Can Large Language Models Act as Ensembler for Multi-GNNs?

Hanqi Duan[1], Yao Cheng[1], Jianxiang Yu[1], Yao Liu[1,*], Xiang Li[1,*]

[1]East China Normal University,

Correspondence: liuyao@cc.ecnu.edu.cn (Yao Liu[*]) and xiangli@dase.ecnu.edu.cn (Xiang Li[*])

Abstract

Graph Neural Networks (GNNs) have emerged as powerful models for learning from graph-structured data. However, GNNs lack the inherent semantic understanding capability of rich textual node attributes, limiting their effectiveness in applications. On the other hand, we empirically observe that for existing GNN models, no one can consistently outperforms others across diverse datasets. In this paper, we study whether LLMs can act as an ensembler for multi-GNNs and propose the LensGNN model. The model first aligns multiple GNNs, mapping the representations of different GNNs into the same space. Then, through LoRA fine-tuning, it aligns the space between the GNN and the LLM, injecting graph tokens and textual information into LLMs. This allows LensGNN to ensemble multiple GNNs and take advantage of the strengths of LLM, leading to a deeper understanding of both textual semantic information and graph structural information. The experimental results show that LensGNN outperforms existing models. This research advances text-attributed graph ensemble learning by providing a robust and superior solution for integrating semantic and structural information. We provide our code and data here: https://github.com/AquariusAQ/LensGNN.

1 INTRODUCTION

Graphs are structured data that captures the interrelations between entities in the real world. To learn from graphs, graph neural networks (GNNs) have been proposed, where graph convolution is introduced (Kipf and Welling, 2016) and message passing is the main mechanism for neighborhood aggregation (Hamilton et al., 2017a; Veličković et al., 2017). GNNs have demonstrated significant success across various applications such as social network analysis (Hamilton et al., 2017a), recommendation systems (Wang et al., 2019), and molecular property prediction (Gilmer et al., 2017).

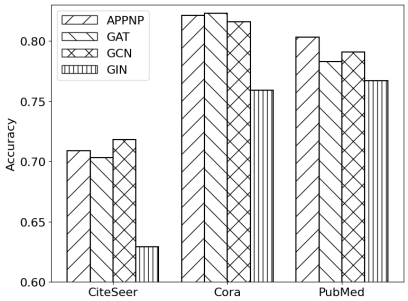

Figure 1: Comparison between GNN models on node classification across different datasets.

Despite the success, two major challenges remain unresolved. First, GNNs, although powerful in capturing graph structures, often lack the inherent semantic understanding capability required to process the rich textual attributes of nodes (Zhao et al., 2019). This can lead to the loss of valuable semantic information during the learning process, limiting the effectiveness of GNNs in applications where node features contain meaningful texts, such as citation networks and social media platforms. Second, while there have been a series of GNNs proposed, no one has been shown to consistently lead others across datasets of various domains. For example, we compare the node classification performance of four representative GNNs, namely, APPNP (Gasteiger et al., 2018), GAT (Veličković et al., 2017), GCN (Kipf and Welling, 2016) and GIN (Xu et al., 2019) on three benchmark datasets (Yan et al., 2023): *Cora*, *Citeseer* and *PubMed*. For fairness, we set the equal number of hidden representation layers and also the same dimensionality size. The results given in Figure 1 demonstrate that the winner varies across the datasets. The challenge of selecting the optimal GNN for a given dataset remains unresolved, as different GNN architectures exhibit varying strengths. This also restricts the wide applicability of GNNs.

To remedy the incapability of GNNs in semantic

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 28875–28894
November 4-9, 2025

understanding, existing works (Qin et al., 2024; Li et al., 2024; Wang et al., 2024; Ye et al., 2024; Fang et al., 2024) have resorted to large language models (LLMs), which have been shown to be prominent in text understanding and generation. The inherent advantage of GNNs in utilizing graph structure, fused with the strength of LLMs in semantic understanding, mutually enhances both models and leads to superior performance in downstream tasks. However, how to select the optimal GNN in different scenarios remains a gap. Since the effectiveness of GNNs varies across datasets, there naturally arises a question: *Can we develop an ensembled model for multi-GNNs?* The ensembled model is expected to integrate the strengths of multiple GNNs and can consistently perform well across datasets. To further leverage the power of LLMs in text understanding, we thus upgrade the question: *Can LLMs act as ensembler for multi-GNNs?*

In this paper, we study whether **L**LMs can act as an **ens**embler for multi-**GNNs** and propose the LensGNN model. Given multiple GNNs, the model ensembles GNNs with LLMs, avoiding the tedious efforts in cherry-picking GNN across datasets. By dynamically integrating the outputs of different GNNs, LensGNN can not only preserve the semantic richness of node attributes, but also optimize the usage of diverse GNNs, which enhances the model's generalizability across a wide range of tasks and datasets. Specifically, since multi-GNNs could generate node embeddings in different low-dimensional spaces, we first align multi-GNNs by alternatively feeding their generated representations into a shared classifier to train them sequentially. After that, we freeze the parameters of GNNs to generate node embeddings and integrate them into the embedding layer of the used LLM, treating them as embeddings corresponding to graph tokens that encapsulate graph structural information. Subsequently, we concatenate graph tokens from multi-GNNs, text tokens from node attributes, and hand-crafted instructions to create a prompt and fine-tune the LLM using LoRA. In this way, we can not only align GNNs and LLMs, but implicitly leverage the power of LLM to ensemble multi-GNNs. Finally, the main contributions of our paper are given as follows:

- We propose a novel method LensGNN to ensemble multi-GNNs with LLMs.
- We introduce an effective graph prompt tuning paradigm with both the multi-GNN alignment and the GNN-LLM alignment.
- We conduct extensive experiments to show the superiority of LensGNN.

2 RELATED WORK

GNNs (Kipf and Welling, 2016; Xu et al., 2019; Veličković et al., 2017; Hamilton et al., 2017b) have been widely used in learning from graph-structured data. Despite their advantages, GNNs struggle to process textual attributes effectively, as traditional GNNs lack the semantic understanding capability that is necessary to handle text-attributed graphs (TAGs). In addition, to tailor a GNN model for a given dataset, existing methods mainly rely on neural architecture search (NAS) (Zoph and Le, 2017). However, it is very computationally expensive, which necessitates new exploration.

Recently, LLMs have revolutionized tasks involving semantic understanding and language generation. Meanwhile, the ensemble of LLMs and GNNs has garnered significant attention, leading to innovative methodologies (Jin et al., 2023). One primary category of approaches takes LLMs as predictors in graph-based tasks, like GraphGPT (Tang et al., 2024), LLaGA (Chen et al., 2024) GraphLLM (Chai et al., 2023) and ENGINE (Zhu et al., 2024). Another notable category is to employ LLMs as encoders, as seen in the works of OFA (Liu et al., 2024), TextGNN (Zhu et al., 2021) and AdsGNN (Li et al., 2021), which focus on optimization strategies for encoding graph data. Additionally, the alignment of LLMs and GNNs has been explored through prediction alignment methods, such as GOFA (Kong et al., 2024), LTRN (Zhang et al., 2021) and GLEM (Zhao et al., 2022). These diverse approaches highlight the evolving landscape of LLM and GNN ensemble. However, there still lack studies exploring multi-GNNs ensembling with LLMs, which is particularly useful to avoid cherry-picking GNNs for a given dataset.

3 PRELIMINARIES

Text-attributed graph. A text-attributed graph is a graph where nodes and edges have textual attributes. In this work, we only consider the case where only nodes have textual attributes. Formally, a text-attributed graph can be defined as $G = (V, E, X^{(v)})$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ represents the set of nodes, and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ represents the set of edges, with $|V| = N$ indicating the total number of nodes in the graph.

$X^{(v)} = \{x_1^{(v)}, x_2^{(v)}, \ldots, x_{|V|}^{(v)}\}$ denote the attributes on nodes, respectively, which are strings. It can be represented as $G = (V, E, \{x_n\}_{n \in V})$. Additionally, we define A as the adjacency matrix of graph G, where the matrix size is $N \times N$. In an unweighted graph, $A(i, j) = 1$ indicates that there is an edge between nodes v_i and v_j, while $A(i, j) = 0$ indicates that there is no edge between them.

Graph Neural Networks. Traditional GNNs are a class of deep learning models designed for handling graph-structured data. The basic architecture of a GNN includes a node representation layer and a message-passing layer. In the node representation layer, each node v is assigned a feature vector x_v. In the message-passing layer, the representation vector $h_v^{(t)}$ of node v is updated after the t-th iteration using the following formula:

$$h_v^{(t)} = \text{UPDATE}\left(\text{AGG}\left(\{h_u^{(t-1)} : u \in \mathcal{N}(v)\}\right), h_v^{(t-1)}\right), \tag{1}$$

where $\mathcal{N}(v)$ denotes the set of neighboring nodes of v. The AGG function is responsible for aggregating the representations of the neighboring nodes, and the UPDATE function combines the aggregated information with the previous state $h_v^{(t-1)}$ of node v to update its current state. Through this iterative process, GNNs are able to learn increasingly rich node representations, capturing both local and global structural information in the graph. The specific implementations of the AGG and UPDATE functions can vary depending on the particular GNN model.

