

Quantizing Text-attributed Graphs for Semantic-Structural Integration

Jianyuan Bo

Singapore Management University
Singapore, Singapore
jybo.2020@phdcs.smu.edu.sg

Hao Wu*

Beijing Normal University
Beijing, China
wuhan@bnu.edu.cn

Yuan Fang*

Singapore Management University
Singapore, Singapore
yfang@smu.edu.sg

Abstract

Text-attributed graphs (TAGs) have emerged as a powerful representation for modeling complex relationships across diverse domains. With the rise of large language models (LLMs), there is growing interest in leveraging their capabilities for graph learning. However, current approaches face significant challenges in embedding structural information into LLM-compatible formats, requiring either computationally expensive alignment mechanisms or manual graph verbalization techniques that often lose critical structural details. Moreover, these methods typically require labeled data from source domains for effective transfer learning, significantly constraining their adaptability. We propose STAG, a novel self-supervised framework that directly quantizes graph structural information into discrete tokens using a frozen codebook. Unlike traditional quantization approaches, our method employs soft assignment and KL divergence guided quantization to address the unique challenges of graph data, which lacks natural tokenization structures. Our framework enables both LLM-based and traditional learning approaches, supporting true zero-shot transfer learning without requiring labeled data even in the source domain. Extensive experiments demonstrate state-of-the-art performance across multiple node classification benchmarks while maintaining compatibility with different LLM architectures, offering an elegant solution to bridging graph learning with LLMs.

CCS Concepts

- Computing methodologies → Artificial intelligence; Unsupervised learning; Neural networks.

Keywords

text-attributed graphs, large language models, few-shot, zero-shot, graph transfer learning

ACM Reference Format:

Jianyuan Bo, Hao Wu, and Yuan Fang. 2025. Quantizing Text-attributed Graphs for Semantic-Structural Integration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25)*, August 3–7, 2025, Toronto, ON, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3711896.3737096>

*Corresponding authors.



This work is licensed under a Creative Commons Attribution 4.0 International License.
KDD '25, Toronto, ON, Canada

© 2025 Copyright held by the owner/authors(s).
ACM ISBN 979-8-4007-1454-2/2025/08
<https://doi.org/10.1145/3711896.3737096>

1 Introduction

Graphs serve as a cornerstone for modeling and understanding complex relationships across diverse domains, from social media [13, 30, 42] and knowledge graphs [3, 54, 60] to recommendation systems [11, 19, 59]. The structural information inherent in graphs is critical for effective graph learning, driving the development of graph neural networks (GNNs) [56]. Meanwhile, many real-world graphs contain textual descriptions, such as paper abstracts in citation networks [65], and product descriptions in co-purchase networks [38]. Such graphs are known as *text-attributed graphs* (TAGs) [61, 63], in which nodes or edges are associated with rich textual content. However, conventional GNNs struggle to effectively utilize raw text in TAGs. With the rise of pre-trained language models (PLMs) [17, 29] and large language models (LLMs) [1, 4, 14, 22, 53, 64], there is growing interest in combining LLMs with graph learning, termed as *GraphLLM* [6]. Among existing GraphLLM studies, processing TAGs represents an important direction due to the abundance of both semantic and structural information in TAGs. Particularly, text attributes can be processed by LLMs to obtain semantically rich initial features, offering significant advantages over traditional shallow features.

Limitations of Prior Work. Despite the potential of GraphLLM in processing TAGs, a fundamental challenge remains: how to effectively unify both semantic and structural information, integrating them into formats that LLMs can utilize. The challenges arise from the misalignment between continuous embedding spaces used for structural encoding and the discrete token spaces native to LLMs. As a result, current approaches particularly struggle with integrating neighborhood structures into the textual semantics in LLM-compatible formats. Graph prompting approaches [35] primarily use LLMs for semantic feature extraction, then manually construct prompt nodes or ego-networks to capture neighborhood structures, which are subsequently encoded using GNNs. However, they fail to leverage the powerful inference capabilities of LLMs. Hence, some methods have sought a deeper integration of structure-based GNN embeddings with LLMs, relying on computationally expensive and LLM-specific projectors for alignment [50]. Additionally, such methods often resort to manually designed subgraph verbalization techniques to describe the neighborhood to LLMs (e.g., “central node A, linked to B and C”) [66, 70], which not only hinders scalability to large neighborhoods but also introduces inconsistency and instability across different LLMs [15].

The second challenge lies in cross-dataset transfer learning. To overcome negative transfer and feature misalignment across different datasets [49], additional labeled data in source or target domains are often used. ZeroG [33] fine-tunes PLMs to align structurally-aware embeddings with semantic features based on labeled data

in the source datasets. Similarly, OFA [35] requires training with labeled data from various domains to enable effective transfer learning. The labeling requirement can be costly and significantly constrain adaptability across diverse downstream tasks, while supervised fine-tuning of PLMs is computationally expensive and offers limited generalization across domains [33].

Proposed Work. These challenges point to a critical need for a fundamentally different approach to embedding structural information of TAGs in GraphLLMs. Directly mapping graph embeddings into discrete token spaces would eliminate the need for expensive alignment mechanisms and manual graph verbalization, enabling native compatibility across different LLMs while integrating both structural and semantic information.

Inspired by vector quantization techniques like VQ-VAE [55], which quantize images into discrete tokens for LLM processing [71], we address the challenges of GraphLLM through a novel approach that fundamentally shifts from continuous embeddings to discrete tokens for TAGs. Our key insight is using quantization techniques to encode nodes into discrete tokens that seamlessly integrate structural and semantic information. However, quantizing graph structural information presents unique opportunities that distinguish our work from quantization in computer vision. First, unlike in computer vision, where feature maps provide natural structures for tokenization [71], graphs lack inherent patterns for encoding neighborhood information to achieve semantic-structural integration. Second, naïve quantization approaches risk overfitting to pre-training datasets, potentially hindering transferability. Overfitting is less common in computer vision, which benefits from large-scale, diverse pre-training datasets [8, 34]. In contrast, graph domains often lack such comprehensive pre-training data, making cross-dataset transfer challenging without additional labeled data.

To overcome these issues in the quantization of TAGs, we propose a framework called **Soft Tokenization for Text-attributed Graphs (STAG)**. During pre-training, a GNN first learns node representations that capture structural information. These structure-based node embeddings are then quantized into discrete tokens with self-supervised learning objectives: (1) a reconstruction loss for preserving semantic information, and (2) a contrastive loss for capturing neighborhood structural information. This design ensures effective preservation of both semantic and structural information without requiring labeled data. More specifically, we design a *soft assignment strategy* to map each node to a distribution of tokens, compensating for the lack of explicit tokenization structures in graphs. In contrast, computer vision employs a hard assignment, where each feature map or patch is naturally mapped to a single token. The soft assignment also prevents overfitting to specific tokens and thus improves transferability across domains. Furthermore, we incorporate a Kullback-Leibler (KL) divergence loss to guide the quantization process, enhancing the alignment between structural and semantic representations in the absence of labeled data.

During inference, STAG can flexibly work with LLMs by providing quantized tokens as prompts in zero- or few-shot settings, or function independently by using the learned embeddings through prompt tuning or linear probing in few-shot settings. It also supports both single- and cross-dataset learning, requiring no source labels in the latter, unlike previous approaches [35].

Contributions. Our key contributions advance the state of the art in GraphLLM in three significant ways. (1) We propose a novel quantization approach for TAGs that bridges the gap between continuous graph embeddings and discrete LLM token spaces. (2) We develop a unified framework that supports diverse learning paradigms with or without LLMs in both single- and cross-dataset scenarios. In particular, in cross-dataset learning, we achieve true zero-shot learning without requiring any labeled data for the source dataset. (3) We conduct extensive empirical validation, demonstrating the superior performance of STAG across multiple graph benchmarks.

2 Related Work

In this section, we briefly review related work on GraphLLM and vector quantization.

Graph Learning with LLMs. Recent advances in large language models (LLMs) have inspired several approaches to combine LLMs with graph learning. GraphGPT [50] converts graph structures into natural language descriptions for LLM processing, while GIMLET [69] proposes a unified graph-text language model for zero-shot learning, though both either lose structural information or require extensive architectural modifications. OFA [35] and Prodigy [26] construct prompts to leverage PLMs' semantic representations, but struggle to fully utilize LLMs' capabilities. While GraphText [70] attempts to bridge graph-LLM semantic spaces through learned projections, it requires computationally expensive alignment procedures. TAPE [18] takes a different approach by using LLM-generated explanations as node features for GNNs, achieving state-of-the-art performance efficiently. Despite these advances, fundamental challenges persist in preserving both structural and semantic information while maintaining cross-architecture compatibility. Our work addresses these limitations through a novel quantization-based approach that enables direct processing of graph data by frozen LLMs without compromising structural integrity.

