
SSTAG: Structure-Aware Self-Supervised Learning Method for Text-Attributed Graphs

Ruyue Liu

Institute of Information Engineering, CAS
School of Cyberspace Security, UCAS
liuruyue@iie.ac.cn

Rong Yin*

Institute of Information Engineering, CAS
School of Cyberspace Security, UCAS
yinrong@iie.ac.cn

Xiangzhen Bo

Wuhan University of Technology
353145@whut.edu.cn

Xiaoshuai Hao

Xiaomi EV
haoxiaoshuai@xiaomi.com

Yong Liu

Renmin University of China
liuyonggsai@ruc.edu.cn

Jinwen Zhong

Institute of Information Engineering, CAS
zhongjinwen@iie.ac.cn

Can Ma

Institute of Information Engineering, CAS
macan@iie.ac.cn

Weiping Wang

Institute of Information Engineering, CAS
wangweiping@iie.ac.cn

Abstract

Large-scale pre-trained models have revolutionized Natural Language Processing (NLP) and Computer Vision (CV), showcasing remarkable cross-domain generalization abilities. However, in graph learning, models are typically trained on individual graph datasets, limiting their capacity to transfer knowledge across different graphs and tasks. This approach also heavily relies on large volumes of annotated data, which presents a significant challenge in resource-constrained settings. Unlike NLP and CV, graph-structured data presents unique challenges due to its inherent heterogeneity, including domain-specific feature spaces and structural diversity across various applications. To address these challenges, we propose a novel structure-aware self-supervised learning method for Text-Attributed Graphs (SSTAG). By leveraging text as a unified representation medium for graph learning, SSTAG bridges the gap between the semantic reasoning of Large Language Models (LLMs) and the structural modeling capabilities of Graph Neural Networks (GNNs). Our approach introduces a dual knowledge distillation framework that co-distills both LLMs and GNNs into structure-aware multilayer perceptrons (MLPs), enhancing the scalability of large-scale TAGs. Additionally, we introduce an in-memory mechanism that stores typical graph representations, aligning them with memory anchors in an in-memory repository to integrate invariant knowledge, thereby improving the model's generalization ability. Extensive experiments demonstrate that SSTAG outperforms state-of-the-art models on cross-domain transfer learning tasks, achieves exceptional scalability, and reduces inference costs while maintaining competitive performance.

*Corresponding author.

1 Introduction

In recent years, large-scale pre-trained models have achieved revolutionary breakthroughs in natural language processing (NLP) [1] and computer vision (CV) [2], demonstrating remarkable cross-domain generalization capabilities [3]. However, the prevailing paradigm in graph learning remains confined to training dedicated models for individual graph datasets [4, 5]. This single-graph modeling approach suffers from two major limitations: (1) models are typically restricted to single or narrowly defined tasks, lacking the ability to transfer knowledge across different graphs; and (2) model performance heavily depends on the scale of annotated data, yet acquiring high-quality labels is often costly and time-consuming, creating a significant bottleneck in low-resource scenarios.

The success of foundation models in language and vision stems from their domain invariance, supported by unified lexical or pixel spaces. In contrast, developing graph foundation models is challenging due to the intrinsic heterogeneity of graph-structured data. Graphs exhibit domain-specific node and edge types, with diverse feature and label spaces, making cross-domain alignment difficult. Moreover, structural diversity, such as acyclic citation networks versus multi-relational cyclic knowledge graphs, further complicates knowledge transfer across domains. To address these challenges, we leverage text as a unified representation medium for graph learning. Many real-world graphs are inherently text-attributed, where raw textual features provide a domain-agnostic semantic space. Large language models (LLMs) excel in semantic understanding and reasoning [6, 7, 8], but struggle with topological reasoning [9, 10], where graph neural networks (GNNs) excel. Conversely, GNNs lack the open-world knowledge embedded in LLMs, motivating a unified framework that bridges their complementary strengths.

To bridge this gap, we propose a novel **Structure-aware Self-supervised learning method for Text-Attributed Graphs**, called **SSTAG**. Specifically, to learn transferable invariants across graphs and tasks, we design a generic template that unifies various tasks by contextualizing the nodes, edges, and graphs for which we make predictions. For node or edge-level tasks on large-scale graphs, we employ the Personalized PageRank (PPR) algorithm to sample subgraphs, which mitigates the differences in graph structure across domains and enhances the scalability of the model. Additionally, we introduce a new pre-training objective of co-distilling language models (LMs) and graph neural networks (GNNs) into structure-aware multilayer perceptrons (MLPs), specifically tailored for self-supervised learning on large-scale task-attribute graphs (TAGs). This approach offers a dual advantage: (1) Through multimodal distillation, the MLP absorbs both the structural modeling capabilities of GNNs and the semantic reasoning abilities of LLMs. (2) The lightweight MLP circumvents the high computational overhead of LLMs, making it more suitable for practical deployment. This two-stage knowledge transfer paradigm not only overcomes the domain limitations of single graph models but also mitigates the structural processing limitations inherent in pure LLM approaches.

To summarize, our main contributions are as follows:

- (1) We propose a general-purpose graph learning framework that unifies node-, edge-, and graph-level prediction tasks within a single architecture. The unified design enables flexible adaptation and effective knowledge transfer across heterogeneous graph domains and diverse downstream tasks, overcoming the limitations of task-specific and domain-isolated models.
- (2) We design a novel self-supervised pretraining objective that distills complementary knowledge from large language models (LLMs) and graph neural networks (GNNs) into a structure-aware multi-layer perceptron (MLP), combining semantic reasoning with structural understanding while ensuring efficient inference.
- (3) Extensive experiments conducted on multiple benchmark datasets demonstrate the superiority of our proposed SSTAG framework: (a) it outperforms state-of-the-art baselines on cross-domain transfer learning tasks; (b) it exhibits remarkable scalability on large-scale graphs compared to existing GNN and LLM-based methods; (c) it significantly reduces inference cost while maintaining competitive performance.

2 Related Work

Representation Learning on TAGs Research on Text-Attributed Graphs (TAGs) lies at the intersection of graph machine learning and natural language processing. Early approaches focused on

shallow text-based enhancements for graph embeddings [11, 12], where textual features are treated as auxiliary node attributes within traditional graph algorithms. While computationally efficient, these methods fail to capture the deep semantic interplay between textual content and graph structures. Another class of graph learning models based on TAGs are LLMs-only approaches, such as LLaGA [13] and GraphGPT [14]. These methods leverage instruction tuning to map graph-structured data into the embedding space of large language models. The emergence of graph neural networks [15] revolutionizes TAGs processing by enabling end-to-end representation learning. For example, TAPE [16] leverages large language models to generate explanatory node descriptions, which are then used as enriched features for training GNNs. Graph-LLM [17] converts graph structures into textual sequences for downstream prediction via LLMs. Das et al. [18] explore the integration of graph data with LLMs, along with the influence of multi-modal representations. CaR [19] extracts textual captions from molecular SMILES strings using LLMs and feeds them into another language model for fine-tuning. However, they primarily rely on supervised training, which limits their applicability in low-resource or unlabeled scenarios.

Self-Supervised Learning on Graphs Self-supervised learning has emerged as a compelling paradigm for learning representations from graph-structured data without the need for explicit labeling. Existing work in this area can be broadly classified into two main categories: contrastive learning methods and generative methods. Contrastive learning methods aim to learn graph representations by maximizing the similarity between positive pairs while minimizing the similarity between negative pairs. GraphCL [20] has significantly advanced contrastive learning techniques by introducing various graph data augmentation strategies. These methods typically rely on effective strategies for pairing positive and negative samples, along with robust GNN architectures to extract meaningful graph features. More recently, methods like GPA [21] have introduced personalized graph enhancement strategies to further improve the quality of learned representations. Generative methods, on the other hand, focus on learning graph representations by predicting the missing or unobserved parts of the graph. For instance, GraphMAE [22] employs GNN-based encoders and decoders to reconstruct masked node features, while S2GAE [23] uses a similar approach to mask edges within the graph and predict the missing links. However, these methods remain confined to single-graph settings and face significant challenges in achieving cross-domain generalization.

