GraphGPT: Graph Instruction Tuning for Large Language Models

Jiabin Tang[1], Yuhao Yang[2], Wei Wei[2], Lei Shi[3],
Lixin Su[3], Suqi Cheng[3], Dawei Yin[3] and Chao Huang[1,2*]
[1]Musketeers Foundation Institute of Data Science,
[2]Department of Computer Science, University of Hong Kong, [3]Baidu Inc.
Project Page: https://GraphGPT.github.io, **Github**: https://github.com/HKUDS/GraphGPT

ABSTRACT

Graph Neural Networks (GNNs) have advanced graph structure understanding via recursive information exchange and aggregation among graph nodes. To improve model robustness, self-supervised learning (SSL) has emerged as a promising approach for data augmentation. However, existing methods for generating pre-trained graph embeddings often rely on fine-tuning with specific downstream task labels, which limits their usability in scenarios where labeled data is scarce or unavailable. To address this, our research focuses on advancing the generalization capabilities of graph models in challenging zero-shot learning scenarios. Inspired by the success of large language models (LLMs), we aim to develop a graph-oriented LLM that can achieve high generalization across diverse downstream datasets and tasks, even without any information available from the downstream graph data. In this work, we present the GraphGPT framework that aligns LLMs with graph structural knowledge with a graph instruction tuning paradigm. Our framework incorporates a text-graph grounding component to establish a connection between textual information and graph structures. Additionally, we propose a dual-stage instruction tuning paradigm, accompanied by a lightweight graph-text alignment projector. This paradigm explores self-supervised graph structural signals and task-specific graph instructions, to guide LLMs in understanding complex graph structures and improving their adaptability across different downstream tasks. Our framework is evaluated on supervised and zero-shot graph learning tasks, demonstrating superior generalization and outperforming state-of-the-art baselines.

1 INTRODUCTION

Graph neural networks (GNNs) have emerged as a powerful framework for analyzing and learning from graph-structured data [5, 27], enabling advancements in various domains, such as social network analysis [31, 65], recommender systems [10, 42], and biological network analysis [7, 25]. One of the key benefits of GNNs is their ability to capture the inherent structural information and dependencies present in graph data. By leveraging message passing and aggregation mechanisms, GNNs can effectively propagate and combine information across the graph, enabling them to model complex relationships and make accurate predictions.

In recent years, various GNN architectures have introduced innovations in how information is exchanged and aggregated among graph nodes. For example, graph convolutional network (GCNs) [17, 22] adapt convolutional operations to the graph domain, enabling effective feature representations. Graph attention networks (GATs) [39, 43] leverages attention mechanisms to assign different weights to neighboring nodes, allowing for more fine-grained information aggregation. Graph transformer networks (GTNs) [15, 60] incorporate self-attention and positional encoding to capture global dependencies and structural patterns in the graph. However, a notable limitation of many GNN approaches is their heavy reliance on supervised learning, which can lead to inadequate robustness and generalization when confronted with sparse and noisy data.

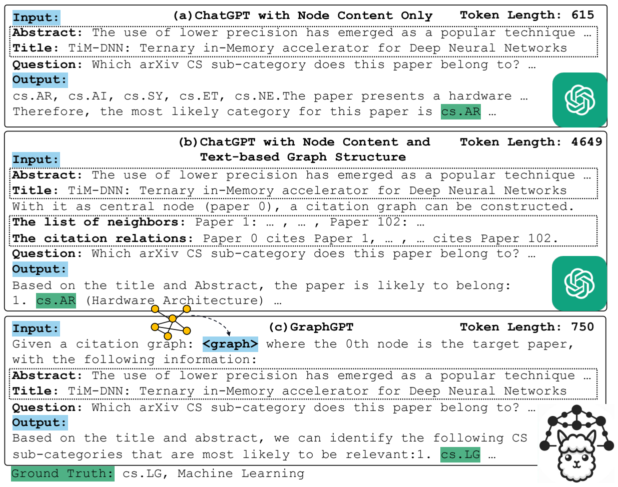
To enhance the generalization ability of GNNs, self-supervised learning (SSL) has emerged as a promising approach in graph representation learning. It aims to pre-train a robust graph model using auxiliary tasks on unlabeled graph data. The idea is to leverage the inherent structure and patterns within the graph itself to create meaningful self-supervisory signals. SSL-enhanced graph learning methods exhibit two primary paradigms: contrastive SSL and generative SSL. Within contrastive SSL, the emphasis lies on learning representations by contrasting positive and negative samples, with notable advancements of DGI [40] and GCA [67]. Conversely, generative SSL focuses on generating synthetic samples that closely resemble the original graph structures with masked autoencoders, exemplified by techniques like GraphMAE [12] and S2GAE [35].

While these methods aim to generate graph embeddings that are generalizable to different downstream tasks, they often require a fine-tuning process using labels specific to the downstream graph learning scenarios. However, this reliance on labeled data from downstream tasks can restrict their generalization in practical situations where obtaining high-quality labels may not always be feasible. This limitation is particularly relevant in learning scenarios like cold-start recommendation systems or traffic flow prediction in new cities where accurate labels may be scarce or unavailable.

As a result, the objective of this research is to advance the generalization capabilities of graph models by addressing challenging and practical zero-shot learning scenarios. Inspired by the remarkable success of large language models (LLMs) in natural language processing (NLP) tasks, where they have demonstrated exceptional generalization abilities, this work aims to develop a graph-oriented LLM capable of achieving high generalization across diverse downstream datasets and tasks. However, effectively integrating large language models with graph learning poses non-trivial challenges.

* Corresponding author, chaohuang75@gmail.com.

The Conference, 2024, XXX
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.



```
Input:              (a)ChatGPT with Node Content Only    Token Length: 615
Abstract: The use of lower precision has emerged as a popular technique …
Title: TiM-DNN: Ternary in-Memory accelerator for Deep Neural Networks
Question: Which arXiv CS sub-category does this paper belong to? …
Output:
cs.AR, cs.AI, cs.SY, cs.ET, cs.NE.The paper presents a hardware …
Therefore, the most likely category for this paper is cs.AR …
```

```
                    (b)ChatGPT with Node Content and    Token Length: 4649
Input:                 Text-based Graph Structure
Abstract: The use of lower precision has emerged as a popular technique …
Title: TiM-DNN: Ternary in-Memory accelerator for Deep Neural Networks
With it as central node (paper 0), a citation graph can be constructed.
The list of neighbors: Paper 1: … , … , Paper 102: …
The citation relations: Paper 0 cites Paper 1, … , … cites Paper 102.
Question: Which arXiv CS sub-category does this paper belong to? …
Output:
Based on the title and Abstract, the paper is likely to belong:
1. cs.AR (Hardware Architecture) …
```

```
Input:                         (c)GraphGPT              Token Length: 750
Given a citation graph: <graph> where the 0th node is the target paper,
with the following information:
Abstract: The use of lower precision has emerged as a popular technique …
Title: TiM-DNN: Ternary in-Memory accelerator for Deep Neural Networks
Question: Which arXiv CS sub-category does this paper belong to? …
Output:
Based on the title and abstract, we can identify the following CS
sub-categories that are most likely to be relevant:1. cs.LG …
Ground Truth: cs.LG, Machine Learning
```

Figure 1: Limitation of LLMs in understanding graph structural contexts with heavy reliance on textual data.

- **C1**: Achieving a proper alignment between the structural information of a graph and the language space demands meticulous deliberation and thoughtful consideration.
- **C2**: Effectively guiding LLMs to comprehend the structural information of graphs remains a considerable challenge.
- **C3**: Endowing LLMs with the ability to reason step-by-step is important when tackling complex graph learning tasks.

