LOGIN: A Large Language Model Consulted Graph Neural Network Training Framework

Yiran Qiao[†‡]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, CAS
Beijing, China
yrqiao@gmail.com

Xiang Ao[*†‡]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, CAS
Beijing, China
aoxiang@ict.ac.cn

Yang Liu[†‡]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, CAS
Beijing, China
liuyang2023@ict.ac.cn

Jiarong Xu
School of Management, Fudan University
Shanghai, China
jiarongxu@fudan.edu.cn

Xiaoqian Sun[‡]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, CAS
Beijing, China
sunxiaoqian@ict.ac.cn

Qing He[†‡]
Institute of Computing Technology, CAS
University of Chinese Academy of Sciences, CAS
Beijing, China
heqing@ict.ac.cn

Abstract

Recent prevailing works on graph machine learning typically follow a similar methodology that involves designing advanced variants of graph neural networks (GNNs) to maintain the superior performance of GNNs on different graphs. In this paper, we aim to streamline the GNN design process and leverage the advantages of Large Language Models (LLMs) to improve the performance of GNNs on downstream tasks. We formulate a new paradigm, coined "LLMs-as-Consultants", which integrates LLMs with GNNs in an interactive manner. A framework named **LOGIN** (**L**LM c**O**nsulted **G**NN tra**IN**ing) is instantiated, empowering the interactive utilization of LLMs within the GNN training process. First, we attentively craft concise prompts for spotted nodes, carrying comprehensive semantic and topological information, and serving as input to LLMs. Second, we refine GNNs by devising a complementary coping mechanism that utilizes the responses from LLMs, depending on their correctness. We empirically evaluate the effectiveness of LOGIN on node classification tasks across both homophilic and heterophilic graphs. The results illustrate that even basic GNN architectures, when employed within the proposed LLMs-as-Consultants paradigm, can achieve comparable performance to advanced GNNs with intricate designs. Our code is available at https://github.com/QiaoYRan/LOGIN.

[*]Corresponding author.
[†]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS).
[‡]Key Lab of AI Safety, Chinese Academy of Sciences. Xiang Ao is also at Institute of Intelligent Computing Technology, Suzhou, China.

WSDM '25, March 10–14, 2025, Hannover, Germany

ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703488

CCS Concepts

• **Information systems** → **Data mining**.

Keywords

Large Language Model; Graph Neural Network

ACM Reference Format:
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 Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25), March 10–14, 2025, Hannover, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3701551.3703488

1 Introduction

Graph structures, due to their extensive ability to represent individual entities and respective interactions, are widely used in fields such as recommendation [3, 38, 60], anomaly detection [29, 34, 68], drug discovery [36, 48, 54], and etc [6, 20]. Recently, thanks to the advent of graph neural networks (GNNs) [18, 28, 52] and their demonstrated powerful capabilities, graph machine learning has become a highly active field drawing significant research attention.

Among these existing works, most of them strive to design different architectures of GNNs to adapt to distinct graph types. The reason for that is the broad real-world sources of graph data induce considerable diversity across different graphs. One of the basic categorizations, for example, could be the case of homophily and heterophily [45]. Classic GNNs, due to their fundamental mechanisms of message passing and aggregation [4], have demonstrated superior performance on homophilic graphs but fail to generalize to heterophily scenarios where dissimilar nodes are connected [33, 69]. Consequently, researchers have committed to developing various structures, aiming at improving the effectiveness of GNNs on particular types of graphs [30, 33, 42]. Despite their various routes of the existing works, these methods hold a similar methodology:

Design a specialized variant of graph neural networks to tackle analysis tasks on specific types of graphs.

However, designing GNN variants for real-world applications often involves extensive trial and error, requiring both theoretical understanding and practical business insights. While tailored for specific scenarios, these variants generally don't deviate significantly from classic GNN models in terms of fundamental mechanisms [70]. This raises the question of whether classic GNNs can be optimized to address various scenarios effectively.

In this paper, our objective is to empower classic GNNs for consistently superior performance on graphs with varying characteristics. To this end, an influential off-shelf tool, the Large Lanuange Model (LLM), is adopted to help us achieve the goal. LLMs currently have not only achieved remarkable advancements in natural language processing [7, 51], but have also buoyed promising insights in other domains, such as computer vision [2, 32, 35] and recommendation systems [31, 44, 71], etc. This success can be attributed to the wealth of open-world knowledge stored in their large-scale parameters [40, 66] and the emerging capabilities in logical reasoning [7, 24].

Analogously, we aim to leverage the advantages of LLMs to enhance graph machine learning, particularly the capacities of GNNs. Some pioneering research has explored the potential of LLMs on graph data, which can be typically categorized into LLMs-as-Predictors and LLMs-as-Enhancers, as shown in Fig. 1 (a) and (b). For the first kind, taking the node classification task as an example, some researchers harness the zero-shot and few-shot reasoning ability of LLMs to directly obtain node labels [14, 17, 53, 64]. Others apply instruction tuning or prefix tuning to adapt LLMs specifically for graph tasks [8, 50, 59]. However, most of them disregard the advantages of GNNs, and tuning LLMs may require substantial computing resources as well. For the latter one, LLMs-as-Enhancers, the LLMs are commonly utilized to enhance nodes' semantic features [11, 13, 22] or to refine local topological structures [49]. This kind of route is straightforward to apply LLMs in data preprocessing as a one-time enhancer, failing to couple the LLMs and GNNs interactively.

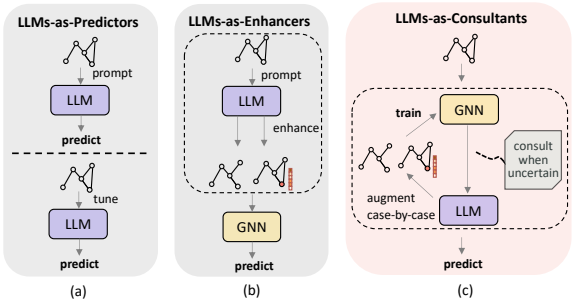

Figure 1: The paradigms for integrating LLMs with graphs.

Different from the existing works, we formulate a new paradigm that integrates LLMs with GNNs in an interactive manner, and we coin it "LLMs-as-Consultants" (c.f. Fig. 1 (c)), following which, LLMs could explicitly contribute to the training process of GNN.