3 Preliminaries

Text-Attributed Graphs Given a text-attributed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}_\mathcal{V}, \mathcal{T}_\mathcal{E}, \mathbf{A}\}$ with N nodes, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ represents the set of edges. For each node $v \in \mathcal{V}$, there is an associated text $t_v \in \mathcal{T}_\mathcal{V}$ that represents the node-level textual information. For each edge $e_{vu} \in \mathcal{E}$ connecting nodes v and u , there is an associated text $t_{e_{vu}} \in \mathcal{T}_\mathcal{E}$ that represents the edge-level textual information. The adjacency matrix is denoted as $\mathbf{A} \in \mathbb{R}^{N \times N}$. In this work, we focus on self-supervised learning on text-attributed graphs (TAGs). Specifically, the goal is to pre-train a mapping function $f_\theta : \mathcal{T}_\mathcal{V} \times \mathbf{A} \rightarrow \mathbb{R}^d$ or $\mathcal{T}_\mathcal{E} \times \mathbf{A} \rightarrow \mathbb{R}^d$ such that the semantic information in $\mathcal{T}_\mathcal{V}$ or $\mathcal{T}_\mathcal{E}$ and the topological information in $\mathbf{A}$ can be efficiently captured in a d -dimensional space in a self-supervised manner.

Graph Neural Networks For graph-structured data, Graph Neural Networks (GNNs) are commonly used to instantiate f_g . Specifically, the objective of GNNs is to update node representations by aggregating messages from their neighbors, as expressed by the following equation: $h_v^{(k)} = \text{COM} \left(h_v^{(k-1)}, \text{AGG} \left(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\} \right) \right)$, where $h_v^{(k)}$ represents the representation of node v at the k -th layer, and $\mathcal{N}(v) = \{u \mid \mathbf{A}_{v,u} = 1\}$ is the set of one-hop neighbors of node v . In particular, we have $h_v^{(0)} = x_v$, where $x_v = \text{Emb}(t_v) \in \mathbb{R}^F$ is a F -dimensional feature vector extracted from the textual attributes t_v of nodes, and $\text{Emb}(\cdot)$ denotes the embedding function. The AGG function is used to aggregate features from the neighbors, while the COM function combines the aggregated neighbor information with the own node embedding from the previous layer.

4 Proposed Method

In this section, we propose SSTAG, a novel framework designed to learn robust and informative graph representations by integrating structural and textual signals in a self-supervised manner. As shown in Figure 1, the proposed method comprises three key components: the Unified Graph Task (UGT)

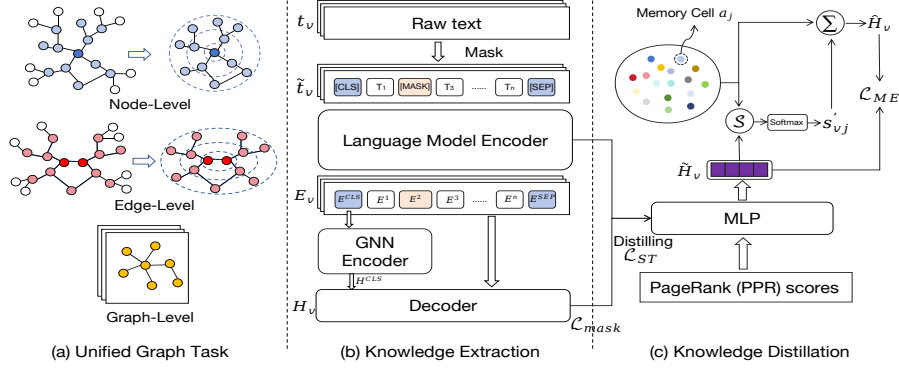


Figure 1: Overview of the proposed SSTAG framework.

module, the Knowledge Extraction from LLM (KEL) module, and the Knowledge Distillation(KD) module. Given a text-attributed graph, SSTAG first constructs a generic and task-agnostic self-supervised objective via the UGT module, which encodes both node structure and attribute semantics. Subsequently, the KEL leverages a LLM to capture high-level semantic representations from the node-associated textual attributes. These representations are aligned with graph-based representations obtained from a GNN. To effectively bridge the modality gap between language and graph features, we introduce a Knowledge Distillation module that transfers the complementary knowledge from both the LLM and the GNN into a lightweight MLP, enabling efficient downstream adaptation. Finally, the pre-trained SSTAG model can be fine-tuned for various downstream tasks at different granularity levels, such as node classification, link prediction, and graph classification.

4.1 Unified Graph Task

Graphs from different domains often exhibit diverse structural patterns and serve distinct application scenarios and task objectives. To address this heterogeneity, graph learning tasks are typically categorized into three levels based on structural granularity: node-level tasks, edge-level tasks, and graph-level tasks. Recent studies suggest that subgraph-based representations offer notable advantages. On one hand, they enhance the expressive capacity of models by incorporating richer local structures [24, 25]; on the other, they enable standardized task formulation across different levels [26]. Motivated by this, we adopt a unified representation format that leverages target nodes along with their corresponding context subgraphs.

Node-Level Tasks We design a subgraph sampling strategy that integrates the Personalized PageRank (PPR) algorithm [27]. For a given node v , its importance score π_v is computed as follows:

$$\pi_v = \alpha(\mathbf{I} - (1 - \alpha)\tilde{\mathbf{A}})^{-1}\mathbf{e}_v, \quad (1)$$

where $\mathbf{I}$ is the unit matrix, $\tilde{\mathbf{A}}$ denotes the normalized adjacency matrix, α is the teleport factor, and $\mathbf{e}_v$ is a one-hot vector corresponding to node v . During sampling, the probability of selecting a node u at the k -hop neighborhood is proportional to its relative importance score:

$$p_k(u) = \frac{\pi_{vu}}{\sum_{w \in \mathcal{N}_k(v)} \pi_{vw}}, \quad (2)$$

where $\mathcal{N}_k(v)$ denotes the set of k -hop neighbors of node v . Once the sampling is complete, we construct the subgraph by extracting all edges among the selected nodes. It ensures a higher probability of including structurally important nodes while preserving the local neighborhood structure.

Edge-Level Tasks For a target edge (u, v) , we first apply the node-level subgraph sampling strategy independently to each endpoint, generating two subgraphs $\mathcal{G}_u$ and $\mathcal{G}_v$. The final subgraph representation for the edge is obtained by taking the union of the two:

$$\mathcal{G}_{(u,v)} = \mathcal{G}_u \cup \mathcal{G}_v. \quad (3)$$

This approach effectively captures both the local context around each endpoint and the structural characteristics of the edge itself, making it well-suited for link prediction and other edge-level tasks.

Graph-Level Tasks For graph-level prediction tasks such as molecular property prediction, each graph instance is treated as a complete data sample without additional subgraph sampling. This is because the graph itself already represents a self-contained unit of information.

4.2 Knowledge Extraction from LLM

Most existing self-supervised learning methods for graphs adopt GNNs as their backbone architecture and rely on pre-processed node feature vectors as input [22, 28, 29]. However, these approaches often fall short of capturing the rich semantic information embedded within graphs, particularly when dealing with nodes that carry complex textual attributes. As previously discussed, Large Language Models excel at understanding and processing textual information, having been trained on diverse and extensive corpora. This enables them to acquire broad and transferable knowledge for interpreting natural language attributes in graph data. To fully exploit the complementary strengths of structural and textual information, we propose an end-to-end self-supervised learning framework for TAGs. Our method integrates a pre-trained Language Model and a GNN in a cascaded architecture that serves as a teacher model, enabling joint modeling of semantic and structural features. Specifically, we employ Sentence Transformers (ST) [30] as the language model and GCN [15] as the graph encoder. These two components collaboratively capture both the semantic content and topological structure of TAGs.