To gain a deeper understanding of the limitations associated with directly prompting LLMs using purely text-based prompts for graph structure modeling, we provide illustrative examples in Figure 1. These examples facilitate a comparative analysis between our GraphGPT framework and the ChatGPT approach. We focus on a representative node classification task, where the objective is to predict the category of a given paper. In Figure 1 (a) and Figure 1 (b), we showcase the prediction results for two scenarios using ChatGPT: (1) utilizing only the input node textual data, and (2) employing text-based graph structure-aware prompts inspired by the prompt designs in recent studies [2, 6]. These figures highlight the potential limitations that arise when relying solely on text-based prompts for graph structure modeling, as evidenced by the incorrect paper node classification results presented. In contrast, our GraphGPT framework effectively addresses these limitations by preserving and leveraging the graph structural information, as shown in Figure 1 (c). It enables accurate identification of the paper category, in understanding the underlying graph structure.

Additionally, the utilization of text-based structural prompts leads to an increase in token size, which presents challenges in practical scenarios. Longer token sequences incur higher computational and memory costs, making it less feasible for real-world applications. Furthermore, existing LLMs have token limits, which further restrict the applicability of longer text-based prompts for large-scale graph structure modeling. These limitations emphasize the necessity for more efficient and scalable approaches that can effectively incorporate graph structural information into LLMs.

Contributions. To address these challenges, we propose a novel framework called GraphGPT, which aims to align Large Language Models (LLMs) with Graphs using a carefully designed graph instruction tuning paradigm. (**C1**) Our framework introduces a text-graph grounding paradigm as the initial step to align the encoding of graph structures with the natural language space. By incorporating textual information in a contrastive manner, we enable effective alignment of graph structure information within language models. (**C2**) In our proposed dual-stage graph instruction tuning paradigm, we leverage self-supervised signals through the graph matching task, which is derived from unlabeled graph structures, to serve as instructions for guiding model tuning of LLMs. By incorporating this self-supervised instruction tuning, the language model acquires domain-specific structural knowledge related to graphs, thereby enhancing its understanding of graph structures. To further customize the LLM's reasoning behavior for diverse downstream graph learning tasks, the second stage of our graph instruction tuning paradigm involves fine-tuning the LLM with task-specific graph instructions, to improve the model's adaptability. (**C3**) By incorporating the Chain-of-Thought (COT) distillation into our framework, GraphGPT enhances its step-by-step reasoning abilities and improves its performance in the face of distribution shift.

In summary, our work makes the following contributions:

- This work aims to align graph domain-specific structural knowledge with the reasoning ability of Large Language Models (LLMs) to improve the generalization of graph learning.
- Our approach aims to align LLMs with Graphs through a graph instruction tuning paradigm. This paradigm incorporates self-supervised instruction tuning, enhancing the LLM's comprehension of graph structural knowledge and its reasoning capabilities. Additionally, we introduce task-specific instruction tuning to improve the model's adaptability across diverse graph tasks.
- We evaluate our proposed GraphGPT on supervised and zero-shot graph learning tasks. We conduct thorough analyses of its component-wise effects and generalization ability. By comparing it with state-of-the-art baselines, we demonstrate the superior generalization power of our approach across various settings.

2 PRELIMINARIES

Graph-structured Data. represents information as a collection of entities (nodes) and the relationships (edges) between them. A graph is formally denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, encompassing several core components. The node set $\mathcal{V}$ represents a collection of nodes, with $|\mathcal{V}| = N$ indicating the total number of nodes in the graph. The edge set $\mathcal{E}$ characterizes the relationships or connections between the nodes. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ encodes the topology of the graph, with each element $A_{i,j}$ indicating the presence or absence of an edge between nodes i and j. The feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$ contains the attribute or feature information associated with each node, where F represents the dimensionality of features.

Graph Neural Networks. have emerged as a powerful framework for representation learning from graph-structured data. Unlike traditional neural networks that operate on grid-like data, GNNs can effectively capture and model the complex relationships and dependencies present in graphs. Specifically, GNNs leverage the inherent structure of the graph, consisting of nodes and edges, to learn expressive node representations through iterative message propagation and aggregation operations, presented as follows.

$$m_v^{(l)} = \text{Propagate}^{(l)}(\{h_u^{(l-1)} : u \in \mathcal{N}(v)\}),$$
$$h_v^{(l)} = \text{Aggregate}^{(l)}(h_v^{(l-1)}, m_v^{(l)}) \tag{1}$$

After applying the message passing and aggregation mechanism in Graph Neural Networks (GNNs), the encoded feature vector of node v at the l-th layer is denoted as $h_v^{(l)}$. The Propagate$^{(l)}$ function performs message passing by aggregating information from the neighboring nodes of v at the l-th layer. The Aggregate$^{(l)}$ function then combines the aggregated information with the previous layer's representation of node v to generate the updated representation $h_v^{(l)}$. By encoding graph structural information with the learned representations, GNNs can be customized for various downstream graph learning tasks, such as node classification and link prediction.

3 METHODOLOGY

3.1 Structural Information Encoding with Text-Graph Grounding

To enhance the understanding of graph structural information by large language models, our framework emphasizes aligning the encoding of graph structures with the natural language space. This alignment aims to enable language models to effectively comprehend and interpret the structural elements of the graph, leveraging their inherent language understanding capabilities. To achieve this objective, we introduce a text-graph grounding paradigm that generates prompts designed to preserve the graph's structural context for language models. This paradigm acts as a bridge, connecting the semantic understanding of textual information with the inherent structural relationships found within the graph.

In our GraphGPT, we design the graph encoder to be highly flexible, allowing it to leverage a wide range of backbone GNN architectures obtained from diverse graph pre-training paradigms. We incorporate a message-passing neural network architecture, which can be a graph transformer [60] or a graph convolutional network [17], as the structure-level pre-trained graph model. In each message-passing step, the graph encoder aggregates information from neighboring nodes, considering their relationships:

$$\mathbf{H}^{(l)} = \sigma\left(\tilde{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}\right) \tag{2}$$

The self-loop adjacency matrix, denoted as $\tilde{\mathbf{A}}$, is obtained by adding the identity matrix $\mathbf{I}$ to the original adjacency matrix $\mathbf{A}$. $\mathbf{W}$ is the parameter matrix. This matrix captures the self-connections and local connectivity of nodes in the graph. $\sigma(\cdot)$ is the non-linear activation. $\mathbf{H}^{(l)}$ is the graph representations at the l-th layer.

Text-Structure Alignment. To align graph structure information within Language Models (LLMs) more effectively, our main focus is to explore the encoding of graph structures that can collaborate well with LLMs. Inspired by prior works [30, 48], we incorporate textual information into the graph structure encoding process in a contrastive manner. In our approach, we directly integrate a graph encoder with pre-trained parameters into our GraphGPT framework, enabling seamless integration of the graph encoder's capabilities in our framework. Formally, let $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ represent a graph with raw textual contents $\mathbf{C} = c_i \in \mathbb{R}^{l_i \times d}, 1 \leq i \leq N$ for N nodes, where l_i denotes the length of the textual content for the i-th node. We obtain encoded graph representations $\hat{\mathbf{H}} \in \mathbb{R}^{N \times d}$ and encoded text representations $\hat{\mathbf{T}} \in \mathbb{R}^{N \times d}$ as follows:

$$\mathbf{H} = f_{\mathbf{G}}(\mathbf{X}), \mathbf{T} = f_{\mathbf{T}}(\mathbf{C}), \hat{\mathbf{H}} = \text{norm}(\mathbf{H}), \hat{\mathbf{T}} = \text{norm}(\mathbf{T}) \tag{3}$$

We utilize the graph encoder, denoted as $f_{\mathbf{G}}$, to take the graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ as input and generates structure-level graph representations. We employ a text encoder, such as a transformer or Bert, denoted as $f_{\mathbf{T}}$, to encode the raw textual contents $\mathbf{C}$ associated with the nodes. This step produces encoded text representations of nodes. We further apply row-wise L2 normalization using the norm function. Formally, the text-structure alignment with the cross-modalities is conducted as follows:

$$\Gamma_i = (g_i^{(1)}(\hat{\mathbf{H}}) g_i^{(2)}(\hat{\mathbf{T}})^\top) \cdot \exp(\tau)$$
$$\mathcal{L} = \sum_i \frac{1}{2}\lambda_i(\text{CE}(\Gamma_i, \mathbf{y}) + \text{CE}(\Gamma_i^\top, \mathbf{y})) \tag{4}$$

In our text-graph grounding, we use the label $\mathbf{y} = (0, 1, \cdots, n-1)^\top$ for the contrastive alignment objective. The transformation functions for different grounding designs are denoted as $g_i^{(1)}$ and $g_i^{(2)}$. We employ a graph transformer [61] as the graph encoder and a vanilla transformer [38] as the text encoder.

