Compressing LLM Knowledge into Graph Representations for Text-attributed Graphs Learning

Runhuai Chen[1], Dian Shen[1*], Dandan Zhang[2]
Kaihong Huang[1], Linghui Meng[1], Beilun Wang[1]
[1]Southeast University
[2]Hangzhou Dianzi University
[*]**Correspondence:** dshen@seu.edu.cn

Abstract

Text-attributed graphs (TAGs) require jointly modeling relational structure and node-level text. Existing GNN-LLM approaches perform by incorporating large language models at inference time for processing the text attributes, resulting in costly deployment. More fundamentally, LLM knowledge is typically used in a sample-wise manner, leading to inefficient utilization across graph instances. In this work, we study how interactions with LLM embedding spaces affect graph representations, and show that projecting into the LLM space can learn better GNNs. That is to say, the knowledge encoded in LLM embeddings can be compressed into graph representations. Based on this insight, we propose a framework that internalizes LLM knowledge within graph models and supports inference-efficient TAG learning. Our framework employs a hierarchical Proxy-Purifier module with distribution-level regularization, using LLM embeddings only as training-time guidance. With this module, the model operates TAGs without invoking LLMs, achieving high efficiency as standard GNNs without LLMs. Notably, experiments on five popular TAG tasks further demonstrate that our method can also achieve consistent performance gains, in comparison to existing GNN-LLM approaches.

1 Introduction

Text-attributed graph (TAG) learning requires models to jointly reason over structured relations and natural language descriptions, making them a natural extension of language understanding to structured reasoning settings (Hamilton et al., 2017a; Zhou et al., 2020). In such scenarios, linguistic information provides semantic context, while graph structure encodes relational constraints that enable reasoning beyond isolated text.

A common approach for TAG learning is to combine graph neural networks (GNNs) (Kipf and Welling, 2017; Velickovic et al., 2018; Hamilton et al., 2017b; Xu et al., 2019), which model relational structure, with textual encoders that process node-associated descriptions. With the emergence of large language models (LLMs), recent methods further incorporate LLMs into this paradigm, leveraging their strong representational capacity to enhance graph learning (Zhao et al., 2023; Liu et al., 2024; He et al., 2024), and achieving notable empirical gains on various TAG tasks.

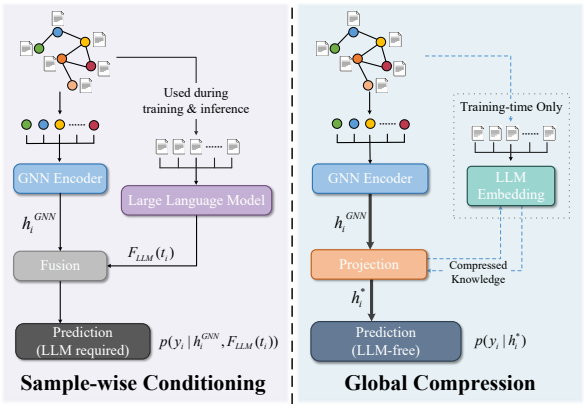


Figure 1: **Comparison of LLM knowledge utilization paradigms in graph learning.** Left: sample-wise LLM conditioning. Right: global knowledge compression with LLM-free inference.

Despite these advances, existing GNN-LLM approaches face two fundamental challenges in practice. First, they usually require computing LLM representations for each sample at inference time, leading to costly deployment pipelines (Li et al., 2024; Ren et al., 2024). More fundamentally, the way LLM knowledge is utilized in these methods remains inefficient. In most cases, LLM knowledge is consumed in a sample-wise manner, where LLM representations are repeatedly used for individual instances rather than being compressed into a reusable form that can support generalization across instances (Tang et al., 2024; Wang et al., 2023; Zhu et al., 2024). Figure 1 contrasts this sample-wise conditioning paradigm with an alter-

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 30303–30318

native global knowledge compression perspective.

From a modeling perspective, we formalize existing GNN-LLM approaches as relying on sample-wise conditioning,

$$p(y_i \mid G, t_i) = p\big(y_i \mid h_i^{\mathrm{GNN}}, \mathcal{F}_{\mathrm{LLM}}(t_i)\big),$$

where the LLM encoder $\mathcal{F}_{\mathrm{LLM}}(\cdot)$ must be invoked independently for each instance. As a result, LLM knowledge appears explicitly at prediction time and is not accumulated into the graph representation itself.

This motivates us to explore whether LLM knowledge can be compressed into graph representations, instead of being utilized through sample-wise conditioning. In modeling terms, we frame this question as an internalized representation transformation process,

$$h_i^* = \Phi\big(h_i^{\mathrm{GNN}}; \mathcal{F}_{\mathrm{LLM}}\big), \quad p(y_i \mid G) = p(y_i \mid h_i^*),$$

where LLM interactions guide representation learning and are removed at inference time.

By examining GNN-LLM interactions from both representation-level geometry and task-level effects, we observe consistent patterns across datasets. Projecting GNN representations into the LLM embedding space leads to more organized embedding geometry and more stable decision behavior. These improvements reveal that the knowledge encoded in LLM embedding spaces can be compressed into graph representations, rather than being used in a sample-wise manner.

Motivated by this insight, we propose a trainable framework that explicitly treats GNN-LLM interaction as a process of knowledge compression. Rather than repeatedly consuming LLM representations for individual instances, our method compresses information encoded in the LLM embedding space into graph representations through a projection module and a controlled refinement mechanism that preserves essential instance-level distinctions. A distribution-level regularization strategy is further introduced to stabilize the compression process.

Under this framework, LLMs are used only during training as supervisory signals to guide knowledge compression. At inference time, the model operates without invoking LLMs, relying solely on graph representations. Experiments on five real-world TAG benchmarks, our method delivers consistent performance gains with an average Macro-F1 improvement of 2.7%, while maintaining inference efficiency comparable to standard GNNs.

Our contributions are summarized as follows:

- We analyze GNN-LLM interactions in text-attributed graph learning, showing that interactions with LLM embedding spaces improve graph representations and support effective knowledge compression.
- We propose a trainable framework that compresses LLM knowledge into graph representations during training, while supporting fully LLM-free inference.
- We introduce a distribution-level regularization strategy that stabilizes the compression process and preserves global structure.
- Extensive experiments on standard TAG benchmarks demonstrate consistent performance gains while significantly reducing inference-time cost.

2 Related Work

2.1 Text-Attributed Graph Learning

Text-attributed graphs (TAGs), where nodes are associated with both relational structure and textual descriptions, are widely studied in applications such as citation networks, knowledge graphs, and social platforms. Early TAG methods typically encode textual attributes using bag-of-words or TF-IDF representations and integrate them into graph models through shallow fusion mechanisms, including linear text models combined with graph-based matrix factorization or early GCN-style architectures (Yang et al., 2016; Tang et al., 2015; Perozzi et al., 2014; Grover and Leskovec, 2016).

Subsequent work improves representational capacity by incorporating pretrained word embeddings or neural text encoders into graph neural networks. Representative models such as TADW and TextGCN jointly model textual and structural information for node-level tasks (Yang et al., 2015; Yao et al., 2019). While these methods outperform purely structural baselines, textual attributes are still treated as auxiliary features and rely on task-specific encoders.

Recent surveys on graph foundation models point out that traditional TAG methods remain limited in leveraging external knowledge beyond the observed corpus, with inductive biases largely governed by graph topology and local feature statistics (Ren et al., 2024; Hamilton et al., 2017a; Zhou et al., 2020). These limitations motivate the inte-

gration of large pretrained language models into TAG learning.