4 METHODOLOGY

This section introduces the main steps in LensGNN, which includes aligning multi-GNNs and ensembling multi-GNNs with LLM. The overall model procedure is given in Fig. 2.

4.1 Aligning multi-GNNs

Given multiple GNNs, they could generate node representations in different low-dimensional spaces. Therefore, before ensembling GNNs, we need to first align them.

Node feature initialization. To utilize the rich semantic information of textual features, we use pre-trained language models to initialize them. Specifically, given node v_i with feature vector x_i, we use `Sentence-BERT` (Reimers, 2019) to get its initial embedding vector $\tilde{x}_i$ by:

$$\tilde{x}_i = \text{Sentence-BERT}\left(x_i\right). \tag{2}$$

This process allows texts of any dataset and length to be converted into the same feature space.

GNN representations. Next, the model takes the text representations obtained from Eq. 2 for each node and feeds them into multiple GNNs. For the k-th GNN, we denote the node embedding matrix $H^{[k]}$ generated by the final layer as:

$$H^{[k]} = \text{GNN}^{[k]}(A, \tilde{X} \mid \Theta_{G_k}), \tag{3}$$

where $\tilde{X}$ is the initial node embedding matrix from Eq. 2 and Θ_{G_k} is the learnable parameters of the k-th GNN. For each node v_i, let $h_i^{[k]}$ be its embedding vector, which is the i-th row in $H^{[k]}$.

Reshape node representations. The output of each GNN is then connected to a linear layer, which transforms node representations into the dimensionality of hidden embeddings in LLMs. This step paves the way for the subsequent alignment of GNN and LLM. Details will be given in the next section. For each node v_i in the k-th GNN, its reshaped embedding vector $\tilde{h}_i^{[k]}$ is denoted as

$$\tilde{h}_i^{[k]} = \text{Linear}^{[k]}\left(h_i^{[k]} \mid \Theta_{L_k}\right). \tag{4}$$

Note that the shape of $\tilde{h}_i^{[k]}$ is a one-dimensional vector of length $t \times e$, where t is a hyperparameter indicating the number of graph tokens each node representation is mapped to. The dimensionality e comes from the hidden embedding layer of the LLM used, where each token is mapped to an embedding of shape $(1, e)$.

Multi-GNN alignment. To align node representations from multiple GNNs, we next feed them into a shared MLP, which serves as the classifier. During the training time, we alternatively input representations from different GNNs into the classifier and train them sequentially. This training approach allows node representations from multi-GNNs to be aligned, which facilitates the ensemble of multi-GNNs. For better alignment, we freeze this MLP starting from a certain training epoch. After training, the parameters of GNNs and the linear layer in Eq. 4 will be frozen, which are then used in aligning GNNs and LLMs. The overall procedure for GNN alignment is summarized in the top of Fig. 2.

In summary, during the multi-GNN alignment step, we train each GNN to ensure that their outputs reside in the same vector space. By incorporating the node feature initialization step that utilizes language pre-trained models, our model gains the ability to extract semantic and structural information from the nodes in the text graph. Building

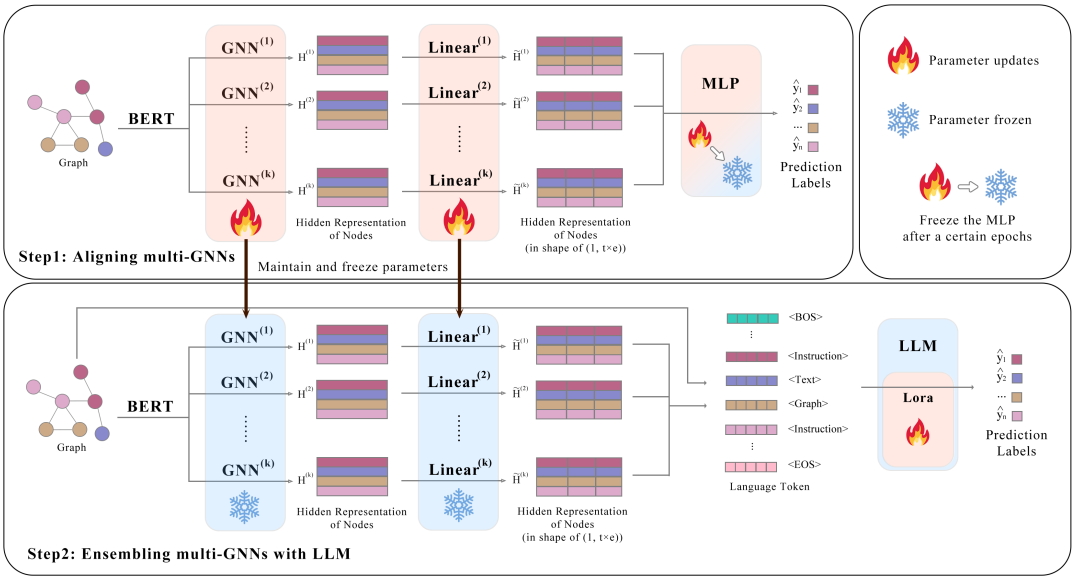

Figure 2: The overall process of LensGNN.

on the full potential of GNNs, we map the dimensions of the GNN representations to the dimensions required by the LLM embeddings, thereby maximizing the ensemble of extensive graph structural information into the LLM in the subsequent steps.

4.2 Ensembling multi-GNNs with LLM

After GNNs are trained, we can get node representations capturing rich graph structural information. Although LLMs have shown competitive capability of text semantic understanding, they are ineffective in understanding graph structure (Guo et al., 2023). Therefore, in the second stage, we empower LLMs to comprehend graph representations. Further, with specially designed prompts, we enable these models to implicitly perform ensembling.

Align GNNs and LLMs. The key step in enabling LLMs to understand GNN tokens lies in aligning GNN representations, which encapsulate rich structural information, with the semantic space required by LLMs. This process necessitates fine-tuning LLMs. However, due to the extensive number of parameters in these models, fine-tuning requires significant computational resources, which is challenging for practical applications. As a result, existing approaches often freeze the parameters of LLMs and train an additional linear layer to act as a projector (Tang et al., 2024), mapping GNN representations into the semantic space. While the method makes training feasible and yields some effectiveness, the expressive capacity of the linear layer is limited, leading to the difficulty in fully leveraging the semantic understanding capabilities of LLMs.

To address the problem, recent advancements in LoRA (Low-Rank Adaptation) (Hu et al., 2021) training have been proposed to directly fine-tune LLMs for alignment. LoRA is an efficient fine-tuning technique for large pre-trained models, and introduces low-rank matrices to simulate the effects of full parameter tuning, significantly reducing the number of training parameters and computational resources required. This allows for effective customization of LLMs even with limited resources. Thus, we adopt LoRA training to fine-tune the LLMs, enabling them to comprehend GNN representations. Specifically, given an instruction prompt, we reserve several *dummy tokens* in the prompt text at the appropriate positions where the GNN representations need to be inserted. For example, in the prompt '(GNN TYPE: GCN, REPRESENTATIONS: $<$ GRAPH_TOKEN$_1$ $>$,$<$ GRAPH_TOKEN$_2$ $>$,...,$<$ GRAPH_TOKEN$_t$ $>$), WHICH CATEGORY DOES IT BELONG TO?', $<$ GRAPH_TOKEN$_i$ $>$ denotes the i-th dummy token. Then, for each token in the prompt text, LLM can generate an embedding vector of size e . These vectors can form an embedding for the whole prompt text in the format of $[\ldots, E'_1, E'_2, \ldots, E'_t, \ldots]$, where E'_i is the embedding vector of $<$ GRAPH_TOKEN$>_i$. After that, we overwrite the embeddings of dummy tokens by GNN representations. From Eq. 4, we can generate a GNN representation vector of length

$t \times e$, which can be further segmented into t one-dimensional vectors of length e, denoted as $[E_1^G, E_2^G, \ldots, E_t^G]$. These vectors serve as the embeddings of t graph tokens. After substitution, the raw embedding vector of the prompt text becomes $[\ldots, E_1^G, E_2^G, \ldots, E_t^G, \ldots]$. We next feed it into LLM and use LoRA for fine-tuning. In this way, we can automatically leverage the language understanding capability of LLM, and implicitly align structure and semantics.

Prompt design and Multi-GNN ensembling. After aligning graph structure and semantics, LLM can understand graph representations. We next design prompts and take LLM as the ensembler to integrate multi-GNNs. Our goal is to leverage the strengths of both multi-GNNs and LLM to output more accurate predictions.