3.2 Dual-Stage Graph Instruction Tuning

The dual-stage graph instruction tuning paradigm proposed in this work builds upon the concept of instruction tuning, which has been recently introduced to enhance the adaptability of language models for specific domains [45]. In this paradigm, we aim to align the language capacity of the model with the nuances of graph learning tasks, enabling the language model to generate more accurate and contextually appropriate responses for graph-structured data.

3.2.1 **Self-Supervised Instruction Tuning.** In the first stage of our graph instruction tuning paradigm, we introduce self-supervised instruction tuning. This mechanism allows us to inject graph domain-specific structural knowledge into the language model, improving its reasoning abilities and enabling it to effectively comprehend the contextual information embedded in the graph's structure. To achieve this, we incorporate self-supervised signals derived from unlabeled graph structures as instructions for model tuning. In particular, we design a structure-aware graph matching task that guides the language model in distinguishing between different graph tokens using natural language tokens. This instruction task plays a crucial role in accurately associating graph tokens with their corresponding textual descriptions, thereby deepening the model's understanding of the graph with the provided guidance.

Instruction Design. The instruction for our graph matching task consists of three components: i) graph information, ii) human question, and iii) GraphGPT response. In this task, we treat each node in the graph as a central node and perform h-hops with random neighbor sampling, resulting in a subgraph structure. The natural language input for the LLM is the human question. In the context of the graph matching task, the instruction includes the indicator token `<graph>` and a shuffled list of node text information. For example, in a citation graph, the node text information corresponds to paper titles. The objective of the LLM in the graph matching task is to align each graph token with its corresponding node text

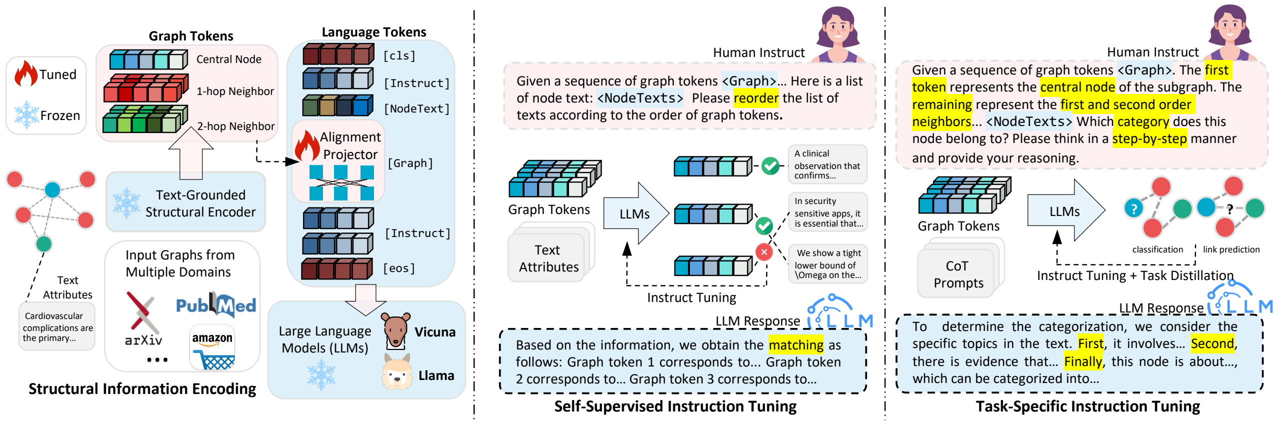

Graph Tokens
Language Tokens
Tuned
Frozen
Central Node
1-hop Neighbor
2-hop Neighbor
[cls]
[Instruct]
[NodeText]
Alignment Projector
[Graph]
[Instruct]
[eos]
Text-Grounded Structural Encoder
Text Attributes
Cardiovascular complications are the primary...
Input Graphs from Multiple Domains
arXiv
PubMed
amazon
Large Language Models (LLMs)
Vicuna
Llama
Structural Information Encoding
Human Instruct
Given a sequence of graph tokens <Graph>... Here is a list of node text: <NodeTexts> Please reorder the list of texts according to the order of graph tokens.
Graph Tokens
Text Attributes
LLMs
A clinical observation that confirms...
In security sensitive apps, it is essential that...
We show a tight lower bound of \Omega on the...
Instruct Tuning
LLM Response
Based on the information, we obtain the matching as follows: Graph token 1 corresponds to... Graph token 2 corresponds to... Graph token 3 corresponds to...
Self-Supervised Instruction Tuning
Human Instruct
Given a sequence of graph tokens <Graph>. The first token represents the central node of the subgraph. The remaining represent the first and second order neighbors... <NodeTexts> Which category does this node belong to? Please think in a step-by-step manner and provide your reasoning.
Graph Tokens
CoT Prompts
LLMs
classification
link prediction
Instruct Tuning + Task Distillation
LLM Response
To determine the categorization, we consider the specific topics in the text. First, it involves... Second, there is evidence that... Finally, this node is about..., which can be categorized into...
Task-Specific Instruction Tuning

Figure 2: The overall architecture of our proposed GraphGPT with graph instruction tuning paradigm.

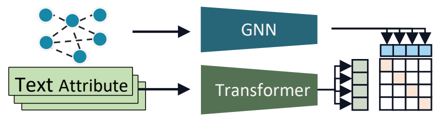


Figure 3: Workflow of text-structure alignment.

information. This requires reordering the node text information list based on the sequence of graph tokens, effectively associating each graph token with its relevant textual description.

Tuning Strategy. To optimize the tuning process efficiently, we propose a strategy that incorporates a **Lightweight Alignment Projector**. During training, we keep the parameters of both the LLM and the graph encoder fixed, focusing solely on optimizing the parameters of the projector $f_{\mathbf{P}}$. After training, we assume that the projector has successfully learned to map the encoded graph representation to graph tokens, while the LLM excels at aligning these tokens with diverse node text information. To align the graph tokens with the language tokens, we employ a projector $f_{\mathbf{P}}$, which can be as simple as a single linear layer. This projector establishes the correspondence between the graph tokens and the language tokens. By replacing the indicator token <graph> in the original nature language token sequence, the aligned graph tokens create a modified token sequence for the large language model. This modified sequence, denoted as {<graph_begin>, <graph_token>$_1, \cdots,$ <graph_token>$_n$, <graph_end>}, corresponds to the number of nodes n in the graph associated with the given prompt. Given that the graph matching process is unsupervised, we have the opportunity to leverage a vast amount of unlabeled graph data from different domains, to enhance the generalizability of the learned projector.

3.2.2 **Task-Specific Instruction Tuning.**

In the second stage, we propose task-specific instruction tuning. This step is designed to customize the model's reasoning behavior to meet the specific constraints and requirements of different graph learning tasks, such as node classification or link prediction. By fine-tuning the LLM using task-specific graph instructions, we guide the model to generate responses that are better suited for the particular graph learning task at hand. This further improves the model's adaptability and performance in handling various graph learning tasks.

Instruction Design. We adopt a similar instruction template, which consists of three parts. To generate graph information for each node, we employ the same neighbor sampling approach used in the first stage. This approach ensures that relevant graph information is captured, with each node acting as the central node. For the node classification task, the human question instruction contains both the indicator token <graph> and specific text information about the central node. This instruction prompts the language model to predict the category of the central node based on both the graph structure data and the accompanying text information. An example of the instruction data for different tasks can be seen in Figure 4, providing a visual representation of how the instruction is structured and presented to the language model.

Tuning Strategy. In the second stage of training, we utilize the parameters of the structure-aware projector that were trained in the first stage as the initial state. This allows us to conduct instruction tuning specifically for downstream tasks. During this training process, we keep the parameters of the language model (LLM) and graph encoder fixed, focusing solely on optimizing the parameters of the projector from the previous stage. By doing so, we ensure that the LLM further aligns with the requirements of downstream tasks, enhancing its ability to comprehend and interpret graph structures.