Inspired by the recent success of masked modeling techniques in natural language processing [31, 32], we design a text-based masked autoencoder framework to enable large-scale self-supervised pretraining on TAGs. By randomly masking portions of node textual attributes and requiring the model to recover the missing content based on contextual and neighborhood information, our approach effectively guides the model to learn latent semantic correlations and structural patterns. This pretraining strategy significantly enhances the model’s generalization ability and expressiveness for a variety of downstream tasks.

Masking Strategy During training, each batch processes a (sub)graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}_{\mathcal{V}})$, where $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ the set of edges, and $\mathcal{T}_{\mathcal{V}}$ the textual features associated with each node. To prepare the textual input, the text of each node is augmented with special tokens: a [CLS] token is added at the beginning to serve as the aggregate representation of the sentence (and thus the node), and a [SEP] token is appended at the end to indicate the end of the sequence.

Let t_v denote the raw textual feature of node $v \in \mathcal{V}$. After tokenization and augmentation, the tokenized input sequence becomes: $t_v = [\text{[CLS]}, T_1, T_2, \dots, T_{n_v}, \text{[SEP]}]$, where T_i are the tokens of the textual input and n_v is the number of tokens for node v . To enable self-supervised learning, we apply a token-level masking strategy inspired by masked language modeling. A subset of the tokens in each t_v is randomly selected and replaced with a special [MASK] token. This process is governed by a stochastic masking function $\mathcal{M}(\cdot)$, which determines which positions to mask in each token sequence. Formally, for each token sequence t_v , we generate a masked version $\tilde{t}_v$ such that:

$$\tilde{t}_v = \mathcal{M}(t_v) = [\text{[CLS]}, \tilde{T}_1, \tilde{T}_2, \dots, \tilde{T}_{n_v}, \text{[SEP]}], \quad (4)$$

where some $\tilde{T}_i$ are replaced with [MASK] tokens while others remain unchanged. The model is then trained to reconstruct the original tokens at the masked positions based on the surrounding textual context and the structural neighborhood encoded by the GNN. This encourages the model to learn deep semantic representations that are sensitive to both local graph topology and node-specific language attributes.

Encoder The teacher model comprises a language model (LM) f_{LM} and a graph neural network (GNN) f_{GNN} . For each node $v \in \mathcal{V}$, the masked textual feature sequence $\tilde{t}_v$ is encoded by the LM to obtain hidden representations:

$$\mathbf{E}_v = f_{\text{LM}}(\tilde{t}_v), \quad (5)$$

where $\mathbf{E}_v \in \mathbb{R}^{(n_v+2) \times d}$ is to the output embeddings of n_v subword tokens along with special tokens such as [CLS] and [SEP]. The embedding of the [CLS] token, denoted by $\mathbf{E}^{\text{cls}} \in \mathbb{R}^{|\mathcal{V}| \times d}$, is extracted as the initial representation of nodes. To incorporate structural information, $\mathbf{E}^{\text{cls}}$ is propagated through the GNN f_{GNN} over the adjacency matrix $\mathbf{A}$, yielding the fused representation $\mathbf{H}^{\text{cls}}$:

$$\mathbf{H}^{\text{cls}} = f_{\text{GNN}}(\mathbf{A}, \mathbf{E}^{\text{cls}}). \quad (6)$$

For each node v , we concatenate the textual embedding $\mathbf{E}_v$, obtained from the masked forward pass, with the GNN-enhanced [CLS] token representation $\mathbf{H}_v^{\text{cls}}$, followed by a linear transformation:

$$\mathbf{H}_v = \text{Linear}(\mathbf{E}_v \oplus (\mathbf{H}_v^{\text{cls}} \otimes \mathbf{1}_{n_v+2}^{\top})), \quad (7)$$

where $\mathbf{1}_{n_v+2} \in \mathbb{R}^{n_v+2}$ is a column vector of ones. The outer product $\mathbf{H}_v^{\text{cls}} \otimes \mathbf{1}_{n_v+2}^\top$ replicates the graph-aware node representation across all token positions, which matches the dimensionality of $\mathbf{E}_v \in \mathbb{R}^{(n_v+2) \times d}$. The symbol $\oplus$ denotes horizontal concatenation, resulting in a fused representation of shape $(n_v + 2) \times 2d$. The linear layer projects this fused matrix back to the original embedding space: $\text{Linear}(\cdot) : \mathbb{R}^{2d} \rightarrow \mathbb{R}^d$. Finally, a language modeling head (MLMHead), implemented as a multi-layer perceptron (MLP), maps the transformed embeddings into the vocabulary space to produce token-level prediction probabilities: $\mathbf{P}_v = \text{MLMHead}(\mathbf{H}_v)$.

4.3 Knowledge Distillation

To enable efficient and scalable deployment, we design a lightweight student model that approximates the teacher’s representations while preserving both semantic and structural information. Unlike the teacher model, which relies on explicit message passing, the student model incorporates graph structure implicitly through feature augmentation, thus significantly reducing computational overhead.

The student model adopts a structure-aware multilayer perceptron (MLP) to approximate the teacher’s representations. For masked textual feature sequence of each node $\tilde{t}_v$, the input to the student model is constructed by augmenting the [CLS] embedding $\mathbf{E}_v^{\text{cls}} \in \mathbb{R}^{1 \times d}$ with its corresponding Personalized PageRank (PPR) scores $p_v \in \mathbb{R}^{1 \times d_p}$ relative to its subgraph neighbors. Specifically, the PPR scores encode the relative importance of neighboring nodes and thereby inject structural information into the input features. The node representation is obtained by applying f_{MLP} over the concatenated features:

$$\tilde{\mathbf{H}}_v = f_{\text{MLP}}([\mathbf{E}_v^{\text{cls}} \parallel p_v]), \quad (8)$$

where $\parallel$ represents vector concatenation. By leveraging PPR-based structural priors, the student model can efficiently capture graph topology without relying on explicit message passing, enabling lightweight yet structure-aware representation learning.

Memory Bank To extract representative and diverse features, we introduce a memory bank that stores a set of prototypical representations throughout training. The memory bank comprises L fixed-size memory anchors $\{\mathbf{a}_j \in \mathbb{R}^d\}_{j=1}^L$, where each anchor serves as a prototype capturing typical embedding patterns of (sub)graphs.

Given a node v , we compute an activation score s_{vj} for each memory anchor $\mathbf{a}_j$, which quantifies the similarity between the input embedding and the stored prototypes. Specifically, the memory anchors $\mathbf{a}_j$ are initialized from a uniform distribution, following standard embedding initialization practices to ensure stable variance and prevent early model collapse.

During training, the memory anchors are progressively refined through attention-based interactions with incoming graph representations. The activation score s_{vj} is computed as:

$$s_{vj} = \mathcal{S}(\tilde{\mathbf{H}}_v, \mathbf{a}_j), \quad (9)$$

where $\mathcal{S}(\cdot, \cdot)$ denotes a distance or similarity metric. We then apply a softmax function over the L activation scores to obtain normalized scores s'_{vj} :

$$s'_{vj} = \frac{e^{s_{vj}}}{\sum_{k=1}^L e^{s_{vk}}}, \quad \hat{\mathbf{H}}_v = \sum_{j=1}^L s'_{vj} \mathbf{a}_j, \quad (10)$$

where $\hat{\mathbf{H}}_v \in \mathbb{R}^{1 \times d}$ denotes the reconstructed node embedding. The memory bank preserves invariant and semantically meaningful knowledge across training instances. By aligning graph embeddings with prototypical memory anchors, the model is encouraged to focus on stable and consistent features, mitigating overfitting and enhancing generalization to unseen graphs. This mechanism strengthens the model’s robustness and predictive capability, particularly in diverse or noisy graph scenarios.