The designed prompts include three major components: *task instruction*, *node text* and *graph token*. Specifically, task instruction consists of task specifications and also instructions that guide LLMs to ensemble multi-GNNs. Node text is used to inject textual attributes of target nodes and their neighbors. Due to the input length limitation, for each target node, we only consider up to 20 adjacent nodes. When the number of adjacent nodes for the target node exceeds 20 or when the total token count of the prompt surpasses the predefined limit, a random sampling of the adjacent nodes is performed. Further, graph tokens are derived from multi-GNNs, which contain rich graph structural information. After prompts are deigned, they are fed into LLMs to predict labels. Formally, we have:

$$o = LLM(x_{\text{instruction}}, x_{\text{node-text}}, \{x_{\text{GNN}_i}\}_{i=1}^{k}),$$

where o is the predicted label, $x_{\text{instruction}}$ denotes the instruction text, $x_{\text{node-text}}$ is node textual descriptions and x_{GNN_i} denotes the graph representation (graph token) generated by the i-th GNN. Details on the prompt template used in our experiments are given in Appendix D. Therefore, the two steps: aligning GNNs and LLMs, and ensembing multi-GNNs can be trained simultaneously. In our experiments, we merge them into one step.

5 EXPERIMENT

5.1 Experimental settings

Datasets. We use eight benchmark datasets, comprising five node classification datasets: Cora, PubMed, ogbn-arXiv, Citeseer and Wiki-CS, and three molecular graph classification datasets (Wu et al., 2018): BACE, BBBP and ClinTox. Details on these datasets are provided in the Appendix A.

Baselines. We compare our method with 23 baselines: MLP, GNN models, ensemble models, LM-based models and LLM-based models. Specifically, GNN models include GCN, GAT, GIN, GraphSAGE (Hamilton et al., 2017a) and Graphormer (Ying et al., 2021). Ensemble models include Bagging (Breiman, 1996), Stacking (Wolpert, 1992) and AdaBoost (Freund and Schapire, 1997). LM-based models consist of BERT (Devlin, 2018), SentenceBERT (Reimers, 2019), DeBERTa (He et al., 2020), SciBERT (Beltagy et al., 2019) and MedBERT (Feihong Yang, 2021). LLM-based models include DGTL (Qin et al., 2024), SNS-GPT4 (Li et al., 2024), GAugLLM (Fang et al., 2024) and Baichuan2-13B (Yang et al., 2023), which all employ 13B LLMs. We also compare LensGNN with other 7B-LLM-based models: GraphGPT (Tang et al., 2024), LLaGA (Chen et al., 2024), InstructGLM (Ye et al., 2024), OFA (Liu et al., 2024) and GOFA (Kong et al., 2024). Due to the space limitation, details on these baselines are given in Appendix B.

Setup. We use Baichuan2 and InternLM2.5 (Cai et al., 2024) as the backbone LLM for LensGNN. In the experiments, we ensemble three widely adopted GNN models: GCN, GAT and GIN, each with two layers. For GNN models, we use node representations obtained from the pre-trained SentenceBERT (Reimers, 2019) as input. The additional hyperparameter analysis experiment is detailed in Appendix G. The analysis of input length is presented in Appendix I. For more details on experiment setup, refer to Appendix C. Detailed hyperparameter settings are provided in Appendix J. Additionally, in Appendix F, we include supplementary experiments utilizing LLama2-7B as the backbone.

5.2 Node classification results

Since the large number of baselines report their results on different datasets and settings, we show the classification results in three tables, respectively. Table 1 compares LensGNN with MLP, GNN models and small language models. Tables 2 summarizes the results against LLM-based models. Table 3 evaluates the performance of LensGNN in the 20-shot setting. Table 4 provides a comparison between LensGNN and methods that ensemble GCN, GAT and GIN. Additionally, we conduct one-shot experiments, which are discussed in Appendix H. From these tables, we observe:

Model type	Model	Cora	PubMed	ogbn-arXiv	Citeseer	Wiki-CS
MLP	MLP	66.42	82.40	61.47	71.13	68.41
GNN	GCN	85.97	83.78	70.81	77.74	77.15
	GAT	86.71	83.59	70.05	78.21	78.94
	GIN	85.60	82.03	67.03	75.07	68.97
	GraphSAGE	84.87	87.01	70.35	73.61	79.56
	Graphormer	80.41	88.75	72.81	71.28	72.07
Language Model	BERT	80.15	93.91	72.78	73.17	78.33
	SentenceBERT	78.82	92.49	71.42	72.79	77.92
	DeBERTa	77.79	93.45	72.90	73.13	75.11
	SciBERT	83.21	95.26	73.11	77.74	76.83
	MedBERT	77.31	93.94	72.09	74.29	74.04
LLM Ensembler	LensGNN-[GCN+GAT]	88.56	94.11	75.78	78.05	18.41
	LensGNN-[GCN+GIN]	**91.88**	94.47	75.89	76.64	**83.47**
	LensGNN-[GAT+GIN]	88.19	95.43	74.31	78.99	82.98
	LensGNN-ALL	90.40	**95.68**	**75.91**	**79.31**	81.78

Table 1: Comparison on classification accuracy (%) with MLP, GNN models and LM-based methods. LensGNN utilizes Baichuan2-13B-chat as the backbone LLM. We highlight the best score on each dataset in bold and underline the runner-up's. Here, LensGNN-ALL refers to LensGNN-[GCN+GAT+GIN].

(1) LensGNN consistently outperforms all the GNN models and LM-based methods across all the datasets. For GNN models, while they initialize node features with SentenceBERT to capture text semantics, LLMs have much stronger capability in text understanding and generation, which explains the advantage of LensGNN. For LM-based models, in addition to the weak expressiveness of small language models, they ignore the rich structural information of graphs, degrading their performance.

(2) LensGNN achieves better performance than LLM-based models, even in the 20-shot setting. In particular, we see that Baichuan2-13B performs very poorly. We speculate this is because LLMs that have not been fine-tuned are not suitable for node classification tasks on graph data. Different from other LLM-based models, LensGNN ensembles the strengths of multiple GNNs and employs two-phases of alignment, which explains its superior performance.

(3) LensGNN leads standard ensemble methods across all datasets. This is because when ensembling multiple GNNs, standard ensemble methods can only leverage graph structural information. However, LLMs as ensembler have strong semantic understanding capability that further improves the model performance.

5.3 Graph classification results

We further evaluate the effectiveness of LensGNN on the graph classification task. We use classification accuracy (ACC) and AUC scores as the evaluation metrics. The results are given in Table 5. From

Model	Cora	PubMed	ogbn-arXiv
GraphGPT-MIX-7B	-	74.16	64.76
LLaGA-HO-7B(GENERAL)	-	94.45	75.01
InstructGLM-Llama-7B	87.08	93.84	75.70
DGTL	81.10	87.10	-
SNS-GPT4	82.50	93.80	74.40
GAugLLM	-	83.68	74.15
Baichuan2-13B	13.65	36.04	4.79
LensGNN*-InternLM2.5-7B	90.03	95.13	74.24
LensGNN*-Baichuan2-7B	89.85	95.08	75.88
LensGNN*-Baichuan2-13B	**91.88**	**95.68**	**75.91**

Table 2: Comparison on classification accuracy (%) with LLM-based models. * denotes the best variant of our model and - means that the results are missing from their original papers. The best variant model is given in Appendix E.

Model	Cora	PubMed
GCN	73.5	68.0
GAT	72.8	68.1
GIN	75.7	69.3
OFA	75.34	77.89
GOFA	77.08	87.33
LensGNN*-Baichuan2-7B	78.09	**89.19**
LensGNN*-InternLM2.5-7B	**79.93**	87.75

Table 3: Classification accuracy (%) in 20-shot setting.

Model	Cora	PubMed	Citeseer	WikiCS
Bagging	87.16	82.67	77.35	76.97
Stacking	86.25	84.85	78.08	74.12
AdaBoost	84.91	83.36	76.09	75.41
LensGNN-ALL	**91.88**	**95.68**	**79.31**	**83.47**

Table 4: Comparison with ensemble methods (%).

Model	BACE		BBBP		ClinTox	
	ACC	AUC	ACC	AUC	ACC	AUC
GCN	74.01	84.92	84.14	79.00	90.93	87.23
GAT	74.67	80.85	76.34	66.12	91.61	86.37
GIN	68.09	**85.66**	83.65	79.02	86.91	85.80
LensGNN-[GCN+GAT]	73.02	72.67	86.09	81.29	98.99	94.00
LensGNN-[GCN+GIN]	75.32	74.59	88.04	83.27	**99.32**	96.00
LensGNN-[GAT+GIN]	76.31	75.99	86.09	81.64	98.99	95.81
LensGNN-ALL	**80.59**	80.33	**88.53**	**85.37**	98.99	**97.63**

Table 5: Graph classification results.

the table, LensGNN outperforms GNN baselines in most cases, which shows that combining GNNs and LLMs is inspiring. Further, LensGNN-ALL generally performs better than its variants. This verifies that leveraging strengths of multiple GNNs is useful. Surprisingly, we find that on BACE, LensGNN and its variants achieve smaller AUC scores than GNN models. We step into the datasets and observe that BACE is a label-balanced dataset while BBBP and ClinTox are highly imbalanced. Note that, for balanced dataset, ACC is more convincing than AUC score, while AUC score is a better indicator than ACC on imbalanced dataset. Therefore, the ACC advantage of LensGNN on BACE and the AUC leads on other two datasets have evidently demonstrated the superiority of our method.