After completing the two training stages as described above, we have confidence that our GraphGPT has acquired the capability to comprehend the given graph structure and perform downstream tasks on the provided graph. The training process involving instruction tuning and the freezing of specific model parameters has refined the model's understanding of graph structures, enabling it to effectively tackle various tasks associated with the given graph.

3.3 Chain-of-Thought (CoT) Distillation

When faced with diverse graph data, language models may encounter new or unfamiliar patterns and structures. This distribution shift can pose challenges in generating accurate and coherent responses, especially when the number of node classes varies across different types of graph data. To address this challenge and boost accuracy in the presence of distribution shift, it is essential to equip our GraphGPT with step-by-step reasoning abilities. In this regard, we propose utilizing the Chain-of-Thought (COT) technique [47], which explicitly models the flow of thoughts and reasoning steps. By incorporating COT, our language model improves the coherence and consistency of generated text. It enables the model to follow a logical progression of ideas, enhancing its ability to understand and reason about the given graph data.

Graph Information: **<graph>: Central Node:** 68442, **Edge index:** [[…src node…],[…dst node…]], **Node list:** […] **Graph Matching**
Human Question: Given a sequence of graph tokens **<graph>** that constitute a subgraph of a citation graph, …. Here is a list of paper titles: 1. … 2. …, please reorder the list of papers according to the order of graph tokens.
GraphGPT Response: Based on the given graph tokens and the list of paper titles, we obtain the matching of graph tokens and papers: Graph token 1 corresponds to smt based induction methods for timed systems. Graph token 2 corresponds to …

Graph Information: **<graph>: Central Node:** 2, **Edge index:** [[…src node…],[…dst node…]], **Node list:** […] **Node Classification**
Human Question: Given a citation graph: **<graph>** where the 0th node is the target paper, with the following information: Abstract: … Title: … Question: Which arXiv CS sub-category does this paper belong to? …
GraphGPT Response: cs.IT, cs.LG, cs.SP, cs.CV, cs.NA. The paper discusses the Restricted Isometry …. So, it is likely to belong to cs.IT…

Graph Information: **<graph>: Central Node 1:** 8471, **Edge index 1:** [[…src node…],[…dst node…]], **Node list 1:** […] **Link Prediction**
<graph>: Central Node 2: 19368, **Edge index 2:** [[…src node…],[…dst node…]], **Node list 2:** […]
Human Question: Given a sequence of graph tokens: **<graph>** that constitute a subgraph of a citation graph, …. Abstract: … Titile: … and the other sequence of graph tokens: **<graph>**, … Abstract: … Title: …, are these two central nodes connected? Give me an answer of "yes" or "no".
GraphGPT Response: Yes, they are connected. Based on the first paper, …. And the second paper proposes ….

Figure 4: Our instruction designs for graph matching task (upper), node classification (middle) and link prediction (lower).

However, incorporating the Chain-of-Thought (COT) technique can be challenging due to the influence of model parameter scale [32]. To overcome this, we draw inspiration from previous research [32] and employ a distillation approach to extract valuable knowledge from a closed-source, powerful language model like ChatGPT (with over 200 billion parameters). This enables us to generate high-quality COT instructions and enhance our model's COT reasoning capabilities while avoiding an increase in parameters.

COT Distillation Paradigm. Our approach involves designing tailored Chain-of-Thought (COT) prompts specifically for node-specific tasks. For the node classification task within a citation graph, we provide the abstract, title of the paper represented by the node, and a description of the classification task as part of the input. We then employ the GPT-3.5 language model (LLM) in our implementation to perform step-by-step reasoning. We prompt the LLM to arrive at the final answer by engaging in a sequential thought process. In the generated output, the LLM not only provides predictions for the node classes but also offers detailed explanations for each prediction. This ensures that the model's reasoning and decision-making process are transparent and comprehensible. To further improve the performance, we integrate the generated COT instruction data with the previously designed instructions for the task-specific instruction tuning stage. With the integrated instructions, we proceed with the proposed instruction tuning paradigm.

4 EVALUATION

We conduct experiments to validate the effectiveness of our framework in various settings and address key research questions.

- **RQ1:** How does the proposed GraphGPT framework perform in both supervised and zero-shot graph learning settings?
- **RQ2:** What is the generalization ability of our model in handling multiple tasks without experiencing catastrophic forgetting?
- **RQ3:** What is the contribution of various key components in the proposed GraphGPT framework to its overall performance?
- **RQ4:** How scalable and efficient is our GraphGPT framework?

4.1 Experimental Settings

4.1.1 **Data Descriptions.** We evaluate the performance of our GraphGPT using three datasets: OGB-arxiv, PubMed, and Cora. The OGB-arxiv dataset [13] represents a directed graph that captures the citation network among computer science arXiv papers indexed by MAG [41]. Each paper in the dataset is associated with a research category, manually labeled by the authors and arXiv moderators. These research categories are selected from a set of 40 subject areas. The PubMed dataset [9] consists of 19,717 scientific publications on diabetes obtained from the PubMed database. The publications are categorized into Experimental induced diabetes, Type 1 diabetes, and Type 2 diabetes. Additionally, the dataset includes a citation network with 44,338 links. The Cora dataset [48] comprises 25,120 research papers connected through citations. We utilize an expanded version of the Cora dataset, which is larger and has more classes (70 in total) compared to previous versions [17].

4.1.2 **Evaluation Protocols.** To ensure compatibility and enable comparison across different datasets, we map the node features into the same vector space. We achieve this by encoding the raw text information using a pre-trained BERT model [4]. In our experiments, we divide the Cora and PubMed datasets into three parts: training, validation, and testing, following a ratio of 3:1:1. This partitioning scheme aligns with the approaches described in the works [9, 48]. For the OGB-arxiv dataset, we adhere to the public split setting [13], which employs a ratio of 6:2:3 for training, validation, and testing. To evaluate the performance of our model, we utilize three commonly adopted evaluation metrics: Accuracy and Macro F1 for node classification, and AUC (Area Under the Curve) for link prediction.

4.1.3 **Baseline Methods.** In our performance comparison, we consider various state-of-the-art methods for comprehensive evaluation. (i) The first category includes MLP, which employs a Multilayer Perceptron for node representation. (ii) The second category comprises representative graph neural encoders, including GraphSAGE [8], GCN [17], GAT [39], and RevGNN [21]. (iii) The third category focuses on the self-supervised approach DGI [40] for graph learning. (iv) The fourth category explores knowledge distillation-enhanced GNNs, with GKD [54] and GLNN [63] as notable methods. (v). The fifth category showcases recently proposed strong graph transformer networks, with NodeFormer [50] and DIFFormer [49] as competitors. (vi) Lastly, we consider open-sourced LLMs, such as Baichuan-7B, vicuna-7B-v1.1, and vicuna-7B-v1.5 as baselines for understanding text-attributed graph data. For more detailed descriptions of the baselines, please refer to the Appendix.

4.1.4 **Implementation Details.** For our model implementation, we primarily utilize the PyTorch and Transformers libraries. We employ Vicuna-7B-v1.1 and Vicuna-7B-v1.5 as the base models for our approach. The batch size is set to 2 on each GPU, and the learning rate is $2e^{-3}$. We apply a warmup ratio of $3e^{-2}$ and set the maximum input length of the Large Language Model (LLM) to 2048. The training process is carried out for 3 epochs. In the stage of task-specific instruction tuning, we explore the performance of

Table 1: Performance comparison of various methods on node classification under both supervised and zero-shot settings.