2.2 GNN-LLM Integration for TAGs

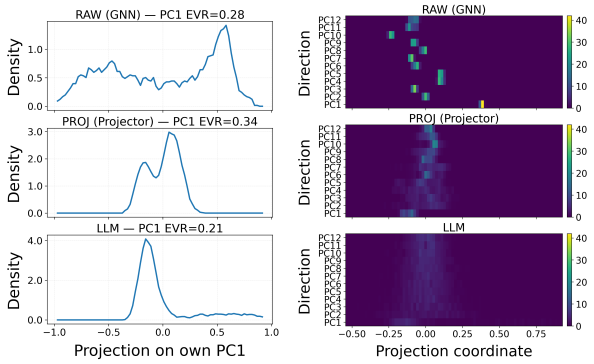
Recent work explores integrating graph neural networks with large language models to exploit semantic knowledge learned through large-scale pretraining. Existing GNN-LLM methods can be broadly categorized by the role of the LLM in the pipeline (Li et al., 2024; Ren et al., 2024).

LLM as Enhancer. A common paradigm treats LLMs as semantic enhancers that generate contextualized embeddings for node-associated texts, which are then incorporated into GNNs as enriched node features. These methods improve semantic representation quality through feature concatenation or lightweight fusion modules (Chai et al., 2023; He et al., 2024; Liu et al., 2024). However, they typically require recomputing LLM embeddings at inference time and rely on sample-wise consumption of LLM representations, leading to high deployment cost and a lack of knowledge reuse (Li et al., 2024).

LLM as Predictor. Another line of work employs LLMs directly as predictors for graph-related tasks such as node classification, link prediction, or subgraph reasoning (Tang et al., 2024; Wang et al., 2023; Chai et al., 2023). Related methods in this direction also explore prompting, instruction tuning, or lightweight LLM-centric adaptation to better exploit pretrained LLM priors for graph learning (Tang et al., 2024; Huang et al., 2024). While flexible, these approaches still keep the LLM within the task-solving pipeline and typically depend on repeated LLM inference or LLM-centric adaptation, rather than explicitly internalizing knowledge into compact graph representations (Li et al., 2024).

LLM as Aligner. LLM-as-aligner approaches jointly train GNNs and LLMs as parallel components, coordinating structural and linguistic representations through unified or iterative optimization frameworks (Zhao et al., 2023; Jin et al., 2023; Zhu et al., 2024). Although effective for multimodal alignment, these methods primarily focus on representation consistency and often retain dependence on LLM components during training or inference, rather than compressing LLM knowledge into standalone graph models.

Overall, existing GNN-LLM approaches differ in integration strategies but share a common limitation: LLM knowledge is not sufficiently compressed into graph representations, and remains tied to sample-wise usage or external LLM components. This motivates the need for more effective mechanisms to compress and reuse LLM knowledge in graph learning.



(a) 1D density along representation PCA directions. (b) Density under LLM-induced PCA directions.

Figure 2: **Representation space analysis on the Cora dataset.** (a) PC1 density of node embeddings with explained variance ratio (EVR). (b) Density patterns under LLM-induced PCA coordinates.

Table 1: Linear probe performance on frozen RAW and PROJ representations.

Dataset	Representation	Acc (%) ↑	F1 (%) ↑
Cora	RAW (GNN)	86.50	84.82
	PROJ (Projected)	**87.04**	**85.32**
Citeseer	RAW (GNN)	76.13	67.49
	PROJ (Projected)	**77.22**	**73.66**
PubMed	RAW (GNN)	89.28	88.75
	PROJ (Projected)	**89.53**	**89.07**

3 Understanding Projection-Induced Structure from LLM Embeddings

In this section, we present an empirical analysis of how interactions between GNN representations and LLM embedding spaces lead to improved graph representations, independent of downstream task architectures, and whether such reorganization can serve as a mechanism for extracting global knowledge from LLMs. To examine these interaction-induced effects in a controlled manner, we use a simplified projection-based diagnostic setup. The projected representations, obtained by mapping GNN embeddings into the LLM embedding space, are used only for diagnostic analysis here and do not correspond to the complete framework developed later in Section 4.

3.1 Representation Space Analysis

We first examine how projection into the LLM embedding space affects the geometry of graph representations at the embedding level. Specifically, we compare three types of representations: raw GNN embeddings (RAW), projected GNN embeddings (PROJ), and LLM embeddings. Here, PROJ refers to the representations obtained by mapping GNN embeddings into the LLM embedding space. These representations are used for diagnostic analysis in the following subsections. Detailed settings of this projection-based diagnostic setup are provided in Appendix B.

Geometry Reshaping under Projection. Figure 2(a) compares RAW, PROJ, and LLM representations on Cora using label-free diagnostics. For each representation space, we perform PCA independently and project node embeddings onto the first principal component (PC1), reporting the explained variance ratio (EVR). RAW shows a diffuse PC1 density with low explained variance, while PROJ exhibits a sharper and more organized PC1 structure with higher EVR, indicating that projection induces global geometric reorganization rather than noise-dominated embeddings. Importantly, this reorganization does not imply collapse into the LLM space, but reflects a structured reshaping that preserves graph-specific variability.

Representations under LLM-induced Coordinates. To further examine how interaction with the LLM embedding space reshapes graph representations, we analyze all representations under a shared coordinate system defined by LLM embeddings. We compute principal directions from LLM embeddings and use them as projection axes, onto which RAW and PROJ representations-mapped to the LLM dimensionality and centered by the LLM meanare projected.

As shown in Figure 2(b), PROJ embeddings form a more continuous density band around the central coordinates across multiple LLM principal directions, whereas RAW embeddings appear as more scattered and isolated high-density blocks. This suggests improved compatibility with the global structure encoded by the LLM space, without enforcing direct semantic alignment.

Taken together, these results suggest that projection enables the internalization of LLM-induced distributional priors, reorganizing graph representations toward more structured global geometry. Additional results on Citeseer and PubMed are provided in Appendix A.

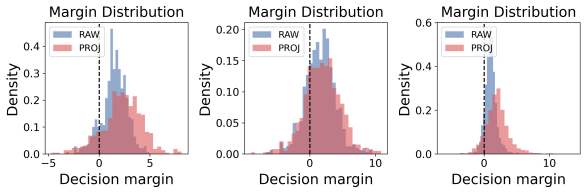


Figure 3: **Decision margin distributions of RAW and PROJ representations.** PROJ exhibits a consistent rightward shift, indicating more stable decisions.

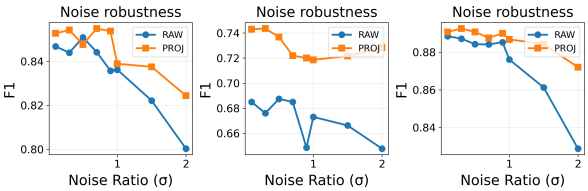


Figure 4: **Robustness of RAW and PROJ representations under feature noise.** PROJ degrades more slowly, indicating improved robustness.

3.2 Task-Level and Robustness Analysis

We next examine whether the projection-induced geometric reorganization observed in Section 3.1 leads to improved decision behavior and robustness at the task level. All evaluations in this section are conducted as diagnostic probes of representation quality, using the same lightweight linear classifier trained on frozen representations. This setting isolates the effect of representation geometry from architectural or optimization differences.

Linear Probe Evaluation. As a first diagnostic, we evaluate representation quality using a linear probe for node classification. A single linear classifier is trained on frozen RAW and PROJ representations for each dataset.