Under this paradigm, we propose a framework named **LoGIN**, short for **L**LM c**O**nsulted **G**NN tra**IN**ing, empowering the interactive utilization of LLMs within the GNN training process. As an interactive approach, the crucial issues of LoGIN lie in what GNNs should deliver to LLMs and how to feed the LLMs' responses back to GNNs. First, for LLMs' inputs, we delve into prompt engineering to craft concise prompts for spotted nodes, which carry comprehensive semantic and topological information. Second, to utilize the responses from LLMs, we devise a complementary coping mechanism depending on their correctness. Specifically, compared to ground truth labels, when LLMs predict correctly, we update node features to obtain semantic enhancement. Otherwise, we impute the misclassification to the potential presence of local topological noises, hence performing structure refinement. Besides, particular criteria may serve in the selection of essential nodes to reduce complexity. In our implementation, we adopt GNN predictive uncertainty to assess the necessity of consulting LLMs.

We explore the effectiveness of LoGIN on node classification tasks across both homophilic and heterophilic graphs. By empirical studies, we illustrate that even classic GNNs, when employed within the proposed LLMs-as-Consultants paradigm, can achieve comparable performance to advanced GNNs with intricate designs.

The contributions of this paper can be summarized as follows.

- To our knowledge, we are the first to propose the LLMs-as-Consultants paradigm of graph machine learning. Different from previous works, we integrate the power of LLMs interactively into GNN training.
- Under this paradigm, we propose the **L**LM c**O**nsulted **G**NN tra**IN**ing (**LoGIN**) framework. This framework can be considered as a synthesis of previous methodologies, with a particularly tailored feedback strategy concerning the correctness of responses.
- Experiments on six node classification tasks with both homophilic and heterophilic graphs demonstrate the effectiveness and generalizability of LoGIN.

2 Related Works

2.1 GNN Variants

Early GNNs [18, 28, 52] were designed for typical graph-structured data like citation networks, which are homophilic, homogeneous, and class-balanced. Homophily means that nodes with similar attributes are more likely to connect [19, 45]. Homogeneous graphs contain the same types of entities and relations [56, 62]. Class balance refers to an even distribution of classes within a graph [26, 65]. Consequently, classic GNNs struggle with performance degradation when generalized to heterophilic [1, 5, 10, 58], heterogeneous [15, 56, 61, 62], and class-imbalanced graphs [9, 34, 41, 43, 47, 65]. However, these three characteristics are common in real-world graphs. To maintain the superiority of GNN performance in such scenarios, researchers have tailored classic GNNs into specific variants.

GNNs Designed for Heterophily. For heterophily, since similar nodes may not tend to bond, two major designs for GNN variants are neighbor extension and GNN architecture refinement. For neighbor extension, MixHop [1] and H2GCN [69] aggregate information across multi-hop ego-graphs. For GNN architecture refinement, JKNet [58], H2GCN [69], and GCNII [10] combine intermediate representations from each layer, allowing flexible adaptation of neighborhood ranges to individual nodes. ACMGCN [37] designs adaptive filters for spectral GNNs to capture high-frequency signals.

GNNs Designed for Heterogeneity. For GNN variants targeting heterogeneity, suitable aggregation mechanisms that better fuse the heterogeneous information were proposed [55]. To name some, HAN [56] develops a hierarchical attention mechanism to capture both the structural and semantic information within heterogeneous graphs. HetGNN [62] uses bi-LSTM to aggregate neighbor embeddings and learn deep interactions among heterogeneous nodes. GTN [61] proposes an aggregation function to automatically discover suitable metapaths during message passing.

GNNs Designed for Class Imbalance. For class imbalance, the goal is to design a GNN variant that effectively handles both majority and minority classes [26]. For example, GraphSMOTE [65] uses synthetic oversampling in the embedding space to enhance minority class representation. ImGAGN [43] and GraphENS [41] generate synthetic minority nodes, while PC-GNN [34] uses a label-balanced sampler for sub-graph training and a neighborhood sampler to balance the local topology.

Theoretically, these GNN variants are not fundamentally different from the classic GNNs [70]. Practically, designing them to work requires extensive manual experimentation over a long period. Our work aims to leverage LLMs to optimize GNNs, enhancing the performance of GNNs across various graphs, and ultimately matching or even surpassing these intricately designed GNN variants.

2.2 LLMs for Graphs

The emergence of LLMs has inspired many explorations of utilizing LLMs for graph-structured data, mainly in two paradigms: LLMs-as-Predictors and LLMs-as-Enhancers.

LLMs-as-Predictors. Representative methods following this paradigm, e.g. NLGraph [53], GPT4Graph [17], GraphQA [14] and GraphText [64], attempt to harness the zero-shot and few-shot ability, along with the in-context learning ability to solve graph tasks by describing the graph structural topology in natural language. However, the complex global graph structure can hardly be compressed in a token-limited prompt, thereby utilizing LLMs solely only achieves the performance far from desired. Besides directly prompting LLMs, GraphLLM [8] conducts prefix tuning on an open-source LLM by concatenating graph-specific prefixes to its attention layers. InstructGLM [59] and GraphGPT [50] adapt LLMs for graph downstream tasks through instruction tuning, employing natural language and a graph-text aligner to express graph structural information, respectively. Nevertheless, neglecting the authenticated power of existing GNNs results in only moderate performance or substantial computing resource consumption.

LLMs-as-Enhancers. Different from the former paradigm, the LLMs-as-Enhancers paradigm incorporates LLMs to enhance input graphs before GNN training. For example, TAPE [22] and GraphLLM [11] enhance GNN node features by prompting LLMs with node texts. SimTeG [13] applies LoRA [23] for parameter-efficient fine-tuning of LLMs on graph textual corpora to improve node representations. LLM-TSE [49] explicitly instructs LLMs to produce a similarity score for two texts of two nodes, which subsequently leads to edge pruning. This methodology employs LLMs only as a one-time data preprocessor before GNN training.

In contrast to these two paradigms, our LLMs-as-Consultants paradigm integrates LLMs directly into GNN training, consulting LLMs on a subset of uncertain nodes identified by GNNs, to enhance GNN performance with acceptable additional resource usage.

3 Preliminaries

3.1 Text-Attributed-Graphs (TAGs)

Text-attributed graphs (TAGs) are widely used in previous research on LLMs for graphs. A TAG can be formulated as:

$$\mathcal{G} = \left(\mathcal{V}, \mathbf{A}, \{s_n\}_{n\in\mathcal{V}}\right), \tag{1}$$

where $\mathcal{V}$ is a set of N nodes, $\mathbf{A} \in \{0,1\}^{N\times N}$ denotes the adjacency matrix, and $s_n \in \mathcal{D}^{L_n}$ is the text attached to node $n \in \mathcal{V}$, with $\mathcal{D}$ as the word dictionary, and L_n as the text length. Note that our proposed framework is not limited to traditional TAGs that literally have texts as original attributes. In fact, most entities and their relations can be modeled and processed as graphs, and their characteristics can be expressed in text form.