Dataset	Arxiv-Arxiv		Arxiv-PubMed		Arxiv-Cora		(Arxiv+PubMed)-Cora		(Arxiv+PubMed)-Arxiv	
Model	Accuracy	Macro-F1	acc	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
MLP	0.5179	0.2536	0.3940	0.1885	0.0258	0.0037	0.0220	0.0006	0.2127	0.0145
GraphSAGE	0.5480	0.3290	0.3950	0.1939	0.0328	0.0132	0.0132	0.0029	0.1281	0.0129
GCN	0.5267	0.3202	0.3940	0.1884	0.0214	0.0088	0.0187	0.0032	0.0122	0.0008
GAT	0.5332	0.3118	0.3940	0.1884	0.0167	0.0110	0.0161	0.0057	0.1707	0.0285
RevGNN	0.5474	0.3240	0.4440	0.3046	0.0272	0.0101	0.0217	0.0016	0.1309	0.0126
DGI	0.5059	0.2787	0.3991	0.1905	0.0205	0.0011	0.0205	0.0011	0.5059	0.2787
GKD	0.5570	0.1595	0.3645	0.2561	0.0470	0.0093	0.0406	0.0037	0.2089	0.0179
GLNN	0.6088	0.3757	0.4298	0.3182	0.0267	0.0115	0.0182	0.0092	0.3373	0.1115
NodeFormer	0.5922	0.3328	0.2064	0.1678	0.0152	0.0065	0.0144	0.0053	0.2713	0.0855
DIFFormer	0.5986	0.3355	0.2959	0.2503	0.0161	0.0094	0.0100	0.0007	0.1637	0.0234
baichuan-7B	0.0946	0.0363	0.4642	0.3876	0.0405	0.0469	0.0405	0.0469	0.0946	0.0363
vicuna-7B-v1.1	0.2657	0.1375	0.5251	0.4831	0.1090	0.0970	0.1090	0.0970	0.2657	0.1375
vicuna-7B-v1.5	0.4962	0.1853	0.6351	0.5231	0.1489	0.1213	0.1489	0.1213	0.4962	0.1853
GraphGPT-7B-v1.1-cot	0.4913	0.1728	0.6103	0.5982	0.1145	0.1016	0.1250	0.0962	0.4853	0.2102
GraphGPT-7B-v1.5-stage2	**0.7511**	**0.5600**	0.6484	0.5634	0.0813	0.0713	0.0934	0.0978	0.6278	0.2538
GraphGPT-7B-v1.5-std	0.6258	0.2622	**0.7011**	**0.6491**	0.1256	0.0819	0.1501	0.0936	0.6390	0.2652
GraphGPT-7B-v1.5-cot	0.5759	0.2276	0.5213	0.4816	**0.1813**	**0.1272**	**0.1647**	**0.1326**	**0.6476**	**0.2854**
p-val	$2.26e^{-9}$	$1.56e^{-10}$	$2.22e^{-7}$	$1.55e^{-9}$	$1.04e^{-9}$	$9.96e^{-6}$	$7.62e^{-8}$	$1.97e^{-7}$	$1.5\mathrm{e}^{-13}$	$4.63e^{-6}$

the model under different data mixtures by adopting various combinations of instruction data. The hyperparameter settings remain constant, except for the number of training epochs, which is set to 2 in this stage. The alignment projector parameters fine-tuned in the self-supervised instruction tuning stage are used as the initial parameters for the projector in the second tuning stage. For the evaluation of most baselines, we utilize their publicly available code. We employ a grid-search strategy based on their default hyperparameter settings to ensure a comprehensive evaluation. For further implementation details, please refer to our released source code.

4.2 Overall Performance Comparison (RQ1)

We conduct experiments on the node classification task, evaluating both supervised and zero-shot scenarios. The overall performance is presented in Table 1. **Supervised Task Settings**: We train the models on a specific dataset and evaluated their performance on the corresponding test set (*e.g.*, training on Arxiv-Arxiv and testing on the Arxiv test set). **Zero-Shot Task Settings**: We train the models on a specific dataset and test them on other datasets without any additional training (*e.g.*, training on Arxiv-PubMed and testing on the PubMed dataset). To account for variations in the number of classes across different datasets, we employed a classifier trained with transfer data, typically a linear layer, when testing GNN-based models. In Table 1, "-7B-" represents the parameter scale, while "-v1.1-" and "-v1.5-" indicate different versions of the base Vicuna model. "-stage2" indicates that only the second stage tuning is adopted. "-std" and "-cot" denote the use of the standard and generated COT instruction datasets, respectively.

Obs.1: Overall Superiority of our GraphGPT. Our graph LLM consistently outperforms various state-of-the-art baselines in both supervised and zero-shot scenarios. Notably, even recently developed strong GNN-based models, such as NodeFormer, DIFFormer, and GKD, exhibit good structural modeling capabilities in the supervised setting. However, when transferred to new datasets without further training, their performance significantly declines. In contrast, our GraphGPT not only surpasses all state-of-the-art methods in supervised tasks but also achieves a remarkable 2-10 times increase in accuracy in the zero-shot graph learning scenario.

Additionally, LLM-based solutions, like Baichuan-7B and Vicuna-7B maintain stable performance across different datasets. However, they are limited to making predictions solely based on text information. In contrast, our GraphGPT effectively preserves graph structured information, offers a more comprehensive solution for graph learning tasks. These improvements can be attributed to two key factors: i) Through our dual-stage graph instruction tuning, our method aligns the graph tokens, which contain rich structural information encoded by the graph encoder, with the natural language tokens. This alignment allows the LLM to retain and understand the inherent structural characteristics of the graph data. ii) Our framework facilitates mutual enhancement between the graph encoder and LLM. The introduction of graph tokens fills the gap in the LLM's structural understanding, enabling it to incorporate and reason about the graph's structural information.

Obs.2: Benefits with Structure-aware Graph Matching. The presence of the first stage, which involves self-supervised graph matching tasks for instruction tuning, plays a crucial role in enhancing the zero-shot transferability of our GraphGPT. The first stage focuses on aligning the graph tokens, which encode rich structural information, with the language tokens. This alignment enables the model to develop a deeper understanding of the inherent structural characteristics of the graph data. Without the first stage, if we only conduct the second stage of task-specific instruction tuning, the model tends to be more prone to overfitting on the specific dataset. In such cases, the model's performance may be heavily reliant on dataset-specific patterns and characteristics, rather than a genuine understanding of the underlying graph structure. This can limit the model's ability to generalize to new, unseen datasets.

Obs.3: Benefits with COT Distillation. The "-std" and "-cot" variants indicate that the use of COT distillation substantially benefits

Table 2: Performance comparison of various instruction mixtures in supervised learning on the Arxiv dataset and the zero-shot setting on the Cora dataset for node classification.

Dataset	Supervision. on Arxiv		Zero Shot on Cora	
Model	Acc	Macro-F1	Acc	Macro-F1
MLP	0.5179	0.2536	0.0220	0.0006
GraphSAGE	0.5480	0.3290	0.0132	0.0029
GCN	0.5267	0.3202	0.0187	0.0032
GAT	0.5332	0.3118	0.0161	0.0057
RvGNN	0.5474	0.3240	0.0217	0.0016
DGI	0.5059	0.2787	0.0205	0.0011
GKD	0.5570	0.1595	0.0406	0.0037
GLNN	0.6088	0.3757	0.0182	0.0092
NodeFormer	0.5922	0.3328	0.0144	0.0053
DIFFormer	0.5986	0.3355	0.0100	0.0007
baichuan-7b	0.0946	0.0363	0.0405	0.0469
vicuna-7B-v1.1	0.2657	0.1375	0.1090	0.0970
vicuna-7B-v1.5	0.4962	0.1853	0.1489	0.1213
Arxiv-std + PubMed-std	0.6390	0.2652	0.1501	0.0936
Arxiv-cot + PubMed-cot	0.6476	0.2854	0.1647	0.1326
Arxiv-mix + PubMed-mix	0.6139	0.2772	0.1544	0.1048
Arxiv-std + PubMed-std + Link	0.5931	0.2238	**0.1847**	**0.1579**
Arxiv-mix + Pubmed-mix + Link	**0.6874**	**0.3761**	0.1836	0.1494

more complex graph learning tasks. Models tuned with the standard instruction dataset can already achieve prominent results when transferred to simpler tasks, such as the PubMed dataset with 3 classes, with an accuracy of 0.7011 for Arxiv-PubMed. However, their performance tends to be mediocre when applied to complex tasks like the Cora dataset with 70 classes. By leveraging the powerful reasoning capabilities of the closed-source model (GPT-3.5) through COT distillation, our model can integrate this knowledge and significantly enhance its performance on complex graph tasks.