Table 1 reports accuracy and macro-F1 scores. Across all datasets, PROJ representations consistently outperform RAW embeddings, indicating that projection-induced reorganization improves downstream separability even under a minimal classifier. This suggests that the benefits of projection are not tied to specific task heads, but arise from intrinsic changes in representation structure.

Decision Margin Distribution. To further examine decision stability, we analyze the distribution of classification margins produced by linear classifiers trained on RAW and PROJ representations. For each node, the margin is defined as the difference between the logit of the ground-truth class and the maximum logit of other classes.

As shown in Figure 3, PROJ representations exhibit a clear rightward shift in margin distributions

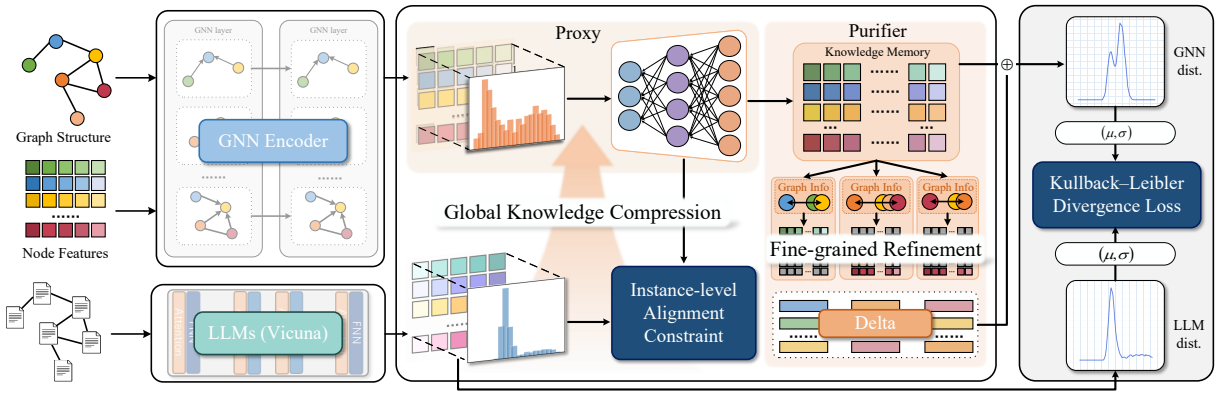


Figure 5: Overall framework of the proposed **Hierarchical Proxy-Purifier** model.

compared to RAW embeddings, with fewer samples near or below zero. This indicates more confident and stable decisions induced by projection, consistent with improved global organization of the representation space.

Robustness under Feature Noise. We further evaluate robustness by injecting controlled Gaussian noise into node features at test time and measuring performance degradation. Figure 4 reports F1 scores under increasing noise ratios for both RAW and PROJ representations.

Across all datasets, PROJ embeddings degrade more slowly than RAW embeddings as noise increases, retaining substantially higher performance over a wide range of perturbation strengths. This behavior suggests that projection-induced structure filters out noise variability, leading to improved robustness to feature perturbations.

Taken together, these task-level analyses demonstrate that projection into the LLM embedding space yields tangible improvements in decision stability and robustness, even when evaluated with frozen representations and lightweight classifiers. These effects align with the geometric reorganization observed in Section 3, reflecting both global structural changes in representation geometry and local improvements in decision behavior. This consistency supports the view that projection internalizes useful distributional priors rather than acting as a superficial feature transformation. Motivated by these projection-based diagnostic observations, we next develop a full training framework that explicitly regulates and compresses such inductive bias during training.

4 Method

In this section, we present an inference-efficient framework that compresses LLM-induced knowledge into graph representations through GNN-LLM embedding interactions.

Motivated by the diagnostic findings in Section 3, which reveal complementary global and instance-level effects, we adopt a hierarchical design implemented via a Proxy-Purifier module and optimized with a two-stage training strategy.

4.1 Problem Setup

We consider a text-attributed graph (TAG) $G = (V, E, X, T)$, where each node $v_i \in V$ is associated with a structural feature vector $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and a text embedding $\mathbf{t}_i \in \mathbb{R}^{d_t}$ extracted from a pretrained LLM.

A GNN encoder $f_\theta(\cdot)$ maps the graph to node representations:

$$\mathbf{z}_i = f_\theta(X, E)_i, \quad \mathbf{z}_i \in \mathbb{R}^{d_z}. \tag{1}$$

Unlike prior GNN-LLM approaches, we assume that text embeddings $\{\mathbf{t}_i\}$ are available only during training. Our goal is to compress LLM-induced knowledge into graph representations that can be directly used for downstream tasks without requiring LLM access at inference time.

4.2 Knowledge Compression via Projection

We view graph-text interaction not as representation matching, but as compressing LLM-induced inductive bias into graph representations.

LLM embeddings encode global regularities shaped by large-scale pretraining. Projection into the LLM embedding space serves as the core mechanism to compress such bias during training, so that it is retained in model parameters and does not require invoking LLMs at inference.

Effective compression requires representations that are both (i) **structure-preserving**, capturing global distributional regularities, and (ii) **instance-aware**, allowing node-level adjustment. This motivates a hierarchical design that integrates global bias injection with local refinement.

4.3 Hierarchical Proxy-Purifier Module

Projection into the LLM embedding space improves graph representations at both global and instance levels. Based on this observation, we introduce a hierarchical Proxy-Purifier module.

Proxy: Global Inductive Bias Projection. The proxy module $g_\phi(\cdot)$ projects GNN embeddings into the LLM embedding space:

$$\hat{\mathbf{p}}_i = g_\phi(\mathbf{z}_i), \quad \hat{\mathbf{p}}_i \in \mathbb{R}^{d_t}, \tag{2}$$

where $\mathbf{z}_i \in \mathbb{R}^{d_z}$ denotes the graph representation of node v_i.

The proxy injects coarse-grained inductive bias by leveraging the geometry of the LLM embedding space to compress global structural regularities into graph representations.

Purifier: Instance-level Refinement. While the proxy captures global structure, it may not fully account for node-specific graph context. We therefore introduce a purifier module $h_\psi(\cdot)$ to perform instance-level refinement:

$$\mathbf{p}_i = \hat{\mathbf{p}}_i + h_\psi([\mathbf{z}_i; \hat{\mathbf{p}}_i]), \tag{3}$$

where $[\cdot;\cdot]$ denotes vector concatenation.

The purifier jointly conditions on the original graph embedding $\mathbf{z}_i$ and its proxy projection $\hat{\mathbf{p}}_i$, enabling fine-grained adjustment of the injected inductive bias based on graph-specific information. Importantly, the purifier does not rely on text embeddings, allowing LLMs to serve solely as training-time guidance without participating in inference. The detailed implementation configurations of the Proxy-Purifier module are provided in Appendix C.

4.4 Training Objective

The Proxy-Purifier module is trained using a combination of instance-level correspondence constraints and distribution-level regularization, both designed to shape graph representations according to LLM-induced inductive bias.

Instance-level Alignment Constraint. We impose an instance-level constraint between the refined embedding $\mathbf{p}_i$ and the corresponding text embedding $\mathbf{t}_i$ using a mean squared error objective:

$$\mathcal{L}_{\text{inst}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \|\mathbf{p}_i - \mathbf{t}_i\|_2^2, \tag{4}$$

where $\mathcal{A}$ denotes the node set used for alignment.

This loss guides representation compression during training, without enforcing instance-wise matching at inference time.

Table 2: Dataset statistics.