Graph-to-LLM Representation. The challenge of representing graphs for LLM processing has led to diverse solutions. Traditional approaches rely on graph verbalization [66, 70], which faces scalability issues with large graphs. More structured representations have emerged through code-like formats and adjacency tables [12, 15, 58], though they struggle to balance structural completeness with LLM compatibility. Recent embedding-fusion methods [20, 52] better preserve structural information by integrating GNN embeddings with LLM representations, but require expensive alignment mechanisms. Position-aware approaches offer another perspective, with GIMLET [69] using generalized position embeddings and LINKGPT [21] employing pairwise encoding to capture structural relationships. However, these methods face challenges in cross-architecture consistency and computational efficiency at scale.

Vector Quantization. Vector quantization has evolved significantly since the introduction of VQ-VAE [55], which pioneered neural discrete representation learning. Subsequent works have enhanced this framework through hierarchical structures in VQ-VAE-2 [45], adversarial training in VQ-GAN [10], and residual quantization in RQ-VAE [32]. The technique has found success across various domains, from audio compression in SoundStream [67] to vision-language models like DALL-E [44] and Stable Diffusion [47]. Recent works such as V2L Tokenizer [71] and LLM-AR [43] have

demonstrated the potential of quantization for enabling LLMs to process visual and action signals. However, applying these techniques to graphs presents unique challenges due to their irregular structure, unlike images and actions which have natural tokenization patterns through feature maps or temporal sequences.

3 Proposed Approach: STAG

We present Soft Tokenization for TAGs (STAG), beginning with the overall framework and then detailing its key stages.

3.1 Overall Framework

A text-attributed graph or TAG is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{R})$, where \mathcal{V} is the set of nodes, $\mathcal{A} \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ represents the adjacency matrix, and \mathcal{R} denotes text attributes associated with the nodes. Operating on TAGs, the workflow of STAG consists of three main stages: (1) initial feature extraction and codebook construction, (2) self-supervised pre-training with semantic-structural integration, and (3) flexible inference with or without LLMs.

As shown in Figure 1(a), we first use a PLM, specifically a sentence transformer [46] to embed raw text attributes into initial node features and construct a codebook from LLM vocabulary. During pre-training in Figure 1(b), we process local subgraphs [23] through a GNN encoder, fuse them with semantic features, and apply soft quantization. The pre-training follows a dual-branch architecture with reconstruction and contrastive objectives. For inference in Figure 1(c), STAG supports both direct classification and LLM-based zero- or few-shot learning. Additionally, prompt tuning in Figure 1(d) further enhances few-shot learning capabilities through a lightweight adaptation mechanism.

3.2 Initial Feature and Codebook Construction

VQ-VAE [55] provides a foundation for encoding continuous data into discrete representations. Given an input \mathbf{x} , the encoder $\xi(\cdot)$ maps it to continuous latent vectors $\mathbf{z}_e = \xi(\mathbf{x})$, which are then quantized by finding their nearest neighbors in a codebook $\mathcal{B} = \{\mathbf{e}_k\}_{k=1}^K$ of K embedding vectors: $\mathbf{z}_q = \text{Quantize}(\mathbf{z}_e) = \mathbf{e}_k$, where $k = \arg \min_j \|\mathbf{z}_e - \mathbf{e}_j\|_2$. The quantized representations \mathbf{z}_q are then passed to a decoder $\omega(\cdot)$ to reconstruct the input.

Building upon this framework for TAGs, we first encode raw text attributes \mathcal{R} into initial node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d_x}$ using a frozen sentence transformer. For our codebook, we filter LLaMA-2's 32,000 subword vocabulary [53] by removing non-English and non-alphabetical entries, and eliminating whitespace-based duplicates. The refined token set $\mathcal{T} = \{t_k\}_{k=1}^K$, which has a size of 15,062, is then embedded using the same frozen transformer to create our codebook $\mathcal{E} = \{\mathbf{e}_k\}_{k=1}^K$ where $\mathbf{e}_k \in \mathbb{R}^{d_x}$, and let $\mathbf{E} = [\mathbf{e}_k^\top]_{k=1}^K \in \mathbb{R}^{K \times d_x}$ be the matrix formed by stacking these embeddings.

However, our approach differs from VQ-VAE in several key aspects. First, unlike VQ-VAE's learnable codebook, our codebook \mathcal{E} remains frozen throughout pre-training to ensure semantic consistency. Second, instead of hard nearest-neighbor quantization, we introduce soft assignment to better handle the lack of natural tokenization structures in graphs. Third, we incorporate semantic-structural fusion and distribution alignment mechanisms to effectively preserve both types of information. These innovations,

detailed in the following sections, enable effective graph representation learning while maintaining LLM compatibility.

3.3 Pre-training with Soft Assignment

To address the challenges of integrating both semantic and structural information into LLM-compatible formats, we develop a self-supervised pre-training approach with feature fusion and soft assignment strategy, ensuring consistency across different LLMs.

3.3.1 Structural-Semantic Feature Fusion. Following GraphMAE's masking strategy [24], we randomly mask nodes $\mathcal{V}' \subset \mathcal{V}$ with a learnable [M] token while keeping the graph structure intact. Our framework processes both original and masked graphs through parallel branches sharing the same feature learning and quantization mechanisms, employing a GNN encoder $e(\cdot)$ to learn structural representations $\mathbf{z}_e = e(\mathbf{x}) \in \mathbb{R}^{d_h}$, where \mathbf{x} is the node feature vector from \mathbf{X} . However, directly quantizing these structural embeddings risks losing the rich semantic information present in the original text attributes.

Therefore, to preserve both structural and semantic information while maintaining computational efficiency, we introduce a parameter-efficient feature fusion module. Given the GNN output \mathbf{z}_e , we first project it to dimension d_x (matching the initial feature dimension) using a linear projection layer $\mathbf{W}_f \in \mathbb{R}^{d_x \times d_h}$. Then, combined with initial node features \mathbf{x} , we compute:

$$\mathbf{z}_f = \phi \cdot \frac{\mathbf{W}_f \mathbf{z}_e}{\|\mathbf{W}_f \mathbf{z}_e\|_2} + \psi \cdot \frac{\mathbf{x}}{\|\mathbf{x}\|_2}, \quad (1)$$

where ϕ and $\psi \in (0, 1]$ are learnable parameters. This design is motivated by two key insights: (1) both features originate from the same text input, allowing for efficient integration without complex transformations, and (2) L2 normalization preserves the directional information of both structural and semantic features.

3.3.2 Soft Assignment with Distribution Alignment. The fused features $\mathbf{z}_f \in \mathbb{R}^{d_x}$ are then quantized into the discrete token space through our semantic-preserving quantization process.

Soft Assignment Strategy. We develop a soft assignment strategy that maps each node to a distribution of tokens, addressing two key challenges: (1) the absence of explicit tokenization structures in graphs, and (2) preventing overfitting to specific tokens to improve transferability across domains. Unlike VQ-VAE, which relies on L2 distance and hard assignment for codebook lookup, we leverage cosine similarity to compute attention weights over the entire codebook, enabling soft token assignments for each fused node representation \mathbf{z}_f :

$$\text{attn}(\mathbf{z}_f) = \text{softmax} \left([\theta(\mathbf{z}_f, \mathbf{e}_k)]_{k=1}^K / \tau_{sa} \right), \quad (2)$$

where $\theta(\mathbf{z}_f, \mathbf{e}_k)$ computes the cosine similarity between \mathbf{z}_f and the k -th codebook embedding, and τ_{sa} is a temperature parameter that controls the softness of the assignment. This cosine similarity-based attention is particularly suitable for our semantic codebook space, as it focuses on directional similarity rather than absolute distances. The quantized representation is then computed as the weighted sum of the codebook embeddings:

$$\mathbf{z}_q = \mathbf{E}^\top \text{attn}(\mathbf{z}_f). \quad (3)$$

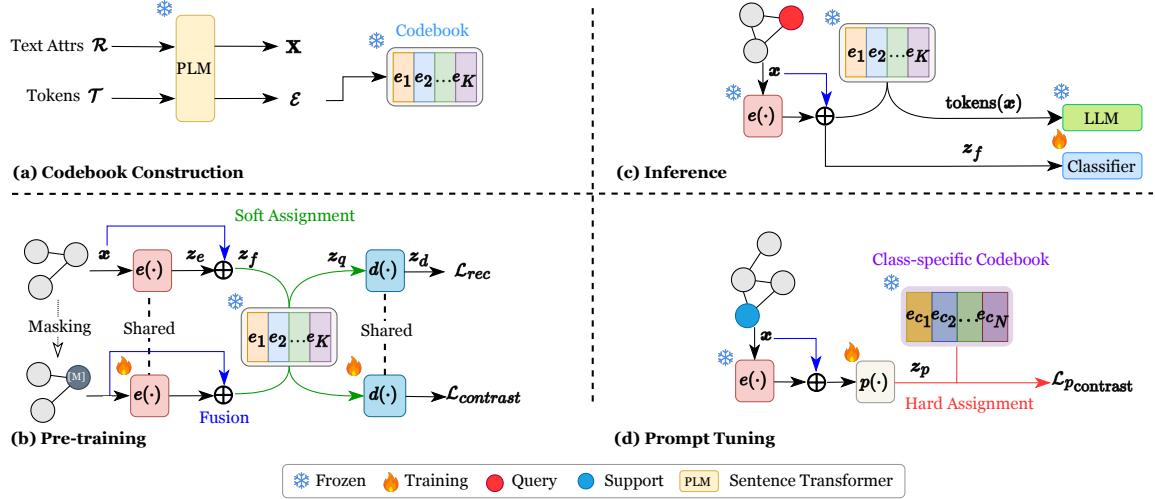


Figure 1: Overview of STAG framework: (a) Feature extraction and codebook construction, (b) Self-supervised pre-training with dual-branch architecture, (c) Inference with or without LLM, and (d) Prompt-tuned inference for few-shot learning.

To prevent encoder outputs from diverging too far from the codebook space during training, we employ a commitment loss:

$$\mathcal{L}_{\text{commit}} = \beta \left(1 - \theta(z_f, \text{sg}[z_q]) \right), \quad (4)$$

where $\text{sg}[\cdot]$ denotes the stop-gradient operator and $\beta \in (0, 2]$ controls the strength of the commitment. While VQ-VAE uses L2-based hard assignment for both lookup and commitment loss, we adopt cosine similarity for semantic space alignment and soft assignments through weighted codebook combinations. Although the frozen codebook ensures consistent semantics across LLMs, the soft assignment alone cannot guarantee that the quantized representations preserve the semantic meaning of the original node text.

Self-supervised Semantic Alignment. To address this challenge, we incorporate a KL divergence loss that aligns the attention distribution of fused features with that of the original semantic features to enhance the alignment between structural and semantic representations in the absence of labeled data:

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(\text{attn}(x) \| \text{attn}(z_f)), \quad (5)$$

where $\text{attn}(x)$ represents the attention distribution computed from the original node features x over the codebook following Eq. (2).

The KL divergence aligns token assignment distributions between fused and original features, enabling effective integration of structural and semantic information by guiding the structure-semantic fused representations to be quantized to semantically meaningful tokens. Unlike Gumbel-Softmax [28], which uses KL divergence to train discrete categorical distributions, our KL loss guides the soft assignment to maintain semantic consistency with respect to the frozen codebook without requiring labels.

3.3.3 Dual-Branch Training Objectives. Our pre-training maintains parallel branches sharing the same quantization mechanism while serving different learning objectives: reconstruction for node-level semantics and contrastive learning for neighborhood structures.

Reconstruction Branch. This branch processes the original graph to preserve node-level semantic information through reconstruction. Given a shared GNN decoder $d(\cdot)$ that maps quantized representations back to the initial feature dimension d_x , we extend the traditional VQ-VAE reconstruction to graphs using GraphMAE’s scaled cosine error (SCE) loss:

$$\mathcal{L}_{\text{rec}} = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \left(1 - \frac{\mathbf{x}_i^\top \mathbf{z}_{d_i}}{\|\mathbf{x}_i\| \cdot \|\mathbf{z}_{d_i}\|} \right)^\gamma, \quad (6)$$

where $\mathbf{z}_d = d(\mathbf{z}_q) \in \mathbb{R}^{d_x}$ is the decoded feature of the quantized representation, and $\gamma \geq 1$ is a scaling factor that controls the penalty for reconstruction errors. Unlike traditional mean squared error used in VQ-VAE, SCE loss avoids sensitivity and low selectivity issues in graph representation learning [23], making it particularly suitable for preserving semantic information during reconstruction.

Contrastive Branch. This branch processes the masked graph to capture neighborhood structural patterns through a contrastive learning objective. For each masked node $v_i \in \widetilde{\mathcal{V}}$, we form positive pairs between its decoded features \mathbf{z}_{d_i} and original features \mathbf{x}_i , with other masked nodes’ decoded features serving as negatives. Following [41], the contrastive loss is defined as:

$$\ell(\mathbf{z}_{d_i}, \mathbf{x}_i) = \log \left(e^{\theta(\mathbf{z}_{d_i}, \mathbf{x}_i)/\tau_c} / \underbrace{\left(e^{\theta(\mathbf{z}_{d_i}, \mathbf{x}_i)/\tau_c} + \sum_{j=1, j \neq i}^{|\mathcal{N}_i|} e^{\theta(\mathbf{z}_{d_i}, \mathbf{z}_{d_j})/\tau_c} \right)}_{\text{positive pair}} \right), \forall v_i \in \widetilde{\mathcal{V}} \text{ and } \mathcal{N}_i \subseteq \widetilde{\mathcal{V}}, \quad (7)$$

where \mathcal{N}_i represents a subset of masked nodes selected as negative samples for v_i , τ_c is the temperature parameter. The overall contrastive loss is $\mathcal{L}_{\text{contrast}} = -\frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v_i \in \widetilde{\mathcal{V}}} \ell(\mathbf{z}_{d_i}, \mathbf{x}_i)$. This contrastive objective compels the model to utilize neighborhood structural information for feature prediction [62], capturing essential graph

topology in quantized representations. The final training objective combines losses from both branches:

$$\mathcal{L} = \underbrace{\frac{1}{2}(\mathcal{L}_{\text{commit}}^{\text{Rec}} + \mathcal{L}_{\text{commit}}^{\text{Contrast}})}_{\text{commitment}} + \underbrace{\mathcal{L}_{\text{rec}}}_{\text{semantic}} + \underbrace{\mathcal{L}_{\text{contrast}}}_{\text{structural}} + \underbrace{\lambda \mathcal{L}_{\text{KL}}}_{\text{alignment}}, \quad (8)$$

where λ is a balancing parameter, and $\mathcal{L}_{\text{commit}}^{\text{Rec}}$ and $\mathcal{L}_{\text{commit}}^{\text{Contrast}}$ are commitment losses applied to reconstruction and contrastive branches, respectively. This multi-objective optimization ensures effective integration of semantic and structural information while ensuring compatibility with discrete token spaces.

3.4 Flexible Inference with or without LLMs

Through effective semantic-structural quantization, our framework bridges continuous graph embeddings and discrete token spaces, enabling both LLM-based and traditional inference strategies.