4.3 Generalization Ability Investigation (RQ2)

In this subsection, we explore the generalization ability of our model by incorporating more instruction data to fine-tune the LLM for effectively handling various types of tasks. Our main results and experimental observations are presented as follows:

More Data Boost Model Transfer Ability. In our preliminary investigation, we examine the influence of data quantity on the transfer capability of our GraphGPT, as illustrated in the "(Arxiv + PubMed)-Cora" column of Table 1. In this experiment, we train models using a combination of the Arxiv and PubMed datasets and perform zero-shot testing on the Cora dataset. The results reveal that by incorporating a relatively smaller PubMed dataset (with 20,000+ items) alongside Arxiv, our GraphGPT exhibits a significant improvement in transfer performance on Cora. In contrast, the transfer performance of GNN-based models, trained separately on Arxiv and PubMed, actually deteriorates.

More Data Yet No Forgetting. We further validate the performance of the combined Arxiv and PubMed instruction data on the original Arxiv data, as demonstrated in the "(Arxiv + PubMed)-Arxiv" column in Table 1. The results indicate that most traditional GNN-based approaches experience a significant decline in performance on Arxiv after iterative training. In contrast, our model exhibits improved performance. We attribute this phenomenon to the occurrence of catastrophic forgetting in GNN-based models, where the structural modeling competence of the model trained solely on the smaller PubMed dataset is compromised. However, our model effectively mitigates this issue through our unified graph instruction tuning paradigm. This enables our model to maintain and even enhance its performance by retaining the generalized graph structure patterns despite incorporating additional data.

Table 3: Module ablation study under both supervised and zero-shot settings to analyze the individual contributions.

Dataset	Arxiv-Arxiv		Arxiv-PubMed		Arxiv-Cora	
Variant	Acc	Mac-F1	Acc	Mac-F1	Acc	Mac-F1
w/o GS	0.4962	0.1853	0.6351	0.5231	0.1489	0.1213
w/o LR	0.5807	0.2462	0.2523	0.1925	0.0050	0.0016
ours	**0.6258**	**0.2622**	**0.7011**	**0.6491**	**0.1813**	**0.1272**

Generalization for Multitasking Graph Learner. Recent studies on instruction tuning suggest that mixing different instruction tuning data can further enhance the performance of Language and Logic Models (LLMs). In this study, we ensure a consistent number of instruction entries and mix different types of instruction data, including standard instruction ("-std"), COT instruction ("-cot"), a blend of standard (50%) and COT (50%) instruction ("-mix"), and link prediction instruction ("Link"). The results are presented in Tables 2 and Table 7 in Appendix. We can observe that effective data mixture solutions can significantly improve the performance of our GraphGPT under various settings. The addition of task-specific instruction for link prediction task notably enhances the performance of our model in node classification. Interestingly, after incorporating node classification, the performance of link prediction also exceeds that of the selected best-performing existing models. After mixing the instructions of different tasks, our model demonstrates the ability to effectively handle various graph learning tasks and transfer its knowledge to other unseen datasets.

4.4 Module Ablation Study (RQ3)

We conduct an ablation study to investigate the individual contributions of different sub-modules of our proposed framework, and the results are reported in Table 4. The observations are as follows:

Effect of Graph Instruction Tuning. In our study, we investigate the benefit of incorporating graph structural information into LLM using the variant "w/o GS." In this variant, we directly adopt the base LLM (specifically, Vicuna-7B-v1.5) to perform node classification on three datasets, without incorporating graph structural information. The results of our study demonstrate that our model significantly outperforms the base model that lacks structural information. This indicates that our graph instruction tuning paradigm enables the LLM to understand the graph structural information more effectively. Importantly, this improvement in performance was achieved without altering the original parameters of the LLM. Instead, it was solely accomplished through our lightweight alignment projector, which aligns graph tokens and natural language tokens through the 1-linear projection operation.

Effect of LLM-enhanced Semantic Reasoning. We conduct further investigations to assess the influence of the LLM's reasoning ability in our GraphGPT by performing supervised and zero-shot predictions using only the default graph encoders. This variant is denoted as "w/o LR". The results of our study indicate that our GraphGPT, which integrates the LLM, significantly enhances the performance of graph encoder, especially in the zero-shot setting.

Table 4: Study on the time and space efficiency of our GraphGPT during both the training and inference stages.

Variants	Training Time	Tuned Parameters	GPU Occupy
Stage-1-tune	OOM	6,607,884,288	OOM
Stage-1-freeze	22:53:33	131,612,672	39517.75
improvement	-	$\downarrow \times 50.21$	-
Stage-2-tune	OOM	6,607,884,288	OOM
Stage-2-freeze	03:44:35	131,612,672	38961.75
improvement	-	$\downarrow \times 50.21$	-

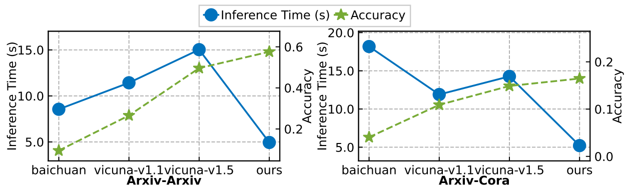


Figure 5: Inference efficiency study of our GraphGPT.

This suggests that the rich semantic information injected by the LLM provides a substantial gain in performance.

4.5 Model Efficiency Study (RQ4)

The study aims to assess the computational efficiency of our model during both the model training and inference stages.

Training Efficiency with Graph Instruction Tuning. Our instruction tuning framework follows a two-stage process where the parameters of both the LLM and the graph encoder are frozen, and only the graph-text alignment projector is tuned. We conduct a comparison between freezing and tuning the LLM parameters in a 4-card 40G Nvidia A100 environment, denoted by "-freeze" and "-tune" respectively. The study analyze the time and space efficiency in terms of training time, the number of tuned parameters, and GPU occupancy (MiB per GPU). Under the same experimental conditions, when tuning LLM parameters, we encounter Out of Memory (OOM) errors even with a batch size of 1. However, by utilizing our tuning strategy, the training process remains stable even with a batch size of 2. Moreover, the number of tuned parameters decreases by more than 50 times compared to the freezing stage.

Model Inference Efficiency. In our exploration, we assess the inference speed and accuracy of our GraphGPT by comparing it with baichuan-7B, vicuna-7B-v1.1, and vicuna-7B-v1.5 LLMs. Using a single 40G Nvidia A100, we measure inference time (seconds per response) on the Arxiv and Cora COT instruction datasets, as shown in Figure 5. Our graph LLM demonstrates superior efficiency and accuracy. Lower inference time doesn't necessarily mean better performance: baichuan-7B yields quick but often incorrect or irrelevant answers, while vicuna-7B-v1.1 and vicuna-7B-v1.5 require longer, complex reasoning steps for better answers. In contrast, our model achieves accurate predictions through a brief reasoning process, enhancing inference efficiency.

4.6 Model Case Study (RQ5)

We conduct a detailed analysis of our model's performance in downstream graph learning tasks and compare it to traditional LLMs using different types of instructions. We prompt both ChatGPT and our GraphGPT with Arxiv data, using only node content (title and abstract), node content with text-based graph structure, and our designed graph instruction. The results, presented in Table 5 and 10 in the appendix, clearly demonstrate that ChatGPT, despite its massive parameter count (over 200B), struggles to make accurate predictions based solely on node text information or node content with the text-based graph structure. This difficulty is particularly pronounced when dealing with papers that have a high degree of cross-disciplinary characteristics, as illustrated in the example of machine learning and hardware architecture. In contrast, our GraphGPT consistently provides accurate predictions and offers reasonable explanations. This is because our GraphGPT accepts a subgraph structure with 103 nodes, allowing it to extract rich structural information from the neighboring nodes' citation relationships, enabling accurate predictions.