Dataset	#Nodes	#Edges	#Classes	d_{avg}
Cora	2,708	5,429	7	3.9
Citeseer	3,312	4,732	6	2.7
PubMed	19,717	44,338	3	4.5
WikiCS	11,701	216,123	10	36.9
arXiv	169,343	1,166,243	40	13.8

Distribution-level Structure Regularization. To further preserve the global statistical structure of the LLM embedding space, we regularize the first- and second-order moments of the aligned representations, encouraging graph representations to internalize the distributional priors encoded in LLM embeddings. Specifically, we approximate the distributions of $\{\mathbf{p}_i\}$ and $\{\mathbf{t}_i\}$ with diagonal Gaussian distributions and minimize their KL divergence:

$$\mathcal{R}_{\text{KL}} = D_{\text{KL}}\big(\mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\sigma}_p^2) \,\|\, \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2)\big). \tag{5}$$

The final alignment objective is:

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{inst}} + \lambda_{\text{KL}} \mathcal{R}_{\text{KL}}, \tag{6}$$

which jointly enforces local correspondence and global distributional structure.

4.5 Two-stage Optimization

We adopt a two-stage training strategy to decouple knowledge compression from task supervision.

Stage 1: Knowledge Compression. In the first stage, we optimize the Proxy-Purifier module using $\mathcal{L}_{\text{align}}$. LLM-derived embeddings are used only in this stage to guide the compression and internalization of LLM-induced inductive bias. The GNN encoder may be fixed or jointly optimized, depending on the setting.

Stage 2: Task-specific Supervision. In the second stage, the alignment module is frozen, and a lightweight task-specific head is trained using a standard supervised loss:

$$\mathcal{L}_{\text{task}} = \ell(\hat{y}, y). \tag{7}$$

This stage trains a task head on the compressed representations to assess their effectiveness, without further modifying the alignment modules.

4.6 Inference

At inference time, the proposed method operates without reliance on LLMs. Given a graph $G =$

Table 3: Node classification results (Accuracy (%), Macro-F1 (%), and test-time latency (ms)) on five TAG datasets. Latency is measured on the test split under the same hardware and batch setting.

Method	Cora			Citeseer			PubMed			WikiCS			arXiv		
	Acc ↑	F1 ↑	Time ↓	Acc ↑	F1 ↑	Time ↓	Acc ↑	F1 ↑	Time ↓	Acc ↑	F1 ↑	Time ↓	Acc ↑	F1 ↑	Time ↓
GCN	86.86	85.08	1.55	76.44	72.83	1.8	89.56	89.05	2.1	85.14	83.74	3.22	59.29	25.19	8.01
GAT	86.4	84.41	2.2	74.38	71.0	2.38	87.28	86.45	2.63	85.4	84.93	3.79	60.56	26.64	10.1
GraphSAGE	86.76	85.83	**1.2**	73.75	71.63	**1.18**	90.57	90.35	**1.14**	85.65	84.2	**1.25**	60.68	26.6	**1.2**
GIN	83.82	81.65	1.23	72.19	70.6	1.2	88.19	87.68	1.17	84.03	83.12	1.43	43.51	8.05	1.22
DIFFormer	74.42	73.46	44.17	58.11	54.67	61.02	74.66	74.57	186.44	75.56	65.35	103.37	46.08	13.47	1394.66
CoBFormer	71.62	69.86	21.95	58.42	54.05	48.26	48.96	46.38	133.22	70.87	67.55	112.86	56.07	20.38	971.81
PATTON	61.63	58.34	10291.32	58.93	55.98	9897.28	83.17	81.66	21918.40	62.88	59.69	59319.62	34.1	25.67	129385.03
ENGINE	88.51	87.14	197.56	78.24	72.84	175.74	91.28	90.96	504.98	83.9	82.69	2849.49	73.84	53.91	8436.35
LOGIN	70.9	67.27	7.17	59.72	51.59	7.49	78.98	78.33	12.34	68.3	52.23	21.89	53.95	25.35	54.00
TAPE	79.93	77.89	7.00	60.33	57.72	6.48	68.91	65.23	11.75	68.03	54.21	35.13	72.51	51.55	132.39
OFA	80.56	78.77	3051.96	70.46	66.85	2850.14	77.46	77.58	15390.47	79.43	77.11	17388.76	74.78	**59.38**	166599.63
LinguGKD	77.18	76.12	216.94	67.23	64.18	1078.99	70.41	35.55	1372.46	71.24	38.25	1902.64	69.83	34.21	10083.79
GraphAdapter	75.82	73.62	118.37	69.80	66.95	104.58	72.50	73.15	186.95	75.71	71.98	201.67	70.89	48.07	3248.41
GraphGPT	44.18	40.62	186.47	28.94	22.17	173.26	66.91	65.68	248.35	55.84	41.37	261.48	60.48	37.92	918.64
Ours	**94.89**	**94.22**	1.96	**82.84**	**80.99**	1.97	**91.38**	**91.16**	2.3	**87.25**	**85.56**	3.47	**75.16**	56.84	8.23

(V, E), node representations are computed as:

$$\begin{aligned} \mathbf{z}_i &= f_\theta(X, E)_i, \\ \hat{\mathbf{p}}_i &= g_\phi(\mathbf{z}_i), \\ \mathbf{p}_i &= \hat{\mathbf{p}}_i + h_\psi([\mathbf{z}_i; \hat{\mathbf{p}}_i]). \end{aligned} \tag{8}$$

The resulting representations $\{\mathbf{p}_i\}$ are fed into downstream task heads. Inference cost is dominated by the GNN encoder, with only lightweight overhead from the proxy and purifier.

5 Experiments

5.1 Experimental Setup

We conduct experiments on multiple benchmarks to evaluate effectiveness, robustness, and inference efficiency.

Datasets. We evaluate our method on widely used real-world text-attributed graph datasets for node classification, including Cora and Citeseer (Sen et al., 2008), PubMed (Namata et al., 2012), WikiCS (Mernyei and Cangea, 2020), and ogbn-arXiv (Hu et al., 2020). These datasets cover diverse scales and graph characteristics. Dataset statistics are summarized in Table 2. We further report results on two additional non-citation TAG benchmarks in Appendix E. These additional evaluations extend our study to broader TAG scenarios and provide a more comprehensive assessment of generalization.

Baselines. We compare our method with two categories of baselines. **GNN baselines** include GCN (Kipf and Welling, 2017), GAT (Velickovic et al., 2018), GraphSAGE (Hamilton et al., 2017b), GIN (Xu et al., 2019), DIFFormer (Wu et al., 2023), and CoBFormer (Xing et al., 2024). **LLM-GNN methods** include PATTON (Jin et al., 2023), ENGINE (Zhu et al., 2024), LOGIN (Qiao et al., 2025), TAPE (He et al., 2024), OFA (Liu et al., 2024), LinguGKD (Hu et al., 2025), GraphGPT (Tang et al., 2024), and GraphAdapter (Huang et al., 2024). All baselines are treated as black-boxes and use the same downstream prediction head to ensure fair comparison.

Hyperparameters and Implementation Details. Hyperparameters are selected on the validation split, and results are averaged over multiple random seeds. Details are provided in Appendix C.

Evaluation Metrics. We report Accuracy and Macro-F1. Inference efficiency is measured by the average test-time latency under the same hardware and batch setting for all methods.

Table 4: Ablation study on key components of the proposed framework (Macro-F1 (%)).

Variant	Cora	Citeseer	PubMed	WikiCS	arXiv
w/o Compression	89.34	67.89	89.12	82.74	27.06
Proxy Only	93.44	78.58	90.86	84.62	45.65
Proxy + Purifier	93.96	80.41	91.03	85.3	56.42
Proxy + KL	93.89	78.94	90.97	84.16	52.28
Full Model	**94.22**	**80.99**	**91.16**	**85.56**	**56.84**

Table 5: Compatibility with different GNN backbones (Macro-F1 (%)).