3.2 Graph Neural Networks (GNNs)

For GNN training, the texts associated with the nodes should be encoded to an embedded space. We represent the node embeddings as $\mathbf{X} \in \mathbb{R}^{N\times D}$, in which each row x_n denotes the corresponding node embedding, with D as its dimension number. For node classification, GNNs aggregate information from a node's neighbors, and then update the node representation with the aggregated information. The k-th layer of a GNN can be formalized as:

$$x_i^{(l)} = f^{(l)}((\mathrm{AGG}^{(l)}_{j\in\mathcal{N}(i)} x_j^{(l-1)}), x_i^{(l-1)}), \tag{2}$$

where $x_i^{(l)} \in \mathbb{R}^D$ denotes the representation of the i-th node in the l-th layer representation, with $\mathcal{N}(i)$ as its neighborhood. In addition, $\mathrm{AGG}^{(l)}$ is a function operator that is differentiable, and $f^{(l)}$ represents the certain structure of the l-th layer GNN.

4 Methodology

4.1 Problem Formulation

In this paper, we explore the integration of LLM consultation into GNN training for the node classification task. We follow the **transductive** setting of node classification: Given some labeled nodes $\mathcal{V}_{\mathcal{L}} \subset \mathcal{V}$ in graph $\mathcal{G} = (\mathcal{V}, \mathbf{A}, \{s_n\}_{n\in\mathcal{V}})$, we aim to classify the remaining unlabeled nodes $\mathcal{V}_{\mathcal{U}} = \mathcal{V}\backslash\mathcal{V}_{\mathcal{L}}$ within the same graph. Formally, the target is to learn a set of GNN parameters $\mathbf{W}$ to predict the unlabeled nodes with the guidance from LLMs:

$$f_{\mathbf{W}|\mathrm{LLM}} : (\mathbf{A}, \{\mathbf{s_n}\}_{\mathbf{n}\in\mathcal{V}}) \to \mathbf{Y}. \tag{3}$$

Note that the ground truth labels of nodes, the pseudo labels predicted by GNNs, and the pseudo labels from LLMs are denoted as $\mathbf{Y}, \hat{\mathbf{Y}}, \hat{\mathbf{Y}}^L \in \{0,1\}^{N\times C}$ respectively, with C as the number of classes.

4.2 Overview of LoGin

Figure 2 demonstrates the framework of LoGin. In this pipeline, we first identify target nodes that need consultation with LLM based on the GNN's predictive uncertainty (c.f. Section 4.3). Then prompts consisting of the associated texts and local structures of these nodes are provided to LLMs for consultation (c.f. Section 4.4). After LLM consultation, a complementary mechanism is applied to

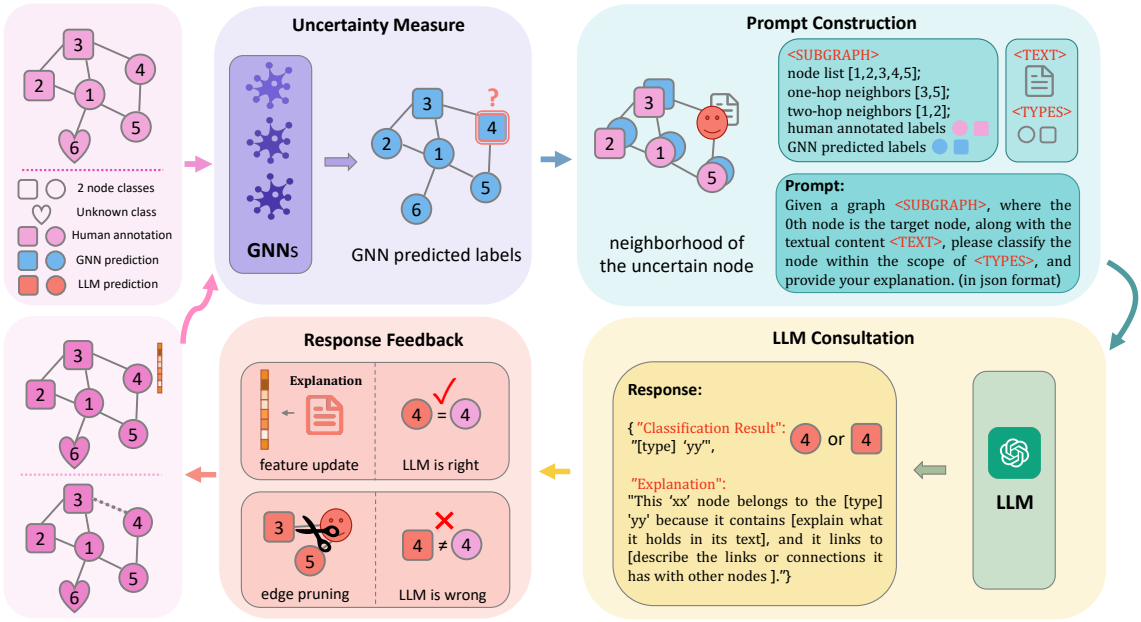

Figure 2: The pipeline of LoGin.

fully utilize the LLMs' responses, regardless of their classification correctness (c.f. Section 4.5).

4.3 Node Selection Based on Uncertainty

An intuitive node selection approach is to consult the LLM with all nodes in the graph. However, such a manner underutilizes the capacity of GNNs, which are proficient at handling simple nodes, and it could be inefficient due to the limited interaction speed with LLMs. Hence, we decide to consult LLMs only with partial difficult nodes, and adopt the variance of the predictions from several GNNs as a surrogate indicator for identifying the hard ones in our LoGin.