5.4 Ablation study

We next systematically conduct an ablation study on Cora, PubMed, and ogbn-arXiv datasets to evaluate the importance of major model components. Specifically, we explore various configurations of "GNN Encoder" (using different GNNs), "Alignment" (whether multiple GNNs are aligned when training GNNs), "With Text" (whether node text is included in prompt) and "With Neighbor" (whether neighborhood information is included or not). The results are presented in Table 6.

Performance of different GNN encoders. Ensembling multiple GNNs noticeably outperforms using a single GNN. For example, on the ogbn-arxiv dataset, the best result from a single GNN is 74.35%, while LensGNN achieves 75.91%. This illustrates that by integrating the strengths of various GNNs, LensGNN can effectively enhance the overall performance.

On Cora and Wiki-CS, the small training sets introduce more noise during training. Since our model relies on an LLM as backbone—and LLMs typically require large-scale data for effective learning—we attribute the underperformance of LensGNN-ALL compared to LensGNN-[GCN+GIN] primarily to the limited training data and elevated noise levels. These factors hinder the LLM from adequately learning to align all three GNNs (GCN, GAT, and GIN) simultaneously.

Impact of Alignment. On the PubMed dataset, the results for unaligned GNN and aligned GNN are 93.96% and 95.68%, respectively, while on the ogbn-arXiv dataset, the results are 73.73% and 75.91%, respectively. This shows that multi-GNN alignment can help LLMs better understand graph tokens. Although the performance of aligned GNN is slightly lower than unaligned GNN on the Cora dataset, this may be due to the insufficient number of training samples.

Importance of Node Text. It can be observed that node text plays a key role in the supervised fine-tuning of LLMs. Through these node texts, the model can achieve a deeper understanding of semantics, resulting in outstanding accuracy.

Role of Neighbors. The textual information of neighbors provides support for LLM's understanding of the semantic information of nodes, thereby improving classification performance.

5.5 Performance with different backbones

To study the impact of backbone models on LensGNN, we use different backbones, including three small LMs: BERT-base (Devlin, 2018), T5-base (Raffel et al., 2023), the encoder-only variant of T5-base, and two other LLMs: Falcon-7B (Almazrouei et al., 2023) and InternLM2.5-7B-chat (Cai et al., 2024). All the results are presented in Table 7. From the table, we see that

(1) Small LMs BERT-base, T5-base, and T5-base (Encoder only) perform well when applied to a single GNN, but the performance drops significantly when ensembling multiple GNN models. This suggests that models with less parameters lack the capability to ensemble multiple GNNs.

(2) For LLMs, they generally perform better than LM backbones. For example, for GAT on Cora, the best result for LM backbones is 81.85%, while that for LLM backbones is 90.03%. This is because LLMs can provide richer text semantics. Further, compared with single GNN, the ensemble of multiple GNNs with LLMs leads to better performance, which verifies that unifying the strengths of multiple GNNs is beneficial for node classification.

GNN Encoder	Alignment	With Text	With Neighbor	Cora (Acc/%)	PubMed (Acc/%)	ogbn-arXiv (Acc/%)
-	-	Yes	1	87.82	93.77	74.47
GCN	-	Yes	1	89.29	94.23	74.35
GAT	-	Yes	1	90.03	94.26	73.37
GIN	-	Yes	1	88.92	94.82	73.26
GCN, GAT, GIN	Yes	No	1	19.92	40.06	34.92
GCN, GAT, GIN	No	Yes	1	**90.77**	93.96	73.73
GCN, GAT, GIN	Yes	Yes	0	84.87	94.21	72.03
GCN, GAT, GIN	Yes	Yes	1	90.40	**95.68**	**75.91**

Table 6: Ablation study results.

Model		Bert-base	T5-base (Encoder only)	T5-base	Falcon-7B	InternLM2.5-7B-chat	Baichuan2-13B
Parameters		110M	110M	220M	7B	7B	13B
Cora	With GCN	81.48	78.52	76.10	85.23	89.29	89.29
	With GAT	81.85	79.26	77.94	80.44	89.66	90.03
	With GIN	**87.50**	**88.24**	**88.24**	87.45	89.29	88.92
	Ensemble All	80.37	80.00	77.57	**89.66**	**90.03**	**90.40**
Pubmed	With GCN	**94.67**	**94.58**	94.03	94.67	94.72	94.23
	With GAT	94.37	94.53	**94.19**	94.37	**95.13**	94.26
	With GIN	94.01	94.17	94.00	94.52	94.87	94.82
	Ensemble All	94.17	93.51	93.95	**94.97**	**95.13**	**95.68**

Table 7: The classification accuracy (%) with different LLMs on Cora and PubMed.

5.6 Traditional methods ensembling LLM and GNNs

We conduct experiments employing traditional ensemble learning methods to ensemble the predictions from both the LLM and GNNs (GCN, GAT and GIN) on 4 datasets. The results are as Table 8:

Method	LLM	Cora	PubMed	Citeseer	Wiki-CS
Bagging	Baichuan2-13B	86.90	86.16	75.55	76.47
Stacking	Baichuan2-13B	85.24	85.70	70.53	67.35
LensGNN*	Baichuan2-13B	**91.88**	**95.68**	**79.31**	**83.47**

Table 8: Comparison with traditional ensemble of LLM and GNNs. (%)

As demonstrated in our experiments, LensGNN consistently outperforms all conventional ensemble methods.

In addition, the observed phenomenon where ensemble learning results underperform a single LLM (as shown in Table 6) can be attributed to the fact that traditional ensemble methods may sometimes yield worse performance than individual models. Potential reasons for this phenomenon include: 1) Strong base learners: Since the base learner (LLM) itself is already a strong model, the random sampling in Bagging may disrupt its original high performance. 2) Large performance gap among base learners: When base learners differ significantly in performance (e.g., LLM vs. GNN), weaker base learners (GNNs) may introduce noise, negatively affecting the meta-learner in Stacking.

5.7 Ensemble more GNN

We further incorporate a larger number of GNNs to evaluate how LensGNN scales and whether the performance continues to improve. The results are presented in Table 9. Overall, we see that with the increase in the number of GNNs, the model performance improves further. However, from the results on Cora and Pubmed, simply adding GNNs will not bring sustained improvement. We speculate that this is because more GNNs will introduce more graph tokens and also noise. The increased input length will bring difficulty in model learning, so does noise.

Based on the empirical findings presented in Table 9, we propose a heuristic recommendation for researchers to select 3-4 high-performing GNNs (e.g., GCN, GAT, GIN) when constructing ensembles for new datasets. Notably, the inclusion of additional GNN models incurs minimal computational overhead, allowing for flexible expansion until validation accuracy approaches saturation—the point at which further performance improvements become negligible. This strategy balances model diversity and efficiency while maximizing predictive accuracy.

5.8 Model efficiency study

Training Efficiency. The training of LensGNN includes two main phases: aligning multi-GNNs and ensembling multi-GNNs with LLM. In the first phase, the time complexity depends on the GNN encoders. Since the used GNNs have a lin-

Variants of LensGNN	Number of GNNs	Cora	PubMed	Citeseer	Wiki-CS
LensGNN-GCN	1	0.9059	0.9490	0.7836	0.8294
LensGNN-[GCN+GAT]	2	0.9003	0.9503	0.7868	0.8306
LensGNN-[GCN+GAT+GIN]	3	0.9114	0.9508	0.7884	0.8334
LensGNN-[GCN+GAT+GIN+GraphSAGE]	4	0.9095	0.9513	0.7852	0.8332
LensGNN-[GCN+GAT+GIN+APPNP]	4	0.9040	0.9500	0.7915	0.8366
LensGNN-[GCN+GAT+GIN+GraphSAGE+APPNP]	5	0.9022	0.9472	0.7931	0.8417

Table 9: Classification accuracy when integrating more GNNs. The backbone LLM is Internlm2.5-7B-chat.

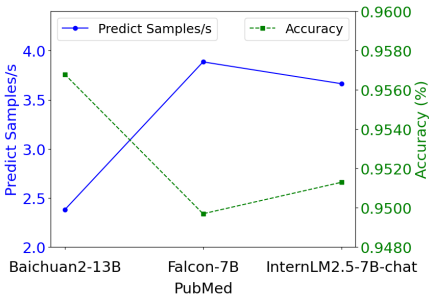

Figure 3: Inference speed and accuracy comparison.

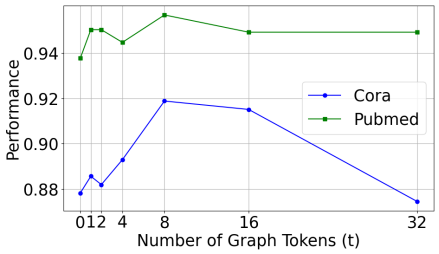

Figure 4: Hyperparameter sensitivity analysis.

ear time complexity w.r.t. the number of nodes in the graph, the training process is efficient. In the second phase, fine-tuning LLMs is the main computational cost. We employ LoRA to reduce the number of parameters to be fine-tuned. For example, for Baichuan2-13B, the trainable parameters constitute only 0.0470% of the total parameters. This allows us to fine-tune the 13B parameter LLM in a single 80G Nvidia A100 environment. Compared to GraphGPT (Tang et al., 2024), the increase of our training computational cost primarily stems from employing more than one GNN. However, in contrast to LLM, the number of parameters in GNN is considerably smaller, which increases marginal training cost.