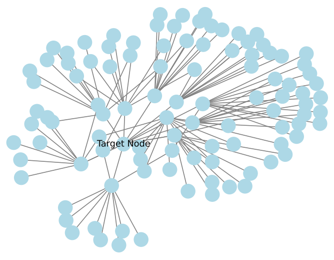
Furthermore, we believe that our approach of using graph tokens to represent the graph structure as input to the LLM is more efficient than the natural language solution. In the case of a 103-node subgraph, our GraphGPT only requires 750 tokens to be fed into the LLM, while the text-based method requires 4649 tokens. This significant reduction in token consumption translates to a substantial decrease in training and inference resource requirements.

5 RELATED WORK

Self-supervised Learning and Pre-training on Graphs. To enhance the robustness of graph models, self-supervised learning (SSL) has been introduced as a powerful technique [14, 16, 24]. It allows GNNs to learn meaningful graph representations without heavily relying on labeled data. The core idea behind self-supervised learning in graph models is to design pretext tasks that leverage the graph's intrinsic properties to generate additional supervision signals [51]. SSL-enhanced graph learning methods can be broadly classified into two main paradigms: contrastive SSL and generative SSL. In particular, i) **Contrastive SSL** focuses on learning representations by contrasting positive and negative samples. Notable methods in this domain include GraphCL [59] and GCA [67]. Recent advancements in contrastive SSL include automated contrastive augmentation (*i.e.*, JOAO [58], AutoGCL [57]), Hyperbolic-Euclidean due space contrasting (*e.g.*, DSGC [55]), or community-aware contrastive learning (*e.g.*, gCooL [20]). ii) **Generative SSL**, on the other hand, focuses on generating realistic samples that resemble the original graph structures. Recent advancements in this line include GraphMAE [11, 12] for feature masking, and S2GAE [35], AutoCF [52] for reconstructing masked edges as SSL tasks.

Prompt-Tuning for Graph Neural Networks. Recent efforts in enhancing the generalization capability of graph neural networks (GNNs) have focused on training GNN models in a self-supervised manner, followed by fine-tuning on specific downstream tasks using prompt-tuning techniques [64]. For example, GPPT [33] is a transfer learning paradigm, where GNNs are first pre-trained on masked edge prediction and then prompted with token pairs for downstream node classification. GraphPrompt [26] aims to handle downstream tasks by integrating pre-training and downstream tasks into a unified task template. Additionally, Sun *et al.* [34] presents a unified prompt format, reformulates tasks to the graph level, and incorporates meta-learning techniques to improve multi-task

Table 5: Comparison of prediction results between our GraphGPT and ChatGPT.

Title: TiM-DNN: Ternary in-Memory accelerator for Deep Neural Networks. (with **102** first-order and second-order neighbors)
Ground-Truth Category: cs.LG, Machine Learning
ChatGPT (Node Content Only): cs.AR, cs.AI, cs.SY, cs.ET, cs.NE. The paper presents a hardware accelerator for deep neural networks (DNNs) that is specifically designed to execute ternary DNNs, which represent weights and activations by signed ternary values. Therefore, the most likely category for this paper is cs.AR (Hardware Architecture), which focuses on the design and analysis of hardware and systems for AI and machine learning applications. The paper also involves the optimization of compute and storage requirements of DNNs, which falls under the category of cs.AI (Artificial Intelligence). Additionally, the paper proposes a programmable, in-memory accelerator, which is a type of embedded system, and therefore, it could be categorized under cs.SY (Systems and Control). Furthermore, the paper discusses the use of ternary values for DNNs, which is an emerging technique to optimize the compute and storage requirements of DNNs, and therefore, it could be categorized under cs.ET (Emerging Technologies). Lastly, the paper proposes a new bit-cell called Ternary Processing Cells (TPCs), which could be categorized under cs.NE (Neural and Evolutionary Computing).
ChatGPT (Node Content and Text-based Graph Structure): Based on the title "TiM-DNN: Ternary In-Memory Accelerator for Deep Neural Networks", the paper is likely to belong to the following arXiv CS sub-categories: 1. cs.AR (Hardware Architecture): The paper proposes a hardware accelerator, TiM-DNN, specifically designed for executing ternary deep neural networks. This falls under the domain of hardware architecture research. 2. cs.NE (Neural and Evolutionary Computing): The paper focuses on accelerating deep neural networks, which is a topic within the field of neural and evolutionary computing. 3. cs.CV (Computer Vision and Pattern Recognition): Deep neural networks are widely used in computer vision tasks, and the TiM-DNN accelerator could potentially be applied to accelerate computer vision algorithms. 4. cs.SY (Systems and Control): The TiM-DNN accelerator is a system-level solution for accelerating deep neural networks, so it could be categorized under systems and control research. 5. cs.DC (Distributed, Parallel, and Cluster Computing): While the paper does not explicitly mention distributed or parallel computing, the TiM-DNN accelerator is designed to optimize the compute and storage requirements of deep neural networks, which are often executed on distributed or parallel computing systems.
GraphGPT: Based on the title and abstract of the paper, we can identify the following arXiv CS sub-categories that are most likely to be relevant:  1. cs.LG (Machine Learning): This sub-category is relevant because the paper discusses the use of ternary DNNs for language modeling and image classification, which are both tasks that involve learning. 2. cs.AI (Artificial Intelligence): This sub-category is relevant because the paper proposes a new accelerator for DNNs, which is designed to optimize the compute and storage requirements of complex DNNs. 3. cs.NE (Numerical Analysis): This sub-category is relevant because the paper discusses the use of ternary DNNs, which involve signed ternary values, and the use of TiM tiles, which are specialized memory arrays that perform massively parallel signed vector-matrix multiplications on ternary values. 4. cs.SY (Systems and Control Theory): This sub-category is relevant because the paper discusses the implementation of TiM-DNN in 32nm technology using an architectural simulator calibrated with SPICE simulations and RTL synthesis. 5. cs.AR (Hardware Architecture): This sub-category is relevant because the paper proposes a new accelerator for DNNs, which is designed to be programmable and in-memory.

performance in graph prompting. Despite these advances, these methods still require further fine-tuning that relies on supervision labels from downstream tasks to ensure accurate learning. In contrast, this work addresses this limitation by introducing a foundational graph model that tackles the more challenging task of zero-shot graph learning. By eliminating the need for label inputs from downstream tasks, this approach allows for a more general and flexible graph learning paradigm in real-world scenarios.

large Language Models. In recent years, LLMs (*e.g.*, ChatGPT [29] and Claude [1]) have gained widespread attention for their remarkable capabilities in various NLP tasks [18, 46]. Based on these unique capabilities of LLMs, many tuning-free prompting techniques have been explored to enhance their generative abilities, such as in-context learning [28] and Chain-of-Thought [47, 56]. With the rise of open-source LLMs, such as Llama [36, 37], ChatGLM [62], and Baichuan [53], technologies for aligning pre-trained LLMs to different specific tasks and human feedback have been proposed, making private LLMs in specific domains possible [19, 44, 45].

While there have been successful attempts to align LLMs with visual information, such as multimodal LLMs [23, 66], the alignment of LLMs with graph structures remains largely unexplored. This research addresses this gap by introducing a dual-stage graph instruction tuning paradigm that effectively aligns the language capacity of LLMs with graph learning. Previous studies [2, 6] have attempted to incorporate graph information into LLMs using natural language, but they have faced challenges in handling complex graph structures and achieving a deep understanding of graphs due to the limitations of relying solely on text-based prompts.

6 CONCLUSION

This work presents an effective and scalable graph large language model, aims at improving the generalization capabilities of graph models. The proposed framework, GraphGPT, injects graph domain-specific structural knowledge into the LLM through a dual-stage graph instruction tuning paradigm. By leveraging a simple yet effective graph-text alignment projector, we enable LLMs to comprehend and interpret the structural components of graphs. Extensive evaluations across different settings demonstrate the effectiveness of our method in both supervised and zero-shot graph learning scenarios. Furthermore, the model exhibits strong generalization abilities, allowing it to handle diverse downstream datasets and tasks without suffering from catastrophic forgetting. A potential avenue for future investigation is exploring pruning techniques to compress redundant or less important parameters of LLM, thereby reducing the overall model size while preserving its performance.

