3 = 5$ to make them to the same scale.

Our framework is built on PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019), leveraging their datasets and functionalities. Experiments are conducted on an Intel(R) Xeon(R) Gold 6248R CPU at 3.00GHz and an Nvidia Tesla V100 GPU with 32GB VRAM. Table 8 shows all relevant parameter settings for our experiments.

E Complexity Analysis

To analyze the computational requirements of CenPre, we consider the complexity of its iterative operations during training while treating the truncated-SVD as a precomputation step. The precomputed structural representations are reused throughout the training process, and their computation does not contribute to the iterative complexity of the framework.

E.1 Spacetime Complexity of CenPre

The space complexity of CenPre during training is determined by:

Table 8: Experimental implementation for node classification, link prediction, and graph classification. In node classification tasks, we set Hidden Size to 64 for the Citation Networks, 128 for the Amazon Co-Buy, and 512 for the arXiv dataset, which has a 4-layer MLP as the downstream classifier.

Parameter	Description	Node Class.	Link Pred.	Graph Class.
LR	Learning Rate	0.001	0.001	0.001
L_2	Weight Decay	5e-4	5e-4	5e-4
p_e	Early Stopping Patience	15	15	15
e_Δ	Early Stopping Min-Delta	1e-5	1e-5	1e-5
p_s	Learning Rate Scheduler Patience	8	8	8
$\mathcal{B}$	Batch size	128	128	1
K	Number of GNN Layers	2	2	2
M	Number of MLP Layers	2/4	2	2
d_h	Hidden Layer Size	64/128/512	64	128
λ_1	Scaling Weight of Node-Level Loss $\mathcal{L}_{node}$	1	1	1
λ_2	Scaling Weight of Graph-Level Loss $\mathcal{L}_{graph}$	1	1	1
λ_3	Scaling Weight of Regularization Loss $\mathcal{L}_{reg}$	5	5	5
E	Training epochs	100	100	300
η_d	Dropout Ratio	0.2	0.2	0.2
$\text{POOL}(\cdot)$	Pooling Function	-	-	mean

- The node embeddings, are stored as a matrix of size $O(nd)$, where n is the number of nodes and d is the dimensionality of the embeddings.
- The adjacency matrix $\mathcal{A}$, which has $O(|E|)$ space complexity for sparse graphs, where $|E|$ is the number of edges.
- The cross-attention mechanism, which requires storing $O(n^2)$ attention weights during alignment.

Combining these, the overall space complexity of CenPre is:

$$O(nd + |E| + n^2) \tag{26}$$

The time complexity of CenPre's iterative operations during training is as follows:

- Node-level importance learning: Computing node degrees from the adjacency matrix $\mathcal{A}$ has $O(|E|)$ complexity. Training the degree predictor $\mathcal{P}_d$ for n nodes requires $O(nd)$ operations per iteration.
- Graph-level importance learning: The cross-attention mechanism aligns graph and structural representations, incurring $O(n^2d)$ complexity due to pairwise attention calculations.
- Graph representation alignment: Computing the L_2-norm alignment between structure-fused and original embeddings requires $O(nd)$ operations.

The total time complexity per training iteration is:

$$O(|E| + nd + n^2d) \tag{27}$$

Note that the truncated-SVD operation is considered a preprocessing step and involves decomposing the adjacency matrix $\mathcal{A}$ to obtain the top k singular vectors. Its time complexity is $O(kn^2)$ and space complexity is $O(kn)$, where k is the number of retained singular values. This cost does not contribute to the iterative training complexity and is incurred only once.

E.2 Time Complexity Comparison with Other Methods

In this subsection, we provide a concise summary of the theoretical time complexity of other methods.

- **GraphCL** costs $O(L|E|d^2 + nd^2 + N^2d)$ per iteration. This includes $O(L|E|d^2)$ for the GNN encoder over L layers, $O(nd^2)$ for the projection head, and $O(N^2d)$ for contrastive loss computation with minibatch size N. Graph augmentation adds minor costs ($O(n)$ to $O(|E_s|)$).
- **GraphMAE** consists of three main components. Masked feature reconstruction, including masking and re-masking, incurs $O(n)$ complexity for n nodes. The GNN encoder and decoder operate over L layers with a complexity of $O(L|E|d^2)$, where $|E|$ is the number

of edges and d is the embedding dimension. The scaled cosine error computation for feature reconstruction adds $O(nd)$. Thus, the overall time complexity per iteration is $O(L|E|d^2 + nd)$.

- **MaskGAE** per iteration includes $O(L|E_{\text{vis}}|d^2)$ for the GNN encoder processing the unmasked graph over L layers, $O(|E_{\text{mask}}|d)$ for the structure decoder reconstructing masked edges, and $O(nd)$ for the degree decoder approximating node degrees, where $|E_{\text{vis}}|$ and $|E_{\text{mask}}|$ are the numbers of unmasked and masked edges, respectively. Overall, the complexity is $O(L|E_{\text{vis}}|d^2 + |E_{\text{mask}}|d + nd)$.
- **TopoGCL** includes three main components. Extended persistence computation, involving simplicial complex operations, has a complexity of $O(N^3)$ for N nodes. The GNN encoder, applied over L layers, incurs $O(L|E|d^2)$ for $|E|$ edges and embedding dimension d. Contrastive loss computation for a minibatch of size M adds $O(M^2d)$. Thus, the overall complexity is $O(N^3 + L|E|d^2 + M^2d)$, with extended persistence dominating for dynamic graphs.
- AutoGCL incurs $O(L|E|d^2)$ for the GNN-based view generator and encoder operating over L layers with $|E|$ edges and embedding dimension d, and $O(N^2d)$ for contrastive loss computation with minibatch size N. Thus, the overall complexity is $O(L|E|d^2 + N^2d)$.

The time complexity comparison highlights the computational efficiency and scalability of CenPre compared to other graph learning methods. CenPre achieves a lower overall complexity of $O(|E| + nd + n^2d)$ by leveraging precomputation for structural representations, making it well-suited for large sparse graphs. In contrast, methods like GraphCL and AutoGCL incur higher costs due to the quadratic dependence on minibatch size ($O(N^2d)$) in contrastive loss computation, which can become a bottleneck for large-scale data. GraphMAE and MaskGAE share similar complexities, but their dependency on the number of masked and unmasked edges ($|E_{\text{vis}}|$ and $|E_{\text{mask}}|$) introduces sensitivity to the masking ratio. TopoGCL stands out with a cubic dependence on the number of nodes ($O(n^3)$) for extended persistence computation, making it less efficient for large graphs, though it provides rich topological insights. Overall, CenPre balances computational demands with performance by focusing on centrality measures and pretraining strategies, offering a practical advantage in terms of scalability and efficiency.

F LIMITATIONS

Despite its advantages, the CenPre framework has certain limitations. Firstly, its reliance on centrality measures may not fully capture complex, multi-faceted structural properties of graphs, potentially leading to sub-optimal representations in dense graphs and heterogeneous networks. Additionally, the computational complexity of calculating centrality measures, especially for large-scale graphs, can be high, impacting scalability and efficiency. Furthermore, the framework's performance may be sensitive to the choice of centrality measures, requiring careful tuning and selection based on specific graph characteristics. Future work should address these limitations by exploring more comprehensive structural descriptors, optimizing computational efficiency for broader applicability, and trying to explore an effective method to exploit the node importance in a more differentiated way.