Inference Efficiency. During inference, all the parameters are frozen. The major cost of model inference comes from LLMs. In our experiments, with a maximum input length of 2047 tokens, the inference speed and accuracy of our method based on Baichuan2-13B, Falcon-7B, and InternLM2.5-7B-chat on the Pubmed dataset is illustrated in Figure 3. Overall, the inference speed ranges from 2 to 4 samples per second. Although a larger LLM results in reduced inference speed, it concurrently yields superior accuracy.

5.9 Hyperparameter sensitivity analysis

We end this section with model sensitivity analysis on the number of graph tokens t, which represents how many tokens each node is mapped into before fed into LLM. We conduct experiments on Cora and Pubmed with varying t values, and the results are shown in Figure 4. From the figure, we see that, as the number of graph tokens increases, the model's performance first rises and then decreases. For both datasets, the best results are achieved when $t = 8$. When t is small, graph token representations cannot well capture the semantic information. On the other hand, when t is large, the representation of graph tokens could be noise-corrupted, which adversely affects the model's understanding on graph structure. Therefore, an appropriate number of tokens helps LLMs effectively capture structural information, enhancing semantic comprehension and leading to better performance.

6 CONCLUSION

We studied the problem of how to ensemble multi-GNNs in this paper and proposed LensGNN, which ensembles multi-GNNs with LLMs. LensGNN adopts two phases of alignment: the multi-GNN alignment aims to map node representations from different GNNs into the same low-dimensional space, while the GNN-LLM alignment injects graph representation as graph tokens into LLM. After that, we performed supervised fine-tuning with LoRA to enhance the LLM's capability in understanding graph topology. Finally, we conducted extensive experiments to show the effectiveness of our ensembling model LensGNN. In particular, the results show that LLMs can serve as effective multi-GNN ensembler, while small LMs cannot.

Limitations

Our method could suffer from several limitations:

First, due to limited computational resources, we do not conduct experiments on LLMs with parameters exceeding 13B. This implies that the upper limit of our method's capabilities has not been fully investigated. For larger-scale LLMs that are utilized in production environments, they could further improve our model's performance.

Second, our method relies on manually crafted prompts to ensemble multi-GNNs. Future studies on prompt design could be a promising direction.

Third, we only evaluated the performance based on node classification and graph classification tasks. Evaluation on other tasks, such as graph dataset generation and graph task interpretability analysis, are valuable research questions.

Ethical Statement

Our work falls under basic research and is not tied to specific applications; therefore, whether our method will be abused and cause negative social impacts depends on the specific applications in which others use our method. In addition, our work does not involve any stakeholders benefiting or being disadvantaged, nor does it involve vulnerable groups. The datasets we used are all commonly used public datasets, posing no privacy risks, and aligned with their intention for scientific research. For the datasets, pre-trained models, and training frameworks utilized, we adhered to their respective open-source licenses: community license for Baichuan-2 (Yang et al., 2023), Apache License 2.0 for InternLM2.5 (Cai et al., 2024) , Falcon-7B (Almazrouei et al., 2023) and Llama-Factory (Zheng et al., 2024)

Acknowledgments

This work is supported by National Natural Science Foundation of China No. 62202172 and No. 42375146.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In *EMNLP*. Association for Computational Linguistics.

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24:123–140.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, Xiaoyi Dong, Haodong Duan, Qi Fan, Zhaoye Fei, Yang Gao, Jiaye Ge, Chenya Gu, Yuzhe Gu, Tao Gui, Aijia Guo, Qipeng Guo, Conghui He, Yingfan Hu, Ting Huang, Tao Jiang, Penglong Jiao, Zhenjiang Jin, Zhikai Lei, Jiaxing Li, Jingwen Li, Linyang Li, Shuaibin Li, Wei Li, Yining Li, Hongwei Liu, Jiangning Liu, Jiawei Hong, Kaiwen Liu, Kuikun Liu, Xiaoran Liu, Chengqi Lv, Haijun Lv, Kai Lv, Li Ma, Runyuan Ma, Zerun Ma, Wenchang Ning, Linke Ouyang, Jiantao Qiu, Yuan Qu, Fukai Shang, Yunfan Shao, Demin Song, Zifan Song, Zhihao Sui, Peng Sun, Yu Sun, Huanze Tang, Bin Wang, Guoteng Wang, Jiaqi Wang, Jiayu Wang, Rui Wang, Yudong Wang, Ziyi Wang, Xingjian Wei, Qizhen Weng, Fan Wu, Yingtong Xiong, Chao Xu, Ruiliang Xu, Hang Yan, Yirong Yan, Xiaogui Yang, Haochen Ye, Huaiyuan Ying, Jia Yu, Jing Yu, Yuhang Zang, Chuyu Zhang, Li Zhang, Pan Zhang, Peng Zhang, Ruijie Zhang, Shuo Zhang, Songyang Zhang, Wenjian Zhang, Wenwei Zhang, Xingcheng Zhang, Xinyue Zhang, Hui Zhao, Qian Zhao, Xiaomeng Zhao, Fengzhe Zhou, Zaida Zhou, Jingming Zhuo, Yicheng Zou, Xipeng Qiu, Yu Qiao, and Dahua Lin. 2024. Internlm2 technical report. *Preprint*, arXiv:2403.17297.

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. *arXiv preprint arXiv:2310.05845*.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024. Llaga: Large language and graph assistant. *Preprint*, arXiv:2402.08170.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. 2024. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. *Preprint*, arXiv:2406.11945.

Jiao Li Feihong Yang, Xuwen Wang. 2021. An exploration and study on the application of bert models in chinese clinical natural language processing. `https://github.com/trueto/medbert`.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

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*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017a. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017b. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Bryan Hooi, et al. 2023. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2(4):8.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

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

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. 2024. Gofa: A generative one-for-all model for joint graph language modeling. *Preprint*, arXiv:2407.09709.

Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 223–232.

Rui Li, Jiwei Li, Jiawei Han, and Guoyin Wang. 2024. Similarity-based neighbor selection for graph llms. *Preprint*, arXiv:2402.03720.

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. *Preprint*, arXiv:2310.00149.

Zequn Liu, Wei Zhang, Yingce Xia, Lijun Wu, Shufang Xie, Tao Qin, Ming Zhang, and Tie-Yan Liu. 2023. Molxpt: Wrapping molecules with text for generative pre-training. *Preprint*, arXiv:2305.10688.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163.

Péter Mernyei and Cătălina Cangea. 2022. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *Preprint*, arXiv:2007.02901.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2024. Disentangled representation learning with large language models for text-attributed graphs. *Preprint*, arXiv:2310.18152.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

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

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. *Preprint*, arXiv:2310.13023.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Atul Kumar Verma, Rahul Saxena, Mahipal Jadeja, Vikrant Bhateja, and Jerry Chun-Wei Lin. 2023. Betgat: An efficient centrality-based graph attention model for semi-supervised node classification. *Applied Sciences*, 13(2):847.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. *Preprint*, arXiv:2402.08785.

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*, pages 165–174.

David H Wolpert. 1992. Stacked generalization. *Neural networks*, 5(2):241–259.

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

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? *Preprint*, arXiv:1810.00826.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan 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*, volume 36, pages 17238–17264. Curran Associates, Inc.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1955–1973.

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

Xinyang Zhang, Chenwei Zhang, Xin Luna Dong, Jingbo Shang, and Jiawei Han. 2021. Minimally-supervised structure-rich text categorization via learning on text-rich networks. In *Proceedings of the Web Conference 2021*, pages 3258–3268.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.

Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. 2019. Semantic graph convolutional networks for 3d human pose regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3425–3435.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, pages 2848–2857.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. 2024. Efficient tuning and inference for large language models on textual graphs. *Preprint*, arXiv:2401.15569.

Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. *Preprint*, arXiv:1611.01578.

A Datasets

We use eight benchmark datasets, comprising five node classification datasets: Cora, PubMed, ogbn-arXiv, Citeseer and Wiki-CS, and three molecular graph classification datasets: BACE, BBBP and ClinTox. Statistics of these datasets are summarized in Table 10.

Dataset	#Graphs	Avg.# Nodes	Avg.# Edges	# Classes
Cora	1	2,708	5,429	7
PubMed	1	19,717	44,338	3
ogbn-arXiv	1	169,343	1,166,243	40
Citeseer	1	3,312	4,732	6
Wiki-CS	1	11,701	216,123	10
BACE	1,513	34.1	73.7	1
BBBP	2,039	23.9	51.6	1
ClinTox	1,491	26.1	55.5	2

Table 10: Datasets statistics.