Recall that Bayesian estimation is typically used to rate the predictive certainty in common neural networks [27]. Under the Bayesian framework, the conventionally fixed GNN parameters $\mathbf{W}$ are considered as random variables following specific distributions. The predictive probability of the Bayesian GNN with parameters $\mathbf{W_b}$ can be defined as Eq. (4).

$$p(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) = \int_{\mathbf{W_b}} p(\hat{\mathbf{Y}} \mid \mathbf{W_b}, \mathbf{A}, \mathbf{X}) p(\mathbf{W_b} \mid \mathbf{A}, \mathbf{X}) \mathrm{d}\mathbf{W_b} \tag{4}$$

Since the true posterior $p(\mathbf{W} \mid \mathbf{A}, \mathbf{X})$ in Eq. (4) is hard to calculate in practice, variational inference uses an arbitrary distribution $q_\theta(\mathbf{W})$ to approximate the posterior. Specifically, we use MC dropout variational inference, in which dropout could serve to perform variational inference. In this method, the variational distribution $q_\theta(\mathbf{W})$ is from a Bernoulli variable $\mathbf{M}_\theta$, representing whether the neurons are on or off, as Eq. (5) shows.

$$\begin{aligned} \mathbf{M}_\theta &\sim \text{Bernoulli}(\theta), \\ q_\theta(\mathbf{W}) &= p(\mathbf{W} \mid \theta) = p(\mathbf{W}_\theta), \\ \mathbf{W}_\theta &= \mathbf{M}_\theta \odot \mathbf{W}_b. \end{aligned} \tag{5}$$

where θ is the dropout rate, and $\mathbf{M}_\theta$ represents a binary mask that controls which neurons in GNNs are off. Then, $\{\hat{\mathbf{W}}_t\}_{t=1}^T$ is a collection of T samples drawn from $\mathbf{W}_\theta$ in a Monte Carlo way. Through the minimization of the loss function defined in Eq. (6) using these weight samples, $\mathbf{W}_b$ can be acquired.

$$\mathcal{L}(\mathbf{W}_b) = -\frac{1}{T} \sum_{t=1}^{T} \mathbf{Y} \log \left(f_{\hat{\mathbf{W}}_t}(\mathbf{A}, \mathbf{X}) \right) + \frac{1-\theta}{2T} \|\mathbf{W}_b\|^2 \tag{6}$$

After $\mathbf{W}_b$ is trained, the model uncertainty score $\mathbf{U}$, which is an N-dimension vector indicating the uncertainty score of each node, is calculated as in Eq. (7).

$$U(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) = \mathrm{Var}(\hat{\mathbf{Y}} \mid \mathbf{A}, \mathbf{X}) \approx \frac{1}{T} \sum_{t=1}^{T} \left(\hat{Y}_t - \frac{1}{T} \sum_{t=1}^{T} \hat{Y}_t \right)^2 \tag{7}$$

4.4 LLM Consultation

After we identify the nodes with the most uncertainty $\mathcal{V}_{uc}$ for LLM consultation, the next step in LoGin is to figure out how to convert the semantic and topological information of these nodes into an understandable format by LLMs. We consult LLMs in a zero-shot node-by-node manner, i.e., constructing a single prompt for each uncertain node, without classification examples provided. Each node prompt is composed of three elements: instruction, input data, and output indicator. The Prompt Construction module in Fig. 2 shows a general prompt example, and we provide the detailed prompts in Appendix A.

Instruction. The instruction explains the node classification task in the context of the graph type, depicting real-world meanings of nodes and edges, with a spectrum of the categories provided.

Input Data. The input data includes information related to the target node n, namely its original text s_n, the two-hop neighborhood $\mathcal{N}_2(n)$ description, and the neighbor labels $\mathbf{Y}_{\mathcal{N}_2(\mathbf{n})}, \hat{\mathbf{Y}}_{\mathcal{N}_2(\mathbf{n})}$

from both human annotations and GNN predictions. Instead of listing edges to express connectivity [11, 17] , we opt to summarize one-hop and two-hop neighbors for the sake of semantic clarity and easier parsing. Note that we only select uncertain nodes from the train set, otherwise, it might bring a data leakage. The target node's labels $y_n, \hat{y}_n$ are also included in its corresponding prompt to provide more information for LLM predictions.

Output Indicator. The output indicator is used to control the output format of LLMs for further analysis, specifically by offering a desired response example in JSON format. Regarding the response content, we aim for the output to include the most probable classification outcome $\hat{y}_n^L$ along with its explanation e_n.

4.5 LLM Response Feedback

After consulting LLMs with all selected uncertain nodes $\mathcal{V}_{uc}$, we have a group of LLM predicted pseudo labels and corresponding explanations $\left\{n \in \mathcal{V}_{uc} : (\hat{y}_n^L, e_n)\right\}$. Next, we aim to maximize the utility of LLM responses and convert the information into signals suitable for GNN processing. The LLM responses can be divided into two types compared to ground truth labels, i.e., the correct predictions where $\hat{y}_n^L = y_n$ and the wrong predictions where $\hat{y}_n^L \neq y_n$. We not only utilize the correct predictions but also the misclassifications, and offer distinct approaches respectively. Incorporating such a complementary coping mechanism endows GNN training with full exploitation of LLM responses.

For clarity in expression, we denote the correct nodes and incorrect nodes as $\mathcal{V}_r$ and $\mathcal{V}_w$ respectively, satisfying $\mathcal{V}_r \cup \mathcal{V}_w = \mathcal{V}_{uc}$ and $\mathcal{V}_r \cap \mathcal{V}_w = \emptyset$.

When LLM is Right. For a right node $n \in \mathcal{V}_r$, we append the explanation e_n parsed from the LLM response to its original text s_n. And we attain its new node embedding $\mathbf{x_n}'$ by encoding the new attached text from e_n:

$$\left\{\mathbf{x'}_\mathbf{n}\right\}_{n \in \mathcal{V}_r} \leftarrow \text{ENC}\left(\{s_n + e_n\}_{n \in \mathcal{V}_r}\right) \tag{8}$$

When LLM is Wrong. For those wrong nodes, the LLM responses are left out since the explanations for incorrect classifications could be unhelpful. Instead, we opt to attribute the misclassification to the topological information, e.g., complex structure patterns and potential noises. Therefore, we heuristically simplify the local structure for the wrong nodes to make it easier for GNNs to learn. Specifically, around a wrong node $n \in \mathcal{V}_w$, we prune edges based on node similarity scores to denoise the local structure. As in representative graph structure learning methods, e.g. GNNGuard [63], we measure similarity d_{ni} between the features of the wrong node n and its neighbor i using cosine similarity:

$$d_{ni} = d^{cos}(\boldsymbol{x}_n,\ \boldsymbol{x}_i) = (\boldsymbol{x}_n \odot \boldsymbol{x}_i) \,/\, (\|\boldsymbol{x}_n\|_2 \, \|\boldsymbol{x}_i\|_2)\,. \tag{9}$$

Then, edges of node n are pruned if their similarity scores fall below a user-defined threshold d_{th}. By pruning edges of the misclassified nodes $\mathcal{V}_w$ as in Eq. (10), we accomplished the topological refinement leveraging the wrong answers from LLMs.

$$\mathbf{A}' \leftarrow \mathbf{1}(\mathbf{D} - d_{th} \cdot \mathbf{I}) \odot \mathbf{M}_w \odot \mathbf{A} \tag{10}$$

Here $\mathbf{D} = \left(d_{ij}\right)_{\mathbf{N}\times\mathbf{N}}$ denotes the similarity matrix, with $\mathbf{M}_w$ as a binary node mask representing $\mathcal{V}_w$, and $\mathbf{1}$ is a binary indicator.