Backbone	Cora	Citeseer	PubMed	WikiCS	arXiv
SAGE w/o Compression	85.83	71.63	90.35	84.2	26.6
SAGE w/ Compression	92.64	80.16	91.02	85.39	55.41
GAT w/o Compression	84.41	71.0	86.45	84.93	26.64
GAT w/ Compression	89.79	80.0	87.34	85.32	54.01
GIN w/o Compression	81.65	70.6	87.68	83.12	8.05
GIN w/ Compression	93.31	79.41	90.9	84.77	39.06
Full Model (Ours)	**94.22**	**80.99**	**91.16**	**85.56**	**56.84**

5.2 Main Results: Performance and Inference Efficiency

Table 3 reports Accuracy, Macro-F1, and test-time latency on five TAG benchmarks.

Our method achieves an average Macro-F1 of

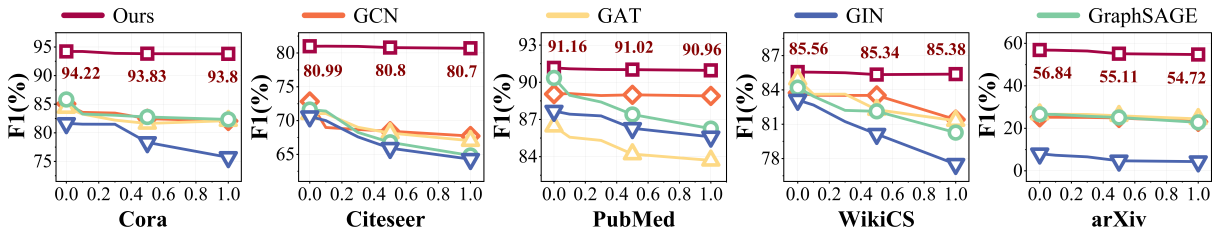


Figure 6: **Robustness to feature noise.** Test-set performance under increasing levels of feature noise across five benchmark datasets (Cora, Citeseer, PubMed, WikiCS, and arXiv).

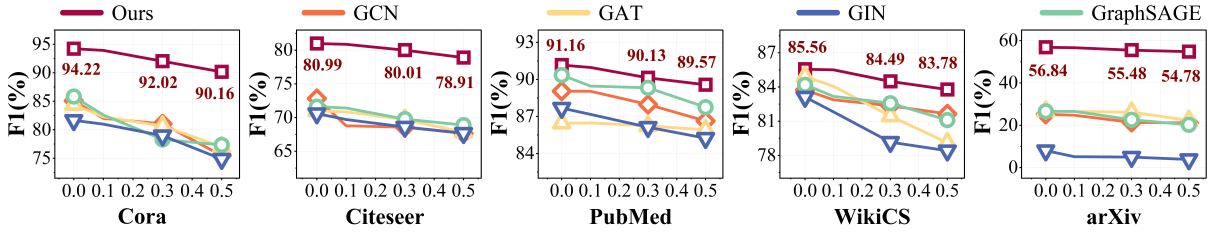


Figure 7: **Robustness to structural perturbations.** Test-set performance under increasing levels of edge deletion across five benchmark datasets (Cora, Citeseer, PubMed, WikiCS, and arXiv).

81.8% across all datasets. Compared to standard GNN baselines, this corresponds to an average improvement of about 9.6% in Macro-F1, and it also outperforms the strongest LLM-GNN baseline by approximately 3.1% on average. Importantly, these gains are obtained with inference latency comparable to conventional GNNs and several orders of magnitude lower than LLM-based methods, confirming the effectiveness of training-time LLM knowledge compression.

5.3 Ablation Studies

We ablate key components of the framework, focusing on the proxy projection, purifier refinement, and distribution-level regularization.

5.3.1 Effect of Compression Components

Table 4 reports the effect of progressively enabling key components. Adding the proxy improves F1 by an average of 7.4%, reflecting effective LLM knowledge internalization. Adding the purifier brings a further average gain of 2.8%, indicating improved instance-level refinement. The full model performs best, supporting the complementary roles of projection, refinement, and distribution-level regularization.

5.3.2 KL-based Distribution Regularization

Table 4 compares variants with and without KL regularization. KL improves performance by an average of 1.4% across datasets, highlighting the role of preserving global distributional structure during compression. We further compare KL with alternative matching losses in Appendix D.

5.4 Robustness Analysis

We evaluate robustness under both feature perturbations and structural perturbations.

Feature Noise. We first evaluate robustness under increasing test-time feature noise. As shown in Figure 6, baseline GNNs degrade rapidly as noise increases, while our method degrades more gradually across datasets. These results suggest that representations learned through LLM-guided compression are generally more robust to feature perturbations and can better preserve predictive performance under noisy conditions.

Structure Perturbations. We further evaluate robustness to structural perturbations by progressively deleting a fraction of edges at test time. As shown in Figure 7, our method remains consistently more stable than standard GNN baselines across all five datasets, exhibiting consistently smaller performance drops as the edge deletion ratio increases. These results indicate that the proposed compression framework improves not only robustness to feature noise but also resilience to structural perturbations in the input graph.

5.5 Compatibility across GNN Backbones

To test whether the framework depends on a specific GNN architecture, we apply it to multiple backbones. Table 5 shows consistent improvements across GNN encoders, indicating that the method is backbone-agnostic.

5.6 Hyperparameter Sensitivity

We analyze sensitivity to key hyperparameters. Figure 8 further shows low sensitivity to the KL

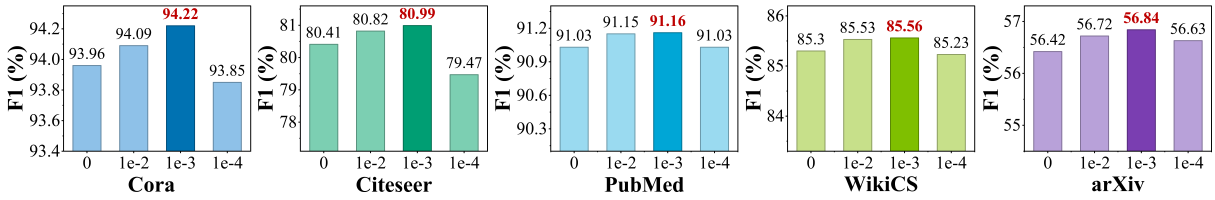


Figure 8: **Sensitivity to KL regularization weight.** Performance across different values of the KL divergence weight on five benchmark datasets (Cora, Citeseer, PubMed, WikiCS, and arXiv).

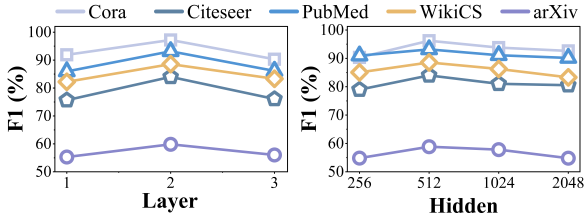


Figure 9: **Sensitivity to purifier hyperparameters.** Performance under different purifier configurations. **Left:** purifier depth. **Right:** hidden dimension.

regularization weight, suggesting that distribution-level regularization provides consistent benefits without careful tuning. Figure 9 shows stable performance across a wide range of purifier depths and hidden dimensions, indicating robustness to purifier capacity.

By design, the proxy module is implemented as a lightweight projection to expose graph representations to the global geometry of the LLM embedding space. We find that its performance is largely insensitive to architectural capacity, supporting the use of a shallow proxy in practice. Additional analysis is provided in Appendix F.