Inference with LLMs. To convert continuous node embeddings into discrete tokens for LLM processing, we first process each node with its local subgraph through our pre-trained GNN encoder and fusion module to obtain its learned embedding. We then compute its attention distribution over the codebook \mathcal{E} and select the $\text{top-}k \in \mathbb{N}^+$ corresponding tokens from \mathcal{T} with the highest attention weights:

$$\begin{aligned} \text{tokens}(\mathbf{x}) &= [t_{m_1}, t_{m_2}, \dots, t_{m_{\text{top-}k}}], \text{ where} \\ m_i &= \underset{j \in \{1, \dots, K\} \setminus \{m_1, \dots, m_{i-1}\}}{\operatorname{argmax}} \text{attn}(z_f), \text{ for } i \in [1, \text{top-}k]. \end{aligned} \quad (9)$$

For N -way k -shot few-shot learning where N represents the number of classes and k is the number of examples per class, we construct a system prompt that includes $N \times k$ support examples, where each example consists of a node's tokens and its corresponding class label. The LLM then predicts the class of a new test node based on its tokens and these support examples. Below is a simplified example using a 3-way 1-shot setting:

System Prompt: You are a node classifier. Given a list of tokens representing a node's features, predict its class from the following options: [Research Paper, Dataset, Software].
Few-shot Examples: Node tokens: [research, methodology, experiment] Class: Research Paper
Node tokens: [benchmark, statistics, collection] Class: Dataset
Node tokens: [implementation, code, library] Class: Software
Test Node: Node tokens: [algorithm, computation, optimization] Predict the class:

Similarly, for zero-shot learning, we directly query the LLM with only the system prompt and test node tokens. Below is a simplified 4-way zero-shot example (see Appendix C for the complete prompt templates for both few-shot and zero-shot):

System Prompt: You are a node classifier. Given a list of tokens representing a node's features, predict its class from the following options: [Research Paper, Dataset, Software, Survey Paper].
Test Node: Node tokens: [algorithm, computation, optimization] Predict the class:

Our quantized tokens integrate both semantic and structural information from input nodes, enabling us to leverage different LLM capabilities effectively. In few-shot learning, we combine the LLM's in-context learning ability (through support examples) with

its semantic understanding to make predictions. For zero-shot learning, we rely purely on the LLM's semantic knowledge, allowing classification without any labeled examples.

Inference without LLMs. For traditional inference, following standard practice in graph self-supervised learning [16, 51, 57, 68], we employ linear probing to evaluate the quality of learned representations. Specifically, we freeze the pre-trained GNN encoder and fusion module, and train only a linear classifier on the fused features z_f . Given a linear classifier $\mathbf{W}_c \in \mathbb{R}^{N \times d_x}$ where N is the number of candidate classes, the prediction is computed as:

$$\hat{y} = \operatorname{argmax}(\operatorname{softmax}(\mathbf{W}_c z_f)). \quad (10)$$

This approach allows direct comparison with other graph SSL methods while maintaining the efficiency of our framework. Beyond these basic inference strategies, our framework enables enhanced few-shot domain transfer through a prompt tuning mechanism that can operate both with and without LLMs.

3.5 Prompt-tuning for Domain Transfer

To enhance domain transfer capabilities in few-shot settings, we develop a prompt tuning mechanism [61] that builds on our quantization framework while switching to hard token assignment. During prompt tuning, we keep our pre-trained GNN encoder and fusion module frozen, and introduce a lightweight prompt network $p(\cdot)$ (two-layer bottleneck neural network) that processes the fused features. Specifically, the prompted features $z_p \in \mathbb{R}^{d_x}$ are computed as: $z_p = p(z_f) \odot z_f$, where \odot denotes element-wise multiplication. For N -way k -shot setting, we obtain class label explanations via LLMs following [18, 35] and embed them using the same frozen sentence transformer to create a frozen class-specific codebook $C = \{e_{c_n}\}_{n=1}^N \in \mathbb{R}^{N \times d_x}$, where e_{c_n} represents the embedded explanation of class c_n .

The prompt tuning is guided by two objectives. First, following our quantization design, a commitment loss ensures the prompted embeddings remain close to C :

$$\mathcal{L}_{p_{\text{commit}}} = \beta_p (1 - \theta(z_p, \operatorname{sg}[z_q])), \quad (11)$$

where β_p controls the strength of the commitment.

Second, with $N \times k$ labeled support examples, we employ a weighted contrastive loss that leverages class labels to guide the tuning:

$$\mathcal{L}_{p_{\text{contrast}}} = -\frac{1}{Nk} \sum_{i=1}^{Nk} \sum_{j=1}^N \theta(c(x_i), c_j) \log \frac{e^{\theta(z_{p_i}, e_{c_j})/\tau_p}}{\sum_{n=1}^N e^{\theta(z_{p_i}, e_{c_n})/\tau_p}}, \quad (12)$$

where $\theta(c(x_i), c_j)$ represents the semantic similarity between the ground truth class $c(x_i)$ of example x_i and class c_j . This weighted objective encourages the prompted embeddings to be similar to their ground truth class embeddings while considering semantic relationships between classes. This design helps capture the natural hierarchy and relationships between classes, leading to more robust few-shot learning.

For inference with LLMs, we quantize the prompted embeddings using our original codebook \mathcal{E} to obtain tokens (we use the same system prompt as in Section 3.4). For inference without LLMs, we directly compute similarity with the class embeddings, where the

most similar class embedding indicates the predicted class:

$$\hat{y} = \operatorname{argmax}_{n \in \{1, \dots, N\}} \theta(z_p, e_{c_n}). \quad (13)$$

This unified prompt tuning approach enables effective domain transfer by leveraging class semantics while maintaining compatibility with both LLM-based and traditional classification paths.

4 Experiments

We conduct comprehensive experiments to evaluate our framework's ability to bridge graph representation learning with LLMs. Specifically, we investigate three key research questions:

- **RQ1:** How effectively does STAG integrate structural and semantic information in few-shot learning?
- **RQ2:** Can our framework enable effective zero-shot generalization across different domains?
- **RQ3:** How flexible is our framework in supporting different LLM architectures?

4.1 Experimental Setup

We evaluate our framework on seven text-attributed graph datasets: five citation networks (Cora Full [39], its pruned subset standard Cora, CiteSeer, PubMed [65], ogbn-arxiv [25]), one web graph (WikiCS [40]), and one co-purchase network (a subset of ogbn-products [25] following [18]). We use sentence transformers [46] to obtain 768-dimensional embeddings as initial node features. For our method, this same model is also used for codebook construction. And we use GAT as both encoder and decoder in our framework.

We compare with three categories of baselines: (1) Traditional graph learning methods including GCN [31] and GAT [56], and self-supervised approaches like DGI [57] and GraphMAE2 [23], which serve as strong baselines for representation learning on graphs; (2) Few-shot graph learning methods including GPPT [48] and G2P2 [61], which are specifically designed for few-shot learning on graphs through prompt tuning; (3) GraphLLM methods like Prodigy [26] and OFA [35], which both require labeled data from source datasets during training for model adaptation.

During pre-training, OFA and Prodigy require labeled data with specific class-splitting strategies, while all other methods use unlabeled data. For ogbn-arxiv, we follow OFA's splitting strategy; for Cora Full and ogbn-products, we create splits with non-overlapping classes. For evaluation, we conduct 5-way 5-shot experiments across 20 random tasks, with a total of 2000 balanced queries distributed across all tasks. For OFA and Prodigy, tasks are created from their test splits (classes unseen during training) to ensure fair evaluation of their transfer learning capabilities. For all other methods that don't require labeled pre-training data, tasks are created from the full dataset. For LLM inference, we use $top-k = 13$ tokens based on empirical performance. All experiments use LLaMA3-8B [9] by default and report classification accuracy. Detailed hyperparameter configurations and hardware specifications are provided in Appendix A.

4.2 RQ1: Cross-dataset Few-shot Learning

Table 1 presents the cross-dataset few-shot learning results across different source-target dataset combinations, where all methods are

pre-trained on either Cora Full or ogbn-products except GCN and GAT. Our framework demonstrates three key advantages. (1) *Effective structural-semantic integration*: Our framework achieves competitive results compared to traditional GNNs and self-supervised methods like GraphMAE2, validating our learned representations. Unlike G2P2 which uses dataset-specific text-graph alignment with a trainable text encoder, STAG prioritizes generalization through a frozen sentence transformer with parameter-efficient alignment modules, showing robust performance while outperforming OFA despite its need for source dataset labels. (2) *Framework versatility*: Prompt tuning enhances performance in both inference paths - with LLM ('STAG+Prompt Tuning') achieving competitive results, and without LLM ('STAG+Prompt Tuning*') achieving state-of-the-art performance on several datasets. (3) *Robust transfer ability*: When pre-trained on ogbn-products, Prodigy and OFA suffer significant performance drops when tested on different target datasets. In contrast, our framework maintains strong performance on citation networks like ogbn-arxiv, validating its ability to capture generalizable graph representations despite domain shifts. When compared to raw text and feature variants, our method outperforms both 'Raw Feat + Quantization' and 'Raw Text' with LLM by effectively incorporating structural information into learned tokens for both inference paths, particularly in Cora Full and ogbn-arxiv datasets despite raw texts' potential information leakage. Results for models pre-trained on ogbn-arxiv are in Appendix B.