In a nutshell, the response feedback stage is divided into two complementary scenarios. When the LLM classifies nodes correctly, we upgrade the original node embeddings with the encoded explanations. Conversely, we impute the misclassifications to the topological noise, hence pruning edges around the wrong nodes.

4.6 Train and Test

After the LLM response feedback stage, LoGIN is set to retrain GNN. When retraining, the GNN aggregates the original node features and those updated with the correct classification grounds. And for nodes misclassified by LLMs, aggregation only happens along the retained edges after structure refinement. In this way, the message passing of GNNs is integrated with LLM intelligence both semantically and topologically.

Figure 2 shows the GNN training process with one-time consultation within the LoGIN framework. Note that this process can be iteratively repeated until the maximum number of iterations is reached or the GNN performance has achieved an acceptable level.

During testing, ground truth labels are inaccessible, so we use the trained GNN without additional LLM consultation for final predictions. Retraining the GNN on LLM-augmented graphs, with enhanced node representations and refined edges, allows it to perform aggregation in a more reliable neighborhood with more accurate semantic information and thus produce improvements.

5 Evaluation

In this section, we investigate the effectiveness of our LoGIN framework on both homophilic and heterophilic graph datasets, to address the following research questions:

- **RQ1**: Does the LoGIN framework achieve performance comparable to state-of-the-art GNNs?
- **RQ2**: How does the complementary coping mechanism for LLMs' responses contribute to the LoGIN framework?
- **RQ3**: How does LoGIN operate over specific nodes based on responses from LLMs?
- **RQ4**: How does LLMs-as-Consultants paradigm perform compared with LLMs-as-Predictors and LLMs-as-Enhancers?
- **RQ5**: Can consulting more advanced LLMs, consulting with more nodes, or applying more advanced GNNs in LoGIN unlock greater potential?

5.1 Experimental Setup

Datasets. We conducted extensive experiments on six datasets: three homophilic graphs and three heterophilic graphs, to demonstrate the versatility and applicability of our proposed LoGIN framework in handling graphs with distinct characteristics. For homophilic graphs, we collected the Cora [39] and PubMed [46] datasets from widely used TAG benchmarks, while Arxiv-23 [22] was recently introduced to eliminate the data leakage unfairness when evaluating the impact of LLMs on graph learning.

For heterophilic graphs, we transformed the commonly-used web-page-link graphs: Wisconsin, Texas, and Cornell [12] into TAGs by sourcing and incorporating the raw texts[1], which were not available previously in graph libraries. Besides, we fine-tuned DeBERTa-base [21] to encode raw texts into node embeddings. The basic statistics of the datasets are displayed in Appendix B.

[1] http://www.cs.cmu.edu/~webkb/

Table 1: Performance comparison for node classification on homophilic and heterophilic graphs. The best results and the second best results among the classic GNNs with and without LoGIN and the advanced models are bold and underlined respectively.

Method			Cora	PubMed	Arxiv-23	Wisconsin	Texas	Cornell
Classic GNNs	MLP	OR	0.6438 ± 0.0331	0.8805 ± 0.0032	0.6759 ± 0.0027	0.8113 ± 0.0718	0.8105 ± 0.0730	0.7538 ± 0.0669
		FT	0.6897 ± 0.0102	0.9486 ± 0.0030	0.7789 ± 0.0023	0.8415 ± 0.0391	0.8211 ± 0.0681	0.8049 ± 0.0602
		LO	0.7063 ± 0.0201	0.9505 ± 0.0036	0.7902 ± 0.0034	**0.8528 ± 0.0588**	**0.8895 ± 0.0820**	**0.8051 ± 0.0618**
	GCN	OR	0.8630 ± 0.0219	0.8635 ± 0.0083	0.6707 ± 0.0040	0.3736 ± 0.0672	0.4579 ± 0.0711	0.4308 ± 0.0664
		FT	0.8683 ± 0.0191	0.9289 ± 0.0069	0.7624 ± 0.0051	0.4415 ± 0.1152	0.5526 ± 0.0832	0.5282 ± 0.0644
		LO	0.8694 ± 0.0177	0.9396 ± 0.0030	0.7703 ± 0.0020	0.5057 ± 0.0430	0.5789 ± 0.0588	0.5231 ± 0.0292
	GraphSAGE	OR	0.8720 ± 0.0216	0.8849 ± 0.0026	0.6864 ± 0.0011	0.6113 ± 0.0662	0.5053 ± 0.0776	0.6051 ± 0.0389
		FT	0.8592 ± 0.0363	0.9472 ± 0.0026	0.7881 ± 0.0019	0.7211 ± 0.1324	0.7579 ± 0.1123	0.7179 ± 0.1189
		LO	**0.8727 ± 0.0219**	0.9511 ± 0.0036	**0.7941 ± 0.0029**	0.7434 ± 0.0930	0.7737 ± 0.1311	0.6872 ± 0.0896
	MixHop	OR	0.8601 ± 0.0281	0.8969 ± 0.0038	0.6774 ± 0.0029	0.5736 ± 0.1183	0.5526 ± 0.1500	0.4974 ± 0.0803
		FT	0.8572 ± 0.0123	0.9493 ± 0.0030	0.7775 ± 0.0036	0.7092 ± 0.1035	0.7421 ± 0.1075	0.6718 ± 0.1397
		LO	0.8624 ± 0.0253	**0.9513 ± 0.0038**	0.7818 ± 0.0040	0.7094 ± 0.0738	0.8158 ± 0.0930	0.7179 ± 0.0314
Advanced GNNs	JK-Net		0.8579 ± 0.0001	0.8841 ± 0.0001	0.7532 ± 0.0012	0.7431 ± 0.0041	0.6649 ± 0.0046	0.6459 ± 0.0075
	H2GCN		0.8692 ± 0.0002	0.8940 ± 0.0001	0.7382 ± 0.0011	**0.8667 ± 0.0022**	**0.8486 ± 0.0044**	**0.8216 ± 0.0023**
	APPNP		0.8539 ± 0.0477	**0.9355 ± 0.0060**	0.7969 ± 0.0143	0.6830 ± 0.0470	0.7368 ± 0.0832	0.6410 ± 0.0480
	GCNII		**0.8833 ± 0.0027**	0.7925 ± 0.0043	0.7847 ± 0.0068	0.7020 ± 0.0037	0.7135 ± 0.0039	0.7405 ± 0.0060
	SGC		0.8509 ± 0.0648	0.8832 ± 0.0055	0.7740 ± 0.0160	0.5321 ± 0.0506	0.5526 ± 0.0811	0.4615 ± 0.0748
	SSP		0.8616 ± 0.0289	0.9178 ± 0.0116	**0.7976± 0.0185**	0.6302 ± 0.0850	0.7000 ± 0.0758	0.6923 ± 0.0314

Compared Methods. We choose classic GNNs as backbones equipped with LoGIN, namely GCN [28], GraphSAGE [18] and MixHop [1], to compare with the advanced state-of-the-art GNNs, including JK-Net [58], H2GCN [69], APPNP [16], GCNII [10], SGC [57] , and SSP [25], to demonstrate that the former can achieve performance on par with the latter. MLP, which predicts without adjacency information, also serves as a base learner combined with LoGIN to show the potential of our framework. We select Vicuna-v1.5-7b as our LLM consultant, an open-source LLM trained by fine-tuning Llama 2 on user-shared conversations.