6 Conclusion

In this work, we study whether LLM knowledge can be leveraged in text-attributed graph learning without requiring inference-time usage to LLMs. Through an empirical analysis of interactions between GNN representations and LLM embedding spaces, we show that even lightweight projection induces systematic geometric and statistical reorganization in graph representations. These effects reflect distributional priors encoded by LLMs that can be internalized during training.

Building on this observation, we propose an inference-efficient framework that compresses such priors into graph models, eliminating the need for inference-time LLM access while retaining useful training-time guidance. Experiments on multiple benchmarks demonstrate consistent performance gains with inference costs comparable to standard GNNs. Overall, this work provides a principled perspective on compressing and internalizing large-model knowledge into graph representations, enabling effective and deployable graph learning on TAGs.

Limitations

This work focuses on node classification on static text-attributed graphs, and the effectiveness of compressing LLM-induced knowledge for other graph learning tasks or dynamic settings remains to be explored. Although we provide additional results on broader real-world TAG benchmarks, the present study does not yet fully characterize cross-dataset or cross-domain transfer. Accordingly, the extent to which the compressed LLM priors generalize beyond within-dataset TAG learning remains unclear. The effect of noisy or irrelevant text attributes is also not systematically studied in the current work. Since the framework uses text embeddings as training-time supervision, the quality and relevance of these signals may influence the overall fidelity of the compressed priors. In addition, the proposed approach relies on pretrained LLM embeddings during training as a source of global distributional priors, and the quality of the compressed knowledge may vary with the choice of language model and the relevance of textual attributes to the underlying graph structure. Finally, although inference is fully LLM-free and efficient, the training stage introduces additional computation for knowledge compression, which may require further optimization when scaling to very large graphs.

Acknowledgments

This work was supported by Guangxi Science and Technology Achievement Transformation Program No.ZG2504240019, Science and Technology Major Special Program of Jiangsu Grants No. BG2024028, the Natural Science Foundation of Jiangsu Province under Grant No. BK20253020, BK20230083, National Natural Science Foundation of China under Grant No.62522205, 62272101, and in part by the Collaborative Inno-

vation Center of Novel Software Technology and Industrialization, the Big Data Computing Center of Southeast University.

References

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *CoRR*, abs/2310.05845.

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 (KDD 2016), San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017a. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017b. Inductive representation learning on large graphs. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, December 4-9, 2017*, pages 1024–1034.

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 *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024), Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shengxiang Hu, Guobing Zou, Song Yang, Shiyi Lin, Yanglan Gan, Bofeng Zhang, and Yixin Chen. 2025. Large language model meets graph neural network in knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2025), Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 17295–17304. AAAI Press.

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 *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS 2020), December 6-12, 2020, virtual*.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024. Can gnn be good adapter for llms? In *Proceedings of the ACM on Web Conference 2024 (WWW 2024), Singapore, May 13-17, 2024*, pages 893–904. ACM.

Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. 2023. Patton: Language model pretraining on text-rich networks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023), Toronto, Canada, July 9-14, 2023*, pages 7005–7020. Association for Computational Linguistics.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, April 24–26, 2017*. OpenReview.net.

Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Wai Kin (Victor) Chan, and Jia Li. 2024. Glbench: A comprehensive benchmark for graph with large language models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2024), Vancouver, BC, Canada, December 10-15, 2024*.

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 *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024), Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Péter Mernyei and Catalina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *CoRR*, abs/2007.02901.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *Proceedings of the 10th International Workshop on Mining and Learning with Graphs*, volume 8, page 1.

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 (KDD 2014), New York, NY, USA, August 24-27, 2014*, pages 701–710. ACM.

Yiran Qiao, Xiang Ao, Yang Liu, Jiarong Xu, Xiaoqian Sun, and Qing He. 2025. Login: A large language model consulted graph neural network training framework. In *Proceedings of the 18th ACM International Conference on Web Search and Data Mining (WSDM 2025), Hannover, Germany, March 10-14, 2025*, pages 232–241. ACM.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh V. Chawla, and Chao Huang. 2024. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2024), Barcelona, Spain, August 25-29, 2024*, pages 6616–6626. ACM.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.*, 29(3):93–106.

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 (SIGIR 2024), Washington DC, USA, July 14-18, 2024*, pages 491–500. ACM.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW 2015), Florence, Italy, May 18-22, 2015*, pages 1067–1077. ACM.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, December 10-16, 2023*.

Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. 2023. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. In *Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. 2024. Less is more: On the over-globalizing problem in graph transformers. In *Proceedings of the 41st International Conference on Machine Learning (ICML 2024), Vienna, Austria, July 21–27, 2024*. OpenReview.net.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 25-31, 2015*, pages 2111–2117. AAAI Press.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016), New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 40–48. JMLR.org.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), The Thirty-First Innovative Applications of Artificial Intelligence Conference (IAAI 2019), The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI 2019), Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7370–7377. AAAI Press.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

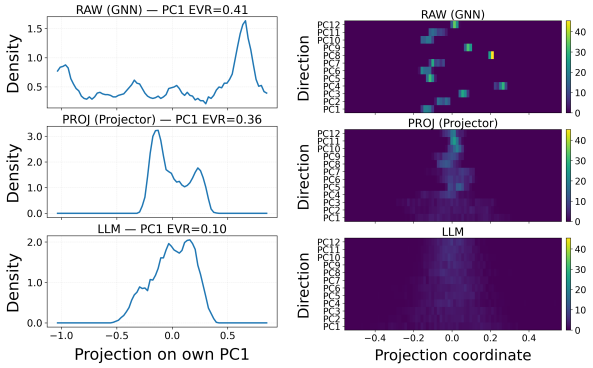
Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI 2024), Jeju, South Korea, August 3-9, 2024*, pages 5734–5742. ijcai.org.

A Additional Representation Analysis

To examine whether the geometric effects observed in Section 3 extend beyond the Cora dataset, we provide additional representation analyses on Citeseer and PubMed. Following the same analysis protocol adopted in Section 3.1, we investigate the distributional geometry of node representations using label-free embedding diagnostics, including principal component projections and multi-directional projection statistics.

Figure 10 and Figure 11 summarize the analysis results on Citeseer and PubMed, respectively. Each figure consists of two subfigures: (a) projection density distributions along the principal directions, and (b) multi-directional projection statistics. Across both datasets, RAW GNN representations exhibit pronounced anisotropy, with variance highly concentrated along a small number of dominant directions. This behavior is reflected by elevated PC1 explained variance ratios (EVR) and sparse activation patterns in the principal component space. In contrast, aligned representations



(a) 1D density along representation PCA directions.

(b) Density under LLM-induced PCA directions.

Figure 10: **Representation space analysis on the Citeseer dataset.** (a) PC1 density of node embeddings with explained variance ratio (EVR). (b) Density patterns under LLM-induced PCA coordinates.

(PROJ) display a more balanced variance distribution and substantially reduced directional dominance, resulting in a geometry that more closely resembles that of LLM embeddings.

The projection density profiles and multi-directional projection heatmaps further reveal this consistent trend. While RAW representations tend to show fragmented and uneven projection patterns, PROJ representations exhibit smoother and more coherent structures, with significantly reduced extreme directional concentrations. These observations closely mirror the results obtained on the Cora dataset in Section 3.1, indicating that the geometric reorganization induced by interaction with the LLM embedding space is not specific to a single dataset, but persists across multiple benchmark graph datasets.