4.4 Optimization Objectives

Mask Loss We adopt the Masked Language Modeling (MLM) objective for training. The underlying intuition behind this design is that the model can learn to reconstruct masked tokens of each node’s text by leveraging the textual information from its neighboring nodes. This encourages the model to simultaneously understand local semantic content and exploit the structural dependencies

within the graph. The training loss is computed using the cross-entropy loss function for each node $v \in \mathcal{V}$, targeting the prediction of the original tokens at the masked positions. The loss is defined as:

$$\mathcal{L}_{\text{mask}} = -\frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \sum_{i=1}^{n_v} \mathbf{I}(v, i) \cdot \log \mathbf{P}_v[i, T_i], \quad (11)$$

where $|\mathcal{V}|$ is the number of nodes, and $\mathbf{I}(v, i)$ is an indicator function, which equals 1 if the i -th token in the tokenized text of node v is a [MASK] token and 0 otherwise. $\mathbf{P}_v[i, T_i]$ denotes the predicted probability of assigning the ground-truth token T_i to the i -th position in the sequence of node v , as output by the model. By minimizing this loss, the model is trained to accurately recover masked tokens using the textual context and the structural information encoded in the graph, thereby fostering more informative and robust node representations.

Consistency Loss In addition to the masked language modeling (MLM) loss, we further introduce a consistency loss to impose regularization constraints on the latent space, thereby enhancing the stability and alignment of learned representations. The consistency loss consists of two components: one enforces alignment between the student model and the teacher model, while the other maintains consistency with memory-based anchors. For student-teacher consistency, we adopt cosine similarity to encourage the student model (typically a lightweight MLP) to produce embeddings close to those generated by the teacher model (the cascaded LM-GNN architecture). Specifically, given the student representation $\tilde{\mathbf{H}}_v$ and the teacher representation $\mathbf{H}_v$ for node v , the loss is formulated as:

$$\mathcal{L}_{\text{ST}} = 1 - \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \left(1 - \frac{\mathbf{H}_v^T \tilde{\mathbf{H}}_v}{\|\mathbf{H}_v\| \cdot \|\tilde{\mathbf{H}}_v\|} \right). \quad (12)$$

The memory consistency loss enables the model to update the corresponding memory anchors, thereby capturing invariant and prototypical knowledge about generalized graph representations. Specifically, the memory consistency loss is defined as:

$$\mathcal{L}_{\text{ME}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \left\| \hat{\mathbf{H}}_v - \tilde{\mathbf{H}}_v \right\|^2, \quad (13)$$

where $\hat{\mathbf{H}}_v$ denotes the original structure-aware graph embedding for node v , and $\tilde{\mathbf{H}}_v$ represents the aligned embedding. By minimizing this loss, the model is encouraged to preserve structural information within the learned representations and refine the memory anchors. This facilitates a more accurate encoding of the essential and invariant characteristics of the graph, thereby enhancing the model’s generalization ability across downstream tasks.

The overall loss is then a sum of these three components:

$$\mathcal{L} = \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{ST}} + \mathcal{L}_{\text{ME}}. \quad (14)$$

By jointly optimizing the MLM loss and the consistency loss, the model is encouraged to capture both semantic and structural information in a stable and generalizable way, leading to more robust node representations for downstream graph tasks.

Remark 4.1. *During inference, only the student model is employed to generate node embeddings. Given an unseen (sub)graph and a set of anchor nodes for which we aim to obtain representations, we first use the language model f_{LM} to encode the raw textual attributes of all nodes in the graph. Subsequently, the [CLS] tokens from each node are passed into the MLP module f_{MLP} to produce propagated representations, which serve as the final embeddings for each node. Finally, we extract the embeddings corresponding to the specified anchor nodes for downstream use.*

5 Experiments

5.1 Experimental Setting

We adopt the widely used linear probing protocol to evaluate the representation learning capability of the self-supervised pretraining models on unseen datasets. Specifically, we train a linear classifier on top of the frozen embeddings produced by the pre-trained models. Both our model and all baseline

Table 1: Experimental results for self-supervised representation learning. We report the accuracy (%) for the node classification task and the ROC - AUC score (%) for the link prediction task. The proposed method and other self-supervised benchmarks are pretrained on ogbn - Paper100M and then evaluated on individual target datasets. The best results are **bold**, and the second best are underlined.

	Node Classification (Accuracy, %)					Link Prediction (ROC-AUC, %)		
	Cora	Pubmed	ogbn-Arxiv	WikiCS	Products	FB15K237	WN18RR	ML1M
GCN	57.62 $\pm$ 0.21	55.18 $\pm$ 0.37	60.85 $\pm$ 0.13	53.24 $\pm$ 0.23	61.95 $\pm$ 0.32	72.52 $\pm$ 0.29	72.05 $\pm$ 0.31	66.64 $\pm$ 0.52
GIN	57.97 $\pm$ 0.45	48.98 $\pm$ 0.21	61.27 $\pm$ 0.25	52.32 $\pm$ 0.36	63.83 $\pm$ 0.15	73.60 $\pm$ 0.33	73.98 $\pm$ 0.39	65.71 $\pm$ 0.37
GAT	66.29 $\pm$ 0.24	57.30 $\pm$ 0.35	63.34 $\pm$ 0.49	50.91 $\pm$ 0.34	64.94 $\pm$ 0.28	72.14 $\pm$ 0.43	72.57 $\pm$ 0.65	66.89 $\pm$ 0.34
GraphCL	72.56 $\pm$ 0.52	67.27 $\pm$ 1.21	62.15 $\pm$ 0.21	55.96 $\pm$ 1.02	72.18 $\pm$ 0.42	65.34 $\pm$ 0.87	68.52 $\pm$ 0.55	67.02 $\pm$ 0.49
BGRL	74.42 $\pm$ 0.81	68.17 $\pm$ 0.22	69.04 $\pm$ 0.14	59.93 $\pm$ 0.35	73.08 $\pm$ 0.28	64.92 $\pm$ 0.36	66.47 $\pm$ 0.43	68.10 $\pm$ 0.22
GraphMAE	73.54 $\pm$ 0.38	68.38 $\pm$ 1.18	68.54 $\pm$ 0.20	54.68 $\pm$ 0.55	72.65 $\pm$ 0.62	62.87 $\pm$ 0.84	70.51 $\pm$ 0.32	68.57 $\pm$ 0.34
GraphMAE2	73.92 $\pm$ 0.64	68.76 $\pm$ 0.55	69.07 $\pm$ 0.27	58.04 $\pm$ 0.47	74.05 $\pm$ 0.33	60.54 $\pm$ 0.39	71.43 $\pm$ 0.11	69.13 $\pm$ 1.01
Graph-LLM	73.88 $\pm$ 0.35	68.62 $\pm$ 0.32	70.11 $\pm$ 0.52	62.16 $\pm$ 0.48	74.02 $\pm$ 0.34	82.47 $\pm$ 0.56	73.46 $\pm$ 0.61	70.21 $\pm$ 0.51
UniGraph	74.65 $\pm$ 0.56	70.84 $\pm$ 0.51	70.89 $\pm$ 0.44	65.47 $\pm$ 0.51	76.58 $\pm$ 0.44	85.01 $\pm$ 0.63	80.55 $\pm$ 0.27	70.02 $\pm$ 0.28
SSTAG (Ours)	75.09 $\pm$ 1.02	72.65 $\pm$ 0.35	72.85 $\pm$ 0.43	68.76 $\pm$ 0.62	78.27 $\pm$ 0.48	88.64 $\pm$ 0.49	82.42 $\pm$ 0.66	71.24 $\pm$ 0.42

Table 2: Experimental results for self-supervised representation learning. We report the ROC - AUC (%) for the graph classification task and RMSE ($\downarrow$) for the graph regression task. “ $\downarrow$ ” indicates that lower RMSE values correspond to better model performance. SSTAG and other self-supervised benchmarks are pretrained on ogbn-Paper100M and then evaluated on individual target datasets.