4.3 RQ2: Zero-shot Generalization

Table 2 presents the 5-way zero-shot classification results across different datasets. Unlike existing methods that require labeled source data during pre-training, our framework achieves true zero-shot learning without using labels from either source or target datasets. All models are pre-trained on Cora Full, with OFA requiring labeled data during pre-training. STAG demonstrates strong zero-shot generalization through two inference paths: (1) *Direct LLM inference*: By effectively embedding both semantic and structural information into LLM-compatible formats, STAG achieves significant improvements over baselines, outperforming OFA by large margins on most datasets. This validates the effectiveness of our quantization strategy in bridging continuous graph embeddings and discrete token spaces, enabling STAG to fully leverage the semantic capabilities of LLMs. (2) *Class-specific codebook inference*: As in Eq. (13), we predict classes by comparing z_f with C . 'STAG + C' achieves even stronger performance, demonstrating the quality of our learned representations. The strong performance across both paths, particularly in challenging domain transfers like WikiCS, demonstrates how our framework effectively captures generalizable graph representations. When compared to raw variants, STAG consistently outperforms both 'Raw Feat' with LLM and 'Raw Feat + C' without LLM by effectively incorporating structural information into learned tokens for both inference paths. Complete results on remaining datasets are provided in the Appendix B.

4.4 RQ3: Flexibility with Different LLMs

Our quantization strategy enables a unique advantage of STAG: the ability to flexibly pair a single pre-trained model with different LLMs during inference. Unlike existing GraphLLM approaches

Table 1: 5-way 5-shot node classification across different datasets (except PubMed: 3-way). Models are pre-trained on Cora Full or ogbn-products with $top-k = 13$ for LLM inference. Gray-shaded rows: supervised baselines trained directly on target datasets; Colored cells: pre-train dataset matches target (blue: Cora Full, green: ogbn-products). Results show accuracy (%) averaged over 20 random tasks, with best results among our variants in bold.

Pre-train data	Method	LLM	Target data						
			Cora	Cora Full	CiteSeer	PubMed	WikiCS	ogbn-arxiv	ogbn-products
Same as target	GCN	✗	76.10 \pm 4.26	82.81 \pm 7.40	59.95 \pm 6.92	66.35 \pm 6.71	70.55 \pm 8.26	76.61 \pm 7.72	80.08 \pm 7.41
	GAT	✗	79.60 \pm 5.00	84.72 \pm 7.83	60.85 \pm 6.78	67.40 \pm 7.18	77.95 \pm 7.81	80.80 \pm 7.99	81.94 \pm 7.07
No pre-train	Raw Text	✓	63.40 \pm 9.07	71.66 \pm 7.84	62.10 \pm 5.35	85.00 \pm 5.48	77.15 \pm 6.92	54.74 \pm 9.21	87.58 \pm 5.48
	Raw Feat + Quantization	✓	54.85 \pm 6.28	73.74 \pm 8.10	56.40 \pm 5.48	48.30 \pm 7.78	73.40 \pm 8.19	63.75 \pm 9.98	65.84 \pm 9.96
	Raw Feat + Linear Probing	✗	70.25 \pm 7.22	81.29 \pm 7.47	63.00 \pm 6.72	68.30 \pm 6.28	78.05 \pm 7.47	83.05 \pm 7.40	77.53 \pm 7.16
Cora Full	DGI	✗	77.05 \pm 5.12	83.32 \pm 8.12	63.85 \pm 5.39	68.20 \pm 7.57	78.65 \pm 6.90	81.30 \pm 8.51	79.90 \pm 7.20
	GraphMAE2	✗	77.70 \pm 6.92	84.74 \pm 7.42	65.25 \pm 5.84	66.35 \pm 6.09	80.95 \pm 4.96	80.04 \pm 8.15	73.93 \pm 7.57
	GPPT	✗	27.16 \pm 7.61	67.90 \pm 12.27	28.66 \pm 7.60	21.53 \pm 10.91	29.00 \pm 8.08	36.92 \pm 10.32	24.32 \pm 5.13
	G2P2	✗	74.90 \pm 7.47	81.10 \pm 7.44	59.65 \pm 9.68	67.85 \pm 8.02	69.90 \pm 10.52	68.75 \pm 10.14	70.97 \pm 10.03
	Prodigy	✗	39.50 \pm 6.75	60.80 \pm 6.38	42.90 \pm 5.02	43.68 \pm 6.91	43.25 \pm 6.91	47.85 \pm 6.89	30.70 \pm 5.94
	OFA	✗	45.95 \pm 4.52	56.95 \pm 5.31	36.80 \pm 5.50	49.40 \pm 4.75	46.45 \pm 4.67	50.80 \pm 4.73	33.60 \pm 4.26
	STAG	✓	67.60 \pm 6.72	80.95 \pm 8.02	62.45 \pm 7.02	54.50 \pm 7.83	79.20 \pm 8.41	71.56 \pm 10.32	69.34 \pm 9.93
	+ Linear Probing	✗	78.50 \pm 5.62	86.04 \pm 6.70	66.70\pm5.36	69.00\pm6.31	84.05\pm5.78	82.99 \pm 8.10	79.62 \pm 7.12
	+ Prompt Tuning	✓	73.30 \pm 4.77	85.20 \pm 7.59	65.40 \pm 5.98	66.20 \pm 5.70	79.45 \pm 7.53	79.18 \pm 8.28	73.94 \pm 9.67
	+ Prompt Tuning*	✗	78.65\pm5.93	86.66\pm7.67	65.80 \pm 7.03	68.25 \pm 6.80	83.55 \pm 5.94	83.57\pm8.30	80.48\pm6.86
ogbn-products	DGI	✗	OOM						
	GraphMAE2	✗	69.30 \pm 5.51	77.74 \pm 7.53	55.90 \pm 7.73	60.05 \pm 6.05	71.15 \pm 7.68	67.70 \pm 9.51	81.08 \pm 8.00
	GPPT	✗	24.52 \pm 4.77	25.30 \pm 6.61	23.76 \pm 4.70	24.70 \pm 10.84	23.32 \pm 5.42	26.52 \pm 6.49	55.18 \pm 11.89
	G2P2	✗	70.70 \pm 9.00	76.35 \pm 8.62	53.10 \pm 9.97	64.95 \pm 9.40	69.20 \pm 9.51	65.55 \pm 9.07	75.52 \pm 10.10
	Prodigy	✗	28.90 \pm 5.38	36.54 \pm 6.04	28.00 \pm 4.56	47.11 \pm 7.68	33.85 \pm 5.82	36.65 \pm 5.17	62.20 \pm 7.29
	OFA	✗	25.85 \pm 3.12	29.70 \pm 5.09	23.05 \pm 3.81	51.30 \pm 5.53	33.80 \pm 4.76	28.95 \pm 4.07	59.65 \pm 5.70
	STAG	✓	59.20 \pm 7.07	73.60 \pm 8.00	54.90 \pm 6.11	48.45 \pm 7.61	76.15 \pm 8.09	68.56 \pm 10.30	79.75 \pm 9.33
	+ Linear Probing	✗	73.65 \pm 7.49	83.14 \pm 6.26	62.30 \pm 6.08	69.85\pm5.89	80.35 \pm 7.17	81.62\pm7.26	83.94 \pm 6.78
	+ Prompt Tuning	✓	68.95 \pm 5.93	80.07 \pm 7.82	60.40 \pm 6.63	63.70 \pm 5.72	78.55 \pm 8.24	74.85 \pm 9.89	82.35 \pm 6.24
	+ Prompt Tuning*	✗	74.90\pm6.54	83.52\pm7.60	63.70\pm6.00	68.45 \pm 6.20	81.40\pm7.14	81.19 \pm 7.68	84.03\pm5.54

✓/✗: LLM usage during inference; **Prompt Tuning***: inference without LLM; **Raw Text**: Use raw text for LLM inference (To fit in the context window of LLM, raw text is truncated); **Raw Feat + Quantization**: Directly quantize raw node features into tokens for LLM inference; **Raw Feat + Linear Probing**: Train a linear classifier on raw node features without any pre-training; OOM: Out-of-memory error during training.

Table 2: 5-way zero-shot classification results ($top-k = 13$).