Taken together, the results presented in this appendix further verify that interaction with the LLM embedding space systematically reshapes the global geometry of graph representations in a consistent and task-agnostic manner. By suppressing spurious dominant directions, the aligned representations effectively internalize the distributional priors encoded in LLM embeddings.

B Analysis Setup and Evaluation Protocols

This appendix details the experimental setup and evaluation protocols used in the task-level analyses presented in Section 3, including diagnostic linear probing, decision margin analysis, and robustness evaluation under feature perturbations.

Diagnostic PROJ Setup. In Section 3, PROJ refers to diagnostic representations obtained by mapping GNN embeddings from a pretrained and frozen GNN encoder into the LLM embedding space using a linear projector. The projector is trained for 200 epochs using Adam with a learning rate of 10^{-3}, weight decay of 5×10^{-4}, and a batch size of 128, with a cosine-regression objective. After training, the projector is kept fixed throughout all analyses.

Frozen Representations and Linear Probes. All task-level analyses are conducted on frozen node representations. Specifically, the pretrained GNN encoder and the alignment projector are kept fixed, and only lightweight linear classifiers are trained on top of the resulting embeddings. Due to dimensional differences between RAW GNN and PROJ representations, separate linear probes are trained for each representation space. All probes share the same architecture and training configuration, differing only in input size.

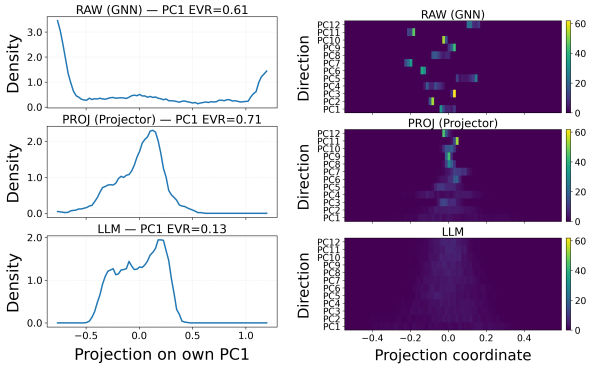
Each linear probe consists of a single fully connected layer mapping node embeddings to class logits. Probes are trained using cross-entropy loss, with the GNN encoder and projector fully frozen throughout. Unless otherwise stated, all probes are trained for 100 epochs using the Adam optimizer with a learning rate of 10^{-3}. No additional regularization or architectural modifications are applied.

Training and Evaluation Splits. Linear probes are trained on the training split and selected based on validation performance, while all reported task-level analyses, including decision margin and robustness evaluations, are conducted on the test split. This protocol ensures that all diagnostic measurements reflect generalization behavior rather than training dynamics.

Decision Margin Definition. Decision margin is defined at the logit level for multi-class classification. For a node i with true label y_i and predicted logits $\mathbf{z}_i \in \mathbb{R}^C$, the margin is computed as:

$$m_i = z_{i,y_i} - \max_{k \neq y_i} z_{i,k}, \tag{9}$$

where z_{i,y_i} denotes the logit corresponding to the ground-truth class. A larger margin indicates a more confident and stable decision, while negative margins correspond to incorrect predictions made with high confidence. All margin statistics in Section 3.2 are computed on the test set.



(a) 1D density along representation PCA directions.

(b) Density under LLM-induced PCA directions.

Figure 11: **Representation space analysis on the PubMed dataset.** (a) PC1 density of node embeddings with explained variance ratio (EVR). (b) Density patterns under LLM-induced PCA coordinates.

Robustness under Feature Perturbations. To assess robustness, we evaluate trained linear probes under additive noise applied directly to the input node features. Specifically, Gaussian noise is added to node features before they are passed through the frozen GNN encoder and projector. For a given noise level σ, perturbations are sampled as $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$, scaled relative to the empirical feature variance.

For each noise level, performance is evaluated on the test split using classification accuracy and macro-F1 score. All robustness results are averaged over 10 independent noise realizations to reduce variance. Importantly, all robustness analyses are performed post hoc and do not affect model training or parameter optimization.

Analysis Independence from Training. All analyses reported in Section 3 are conducted after model training and do not influence the optimization of either the GNN encoder or the alignment module. This design ensures that the observed effects reflect intrinsic properties of the learned representations rather than artifacts of task bias.

C Hyperparameter and Training Details

This appendix summarizes the key training configurations used in the experiments. We report the essential hyperparameters that affect optimization behavior, together with the implementation details most relevant to reproducibility.

GNN Backbone. Node representations are extracted using pretrained GNN encoders. For each dataset, the GNN backbone is trained in advance on the corresponding node classification task using the same dataset, and the resulting checkpoint is reused in all subsequent experiments. During all alignment and downstream stages, the GNN backbone remains frozen.

Compression Modules. The proposed compression framework consists of a proxy module and a purifier operating in the LLM embedding space. In our main experiments, the proxy is implemented as a linear projection from the GNN embedding space to the text-embedding dimension d_t, serving to inject coarse-grained inductive bias from the global geometry of the LLM embedding space. The purifier is implemented as a two-layer MLP with ReLU activation, hidden dimension 512, and dropout 0.1. Taking the concatenated representation $[\mathbf{z}_i; \hat{\mathbf{p}}_i]$ as input, it performs fine-grained instance-level adjustment of the projected representations through residual refinement, where the residual term is scaled element-wise by a learnable vector before being added to $\hat{\mathbf{p}}_i$. Unless otherwise stated, no additional normalization is used inside the proxy or purifier.

Regularization Weights. Unless otherwise stated, the weight of the KL-based distribution regularization is fixed to 10^{-3} throughout.

Two-stage Training Schedule. All experiments follow a two-stage training procedure. In Stage 1, the GNN backbone is frozen and the alignment modules are optimized using LLM embeddings as supervision via mean squared error (MSE) loss. Stage 1 is trained for 200 epochs with a learning rate of 10^{-4}.

In Stage 2, all representation modules are frozen and a lightweight classification head is trained using standard cross-entropy loss. Stage 2 is trained for 300 epochs with a learning rate of 5×10^{-4}. Unless otherwise noted, the classification head takes the aligned representations as input.

Randomness and Reproducibility. All experiments are conducted with fixed random seeds. Unless otherwise stated, results reported in the main text are averaged over multiple runs with different random seeds for each configuration.

D Distribution-Level Regularization

D.1 KL-Based Distribution Regularization

To preserve global structural characteristics of the LLM embedding space during projection-based

Table 6: Comparison of different distribution-level regularization methods (Macro-F1 (%)).

Method	Cora	Citeseer	PubMed	WikiCS	arXiv
KL divergence	**94.22**	**80.99**	**91.16**	**85.56**	**56.84**
L2-MM	94.01	80.41	91.08	85.30	55.42
MMD	93.76	80.54	91.01	85.33	55.36

compression, we introduce a distribution-level regularization term based on Kullback-Leibler (KL) divergence. Rather than enforcing instance-level correspondence, this regularizer encourages the aligned graph representations to preserve the overall distributional statistics of LLM embeddings.

Let $\{\mathbf{p}_i\}$ denote the aligned representations produced by the proxy-purifier module, and let $\{\mathbf{t}_i\}$ denote the corresponding LLM embeddings. For a given alignment pool $\mathcal{S}$, we approximate both sets of representations with diagonal Gaussian distributions by estimating their empirical first- and second-order moments:

$$\boldsymbol{\mu}_g = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\mathbf{p}_i, \quad \boldsymbol{\sigma}_g^2 = \mathrm{Var}_{i\in\mathcal{S}}(\mathbf{p}_i), \qquad (10)$$

and analogously $\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t^2$ for the LLM embeddings.