	Graph Classification (ROC-AUC, %)				Graph Regression (RMSE, $\downarrow$)		
	HIV	BBBP	BACE	MUV	esol	LIPO	CEP
GCN	74.15 $\pm$ 0.26	65.43 $\pm$ 0.33	69.02 $\pm$ 0.38	71.82 $\pm$ 0.26	1.379 $\pm$ 0.034	0.824 $\pm$ 0.034	1.342 $\pm$ 0.036
GIN	74.38 $\pm$ 0.24	66.07 $\pm$ 0.52	69.85 $\pm$ 0.32	72.35 $\pm$ 0.14	1.295 $\pm$ 0.021	0.819 $\pm$ 0.021	1.296 $\pm$ 0.015
GAT	73.82 $\pm$ 0.43	66.82 $\pm$ 0.15	68.51 $\pm$ 0.20	72.06 $\pm$ 0.42	1.324 $\pm$ 0.027	0.821 $\pm$ 0.027	1.305 $\pm$ 0.008
GraphCL	75.55 $\pm$ 0.29	68.74 $\pm$ 0.38	73.64 $\pm$ 0.56	74.27 $\pm$ 0.37	1.304 $\pm$ 0.024	0.763 $\pm$ 0.024	1.326 $\pm$ 0.016
BGRL	75.32 $\pm$ 0.44	67.35 $\pm$ 0.42	75.14 $\pm$ 0.21	75.13 $\pm$ 0.35	1.162 $\pm$ 0.018	0.784 $\pm$ 0.018	1.293 $\pm$ 0.021
GraphMAE	76.13 $\pm$ 0.12	69.51 $\pm$ 0.14	76.28 $\pm$ 0.43	75.88 $\pm$ 0.26	1.116 $\pm$ 0.015	0.754 $\pm$ 0.015	1.288 $\pm$ 0.008
GraphMAE2	77.84 $\pm$ 0.35	71.62 $\pm$ 0.25	77.41 $\pm$ 0.18	77.69 $\pm$ 0.42	1.069 $\pm$ 0.006	0.728 $\pm$ 0.006	1.262 $\pm$ 0.011
Graph-LLM	76.43 $\pm$ 0.20	72.54 $\pm$ 0.37	80.65 $\pm$ 0.33	76.13 $\pm$ 0.31	1.114 $\pm$ 0.024	0.719 $\pm$ 0.024	1.232 $\pm$ 0.009
UniGraph	77.27 $\pm$ 0.31	73.28 $\pm$ 0.30	79.23 $\pm$ 0.26	76.88 $\pm$ 0.52	1.090 $\pm$ 0.032	0.710 $\pm$ 0.032	1.195 $\pm$ 0.012
SSTAG (Ours)	79.52 $\pm$ 0.26	74.38 $\pm$ 0.35	82.06 $\pm$ 0.31	79.86 $\pm$ 0.40	1.043 $\pm$ 0.020	0.698 $\pm$ 0.003	1.186 $\pm$ 0.006

self-supervised methods are first pre-trained on the large-scale citation network ogbn-Paper100M. Subsequently, we evaluate the learned representations on twelve graph datasets spanning five distinct domains. For baselines, we compare our method with the state-of-the-art generative self-supervised methods for graphs: GraphMAE [22] and GraphMAE2 [33], contrastive methods such as GraphCL [20] and BGRL [34], and methods specifically tailored for TAGs, including UniGraph [35] and Graph-LLM [9]. Since most of these baselines are not originally designed for cross-domain evaluation, we use the ST language model to unify the input node features across different graphs. To ensure a fair comparison, all baselines employ GCN as the backbone GNN, consistent with our method. Detailed descriptions of the datasets, baselines and hyperparameter settings can be found in Appendix A.

5.2 Self-Supervised Representation Learning

We comprehensively evaluate the proposed SSTAG framework on four graph representative tasks: node classification, link prediction, graph classification, and graph regression. The overall experimental results are summarized in Tables 1 and 2, from which three key observations can be drawn. (1) Across all tasks and datasets, SSTAG consistently outperforms a wide range of existing graph self-supervised learning methods. This highlights its strong generalization capability in cross-domain graph learning scenarios, where data distributions and structural patterns vary significantly. By capturing richer structural and semantic dependencies, SSTAG generates more discriminative and transferable graph embeddings, enabling it to adapt effectively to unseen graphs. (2) As a standalone pretraining model, SSTAG exhibits remarkable performance when transferred to downstream tasks. In most cases, it achieves results comparable to, or even better than, fully supervised baselines, particularly in low-label regimes where annotated data is scarce. For instance, on the BACE graph classification dataset, SSTAG achieves an accuracy of 82.06% after fine-tuning, which surpasses the supervised baseline by 12.21%. This clearly demonstrates the label efficiency of our method, underscoring its potential in real-world scenarios where labeled graph data is often limited or costly to obtain. (3) Another important finding is that SSTAG benefits from a unified task template, which allows it to seamlessly adapt to tasks of different granularities, ranging from local node-level tasks to holistic graph-level tasks. This flexible adaptation mechanism contributes to its robust performance

Table 3: Ablation studies of key components.

	WikiCS	ogbn-Arxiv	FB15K237	MUV
SSTAG	68.76	72.85	88.64	79.86
W/o $\mathcal{L}_{\text{mask}}$	67.02	70.51	85.84	76.22
W/o $\mathcal{L}_{\text{ST}}$	67.75	71.86	87.12	78.65
W/o $\mathcal{L}_{\text{ME}}$	66.53	71.14	85.96	76.43
W/o GNN	64.34	69.53	84.32	70.57
W/o PPR	68.12	72.37	88.4	79.21

Table 4: Analysis of distillation and inference efficiency on ogbn-Arxiv.

Model	Inference Time (min)	Accuracy (%)	Parameters
GNN	13.4	73.91	7.1B
Distilled MLP	8.7	72.85	22M
Δ Change	$\downarrow$ 35.1%	$\downarrow$ 1.06%	$\downarrow$ 99.7%

Table 5: Analysis of LMs and GNNs choices.

	#parameters	ogbn-Arxiv	FB15K237	MUV
Sentence Transformer [30]	$\sim$ 66M	72.85	88.64	79.86
DeBERTa-v3-base [36]	$\sim$ 184M	72.53	88.83	79.54
E5-large-v2 [37]	$\sim$ 335M	73.21	89.02	80.01
LLaMA-2-7B-hf [38]	$\sim$ 7B	73.68	89.67	80.39
GCN	—	72.85	88.64	79.86
GIN	—	72.43	89.13	80.04
GAT	—	73.02	88.92	80.33

in complex multi-task settings, where heterogeneous objectives must be addressed simultaneously. The ability to transfer knowledge across tasks and granularities further underscores the scalability and versatility of the proposed framework.