Pre-train data	Method	LLM	Target data			
			Cora	Cora Full	WikiCS	ogbn-arxiv
No pre-train	Raw Feat + Q	✓	47.10 \pm 5.98	60.33 \pm 10.88	70.40 \pm 8.88	25.48 \pm 5.54
	Raw Feat + C	✗	62.20 \pm 8.45	77.23 \pm 8.96	73.85 \pm 8.02	72.85 \pm 10.43
Cora Full	G2P2	✗	60.45 \pm 7.58	64.29\pm11.56	50.25 \pm 8.43	19.66 \pm 6.38
	OFA	✗	20.30 \pm 2.93	23.85 \pm 3.58	21.45 \pm 3.99	17.60 \pm 3.74
	STAG	✓	48.05 \pm 6.15	62.63 \pm 11.70	76.25\pm8.48	26.01 \pm 7.52
	STAG + C	✗	66.55\pm7.48	82.90\pm9.52	75.15 \pm 7.81	74.23\pm9.35

Raw Feat + Q: Quantize raw node features into tokens for LLM inference.
+ C: Classification using class-specific codebook.

that require architecture-specific modifications, STAG can seamlessly work with various LLMs, from open-source models like LLaMA2 [53], LLaMA3 [9], and Vicuna [7] to closed-source ones like GPT-4o [27] by converting graph representations into discrete tokens that are universally interpretable across different LLM architectures. Results in Table 3 demonstrate three key patterns (using $top-k = 5$ for cost-efficient inference with closed-source LLMs): (1) **Model size matters**: Larger models consistently achieve better performance, suggesting enhanced semantic understanding capabilities.

Table 3: 5-way 5-shot classification results with different LLMs (pre-trained on Cora Full, $top-k = 5$ for inference).

LLM	Cora Full	WikiCS	ogbn-arxiv	CiteSeer
LLaMA2-7B + PT	76.66 \pm 7.79	79.00 \pm 7.96	65.33 \pm 10.46	54.35 \pm 9.54
	81.05 \pm 7.77	79.90 \pm 7.69	77.42 \pm 10.48	58.45 \pm 8.61
LLaMA2-13B + PT	77.62 \pm 8.67	79.80 \pm 7.30	69.38 \pm 8.83	54.60 \pm 8.79
	81.95 \pm 7.06	80.45 \pm 7.66	77.75 \pm 9.01	57.30 \pm 9.20
Vicuna-7B + PT	74.12 \pm 6.47	80.30 \pm 7.02	64.84 \pm 9.38	49.25 \pm 6.72
	80.77 \pm 6.75	80.10 \pm 7.39	76.95 \pm 9.43	52.25 \pm 8.23
Vicuna-13B + PT	77.76 \pm 8.58	79.35 \pm 7.98	66.03 \pm 9.34	52.25 \pm 6.39
	81.38 \pm 7.65	79.25 \pm 7.50	75.65 \pm 9.59	53.00 \pm 8.16
LLaMA3-8B + PT	79.22 \pm 8.45	78.40 \pm 8.05	70.37 \pm 8.95	61.25 \pm 7.14
	82.88 \pm 8.09	78.35 \pm 7.61	76.71 \pm 10.20	64.20 \pm 7.39
GPT-4o-mini + PT	79.25 \pm 8.42	81.05 \pm 6.80	71.32 \pm 9.13	61.90 \pm 7.22
	83.04 \pm 7.84	81.90\pm6.16	77.51 \pm 9.58	65.90 \pm 7.04
GPT-4o + PT	81.40\pm7.41	81.45\pm7.10	72.75 \pm 8.83	62.95\pm6.61
	83.28 \pm 7.06	81.60 \pm 7.19	78.85\pm9.74	65.90\pm7.03

+ PT: with prompt tuning. 7B, 8B, and 13B indicate the number of parameters in billions.

(2) *Advanced architectures show advantages*: Newer models like GPT-4o and LLaMA3-8B outperform their predecessors, with STAG's LLM inference becoming competitive even with its traditional variants. (3) *Prompt tuning provides consistent gains*: Performance improves across all LLMs, with particularly notable gains in smaller and older models like LLaMA2-7B and Vicuna-7B, demonstrating the effectiveness and robustness of our adaptation strategy. These results validate our framework's flexibility, offering future-proof advantages as more powerful LLMs become available.

4.5 Qualitative Analysis

To provide insights into how STAG processes and represents graph information, we conduct two case studies examining the quantization outputs (using $top-k = 5$ tokens).

Structure-Aware Tokenization Analysis. We analyze a sample node from the ‘Computational Complexity’ class in Cora Full dataset, where our model is pre-trained. This paper, which primarily discusses oracle constructions and isomorphism conjectures, serves as an illustrative example of how STAG effectively integrates structural information into token representations:

Raw Text: “In this paper we demonstrate an oracle relative to which there are one-way functions but every paddable 1-li-degree collapses to an isomorphism type, thus yielding a relativized failure of the Joseph Young Conjecture [JY85]. We then use this result to construct an oracle relative to which the Isomorphism Conjecture is true but oneway functions exist, which answers an open question of Fenner, Fortnow, and Kurtz [FFK96]. Thus, there are now relativizations realizing every one of the four possible states of affairs between the Isomorphism Conjecture and the existence of one-way functions.”

Category: Computational Complexity

Raw Feat + Quantization: isomorphism, oracle, numerable, mutable, schemes
STAG: complexity, algebraic, computation, polynomials, compute
STAG + PT: complexity, computational, computation, algorithms, compute

This example illustrates the effectiveness of our structure-aware tokenization: While the raw text primarily discusses oracle constructions and isomorphism conjectures, making raw quantization focus on surface-level terms (‘isomorphism’, ‘oracle’), STAG successfully captures the paper’s theoretical computer science nature by integrating neighborhood information. The learned tokenization emphasizes core complexity theory concepts (‘complexity’, ‘computation’) and mathematical foundations (‘algebraic’), and the prompted version further refines these tokens toward computational complexity aspects. This demonstrates how our method can effectively identify a node’s domain even when its raw content is not directly indicative of its category.

Intra-class Consistency. To examine how STAG captures class-specific patterns, we analyze the quantization results for multiple nodes from the same class. The following example presents tokenization outputs for five different nodes from the ‘Operating Systems’ category in WikiCS, showing how our method consistently captures class-relevant features while preserving individual paper characteristics:

Node 1: unix, linux, os, kernel, system
Node 2: unix, system, network, interface, protocol
Node 3: unix, system, compiler, terminal, execution
Node 4: unix, linux, kernel, system, filesystem
Node 5: linux, unix, kernel, system, filesystem

Table 4: Ablation study on representative datasets (pre-trained on Cora Full, 5-way 5-shot, $top-k = 13$).

Method	LLM Inference			Linear Probing		
	Cora Full	WikiCS	ogbn-arxiv	Cora Full	WikiCS	ogbn-arxiv
Full Model	80.95±8.02	79.20±8.41	71.56±10.32	86.04±6.70	84.05±5.78	82.99±8.10
¬ Fusion	37.49±6.66	29.10±9.65	29.49±8.01	46.73±6.66	35.05±8.61	34.12±6.79
¬ \mathcal{L}_{KL}	69.74±11.04	57.95±11.48	59.79±8.32	81.83±8.33	75.05±7.21	76.18±9.21
¬ Soft	37.07±8.01	31.55±8.98	28.21±7.63	67.77±9.89	61.35±8.81	50.90±9.53

While all nodes maintain core tokens related to operating systems (‘unix’, ‘system’), each node’s unique focus is preserved through specific technical tokens (‘interface’, ‘terminal’). This demonstrates STAG’s ability to balance class-level consistency with instance-level specificity. The consistent appearance of system-related tokens (‘kernel’, ‘filesystem’) also suggests that our quantization process effectively captures the semantic structure of technical documents beyond simple keyword matching.

4.6 Ablation Studies

To validate our design choices in addressing the technical challenges, we conduct ablation studies by systematically removing key components from our framework. All variants are pre-trained on Cora Full. Table 4 presents results on representative datasets, demonstrating each component’s contribution: (1) *Semantic and Structural Fusion*: Removing the fusion module (‘¬Fusion’) to directly quantize node embeddings z_e leads to substantial performance degradation, even when evaluated on the pre-training dataset Cora Full. This underscores the necessity of integrating semantic and structural information to ensure semantic preservation. (2) *KL Regularization*: Eliminating the \mathcal{L}_{KL} (‘¬ \mathcal{L}_{KL} ’) results in considerable performance deterioration across both inference paths, confirming its importance in preserving semantic similarity during quantization by encouraging nodes to be mapped to semantically related tokens. (3) *Soft Token Assignment*: Replacing soft assignment with hard assignment (‘¬Soft’), which is to quantize into single token during pre-training and inference, severely hampers performance, especially in LLM inference. This demonstrates that soft assignment effectively mitigates overfitting in the quantization process by maintaining distributional information rather than forcing hard decisions, enabling more robust transfer across datasets. These results collectively validate that each component directly addresses a key technical challenge in bridging the gap between graph structures and LLM-compatible representations.