Cora (McCallum et al., 2000) is a standard citation network dataset consisting of 2,708 research papers in the field of machine learning. These papers are categorized into seven classes. The title and abstract of each paper are utilized as the textual attributes of the nodes. The dataset includes 5,429 citation links, constructing a graph structure among the papers. The Cora dataset is commonly used for node classification and evaluating the performance of graph neural network models.

Citeseer (Giles et al., 1998) is a citation network dataset comprising 3,312 research papers primarily from the fields of computer science and information technology. These papers are categorized into six classes based on their research areas. The title and abstract of each paper are used as the textual attributes of the nodes, providing a semantic representation of the content. The dataset includes 4,732 citation links, forming a graph structure that represents the citation relationships between the papers. The Citeseer dataset is widely used for node classification tasks, and it serves as a benchmark for evaluating the performance of graph neural network models.

PubMed (Sen et al., 2008) is another citation network dataset containing 19,717 research papers from the biomedical field. These papers are categorized into three classes. The title and abstract of each paper are utilized as the textual attributes of the nodes. The dataset includes 44,338 citation links, constructing a graph structure among the papers. The PubMed dataset is also used for node classification tasks, particularly in testing large-scale graph neural network models.

ogbn-arXiv (Hu et al., 2020) is part of the Open Graph Benchmark (OGB) and contains 169,343 academic papers scraped from arXiv. These papers are time-ordered by their submission dates and categorized into 40 subject areas. In this dataset, the nodes represent arXiv papers, and the edges represent citation relationships. The dataset includes 1,166,243 citation links, constructing a citation network among the papers. The ogbn-arXiv dataset is commonly used for node classification and studying time-sensitive graph neural network models.

Wiki-CS (Mernyei and Cangea, 2022) is derived from Wikipedia and consists of 11,701 web pages related to computer science topics. These pages are categorized into 10 classes, each representing a different area of computer science such as artificial intelligence, computer architecture, and software engineering. The text content of each page is used as the textual attributes of the nodes, capturing the thematic essence of the pages. The dataset includes 216,123 hyperlinks, which construct a graph structure connecting the web pages. The Wiki-CS dataset is often utilized for node classification tasks and for evaluating graph-based machine learning models.

BACE (Wu et al., 2018) is a collection of inhibitors of human beta-secretase 1 (BACE-1) and provides both quantitative IC50 values and qualitative binary labels indicating the binding results. The BACE dataset comprises 1,513 molecular graphs for the molecular property prediction.

BBBP (Wu et al., 2018) is used for predicting blood-brain barrier permeability (BBBP), which is crucial for determining whether a molecule can cross the blood-brain barrier. The BBBP dataset contains 2,039 molecular graphs for the binary graph classification task.

ClinTox (Wu et al., 2018) is a collection of drugs approved by the FDA and those that have failed clinical trials due to toxicity reasons. The dataset encompasses two classification tasks for 1,491 drug molecules: (1) clinical trial toxicity and (2) FDA approval status.

For Cora, PubMed, Citeseer, BACE, BBBP and ClinTox, we randomly split nodes into 60%, 20%, and 20% for training, validation and testing, and measure the performance of all models on the test set.

For ogbn-arXiv we split the dataset as suggested in (Hu et al., 2020).

For Wiki-CS, we split the dataset as suggested in (Mernyei and Cangea, 2022).

B Baselines

To evaluate the effectiveness of LensGNN, we compare it with the SOTA methods. Details of these baselines are summarized as follows.

(1) MLP: Multilayer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of nodes (neurons) connected by weights and is primarily used for supervised learning tasks, such as classification.

(2) Graph Neural Networks: GCN (Kipf and Welling, 2016) is a fundamental method based on convolutional neural networks which operates directly on graph-structured data. **GAT** (Veličković et al., 2017) computes the hidden representations of each node in the graph by first learning the importance of its neighbors and then aggregating information from them. **GIN** (Xu et al., 2019) develop a simple architecture that is provably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test. **GraphSAGE** (Hamilton et al., 2017a) present a general inductive framework that leverages node feature information (e.g., text attributes) to efficiently generate node embeddings for previously unseen data. **Graphormer** (Ying et al., 2021) is built upon the standard Transformer (Vaswani, 2017) architecture, and could attain excellent results on a broad range of graph representation learning tasks. Graphormer propose several simple yet effective structural encoding methods to help Graphormer better model graph-structured data.

(3) Ensemble Models: Bagging (Breiman, 1996) is an ensemble method that improves model stability and accuracy by training multiple base learners on bootstrap samples and aggregating their predictions through voting or averaging. **Stacking** (Wolpert, 1992) is an ensemble technique that combines predictions from multiple base learners to train a meta-learner, enhancing overall predictive performance through stacked generalization. **AdaBoost** (Freund and Schapire, 1997) is an adaptive boosting algorithm that sequentially trains weak learners, adjusts sample weights, and combines their predictions to create a strong classifier with improved accuracy.

(4) LM Based Models: BERT (Devlin, 2018) is a groundbreaking model in natural language processing. It utilizes the transformer architecture to understand the context of words in a sentence by looking at both their left and right contexts simultaneously. This bidirectional approach enables BERT to capture nuanced meanings and relationships in text. **SentenceBERT** (Reimers, 2019) is a modification of the original BERT (Bidirectional Encoder Representations from Transformers) model designed specifically for generating sentence embeddings. It was introduced to improve the performance of BERT on tasks that require understanding the semantic meaning of entire sentences, such as semantic textual similarity, paraphrase identification, and clustering. **DeBERTa** (He et al., 2020) is a transformer-based language model that improves upon BERT and other models like RoBERTa by introducing several innovations. **SciBERT** (Beltagy et al., 2019) is a BERT-based language model pre-trained on a large multi-domain corpus of scientific publications to achieve state-of-the-art performance on various scientific NLP tasks. **MedBERT** (Feihong Yang, 2021) is a series of BERT and ALBERT models pre-trained on a large Chinese clinical corpus, which outperforms baseline models on named entity recognition and text classification tasks within the Chinese clinical NLP domain.

(5) LLM Based Models: GraphGPT (Tang et al., 2024) is a novel framework that integrates Large Language Models (LLMs) with graph structural knowledge through graph instruction tuning. This innovative approach enables LLMs to comprehend and interpret the structural components of graphs, thereby demonstrating superior generalization in both supervised and zero-shot graph learning tasks. Additionally, GraphGPT employs Baichuan-7B and Vicuna-7B as its backbone models. **LLaGA** (Chen et al., 2024) integrates the capabilities of LLMs with graph-structured data, enabling LLMs to handle complex graph tasks effectively. LLaGA achieves this by reorganizing graph nodes into structure-aware sequences and mapping them into the token embedding space through a versatile projector, demonstrating superior versatility, generalizability, and interpretability across various datasets and tasks. LLaGA employs Vicuna-7B as its backbone model. **InstructGLM** (Ye et al., 2024) use natural language to describe multiscale geometric structure of the graph and then instruction finetune an LLM to perform graph tasks, which enables Generative Graph Learning. InstructGLM employs Llama2-7B as its backbone model. **DGTL** (Qin et al., 2024) is an LLM-first-GNN-later method, which means first using LLMs to process text before training GNNs. DGTL incorporates graph structure information through tai-

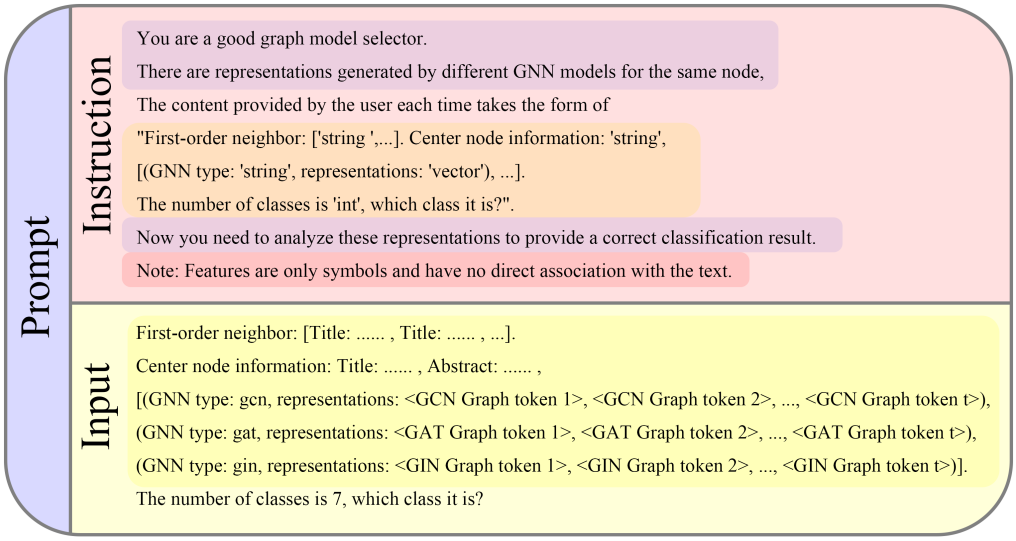

Figure 5: A prompt template for node classification task.