Under this approximation, the KL divergence from the graph-induced distribution to the LLM distribution is computed as:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{2}\sum_d \left(\log\frac{\sigma_{t,d}^2}{\sigma_{g,d}^2} + \frac{\sigma_{g,d}^2 + (\mu_{g,d}-\mu_{t,d})^2}{\sigma_{t,d}^2} - 1\right) \qquad (11)$$

where d indexes feature dimensions and a small constant is added to variances for numerical safety.

The KL regularization term is applied only during Stage 1 alignment and is combined with the representation-level alignment loss using a fixed weighting factor. During Stage 2 downstream training, the alignment module is frozen and no distribution-level regularization is applied. This design ensures that the regularizer shapes the global geometry of the learned representations without interfering with task optimization.

D.2 Comparison with Alternative Distribution Matching

We further examine whether alternative distribution matching strategies can serve as effective substitutes for KL-based regularization. In addition to KL divergence, we consider two commonly used alternatives: (i) L2 moment matching (L2-MM), which penalizes discrepancies between empirical means and variances, and (ii) maximum mean discrepancy (MMD), which measures distributional differences via kernel-based distance metrics.

All alternatives are evaluated under identical experimental settings. Specifically, we fix the same GNN backbone, proxy-purifier module, training protocol, and optimization configuration, and only replace the distribution-level regularization term. This controlled comparison isolates the effect of the regularizer choice itself.

Table 6 reports the results of these replacement experiments. While both L2 moment matching and MMD occasionally yield performance improvements, their effects are less consistent across datasets and tasks. In contrast, KL-based regularization leads to more stable and reliable gains, particularly in terms of macro-F1 score and robustness-related metrics.

These results suggest that KL divergence provides a more effective inductive bias for compressing and preserving global distributional structure. Compared to simpler moment matching, KL explicitly accounts for both variance scaling and mean shifts in a principled probabilistic form, while avoiding sensitivity to kernel choices inherent to MMD. Based on these observations, we adopt KL divergence as the default distribution-level regularizer throughout this work.

Table 7: Node classification results (Accuracy (%), Macro-F1 (%), and test-time latency (ms)) on two additional TAG datasets. Latency is measured on the test split under the same hardware and batch setting.

Method	Instagram			Reddit		
	Acc ↑	F1 ↑	Time ↓	Acc ↑	F1 ↑	Time ↓
GCN	62.20	48.13	**2.97**	64.92	65.81	**5.06**
GAT	59.74	39.15	4.08	65.49	65.84	7.12
GraphSAGE	60.70	50.55	19.76	65.82	61.55	40.46
GIN	64.32	44.29	21.70	66.90	66.34	42.04
DIFFormer	60.35	44.02	156.07	59.88	63.59	400.03
CoBFormer	59.58	42.23	150.37	60.77	61.64	313.41
PATTON	51.24	49.34	26237.76	49.27	46.64	79677.35
ENGINE	**68.22**	54.69	1121.40	64.68	64.66	2719.73
LOGIN	67.56	50.23	11.56	62.81	62.74	15.91
TAPE	64.78	43.94	15.59	62.11	61.97	34.35
OFA	61.27	56.13	12721.26	65.25	65.23	27393.25
LinguGKD	63.94	47.03	1846.52	64.58	61.08	5127.84
Ours	67.70	**57.34**	3.31	**68.33**	**68.06**	5.26

Table 8: Sensitivity to proxy depth (Macro-F1 (%), proxy hidden dimension fixed to 2048).

Proxy Layers	Cora	Citeseer	PubMed	WikiCS	arXiv
1 (default)	**94.22**	**80.99**	**91.16**	**85.56**	**56.84**
2	94.04	80.07	91.08	85.21	56.60
3	94.02	80.54	91.11	85.55	54.97

Table 9: Training-time memory consumption (MB) and per-epoch training time (s), on five TAG datasets, measured under the same hardware setting.

Method	Cora		Citeseer		PubMed		WikiCS		arXiv	
	Memory	Time	Memory	Time	Memory	Time	Memory	Time	Memory	Time
GCN	412	3.8	420	6.2	622	14.7	1062	10.6	2714	123
GAT	432	4.38	440	6.66	808	15.5	1716	11.2	5290	124
GraphSAGE	482	3.84	600	6.56	706	14.94	1376	9.99	1770	118
GIN	488	3.9	616	6.06	674	14.83	1374	10.26	1770	120.6
DIFFormer	406	12.68	444	15.6	720	35.2	572	33	2862	277
CoBFormer	438	10.94	550	13.23	912	24.36	786	26.98	18030	221.31
Ours	1875	8.8	1987	8.6	5385	22.6	3961	25.3	38068	146.1

E Additional Results on More Diverse TAG Benchmarks

To complement the five benchmark datasets reported in the main text, we further evaluate the proposed framework on two additional TAG benchmarks, namely Instagram and Reddit. These additional results extend the evaluation to broader TAG scenarios and provide a more comprehensive assessment of generalization.

Table 7 reports Accuracy, Macro-F1, and test-time latency on these two datasets. Overall, the proposed method remains highly competitive on both benchmarks. On Instagram, although ENGINE achieves the highest accuracy, our method obtains the best Macro-F1 while maintaining inference latency close to standard GNNs and substantially lower than LLM-dependent methods. On Reddit, our method achieves the best accuracy and Macro-F1, further supporting the effectiveness of training-time LLM knowledge compression beyond the benchmarks used in the main text.

F Additional Hyperparameter Sensitivity

The proxy module is designed as a lightweight projection from GNN representations into the LLM embedding space. To examine whether its effectiveness depends on architectural capacity, we vary the depth of the proxy while fixing its hidden dimensionality to 2048, keeping all other components and training settings identical.

As shown in Table 8, performance remains largely stable across different proxy depths on all datasets. This suggests that the proxy primarily serves to expose graph representations to the global geometry of the LLM embedding space, rather than to perform complex nonlinear transformations. These results support the use of a shallow proxy in practice and validate the lightweight design adopted in the main experiments.

G Training Overhead Discussion

Although the proposed framework introduces additional alignment components during training, its overall computational overhead remains moderate and comparable to standard GNN baselines. In this section, we analyze the training-time memory consumption and per-epoch training cost of our method, and compare them with representative GNN architectures.

Our approach leverages pretrained LLM embeddings only as fixed supervision signals in the alignment stage. During training, a lightweight projection layer together with a shallow MLP-based purifier is optimized, while no language model inference or parameter updates are involved. Consequently, the additional training overhead mainly arises from storing the fixed LLM embeddings and optimizing small feed-forward modules, rather than from increased model depth or complex iterative procedures.

Table 9 reports the peak GPU memory usage and per-epoch training time on five benchmark datasets. Compared with standard GNN baselines, our method exhibits a higher memory footprint, which can be primarily attributed to the storage of LLM embeddings used for alignment. Nevertheless, the memory consumption remains within a manageable range and scales consistently with dataset size, preserving the same order of magnitude as GNN-only training.

In terms of training efficiency, the proposed framework achieves per-epoch runtimes that are largely comparable to those of conventional GNN models across all datasets. This is because the introduced alignment modules are lightweight, and their computational cost is negligible relative to the message passing and aggregation operations in the backbone GNN. As a result, the additional objectives do not substantially increase the overall training time.

Overall, these results demonstrate that the proposed knowledge compression framework introduces only moderate training overhead. By confining the use of LLM information to a lightweight alignment mechanism, our method maintains training efficiency close to standard GNN baselines while enabling effective incorporation of LLM-induced inductive bias.