4.7 Computational Efficiency Analysis

To demonstrate the practical viability of our quantization approach, we analyze the computational overhead of STAG’s key components. Our quantization method has time complexity $O(B \times K \times d)$, where B is the total number of nodes in a batch (across all subgraphs), K is the codebook size (15,062 tokens), and d is the embedding dimension (768).

Table 5 presents a detailed breakdown of computational costs for a typical batch containing 64 subgraphs with approximately

Table 5: Computational efficiency breakdown of STAG components (pre-trained on Cora Full).

Component	Time (ms)	Percentage
(a) GNN Encoding	18.4 ± 41.9	57%
(b) Cosine Similarity (Eq. (2))	8.2 ± 3.1	25%
(c) Weighted Combination (Eq. (3))	5.7 ± 3.3	18%
Total Quantization (b+c)	13.9 ± 6.4	43%
Total Processing (a+b+c)	32.3 ± 48.3	100%

3,944 total nodes. The analysis shows that our quantization process adds minimal overhead compared to the core GNN encoding, representing only 43% of the total processing time.

Compared to traditional GraphLLM approaches that require expensive projector networks for embedding alignment [50], our frozen codebook design eliminates the need for additional parameter-heavy components during inference. The quantization overhead is dominated by cosine similarity computation with the codebook, which scales linearly with the number of nodes and remains computationally tractable even for large graphs. Our framework achieves a throughput of approximately 121,920 nodes per second, demonstrating that STAG provides a practical solution for real-world deployment scenarios where computational efficiency is crucial.

4.8 Task Generalization

To demonstrate STAG’s versatility beyond node classification, we evaluate our framework on two additional graph learning tasks: link prediction and edge classification. All experiments use models pre-trained on source datasets without any task-specific training.

Link Prediction. We evaluate zero-shot binary link prediction following the protocol established by LLaGA [5]. Given a pair of nodes, we construct prompts using their quantized tokens and ask the LLM to predict whether an edge exists between them. For the non-LLM variant, we use cosine similarity between node embeddings with a fixed threshold.

Table 6 presents the comparison results. Notably, our model achieves comparable or superior performance to LLaGA despite several key differences: (1) STAG is not pre-trained on link prediction tasks, (2) we use only single-dataset pre-training (Arxiv) compared to LLaGA’s multi-dataset approach (Arxiv + PubMed + Cora), and (3) our method doesn’t use node text during inference, unlike LLaGA.

These results highlight STAG’s strong generalization capabilities and effective representation learning. The superior performance of our linear probing variant demonstrates that our learned embeddings capture meaningful structural relationships that transfer well to link prediction tasks, even without task-specific training.

Edge Classification. We evaluate edge classification on two text-attributed knowledge graphs: WN18RR (5-way) and FB15K237 (20-way) in N-way 5-shot settings. The LLM is prompted with tokenized (head, tail) pairs to predict the relation type. The non-LLM variant trains a linear classifier on concatenated node embeddings. All models are pre-trained on Cora Full.

Table 6: Zero-shot binary link prediction comparison with LLaGA.

Method	Cora	ogbn-products
LLaGA	87.35	92.99
STAG	63.00	92.65
STAG (non LLM)	93.20	96.85

LLaGA uses multi-dataset pre-training (Arxiv + PubMed for Cora, Arxiv + PubMed + Cora for ogbn-products).
STAG uses single-dataset pre-training (Arxiv only).

Table 7: N-way 5-shot edge classification results (pre-trained on Cora Full).

Method	WN18RR	FB15K237
OFA	34.35	19.55
STAG	41.75	56.60
STAG + Linear Probing	58.30	74.80

Table 7 shows that STAG significantly outperforms OFA across both datasets, demonstrating our framework’s effective structural-semantic integration for relation prediction tasks. This superior performance highlights STAG’s ability to capture meaningful relational patterns through quantized representations, even when applied to knowledge graph domains that differ substantially from the citation network used for pre-training. The consistent improvements across both WN18RR and FB15K237 validate the generalizability of our quantization approach beyond traditional graph learning tasks.

5 Conclusion

We presented STAG, a self-supervised quantization-based framework bridging graph representation learning and LLM. Our framework addresses two fundamental challenges: integrating structural and semantic information while maintaining LLM compatibility, and enabling label-free domain transfer. Through soft token assignment and distribution alignment, STAG effectively integrates structural-semantic information, enables robust cross-dataset transfer, and maintains consistent performance across LLM architectures through a shared vocabulary. Extensive experiments demonstrate STAG’s superior performance in both few-shot and zero-shot settings across diverse datasets. Looking forward, STAG could be extended to graph-level tasks like graph classification, as well as link prediction. Additionally, our framework could potentially benefit from advanced LLM architectures and techniques like chain-of-thought reasoning to enhance interpretability and performance. We believe STAG represents a significant step toward more effective and flexible integration of graph learning with LLMs.

Acknowledgments

This research / project is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (Proposal ID:

T2EP20122-0041). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
- [2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.).
- [4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* (2020).
- [5] Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. 2024. LLaGA: Large Language and Graph Assistant. In *International Conference on Machine Learning*.
- [6] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter* (2024).
- [7] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Liannmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. <https://lm-sys.org/blog/2023-03-30-vicuna/>
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- [9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
- [10] Patrick Esser, Robin Rombach, and Björn Ommer. 2021. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.
- [11] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*.
- [12] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a Graph: Encoding Graphs for Large Language Models. In *International Conference on Learning Representations*.
- [13] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*.
- [14] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shiron Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).
- [15] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066* (2023).
- [16] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*.
- [17] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-Enhanced Bert with Disentangled Attention. In *International Conference on Learning Representations*.
- [18] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. In *International Conference on Learning Representations*.
- [19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgc: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*.
- [20] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering. In *Advances in Neural Information Processing Systems*.
- [21] Zhongmou He, Jing Zhu, Shengyi Qian, Joyce Chai, and Danai Koutra. 2024. LinkGPT: Teaching Large Language Models To Predict Missing Links. *arXiv preprint arXiv:2406.04640* (2024).
- [22] Jordan Hoffmann, Sébastien Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556* (2022).
- [23] Zhenyu Hou, Yafei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. 2023. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference*.
- [24] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [25] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in neural information processing systems*.
- [26] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. 2024. Prodigy: Enabling in-context learning over graphs. In *Advances in Neural Information Processing Systems*.
- [27] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).
- [28] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.
- [29] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Naacl-HLT*.
- [30] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.
- [31] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.
- [32] Doyop Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- [33] Yuhang Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024. ZeroG: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference*.
- [35] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2024. One For All: Towards Training One Graph Model For All Classification Tasks. In *International Conference on Learning Representations*.
- [36] Ilya Loshchilov and Frank Hutter. 2017. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.
- [37] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to pre-train graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*.
- [38] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*.
- [39] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* (2000).
- [40] Péter Mernyei and Cătălina Cangea. 2020. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. *arXiv preprint arXiv:2007.02901* (2020).
- [41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).
- [42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.
- [43] Haoxuan Qu, Yujun Cai, and Jun Liu. 2024. Llms are good action recognizers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- [44] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*.
- [45] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*.
- [46] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing*.
- [47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In

- Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.*
- [48] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [49] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- [50] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- [51] Shantanu Thakoor, Corentin Tallec, Mohammad Gheşlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. 2022. Large-scale representation learning on graphs via bootstrapping. In *International Conference on Learning Representations*.
- [52] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2024. Graph neural prompting with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [53] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).
- [54] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*.
- [55] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*.
- [56] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.
- [57] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *International Conference on Learning Representations*.
- [58] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*.
- [59] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*.
- [60] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*.
- [61] Zhihao Wen and Yuan Fang. 2024. Prompt tuning on graph-augmented low-resource text classification. *IEEE Transactions on Knowledge and Data Engineering* (2024).
- [62] Yaochen Xie, Zhao Xu, and Shuiwang Ji. 2022. Self-supervised representation learning via latent graph prediction. In *International Conference on Machine Learning*.
- [63] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyani Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. 2023. A Comprehensive Study on Text-attributed Graphs: Benchmarking and Rethinking. In *Advances in Neural Information Processing Systems*.
- [64] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).
- [65] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.
- [66] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, Yongfeng Zhang, et al. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134* (2023).
- [67] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2021).
- [68] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. In *Advances in Neural Information Processing Systems*.
- [69] Haiteng Zhao, Shengchao Liu, Chang Ma, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. 2023. GIMLET: A Unified Graph-Text Model for Instruction-Based Molecule Zero-Shot Learning. In *Advances in Neural Information Processing Systems*.
- [70] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089* (2023).