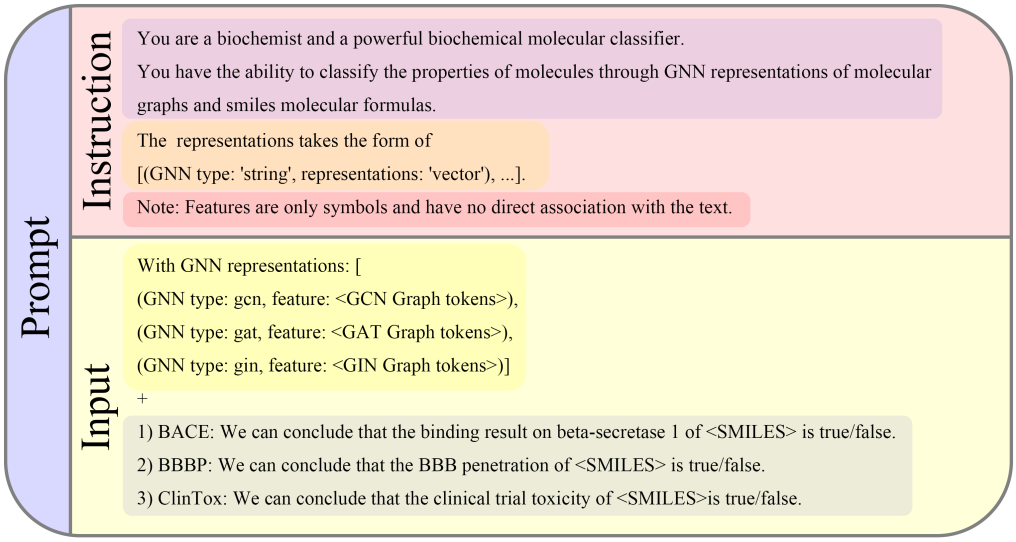

Figure 6: A prompt template for graph classification task.

lored disentangled graph neural network layers, enabling LLMs to capture the intricate relationships hidden in TAGs from multiple structural factors, which is able to enhance the reasoning and predicting capabilities of LLMs for TAGs. DGTL employs Llama2-13B as its backbone model. **SNS-GPT4** (Li et al., 2024) is a specialized version of the GPT-4 model designed for Social Network Services (SNS) applications. It introduces similarity-based neighbor selection to effectively improve the quality of selected neighbors, thereby enhancing graph representation and alleviating issues like over-squashing and heterophily. SNS-GPT4 employs GPT3.5-turbo and GPT4 as its backbone models. Notably, SNS-GPT4 demonstrates better performance than TAPE (He et al., 2023) (another LLM-first-GNN-later approach) in its reported results. **GAugLLM** (Fang et al., 2024) is a novel framework for augmenting TAGs. It leverages advanced large language models like Mistral to enhance self-supervised graph learning. GAugLLM employs Mistral 8*7b, ChatGPT-3.5 and Llama2-13b as its backbone models. **Baichuan2** (Yang et al., 2023) refers to the second iteration of the Baichuan large language model (LLM). It has been trained on an impressive 2.6 trillion high-quality tokens, ensuring a robust understanding of language nuances. Baichuan 2 comes in two main versions: 7B and 13B, both available in Base and Chat configurations, with the latter offering 4-bit quantization

for efficient deployment. **OFA** (Liu et al., 2024) is a pioneering framework that unifies various graph tasks into a single model, enabling it to address classification tasks across different domains and tasks. OFA achieves this by converting graph data into text-attributed graphs (TAGs), using language models to encode diverse text attributes into a common embedding space, and introducing a novel graph prompting paradigm for in-context learning without fine-tuning. OFA employs Llama2 as its backbone model. **GOFA** (Kong et al., 2024) is a novel graph foundation model that integrates the strengths of large language models (LLMs) and graph neural networks (GNNs) to enable joint graph and language modeling. It achieves this by interleaving GNN layers into a frozen pre-trained LLM, allowing for self-supervised pretraining on graph data and fluidity in handling various graph-related tasks. OFA employs Llama2 and Mistral as its backbone models.

C Experiment setup

We implement LensGNN by PyTorch. We primarily use Baichuan2 (Yang et al., 2023) and InternLM2.5 (Cai et al., 2024) as the backbone LLM for LensGNN. In the experiments, we ensemble three widely adopted GNN models: GCN, GAT and GIN, each with two layers. We perform grid search to fine-tune hyperparameters based on the validation set. Details on the search space is given in Table 11. We utilize LoRA+ (Hayou et al., 2024) for fine-tuning. Some fixed LoRA settings include capping each training sample at 2,047 tokens and using half-precision (FP16) for LoRA fine-tuning, with a batch size of 4 per GPU and gradient updates every step. We utilize a cosine-type learning rate scheduler and set the warmup ratio to 0.1. For the training of LLMs, we utilize Llama-Factory (Zheng et al., 2024) as a framework. For GNN models, we use node representations obtained from the pre-trained SentenceBERT (Reimers, 2019) as input. For baselines that report results on the adopted datasets, we directly report the results from their original papers. For those whose results are missing, we leave them blank. For Baichuan2, we directly use it without fine-tuning. We run all the experiments on a server with a single NVIDIA Tesla A100 GPU.

In addition, in the era of LLMs, due to the high cost, it is difficult to run experiments for baselines. For fair comparison, a widely adopted approach is to make comparison on the results for baselines reported from their original papers because their authors have well fine-tuned these models. This explains why only part of results are included in Table 2 and why we did not reimplement some baselines.

Hyperparameter	Search space
learning rate	$\{2.0 \times 10^{-5}, 5.0 \times 10^{-5}, 1.0 \times 10^{-4}\}$
dropout	$[0, 0.25]$
loraplus lr ratio	$\{16.0, 24.0, 32.0\}$
Graph token t	$\{1, 2, 4, 8, 16\}$

Table 11: Hyperparameter Search Spaces.

D Prompt design

The prompt design follows three key principles:

1. **Simultaneous input of textual strings from target node and its adjacent neighbors, graph tokens of the node, and task specification.** Existing studies (Kipf and Welling, 2016; Veličković et al., 2017; Verma et al., 2023) have demonstrated that aggregating the information from both the center node and its adjacent neighbors could contribute to the label prediction. Further, clear task instructions are necessary for accurate prediction.

2. **Differentiation between text tokens and graph tokens.** It directs LLM to correctly distinguish between textual tokens and graph tokens within different segments of the prompt, thus preventing confusion.

3. **Guidance for learning from multi-GNNs:** It leads LLM to implicitly learn how to effectively combine strengths of different GNNs.

The prompt is divided into two parts: Instruction and Input. The former specifies what the LLM should do with the input, defines the format of the input, and highlights the characteristics of the input content. The latter provides the specific text of target node or graph and their multiple GNN representations.

An example of the prompt template can be seen in Figure 5. For graph classification, we follow MolXPT (Liu et al., 2023) to design prompts for molecular graphs as shown in Figure 6.

E Best variants

The best variants in Table 2 and Table 3 are listed in Table 12 and Table 13, respectively.

Backbone LLM	Cora	PubMed	ogbn-arXiv
InternLM2.5-7B	LensGNN-ALL	LensGNN-ALL	LensGNN-[GAT+GIN]
Baichuan2-7B	LensGNN-[GCN+GAT]	LensGNN-ALL	LensGNN-ALL
Baichuan2-13B	LensGNN-[GCN+GIN]	LensGNN-ALL	LensGNN-ALL

Table 12: Best variants in Table 2.

Backbone LLM	Cora	PubMed
Baichuan2-7B	LensGNN-ALL	LensGNN-ALL
InternLM2.5-7B	LensGNN-ALL	LensGNN-[GAT+GIN]

Table 13: Best variants in Table 3.

F LLama2-7B as backbone

We have supplemented our experiments by implementing LLaMA2-7B as the backbone LLM. We conducted these experiments on a medium-scale dataset, PubMed, as a representative, with results presented in Table 14:

Model	PubMed
MLP	82.40
GCN	83.78
GAT	83.59
GIN	82.03
GraphSAGE	87.01
Graphormer	88.75
BERT	93.91
SentenceBERT	92.49
DeBERTa	93.45
GraphGPT-MIX-7B	64.76
LLaGA-HO-7B(GENERAL)	75.01
InstructGLM-Llama-7B	93.84
DGTL	87.10
SNS-GPT4	93.80
GAugLLM	74.15
Baichuan2-13B	36.04
LensGNN*-LLaMA2-7B	**94.59**

Table 14: Performance comparison (%) on PubMed dataset using LLaMA2-7B as backbone. (%)

Experimental results demonstrate that our model with LLaMA2-7B as the backbone also achieves superior accuracy compared to: (1) GNN baselines, (2) LM-based methods, and (3) other LLM-based approaches.

G Additional hyperparameter sensitivity analysis

We further conduct hyperparameter sensitivity analysis on Citeseer and Wiki-CS, whose results are given in Figure 7. From the figure, we see that the model can also achieve good results at $t = 8$ in terms of the overall performance, which is similar as the results in Figure 4.