For the backbones, "OR" refers to the backbone model trained on original shallow node features, "FT" uses node embeddings from the fine-tuned LM, and "LO" denotes the model trained with our LoGIN framework. Besides, in the ablation study, we refer to the feature update and structure refinement operations in the LLM response feedback stage as "F" and "S", respectively.

In addition to advanced GNNs, we also compared LoGIN as an implementation of LLMs-as-Consultants paradigm, with the other existing paradigms, i.e. LLMs-as-Predictors and LLMs-as-Enhancers, respectively.

Implementations. We adopt **accuracy** on the test set to evaluate the node prediction performance. We report the mean accuracy and standard error from five runs with varied random data splits, and all the reported results are statistically significant.

For hyper-parameters, in the uncertainty measure module, we implement Monte Carlo dropout variational inference by running models T times with different neurons off, where T represents the number of weight samples in MC dropout. T is always set to 5 in our experiments. For the ratio γ of uncertain nodes to consult, we adjust γ slightly on each dataset around the proportion of heterophilic nodes shown in Table **??**. In the response feedback stage, we tune the similarity threshold d_{th} between $[0.1, 0.2]$, according to the off-shell tool GNNGuard [63].

5.2 Performance Comparison (RQ1)

To answer RQ1, we evaluate our proposed LoGIN framework on three homophilic and three heterophilic datasets, respectively. Table 1 reports the prediction accuracy scores and standard errors, and we have the following observations.

Comparison with Vanilla Baselines. Firstly, our method consistently outperforms the vanilla baselines trained with the original node features or the LM-finetuned node embeddings in most cases. There are only two exceptions in the Cornell dataset. Since Cornell is a relatively small dataset with only 39 nodes in the test set, as indicated by the standard errors, the experimental randomness is quite high with this small dataset. Nevertheless, LoGIN still helps improve the performance of MixHop on Cornell by 4.6% with a low deviation. Apart from this exception, all listed fundamental baselines trained within our LoGIN framework exceed the vanilla ones in node classification accuracy on the other five datasets, demonstrating the effectiveness of integrating LLMs as consultants into the GNN training process.

Comparison with Advanced GNNs. Secondly, we are able to attain performance comparable to that of advanced GNNs by training fundamental models within the LoGIN framework. It is noteworthy that on PubMed and Texas, respectively known as benchmarks for homophilic and heterophilic graphs, we achieve the highest prediction accuracy among all the compared methods. This finding verifies the generalizability of our method on graphs with distinctive characteristics. For Cora, Arxiv-23, Wisconsin, and Cornell, LoGIN achieve remarkable performance on par with the SOTA GNNs such as H2GCN and SSP as well.

Comparison with the Simplest Model. Thirdly, it draws our attention to the fact that regardless of the feature type and training paradigm we apply, MLP reveals great potential in the node classification task on heterophilic graphs. We believe that the TAGs

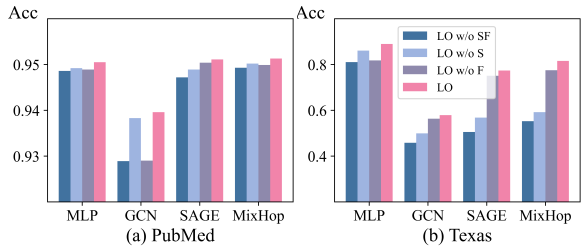

Figure 3: Ablation studies of LOGIN on PubMed and Texas.

with considerably high heterophily may be better regarded as natural language data rather than graph-structured data, for the links contained in these datasets seem only to carry noticeably limited information. Additionally, our LOGIN can clearly enhance MLP, inspiring further research on how to leverage simple existing tools to achieve superb capability with the help of the LLM-as-Consultants paradigm.

5.3 Ablation Study (RQ2)

To answer RQ2, we subdivide the response feedback stage into feature update and structure refinement. We remove each module to demonstrate their contribution to our complementary mechanism for utilizing LLM responses.

We present the results of ablation studies exclusively on the PubMed and Texas datasets, where LOGIN has demonstrated superior performance, as depicted in Fig. 3. Across both datasets, the complete LOGIN pipeline consistently achieves the highest performance, thus affirming the effectiveness and necessity of our complementary design for processing LLMs' responses.

Feature Update. Regarding the feature update component, we observe that it enhances prediction accuracy more effectively for a homophilic graph. Specifically, in the case of Cora, we notice from Fig. 3 (a) that LOGIN without feature update results in a notable decrease in performance on certain occasions compared to the whole pipeline.

Structure Refinement. Regarding the structure refinement component, LOGIN without edge pruning exhibits a significant performance decrease compared to the models with the complete coping mechanism, as shown in Fig. 3 (b). This observation aligns with our intuition, as in heterophilic graphs, the principal challenge for conventional GNNs in achieving generalization stems from the distinctiveness of their structural characteristics.

By removing each component separately, we verify that regardless of the graph type, in this case, the extent of heterophily, our design of the fault-tolerant complementary analysis strategy for LLMs' responses is sound and necessary.

5.4 Case Study (RQ3)

To answer RQ3, we select two individual nodes, respectively from Cora and Wisconsin, as concrete examples to illustrate how LOGIN operates on them. These two nodes are initially recognized as uncertain nodes and misclassified by a pre-trained GNN. Through interaction with an LLM, the operation of feature enhancement or structure refinement is correspondingly conducted, thereby in turn helping the GNN make the right prediction.