- [71] Lei Zhu, Fangyun Wei, and Yanye Lu. 2024. Beyond text: Frozen large language models in visual signal comprehension. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Table 8: Hyperparameter configurations for model pre-training on different datasets.

Dataset	Cora Full	ogbn-arxiv	ogbn-products
mask rate	0.53	0.6	0.74
learning rate	5.0×10^{-5}	2.32×10^{-4}	3.47×10^{-4}
weight decay	1.88×10^{-6}	9.94×10^{-3}	1.57×10^{-3}
epoch	20	16	10
activation	elu	prelu	relu
hidden dim	256	512	512
# layers	3	1	2
# heads	2	2	4
# neg	20	23	16
τ_c	0.831	0.354	0.103
β	1.9	0.58	1.4
λ	1.0	1.0	1.6

Appendices

A Implementation Details

A.1 Hardware Configuration

In our experiments, we used a Linux machine equipped with an AMD EPYC 7763 64-core processor (3.53 GHz) and an NVIDIA L40 GPU (40 GB).

A.2 Model Configuration

Our model is pre-trained using the Adamw optimizer [36], and we employ the Tree-structured Parzen Estimator (TPE) from Optuna [2] for hyperparameter optimization. The codebook E has 15,062 tokens across all datasets. The detailed hyperparameter settings for model pretraining are provided in Table 8.

B Supplementary Experiments and Analysis

B.1 Few-shot Learning Performance

Table 9 presents the few-shot learning results when pre-trained on ogbn-arxiv. And STAG performance trends remain consistent with Cora Full pre-training results.

B.2 Zero-shot Performance

Table 10 presents zero-shot classification results on additional datasets (CiteSeer, PubMed, and ogbn-products). STAG with class-specific codebook maintains strong performance compared to raw feature baselines, demonstrating effective transfer of learned representations even without any target domain examples.

B.3 Subgraph Classification

We evaluate 5-way 5-shot subgraph classification following L2P-GNN [37]. We sample subgraphs around center nodes and use their labels for classification. Subgraph embeddings are obtained by mean-pooling node embeddings across all nodes within each subgraph, which are then quantized and used in prompts similar to our node classification approach.

Table 9: 5-way 5-shot node classification across different datasets (except PubMed: 3-way). Models are pre-trained on ogbn-arxiv with $top-k = 13$ for LLM inference. Colored cells: pre-train dataset matches target (red: ogbn-arxiv). Results show accuracy (%) averaged over 20 random tasks, with best results among our variants in bold.

Pre-train data	Method	LLM	Target data					
			Cora	Cora Full	CiteSeer	PubMed	WikiCS	ogbn-arxiv
ogbn-arxiv	DGI	✗	OOM	OOM	OOM	OOM	OOM	OOM
	GraphMAE2	✗	78.85 \pm 4.78	83.54 \pm 7.63	63.35 \pm 5.82	65.90 \pm 6.82	76.90 \pm 7.70	79.61 \pm 8.04
	GPPT	✗	25.80 \pm 5.20	32.95 \pm 10.99	28.80 \pm 6.20	18.28 \pm 11.14	25.05 \pm 7.84	64.30 \pm 13.19
	G2P2	✗	74.10 \pm 7.00	80.50 \pm 7.07	58.90 \pm 9.66	66.20 \pm 8.02	70.25 \pm 8.14	71.30 \pm 9.15
	Prodigy	✗	48.60 \pm 6.12	60.90 \pm 7.89	44.45 \pm 6.30	55.27 \pm 7.22	57.40 \pm 6.33	46.10 \pm 5.67
	OFA	✗	49.90 \pm 5.67	64.85 \pm 3.82	51.40 \pm 6.05	42.40 \pm 3.68	51.90 \pm 6.15	65.00 \pm 3.54
	STAG	✓	63.50 \pm 7.85	79.32 \pm 7.47	60.05 \pm 6.61	55.00 \pm 7.06	79.10 \pm 8.37	71.99 \pm 7.89
	+ Linear Probing	✗	76.95 \pm 5.91	84.83 \pm 7.22	65.05 \pm 6.26	69.70 \pm 6.73	82.05 \pm 6.96	82.71 \pm 8.11
	+ Prompt Tuning	✓	71.45 \pm 4.97	83.05 \pm 7.95	62.65 \pm 5.07	62.90 \pm 5.86	81.55 \pm 7.55	79.28 \pm 8.62
	+ Prompt Tuning*	✗	76.75 \pm 5.78	85.82 \pm 8.01	65.40 \pm 6.27	70.05 \pm 5.49	83.15 \pm 7.09	82.91 \pm 7.69
<p>✓/✗: LLM usage during inference; Prompt Tuning*: inference without LLM; Raw Text: Use raw text for LLM inference (To fit in the context window of LLM, raw text is truncated); Raw Feat + Quantization: Directly quantize raw node features into tokens for LLM inference; Raw Feat + Linear Probing: Train a linear classifier on raw node features without any pre-training; OOM: Out-of-memory error during training.</p>								

Table 10: 5-way zero-shot node classification results (except PubMed with 3 classes). We report accuracy (%) averaged over 20 random tasks with standard deviation.

Pre-train data	Method	LLM	Target data		
			CiteSeer	PubMed	ogbn-products
No pre-train	Raw Feat + Q	✓	43.90 \pm 5.55	42.35 \pm 2.95	62.42 \pm 10.84
	Raw Feat + C	✗	62.35 \pm 6.55	63.50 \pm 4.14	74.66 \pm 7.68
Cora Full	G2P2	✗	35.40 \pm 5.63	27.45 \pm 3.50	20.11 \pm 7.29
	OFA	✗	23.25 \pm 5.15	33.40 \pm 3.87	19.75 \pm 3.73
	STAG	✓	47.50 \pm 6.61	34.85 \pm 4.89	61.58 \pm 10.35
	STAG + C	✗	62.80 \pm 7.13	61.80 \pm 3.87	73.17 \pm 7.89

Raw Feat + Q: Quantize raw node features into tokens for LLM inference.
+ C: Classification using class-specific codebook.

Table 11: 5-way 5-shot subgraph classification results (pre-trained on Cora Full).

Method	Cora	Cora Full	Arxiv
Raw Feat + Quantization	67.75	78.32	65.18
STAG	69.60	79.25	68.41

As shown in Table 11, STAG consistently outperforms the raw feature quantization baseline across all three datasets. While the improvements are modest, they demonstrate that our structure-aware quantization approach successfully incorporates neighborhood information even at the subgraph level.

These comprehensive evaluations across link prediction, edge classification, and subgraph classification validate STAG’s flexibility in handling different granularities and types of graph learning tasks beyond the primary focus on node classification.

C Few-shot and Zero-shot Prompt Templates

For few-shot classification, we use the following prompt template:

You are an AI assistant tasked with classifying input word sequences into one of the following categories: [candidate classes are inserted here].

You must choose strictly from these categories and no others.

Each category has characteristic patterns shown in its examples.

Here are examples of input sequences and their corresponding categories to guide you:
[Support examples are inserted here]

When given a new input sequence, identify its key patterns and match them to the most similar category from the examples.

If no category is a clear match, choose the closest one.

IMPORTANT: Output only the category name and nothing else.

Input: [tokens of test node inserted here]

For zero-shot classification, we use a simpler prompt template:

You are an AI assistant tasked with classifying input word sequences into one of the following categories: [candidate classes are inserted here].

You must choose strictly from these categories and no others.

When given a new input sequence, classify it into one of the categories.

IMPORTANT: Output only the category name and nothing else.

Input: [tokens of test node inserted here]