5.3 Ablation Studies

Ablation on Key Components We perform an ablation study to evaluate the contribution of each component in SSTAG, with results summarized in Table 3. The variant “W/o $\mathcal{L}_{\text{mask}}$ ” removes the masked modeling objective and trains the model only with the consistency loss, and the performance drop highlights the role of masked modeling in capturing fine-grained structural and semantic dependencies. The variant “W/o $\mathcal{L}_{\text{ST}}$ ” discards the student–teacher consistency while retaining masked modeling and memory-based consistency, and the degradation indicates that the student–teacher design is crucial for stabilizing training and improving cross-view alignment. In contrast, “W/o $\mathcal{L}_{\text{ME}}$ ” eliminates memory-based consistency while keeping masked modeling and student–teacher consistency, and the observed decline suggests that memory-based consistency is effective in preserving long-range dependencies and mitigating representation drift. The setting “W/o GNN” replaces the graph neural network encoder with a standard MLM objective for fine-tuning the language model, followed by distillation into an MLP, and the substantial performance loss demonstrates the necessity of graph-structured message passing for relational reasoning. Finally, “W/o PPR Sampling” substitutes personalized PageRank-based sampling with simple neighborhood sampling, and the reduction confirms that PPR-based sampling provides more informative subgraph contexts and alleviates bias from naive neighborhood expansion. Overall, the consistent degradation across all variants demonstrates that each component is indispensable, and their joint design is critical to the effectiveness and robustness of SSTAG.

Impact of Model Distillation We further evaluate the effectiveness and efficiency of the distilled MLP model in comparison to the original GNN teacher. Specifically, we measure inference accuracy, computational efficiency, and model size. The distilled MLP achieves a 35.1% improvement in inference efficiency and a 99.7% reduction in the number of parameters, while incurring only a minor accuracy drop of 1.06% relative to the GNN. These results demonstrate that our distillation approach preserves the representational power of the teacher model while substantially reducing computational and memory overhead. As shown in Table 4, the distilled MLP consistently maintains high accuracy across multiple downstream tasks, confirming that the co-distillation strategy effectively transfers knowledge from the GNN to the lightweight student model without sacrificing task performance.

Analysis of LMs and GNNs Choices Table 5 presents a comparative analysis of different language models (LMs) and GNNs as backbone architectures. To assess the impact of LMs, we evaluate several widely used pre-trained models. Compared with Sentence Transformers (ST, 110M parameters), larger models such as E5-large-v2 (335M) [37] and LLaMA-2-7B-hf (7B) [38] yield consistent improvements. For instance, replacing ST with LLaMA-2-7B-hf on the ogbn-Arxiv node classifica-

Table 6: Comparison of computational cost and performance of different methods.

Dataset	Method	Pre-training	Downstream Training	Downstream Inference	Accuracy(%)
ogbn-Arxiv	GAT	—	24.6mins	5.8min	63.34
	GraphCL	—	32.6mins	4.9min	62.15
	GraphMAE2	—	5.2h	5.1min	68.76
	Graph-LLM	24.2h	—	12.6min	72.85
	SSTAG (Ours)	22.6h	—	8.7min	72.85

tion task improves accuracy by 0.82%, highlighting the benefit of higher-capacity LMs. However, these gains incur significantly higher computational and memory costs, revealing an inherent trade-off between accuracy and efficiency. Hence, LM selection should balance task requirements and available resources: smaller models suit low-latency or resource-limited settings, while larger ones favor accuracy-oriented applications. For GNNs, models with stronger aggregation capability, such as GraphSAGE and GAT, generally outperform simpler architectures like GCN, indicating that expressive structural encoders complement high-capacity LMs in enhancing downstream performance. Further LM configuration details are provided in Appendix A.

5.4 Efficiency Analysis

The overall time complexity of the proposed method is primarily dominated by the language model (LM), owing to its long-sequence processing. During pretraining, the computational cost is approximately $\mathcal{O}(N \cdot (L^2d + L_t d^2))$, where N denotes the number of nodes, L_t the input sequence length, and d the embedding dimension. The neighborhood aggregation in the graph neural network (GNN) introduces an additional overhead of $\mathcal{O}(Nd^2 + Ed)$, where E is the number of edges; in dense graphs ($E \propto N^2$), this can grow to $\mathcal{O}(N^2d)$. In the student model, explicit message passing is replaced by structure-aware MLPs with PPR-based feature injection, thereby reducing the complexity to $\mathcal{O}(Nd)$. Memory retrieval incurs an additional cost of $\mathcal{O}(NLD)$, where L is the number of memory anchors. Other components, such as masked prediction in multimodal interaction ($\mathcal{O}(n_{\text{masked}} \cdot n_v)$, where n_{masked} is the number of masked tokens) and consistency loss ($\mathcal{O}(Nd)$), introduce relatively minor computational overhead. Overall, the dominant factor remains the LM complexity, while the student model and auxiliary modules are designed to maintain scalability in large-scale graph settings.

As shown in Table 6, the training time and memory overhead of SSTAG are comparable to those of training a language model (LM) using only the masked language modeling (MLM) objective. This suggests that the overall computational cost of our framework is primarily dominated by the LM. Consequently, when using similar LMs, the runtime of SSTAG is on par with other LM-based approaches. SSTAG is designed as a pretraining-centric model, where most of the computational cost is incurred during the pretraining phase. However, it offers a key advantage at inference time by allowing the use of a distilled student model (structure-aware MLP) resulting in significantly lower inference overhead. We further compare the training and inference costs of our model with GNN-based methods. We conduct experiments on two datasets of different scales: ogbn-arXiv and WikiCS. Although SSTAG incurs longer pretraining time, its inference time on downstream datasets is comparable to or even shorter than the combined training and inference time of GNN-based methods. This advantage becomes more pronounced as the size and number of downstream datasets increase. While LMs generally have larger parameter counts, our framework mitigates this drawback by requiring only forward passes during downstream inference, thereby avoiding the additional memory overhead of backpropagation during training.

6 Conclusion

In this work, we propose **SSTAG**, a structure-aware self-supervised framework tailored for text-attributed graphs, aiming to bridge the gap between the structural reasoning strengths of GNNs and the semantic understanding capabilities of LLMs. By leveraging text as a unified medium, SSTAG tackles the challenge of knowledge transfer across heterogeneous graph domains. Our approach introduces a generic prediction template for node-, edge-, and graph-level tasks, along with a novel co-distillation objective that fuses multimodal knowledge into a lightweight, structure-aware MLP. Extensive experiments demonstrate that SSTAG not only achieves superior performance across cross-domain and large-scale settings but also substantially reduces inference costs, making it a promising direction for practical and scalable graph representation learning.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (No.62106259, No.62076234), Beijing Outstanding Young Scientist Program (NO.BJJWZYJH012019100020098), and Beijing Natural Science Foundation (No. 4222029).

References

- [1] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.
- [2] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17980–17989, 2022.
- [3] Zhen Wang, Zhifeng Gao, Hang Zheng, Linfeng Zhang, Guolin Ke, et al. Exploring molecular pretraining model at scale. *Advances in Neural Information Processing Systems*, 37:46956–46978, 2024.
- [4] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. Learning to pre-train graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4276–4284, 2021.
- [5] 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*, pages 1857–1867, 2020.
- [6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [7] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, 18(6):1–32, 2024.
- [8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.
- [9] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.
- [10] Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, 2023.
- [11] Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. *Advances in neural information processing systems*, 36:13308–13325, 2023.
- [12] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE transactions on knowledge and data engineering*, 29(12):2724–2743, 2017.
- [13] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llava: large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, pages 7809–7823, 2024.
- [14] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500, 2024.