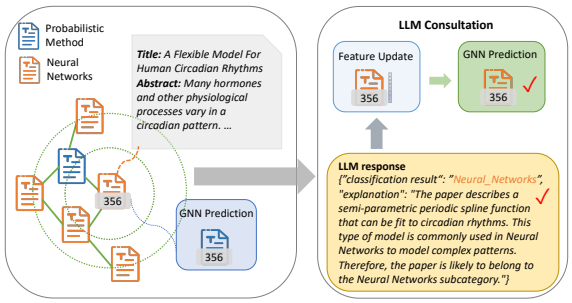

Figure 4: Case study of node 356 in Cora, where the LLM consultation classified right and performed feature update.

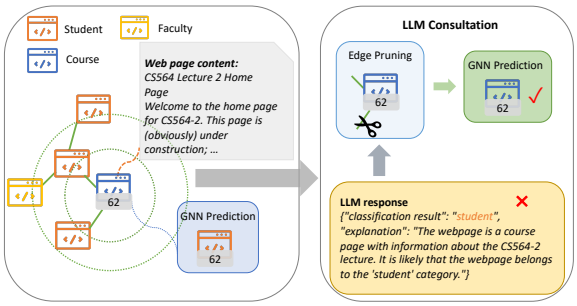

Figure 5: Case study of node 62 in Wisconsin, where the LLM consultation misclassified and pruned edges.

From a Citation Graph: Cora. We present node 356 from Cora as a representative example, whose ground truth label is *Neural Networks*. Unlike other papers, the title and abstract of this paper do not feature its label as a term explicitly, which also poses challenges for human classification. Besides, node 356 only has two one-hop neighbors, one of which is labeled differently as *Probabilistic Method*. This discrepancy may lead to failure in GNNs when processing this node. Nevertheless, thanks to the comprehensive understanding of the paper content facilitated by the parametric knowledge of LLMs, accurate prediction and a concise rationale are generated. Consequently, the subsequent semantic enhancement of node features significantly contributes to the final prediction of the GNN. The consultation process is elaborated in Fig. 4.

From a Web-page-link Graph: Wisconsin. Node 62 from Wisconsin represents a course web page, with content that is clear enough for humans to identify as a *course* homepage. However, due to its misleading neighborhood, where all nodes except itself in its 2-hop ego-graph do not represent course, among which 3 out of 4 are web pages of *students*, pre-trained GNNs cannot directly classify node 62 accurately. Moreover, the LLM consultant also provides the incorrect classification with an illogical rationale as in Fig. 5. This leads to edge pruning around node 62, which contributes to structure denoising that aids the re-trained GNN in making the right choice.

The studies of two interesting cases we encountered in experiments highlight the solidity and efficacy of LOGIN when handling various scenarios, which are consistent with our motivation of designing an LLM-fault-utilized strategy for feedback.

Table 2: Performance comparison among LLMs-as-Predictors, LLMs-as-Enhancers and LLMs-as-Consultants paradigms.

Dataset	Method	LLMs-as-Predictors (vicuna-v1.5-7b)	LLMs-as-Enhancers (TAPE + llama2-13b-chat)	LLMs-as-Consultants (LoGIN + vicuna-v1.5-7b)
Cora	MLP	0.7432 ± 0.0131	0.7675 ± 0.0187	0.7343 ± 0.0841
	GCN		0.8630 ± 0.0101	**0.8759 ± 0.0151**
	GraphSAGE		0.8625 ± 0.0093	0.8699 ± 0.0167
PubMed	MLP	0.7847 ± 0.0632	0.9475 ± 0.0046	**0.9508 ± 0.0040**
	GCN		0.9257 ± 0.0063	0.9401 ± 0.0043
	GraphSAGE		0.9464 ± 0.0033	0.9505 ± 0.0031
Arxiv-23	MLP	0.7547 ± 0.0927	0.7905 ± 0.0041	0.7909 ± 0.0064
	GCN		0.7751 ± 0.0029	0.7734 ± 0.0028
	GraphSAGE		0.7935 ± 0.0029	**0.7961 ± 0.0029**

5.5 Comparison among LLM-based Paradigms (RQ4)

To answer RQ4, we investigate the LLMs-as-Predictors, LLMs-as-Enhancers, and our LLM-as-Consultants paradigms by testing typical methods derived from each. For LLMs-as-Predictors, we prompt vicuna-v1.5-7b [67] to get direct predictions. For LLMs-as-Enhancers, we adopt TAPE [22] equipped with llama2-13b-chat[51].

Note that the accuracy scores in Table 2 differ from those in Table 1, since we take the results of TAPE + llama2-13b-chat from their paper, which are reported for four runs. In Table 2, we also present results for the same four data splits to ensure a fair comparison.

As Table 2 shows, the LLMs-as-Consultants paradigm consistently outperforms the LLMs-as-Predictors paradigm across all datasets. Additionally, compared to the LLMs-as-Enhancers paradigm, our method surpasses the TAPE method equipped with llama2-13b-chat with only one exception, despite it employs an open-source LLM with significantly more parameters and prompts it with all nodes rather than a small subset selected. Our LLMs-as-Consultants paradigm demonstrates greater compatibility with lower time and resource consumption. Comparisons with additional baselines from various LLMs-for-Graphs paradigms are provided in Appendix C.

5.6 Extensive Study (RQ5)

To answer RQ5, we conduct extended studies incorporating more advanced LLMs, consulting a larger number of nodes, and utilizing more advanced GNN backbones respectively.

Consulting more advanced LLMs. We consulted more advanced LLMs: vicuna-v1.5-13b [67] and GPT 3.5-turbo-0125 [7]. Due to the constraints related to computational resources and OpenAI API calling, we provide results solely for Cora, based on two runs as presented in Table 3.

In this experiment, all fundamental baselines with GPT 3.5-turbo-0125 outperform the ones with open-source LLMs. Additionally, the implementation of LoGIN with vicuna-v1.5-13b demonstrates a modest enhancement in predictive accuracy compared with vicuna-v1.5-7b, which has fewer parameters. The trend in predictive accuracy indicates that employing more advanced LLMs within LoGIN framework indeed facilitates performance increase. This underscores the significant potential of the LLMs-as-Consultants paradigm when equipped with more powerful LLMs.

Table 3: Performance comparison among LoGIN with different LLMs on Cora.

LLMs	MLP	GCN	GraphSAGE
Vicuna-v1.5-7b	0.7063 ± 0.0331	0.8694 ± 0.0102	0.8727 ± 0.0201
Vicuna-v1.5-13b	0.7202 ± 0.0201	0.8702 ± 0.0191	0.8739 ± 0.0135
GPT 3.5-turbo-0125	0.8123 ± 0.0254	0.8992 ± 0.0099	0.8856 ± 0.0102

Consulting with more nodes & More advanced GNN backbones. We further explore the effects of consulting with more nodes and employing more advanced GNN architectures. The details are provided in Appendix E. Empirical studies reveal that: (i) increasing the number of nodes in the LLM consultation by 10% leads to a notable improvement in performance; and (ii) LoGIN effectively enhances the performance of more advanced GNNs, such as APPNP and GCNII, extending its benefits beyond classic GNNs.