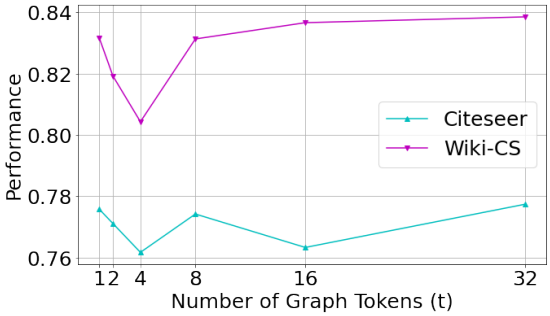

Figure 7: Hyperparameter sensitivity analysis.

Model	Cora	PubMed	Citeseer	Wiki-CS
GCN	18.90	41.55	32.42	28.73
GAT	25.43	49.93	35.06	**40.94**
GIN	20.21	40.71	27.63	32.78
LensGNN	**59.38**	**83.79**	**53.32**	30.47

Table 15: Classification accuracy in one-shot setting. (%)

H One-shot results

We further conduct experiments on one-shot setting, the results are presented in Table 15. From the table, we see that our method still achieves outstanding performance on the Cora, PubMed, and Citeseer datasets, which shows the strong generalizability. For WikiCS, due to the very scare labeled data, GNN ensembling is adversely affected.

I Input length analysis

We investigate the impact of LLM input length on model efficiency and performance across Cora, Citeseer, and Wiki-CS. Specifically, we established different limits on total number of input tokens. When exceeded, some neighbor text is randomly truncated. Within this constraint, more text introduces additional noise but also provides richer useful information, leading to better model performance—albeit at increased computational cost. The experimental results are presented in Figure 8.

Our experiments using Baichuan-7B as the backbone record sample statistics with total token length limits ranging from 0.5K to 1.75K, including average input string length, training time, and test accuracy. While Cora's shorter texts show negligible variations, both Citeseer and Wiki-CS exhibit a clear trend: longer inputs increased training time but also improved accuracy.

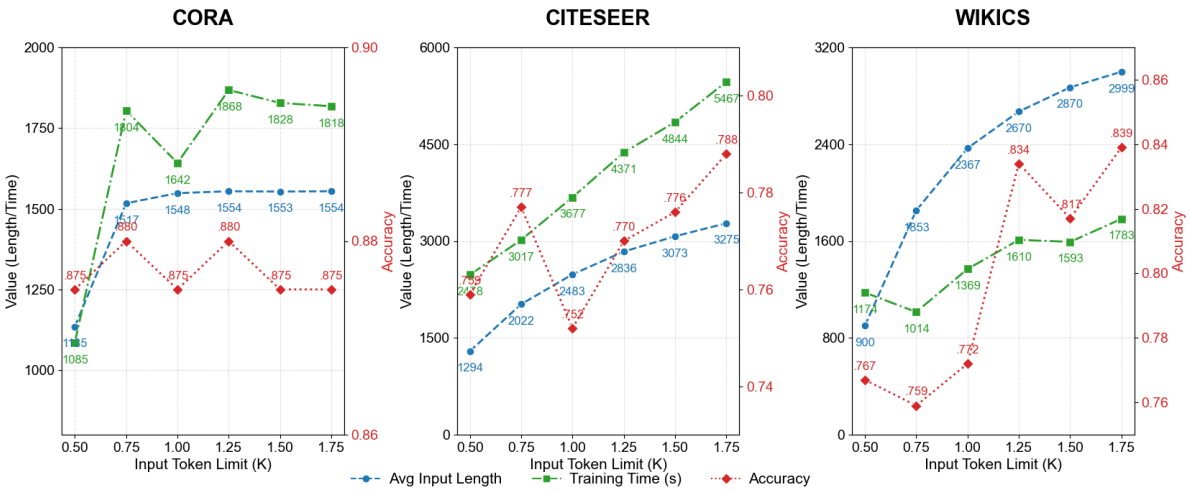

Figure 8: Input length analysis.

J Detailed hyperparameter settings

The details on the setting of hyperparameters in the first and second stages of LensGNN are shown in Tables 16 and 17, respectively. The hyperparameters for the 20-shot experiment are shown in Tables 18 and 19.

Model	Dataset	Hidden size	GNN representation size	Number of layers	Learning rate of GNN encoder	Initial learning rate of the classifier	Dropout Rate	Weight Decay
GCN	Cora	256	40960	2	0.0002	0.0002	0.25	0.001
	Pubmed	256	40960	2	0.0002	0.0002	0.25	0.001
	ogbn-arxiv	256	20480	2	0.01	0.0005	0.2	0
	Citeseer	256	40960	2	0.0001	0.0005	0.25	0.001
	Wiki-CS	256	40960	2	0.0002	0.0005	0.25	0.001
	BACE	256	40960	4	0.0001	0.0005	0.1	0.0005
	BBBP	256	40960	2	0.0001	0.0005	0.1	0.0005
	ClinTox	256	40960	2	0.0001	0.0005	0.1	0.0005
GAT	Cora	256	40960	2	0.0002	0.0002	0.25	0.001
	Pubmed	256	40960	2	0.0002	0.0002	0.25	0.001
	ogbn-arxiv	256	20480	2	0.01	0.0005	0.2	0
	Citeseer	256	40960	2	0.0001	0.0005	0.25	0.001
	Wiki-CS	256	40960	2	0.0002	0.0005	0.25	0.001
	BACE	256	40960	4	0.0001	0.0005	0.1	0.0005
	BBBP	256	40960	2	0.0001	0.0005	0.1	0.0005
	ClinTox	256	40960	2	0.0001	0.0005	0.1	0.0005
GIN	Cora	256	40960	2	0.0001	0.0001	0.25	0.001
	Pubmed	256	40960	2	0.0001	0.0001	0.25	0.001
	ogbn-arxiv	256	20480	2	0.004	0.0005	0.2	0
	Citeseer	256	40960	2	0.00005	0.0005	0.25	0.001
	Wiki-CS	256	40960	2	0.0001	0.0005	0.25	0.001
	BACE	256	40960	4	0.0001	0.0005	0.1	0.0005
	BBBP	256	40960	2	0.0001	0.0005	0.1	0.0005
	ClinTox	256	40960	2	0.0001	0.0005	0.1	0.0005

Table 16: Detailed hyperparameters of aligning multi-GNNs.

Model	Dataset	Lora dropout	Loraplus lr ratio	Training batch size	Learning rate	Early stop epoch	Warmup ratio
Baichuan2-13B	Cora	0.1	16	4	0.0001	3	0.1
	Pubmed	0.1	16	4	0.00005	2	0.1
	ogbn-arxiv	0.1	16	4	0.0001	3	0.1
	Citeseer	0.15	16	4	0.0001	5	0.1
	Wiki-CS	0.1	16	4	0.0001	5	0.1
	BACE	0.1	16	4	0.0001	4	0.1
	BBBP	0.1	16	4	0.0001	3	0.1
	ClinTox	0.1	16	4	0.0001	3	0.1
Baichuan2-7B	Cora	0.15	16	4	0.0001	3	0.1
	Pubmed	0.15	16	4	0.0001	2	0.1
	ogbn-arxiv	0.15	16	4	0.0001	3	0.1
InternLM2.5-7B	Cora	0.1	16	4	0.0001	3	0.1
	Pubmed	0.1	16	4	0.0001	2	0.1
	ogbn-arxiv	0.1	16	4	0.0001	3	0.1
Falcon-7B	Cora	0.1	16	4	0.0001	3	0.1
	Pubmed	0.1	16	4	0.0001	2	0.1

Table 17: Detailed hyperparameters of Ensembling multi-GNNs with LLM (LoRA fine-tune).

Model	Dataset	Hidden size	GNN representation size	Number of layers	Learning rate of GNN encoder	Initial learning rate of the classifier	Dropout Rate	Weight Decay
GCN	Cora	256	32768	2	0.0002	0.0005	0.25	0.001
	Pubmed	256	32768	2	0.0002	0.0005	0.25	0.001
GAT	Cora	256	32768	2	0.0002	0.0005	0.25	0.001
	Pubmed	256	32768	2	0.0002	0.0005	0.25	0.001
GIN	Cora	256	32768	2	0.0001	0.0005	0.25	0.001
	Pubmed	256	32768	2	0.0001	0.0005	0.25	0.001

Table 18: Detailed hyperparameters of aligning multi-GNNs (20-shot).

Model	Dataset	Lora dropout	Loraplus lr ratio	Training batch size	Learning rate	Early stop epoch	Warmup ratio
Baichuan2-7B	Cora	0.15	16	4	0.0001	8	0.1
	Pubmed	0.15	16	4	0.0001	9	0.1
InternLM2.5-7B	Cora	0.1	16	4	0.0001	8	0.1
	Pubmed	0.05	16	4	0.0001	9	0.1

Table 19: Detailed hyperparameters of Ensembling multi-GNNs with LLM (LoRA fine-tune) (20-shot).