- [15] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.
- [16] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.
- [17] Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.
- [18] Debarati Das, Ishaan Gupta, Jaideep Srivastava, and Dongyeop Kang. Which modality should I use - text, motif, or image? : Understanding graphs with large language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 503–519, 2024.
- [19] Chen Qian, Huayi Tang, Zhirui Yang, Hong Liang, and Yong Liu. Can large language models empower molecular property prediction? *arXiv preprint arXiv:2307.07443*, 2023.
- [20] 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.
- [21] Xin Zhang, Qiaoyu Tan, Xiao Huang, and Bo Li. Graph contrastive learning with personalized augmentation. *IEEE Transactions on Knowledge and Data Engineering*, 2024.
- [22] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 594–604, 2022.
- [23] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. 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*, pages 787–795, 2023.
- [24] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.
- [25] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pages 417–428, 2023.
- [26] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024.
- [27] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.
- [28] Ruyue Liu, Rong Yin, Yong Liu, and Weiping Wang. Aswt-sgcn: Adaptive spectral wavelet transform-based self-supervised graph neural network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 13990–13998, 2024.
- [29] Ruyue Liu, Rong Yin, Yong Liu, and Weiping Wang. Unbiased and augmentation-free self-supervised graph representation learning. *Pattern Recognition*, 149:110274, 2024.
- [30] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023.
- [31] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

- [32] Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. Pre-training and prompting for few-shot node classification on text-attributed graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4467–4478, 2024.
- [33] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*, pages 737–746, 2023.
- [34] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Rémi Munos, Petar Veličković, and Michal Valko. Bootstrapped representation learning on graphs. In *ICLR 2021 workshop on geometrical and topological representation learning*, 2021.
- [35] Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins, editors, *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025*, pages 448–459, 2025.
- [36] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.
- [37] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.
- [38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [39] 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.
- [40] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.
- [41] Haiteng Zhao, Shengchao Liu, Ma Chang, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. *Advances in neural information processing systems*, 36:5850–5887, 2023.
- [42] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019*, 2019.
- [43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, 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*, 2018.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: See Section 1.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: See Section 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#) .

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: See Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Section 5.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: See Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See Section 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conducted in the dissertation complies in all respects with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [No]

Justification: This paper is a fundamental study of dynamic graph data, not related to a specific application, and does not address the societal impacts.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This thesis does not present a high risk of misuse or dual use.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Assets used in the paper have been appropriately noted.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: New assets introduced in the document are well documented and available with the asset.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: Crowdsourcing experiments and studies with human subjects are not included in this paper.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Details of Experiments

The supplementary material provides additional details on the experiments section that could not be included in the main manuscript due to page limitations. All experiments were conducted on a Linux server equipped with 945GB of RAM and eight NVIDIA A100 GPUs, each with 40GB of memory. The implementation of our method is available at <https://github.com/Liury925/SSTAG>.

A.1 Datasets

In this section, we describe the datasets used in this work. The overall statistics for each dataset are given in Table 7.

Cora The Cora [9] dataset represents a co-citation graph of academic papers in the field of computer science. In *Graph-LLM*, the authors reconstruct this dataset because the commonly used Cora version in the GNN community relies on bag-of-words features, making it difficult to retrieve the original text. The newly collected Cora dataset contains 2,708 nodes and 10,556 edges, maintaining the same graph structure as the original version.

PubMed The PubMed [26] dataset is a co-citation graph of biomedical research papers focused on diabetes mellitus. The data source and processing procedure follow the same approach as the Cora dataset. After preprocessing, the dataset contains 19,717 nodes and 88,648 edges. For the node classification task, nodes are categorized into three classes: *Diabetes mellitus, experimental*, *Diabetes mellitus, type 1*, and *Diabetes mellitus, type 2*. The standard train/validation/test split consists of 60 training nodes, 500 validation nodes, and 19,157 test nodes.

ogbn-Arxiv The Arxiv [39] dataset is a large-scale citation graph constructed from academic papers published on the arXiv platform. The graph comprises 169,343 nodes and 1,166,243 edges. It is primarily used for the node classification task, where each node corresponds to a paper, and edges represent citation relationships. The dataset includes a total of 40 distinct classes. The standard data split contains 90,941 training, 29,799 validation, and 48,603 test nodes.

ogbn-Papers100M The ogbn-Papers100M [39] dataset is part of the Open Graph Benchmark (OGB) and contains over 111 million nodes and 1.6 billion edges. Each node represents a paper from the Microsoft Academic Graph, and edges denote citation relationships. The task is node classification, where the goal is to predict the field of study for each paper. Due to its massive scale, the dataset is designed to evaluate the scalability and efficiency of graph learning algorithms.

WikiCS The WikiCS [26] is a graph dataset constructed from the English Wikipedia, where nodes correspond to articles and edges represent hyperlink connections. Each article is associated with textual features and is labeled by one of several pre-defined classes. The task is semi-supervised node classification, and it includes 10 different training/validation/test splits, allowing for robust evaluation under few-shot settings.

Products The Products [39] dataset is part of the Amazon co-purchase graph, where nodes are products and edges connect products frequently bought together. It is included in the OGB benchmark as ogbn-products. Each node is associated with a multi-hot encoded feature vector and a category label. The dataset is used for node classification, with over 2 million nodes and 60+ classes.

FB15K237 FB15k237 [26] is a commonly used benchmark in knowledge graph completion tasks. It is a refined version of the original FB15k dataset, which was curated from Freebase. The refinement removes inverse relations to avoid test leakage. The dataset includes entities as nodes and relations as labeled edges, and the primary task is link prediction or knowledge graph completion.

WN18RR WN18RR [26] is a benchmark knowledge graph dataset derived from WordNet. It is a variant of WN18 with inverse relations removed to prevent test leakage. The graph consists of entities and labeled edges representing lexical relationships such as hypernymy and synonymy. It is widely used for evaluating link prediction models in knowledge graphs.

Table 7: Statistics of text-attributed graph datasets.

Dataset	Avg. #N	Avg. #E	#G	Task level	Task(class)	Domain	Split (train/val/test)
Cora	2,708	10,556	1	Node	classification(7)	Citation	140/500/2,068
Pubmed	19,717	88,648	1	Node	classification(3)	Citation	60/500/19,157
ogbn-Arxiv	169,343	1,166,243	1	Node	classification(40)	Citation	90,941/29,799/48,603
ogbn-Papers100M	111,059,956	1,615,685,872	1	Node	classification(172)	Citation	1,196,087/125,265/214,326
WikiCS	11,701	216,123	1	Node	classification(10)	Web link	580/1,769/5,847
Products	54,025	144,638	1	Node	classification(47)	Co-purchase	14,695/1,567/36,982
fb15k237	14,541	310,116	1	Link	classification(237)	Knowledge	272,115/17,535/20,466
WN18RR	40,943	93,003	1	Link	classification(11)	Knowledge	86,835/3,034/3,134
ML1M	9,923	2,000,418	1	Link	classification(5)	Movie rating	850,177/50,011/100,021
HIV	25.51	54.94	41,127	Graph	classification(2)	molecular	32,901/4,113/4,113
BBBP	24.06	51.91	2,039	Graph	classification(2)	molecular	1,631/204/204
BACE	34.09	73.72	1,513	Graph	classification(2)	molecular	1,210/151/152
MUV	24.23	52.56	93,087	Graph	classification(17)	molecular	74,469/9,309/9,309
ESOL	13.29	27.35	1,128	Graph	Regression	molecular	902/113/113
CEP	38.02	41.00	29978	Graph	Regression	molecular	23,982/2,998/2,998
LIPO	27.04	59.00	4,200	Graph	Regression	molecular	3,360/420/420

ML1M The MovieLens-1M [40] dataset is a widely used benchmark for recommender systems. It can be represented as a bipartite user-item interaction graph. Node features include user and item attributes such as age, gender, occupation, and genres. The typical task is rating prediction or top-k recommendation.