Through the above experiments, we demonstrate that incorporating more advanced LLMs, consulting a larger number of nodes, and utilizing more advanced GNN backbones can each enhance the LoGIN framework. This finding highlights the practicality and potential of the LLMs-as-Consultants paradigm.

6 Conclusion

In this work, we propose a new paradigm of leveraging LLMs for graph tasks, coined "LLMs-as-Consultants". Following this paradigm, our LoGIN framework empowers interactive LLM consultation in the GNN training process. We identify uncertain nodes in the GNN pre-training stage, prompt LLMs with rich semantic and topological information compression, and parse LLMs' responses in a not only fault-tolerant but also fault-utilized way to enhance GNN re-training. Extensive experiments on both homophilic and heterophilic graphs illustrate the validity and versatility of our proposed LLMs-as-Consultants paradigm.

Acknowledgments

The research work is supported by National Key R&D Plan No. 2022YFC3303302, Beijing Nova Program 20230484430, the Innovation Funding of ICT, CAS under Grant No. E461060, the National Natural Science Foundation of China under Grant No. 62476263, No. 62406307, and the Postdoctoral Fellowship Program of CPSF under Grant No. GZB20240761.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems* 35 (2022), 23716–23736.

[3] Chumki Basu, Haym Hirsh, and William Cohen. 1998. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. *Proceedings of AAAI-98* (1998).

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[5] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 3950–3957.

[6] Broder, Andrei, Kumar, Ravi, and Janet. 2000. Graph Structure in the Web : Experiments and models. (2000).

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[8] 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* (2023).

[9] Deli Chen, Yankai Lin, Guangxiang Zhao, Xuancheng Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Topology-imbalance learning for semi-supervised node classification. *Advances in Neural Information Processing Systems* 34 (2021), 29885–29897.

[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International conference on machine learning*. PMLR, 1725–1735.

[11] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393* (2023).

[12] Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Seán Slattery. 1998. Learning to extract symbolic knowledge from the World Wide Web. *AAAI/IAAI* 3, 3.6 (1998), 2.

[13] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565* (2023).

[14] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560* (2023).

[15] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*. 2331–2341.

[16] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *International Conference on Learning Representations*.

[17] Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

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

[19] William L Hamilton. 2020. *Graph representation learning*. Morgan & Claypool Publishers.

[20] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[21] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION. In *International Conference on Learning Representations*. https://openreview.net/forum?id=XPZIaotutsD

[22] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. *arXiv preprint arXiv:2305.19523* (2023).

[23] 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. *arXiv preprint arXiv:2106.09685* (2021).

[24] Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403* (2022).

[25] Mohammad Rasool Izadi, Yihao Fang, Robert Stevenson, and Lizhen Lin. 2020. Optimization of graph neural networks with natural gradient descent. In *2020 IEEE international conference on big data (big data)*. IEEE, 171–179.

[26] Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, et al. 2024. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *arXiv preprint arXiv:2403.04468* (2024).

[27] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems* 30 (2017).

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

[29] Christopher Krügel, Thomas Toth, and Engin Kirda. 2002. Service specific anomaly detection for network intrusion detection. In *the 2002 ACM symposium*.

[30] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems* 34 (2021), 20887–20902.

[31] Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. ClickPrompt: CTR Models are Strong Prompt Generators for Adapting Language Models to CTR Prediction. In *Proceedings of the ACM on Web Conference 2024*. 3319–3330.

[32] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *arXiv preprint arXiv:2304.08485* (2023).

[33] Yang Liu, Xiang Ao, Fuli Feng, and Qing He. 2022. UD-GNN: Uncertainty-aware Debiased Training on Semi-Homophilous Graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1131–1140.

[34] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*. 3168–3177.

[35] Yixin Liu, Kai Zhang, Yuan Li, Zhiling Yan, Chujie Gao, Ruoxi Chen, Zhengqing Yuan, Yue Huang, Hanchi Sun, Jianfeng Gao, et al. 2024. Sora: A Review on Background, Technology, Limitations, and Opportunities of Large Vision Models. *arXiv preprint arXiv:2402.17177* (2024).

[36] Yu-Chen Lo, Stefano E Rensi, Wen Torng, and Russ B Altman. 2018. Machine learning in chemoinformatics and drug discovery. *Drug discovery today* 23, 8 (2018), 1538–1546.

[37] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems* 35 (2022), 1362–1375.

[38] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. *Advances in neural information processing systems* 32 (2019).

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

[40] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[41] Joonhyung Park, Jaeyun Song, and Eunho Yang. 2021. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International conference on learning representations*.

[42] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

[43] Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. 2021. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1390–1398.

[44] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation Learning with Large Language Models for Recommendation. In *Proceedings of the ACM on Web Conference 2024*. 3464–3475.

[45] E. M. Rogers and D. K. Bhowmik. 1970. Homophily-heterophily: Relational concepts for communication research. *Public Opinion Quarterly* 34, 4 (1970), 523–538.

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

[47] Jaeyun Song, Joonhyung Park, and Eunho Yang. 2022. TAM: topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*. PMLR, 20369–20383.

[48] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 2022. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*. PMLR, 20479–20502.

[49] Shengyin Sun, Yuxiang Ren, Chen Ma, and Xuecang Zhang. 2023. Large Language Models as Topological Structure Enhancers for Text-Attributed Graphs. *arXiv preprint arXiv:2311.14324* (2023).

[50] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023* (2023).

[51] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[52] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[53] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can Language Models Solve Graph Problems in Natural Language? *arXiv preprint arXiv:2305.10037* (2023).

[54] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature* 620, 7972 (2023), 47–60.

[55] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data* 9, 2 (2022), 415–436.

[56] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *The world wide web conference*. 2022–2032.

[57] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[58] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*. PMLR, 5453–5462.

[59] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134* (2023).

[60] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[61] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in neural information processing systems* 32 (2019).

[62] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 793–803.

[63] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems* 33 (2020), 9263–9275.

[64] Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089* (2023).

[65] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*. 833–841.

[66] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[67] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685* (2023).

[68] Qiwei Zhong, Yang Liu, Xiang Ao, Binbin Hu, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *Proceedings of The Web Conference 2020*. 785–795.

[69] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems* 33 (2020), 7793–7804.

[70] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*. 1215–1226.

[71] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3162–3172.