HIV The HIV [41] dataset is a molecular graph classification dataset from the MoleculeNet benchmark. Each molecule is represented as a graph, where atoms are nodes and bonds are edges. The binary classification task is to predict whether a molecule is active against HIV. The dataset is used to evaluate models in molecular property prediction.

BBBP The BBBP [41] dataset is a binary classification dataset that predicts whether a given compound can penetrate the blood-brain barrier. Each data point is a molecular graph with atom-level features. This dataset is particularly relevant for drug discovery applications and poses a challenge due to its relatively small size and imbalanced labels.

BACE The BACE [41] dataset contains molecular graphs used to predict the binding results of human β -secretase 1 (BACE-1) inhibitors. It is a binary classification task that plays a role in early-stage drug development, especially for Alzheimer’s disease. Graph-based models leverage atom and bond features to make predictions.

MUV The MUV (Maximum Unbiased Validation) [41] dataset is designed to serve as a challenging benchmark for virtual screening. It includes a collection of molecular graphs with multiple binary classification tasks, each corresponding to a biological target. The dataset is highly imbalanced and contains a significant number of decoys, making it suitable for testing model robustness.

ESOL The ESOL [41] dataset is used for regression tasks where the goal is to predict the aqueous solubility of compounds. Molecules are represented as graphs, and the target is a continuous solubility value. This dataset is important for evaluating models in pharmaceutical and materials chemistry.

CEP The CEP (Clean Energy Project) [41] dataset comprises molecular graphs of organic photo-voltaic compounds. Each molecule has a computed power conversion efficiency (PCE), making the task a regression problem. It is one of the largest publicly available molecular property datasets and is critical for materials discovery in renewable energy research.

LIPO The LIPO [41] dataset is a molecular property prediction dataset where the target is the logarithm of the partition coefficient between octanol and water (logP), reflecting the molecule’s lipophilicity. It is a regression dataset used in computational chemistry and drug design, where accurate logP prediction is essential for pharmacokinetics modeling.

A.2 Baselines

GCN Graph Convolutional Network (GCN) [15] introduces convolutional operations into graph-structured data. It aggregates features from a node’s neighbors and itself, enabling effective semi-

supervised learning on graph data. GCN is widely used in node classification tasks and serves as the backbone for many subsequent GNN models.

Code: <https://github.com/tkipf/gcn>

GIN Graph Isomorphism Network (GIN) [42] is designed to have maximum expressive power among GNNs, equivalent to the Weisfeiler-Lehman test for graph isomorphism. By using a summation-based aggregation and MLP update, GIN can effectively distinguish different graph structures.

Code: <https://github.com/weihua916/powerful-gnns>

GAT Graph Attention Network (GAT) [43] applies attention mechanisms to assign different importances to different neighbors during message passing. This allows the model to better capture local structural variations and learn more robust node embeddings.

Code: <https://github.com/PetarV-/GAT>

GraphCL GraphCL [20] is a contrastive self-supervised learning framework for graphs. It generates multiple augmented graph views via structural and attribute perturbations and maximizes agreement between their representations. GraphCL has shown competitive performance in unsupervised graph classification.

Code: <https://github.com/Shen-Lab/GraphCL>

BGRL BGRL [34] is a bootstrap-based self-supervised method that eliminates the need for negative samples. Inspired by BYOL, it uses two networks—an online and a target network—to predict node embeddings across augmented views. This method is memory-efficient and stable on large graphs.

Code: <https://github.com/nerdslab/bgrl>

GraphMAE GraphMAE [22] is a masked autoencoder designed for graphs, inspired by BERT-style pretraining. It masks parts of node features and learns to reconstruct them using a GNN backbone, enabling effective pretraining for downstream tasks.

Code: <https://github.com/THUDM/GraphMAE>

GraphMAE2 GraphMAE2 [33] is an enhanced version of GraphMAE with improved masking strategies and decoder designs. Using GAT as the encoder, it introduces better training stability and performance in graph representation learning.

Code: <https://github.com/THUDM/GraphMAE-v2>

Graph-LLM Graph-LLM [9] is a framework designed to bridge graph representation learning and large language models (LLMs). It introduces a graph-to-text conversion pipeline that transforms graph-structured data into natural language sequences, enabling pretrained LLMs to reason over and extract knowledge from graphs. Graph-LLM supports both node-level and graph-level tasks by prompting the LLMs with rich textual contexts that reflect topological and semantic information. This approach bypasses the need for message passing in traditional GNNs, offering a scalable alternative for graph-based learning.

Code: <https://github.com/CurryTang/Graph-LLM>.

UniGraph UniGraph [35] proposes a unified pretraining framework for graph-level, node-level, and edge-level tasks. By designing a universal contrastive learning objective and architecture, UniGraph generalizes well across diverse graph tasks.

Code: <https://github.com/Graph-COM/UniGraph>

A.3 Hyperparameter Setting

The hyperparameter tuning process in this work is divided into three categories. First, some hyperparameters (such as the number of epochs, learning rate, optimizer, and batch size) are selected empirically based on standard practice. Second, certain hyperparameters (such as the masking rate and the number of memory anchors) are optimized using grid search on the validation split, with metrics like accuracy or AUC depending on the specific task. Third, for parameters with complex interactions, we select the best values based on cross-validation across multiple settings. The selection

criteria and the actual values used in our experiments are summarized in Table 8. It is worth noting that optimal values may vary slightly across different datasets.

Table 8: Summary of hyperparameters and tuning criteria.

Hyperparameter	Value	Tuning Criterion (Search Range)
Mask Rate	0.5	Grid Search (0.1, 0.3, 0.5, 0.7)
Num GNN Layers	3	Empirical
Hidden Size	768	Empirical
PPR Top-k	128	Cross-validation (64, 128, 256)
Learning Rate	2e-5	Empirical
PPR α	0.15	Cross-validation (0.10, 0.15, 0.20, 0.25)
Weight Decay	0.001	Empirical
Batch Size	1024	Empirical
Dropout	0.2	Grid Search (0.1, 0.2, 0.3, 0.4)
Optimizer	AdamW	Empirical
Num Epochs	1	Empirical
Warmup Steps	10%	Empirical
Num MLP Layers	3	Empirical
Memory Anchors	256	Grid Search (64, 128, 256, 512)

A.4 Language Models

Sentence Transformer Sentence Transformers [30] are a family of models that extend pretrained transformers like BERT to generate semantically meaningful sentence embeddings. By fine-tuning on natural language inference and paraphrase datasets using Siamese or triplet networks, Sentence Transformers enable efficient semantic similarity search, clustering, and information retrieval.

HuggingFace: <https://huggingface.co/sentence-transformers>

DeBERTa-v3-base DeBERTa [36] improves BERT and RoBERTa by disentangling the representation of content and position, and using an enhanced mask decoder. The v3 version incorporates further improvements such as better initialization and larger-scale training. DeBERTa-v3-base has around 140M parameters and achieves strong performance on various NLU benchmarks.

HuggingFace: <https://huggingface.co/microsoft/deberta-v3-base>

E5-large-v2 E5-large-v2 [37] (Embedding-from-Embedding) is a dual-encoder model developed by Google for high-quality semantic search and retrieval tasks. It is fine-tuned on a mixture of supervised and unsupervised datasets with contrastive loss to produce universal embeddings. The "large-v2" version contains approximately 355M parameters and supports both query and passage encoding.

HuggingFace: <https://huggingface.co/intfloat/e5-large-v2>

LLaMA-2-7B-hf LLaMA 2 [38] is a series of open foundation language models released by Meta. LLaMA-2-7B-hf is the 7-billion-parameter variant and is suitable for a wide range of NLP tasks, including generation, question answering, and dialogue. The Hugging Face version provides easy integration with the Transformers library.

HuggingFace: <https://huggingface.co/meta-llama/Llama-2-7b-hf>