REFERENCES

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *CoRR* abs/2212.08073 (2022).

[2] Zhikai Chen, Haitao Mao, Hang Li, et al. 2023. Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs. *CoRR* abs/2307.03393 (2023).

[3] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, et al. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[5] Yushun Dong, Ninghao Liu, Brian Jalaian, et al. 2022. EDITS: Modeling and Mitigating Data Bias for Graph Neural Networks. In *WWW*. ACM, 1259–1269.

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

[7] Zhichun Guo, Kehan Guo, Bozhao Nan, Yijun Tian, Roshni G. Iyer, et al. 2023. Graph-based Molecular Representation Learning. In *IJCAI*. 6638–6646.

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[9] Xiaoxin He, Xavier Bresson, et al. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *CoRR* abs/2305.19523 (2023).

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. ACM, 639–648.

[11] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, et al. 2023. GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner. In *WWW*. 737–746.

[12] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Jie Tang, et al. 2022. Graphmae: Self-supervised masked graph autoencoders. In *KDD*. 594–604.

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, et al. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *NeurIPS*.

[14] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *KDD*. 1857–1867.

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. ACM / IW3C2, 2704–2710.

[16] Baoyu Jing, Chanyoung Park, and Hanghang Tong. 2021. Hdmi: High-order deep multiplex infomax. In *WWW*. 2414–2424.

[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.

[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *NeurIPS*.

[19] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, et al. 2023. RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *CoRR* abs/2309.00267 (2023).

[20] Bolian Li, Baoyu Jing, and Hanghang Tong. 2022. Graph communal contrastive learning. In *WWW*. 1203–1213.

[21] Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training Graph Neural Networks with 1000 Layers. In *ICML*. 6437–6449.

[22] Mingkai Lin, Wenzhong Li, Ding Li, Yizhou Chen, and Sanglu Lu. 2022. Resource-Efficient Training for Large Graph Convolutional Networks with Label-Centric Cumulative Sampling. In *WWW*. ACM, 1170–1180.

[23] Haotian Liu, Chunyuan Li, et al. 2023. Visual Instruction Tuning.

[24] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *TKDE* 35, 6 (2022), 5879–5900.

[25] Yunchao Liu, Yu Wang, Oanh Vu, Rocco Moretti, et al. 2023. Interpretable Chirality-Aware Graph Neural Network for Quantitative Structure Activity Relationship Modeling in Drug Discovery. In *AAAI*. 14356–14364.

[26] Zemin Liu, Xingtong Yu, et al. 2023. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*. 417–428.

[27] Xiaojun Ma, Qin Chen, et al. 2022. Meta-Weight Graph Neural Network: Push the Limits Beyond Global Homophily. In *WWW*. ACM, 1270–1280.

[28] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?. In *EMNLP*. 11048–11064.

[29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning (ICML)*. PMLR, 8748–8763.

[31] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. In *KDD*. ACM, 1567–1577.

[32] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling Reasoning Capabilities into Smaller Language Models. In *ACL*. 7059–7073.

[33] Mingchen Sun, Kaixiong Zhou, et al. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *KDD*. 1717–1727.

[34] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks. In *KDD*.

[35] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. 2023. S2GAE: Self-Supervised Graph Autoencoders are Generalizable Learners with Graph Masking. In *WSDM*. 787–795.

[36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023).

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023).

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. 2017. Attention is all you need. In *NeurIPS*, Vol. 30.

[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, et al. 2018. Graph Attention Networks. In *ICLR (Poster)*. OpenReview.net.

[40] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, et al. 2019. Deep Graph Infomax. In *ICLR (Poster)*. OpenReview.net.

[41] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft Academic Graph: When experts are not enough. *Quant. Sci. Stud.* 1, 1 (2020), 396–413.

[42] Xiang Wang, Tinglin Huang, Dingxian Wang, et al. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *WWW*. 878–887.

[43] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, et al. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.

[44] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, et al. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. *CoRR* abs/2306.04751 (2023).

[45] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *ACL*. 13484–13508.

[46] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Jeff Dean, William Fedus, et al. 2022. Emergent Abilities of Large Language Models. *Trans. Mach. Learn. Res.* 2022 (2022).

[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*.

[48] Zhihao Wen and Yuan Fang. 2023. Augmenting Low-Resource Text Classification with Graph-Grounded Pre-training and Prompting. In *SIGIR*.

[49] Qitian Wu, Chenxiao Yang, et al. 2023. DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion. In *ICLR*.

[50] Qitian Wu, Wentao Zhao, et al. 2023. NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification. *CoRR* abs/2306.08385 (2023).

[51] Jun Xia, Lirong Wu, Jintao Chen, et al. 2022. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *WWW*. 1070–1079.

[52] Lianghao Xia, Chao Huang, Tao Yu, Ben Kao, et al. 2023. Automated Self-Supervised Learning for Recommendation. In *WWW*. 992–1002.

[53] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, et al. 2023. Baichuan 2: Open Large-scale Language Models. *CoRR* abs/2309.10305 (2023).

[54] Chenxiao Yang, Qitian Wu, and Junchi Yan. 2022. Geometric Knowledge Distillation: Topology Compression for Graph Neural Networks. In *NeurIPS*.

[55] Haoran Yang, Hongxu Chen, Shirui Pan, Lin Li, Philip S Yu, and Guandong Xu. 2022. Dual space graph contrastive learning. In *WWW*. 1238–1247.

[56] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *CoRR* abs/2305.10601 (2023).

[57] Yihang Yin, Qingzhong Wang, et al. 2022. Autogcl: Automated graph contrastive learning via learnable view generators. In *AAAI*, Vol. 36. 8892–8900.

[58] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *ICML*. PMLR, 12121–12132.

[59] Yuning You, Tianlong Chen, Yongduo Sui, et al. 2020. Graph contrastive learning with augmentations. In *NeurIPS*, Vol. 33. 5812–5823.

[60] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2019. Graph Transformer Networks. In *NeurIPS*. 11960–11970.

[61] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. In *NeurIPS*, Vol. 32.

[62] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, et al. 2023. GLM-130B: An Open Bilingual Pre-trained Model. In *ICLR*.

[63] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2022. Graph-less Neural Networks: Teaching Old MLPs New Tricks Via Distillation. In *ICLR*.

[64] Wen Zhang, Yushan Zhu, Mingyang Chen, Yuxia Geng, Yufeng Huang, Yajing Xu, Wenting Song, and Huajun Chen. 2023. Structure Pretraining and Prompt Tuning for Knowledge Graph Transfer. In *WWW*. 2581–2590.

[65] Yanfu Zhang et al. 2022. Robust Self-Supervised Structural Graph Neural Network for Social Network Prediction. In *WWW*. ACM, 1352–1361.

[66] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models. *arXiv preprint arXiv:2304.10592* (2023).

[67] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *WWW*. 2069–2080.

A APPENDIX

In the appendix, we provide a summary of the statistical information of our datasets in Section A.1. Furthermore, in Section A.2, we offer detailed descriptions of the baseline methods used for comparison in our experiments. To facilitate the understanding of the instruction templates used for different tasks and datasets, we present them in Section A.4. These templates serve as guidelines for formulating instructions that are specific to the task and dataset at hand. In Section A.3, we present the implementation details of the text-graph grounding process. Finally, in Section A.5, we provide supplementary experimental results.

A.1 Statistical information of datasets

The detailed statistical information of datasets is shown in Table 6.

Table 6: Statistical information of datasets.

Dataset	# Nodes	# Edges	# Classes
ogb-arxiv	169,343	1,166,243	40
PubMed	19,717	44,338	3
Cora	25,120	182,280	70

Table 7: Performance comparison of various instruction mixtures for link prediction on the PubMed dataset.

Dataset	PubMed	
Model	AUC	AP
MLP	0.5583	0.5833
GAT	0.5606	0.6373
GraphSAGE	0.5041	0.5813
RevGNN	0.4538	0.5083
Node2Vec	0.6535	0.6885
w/o Link	0.5010	0.5005
only Link	0.6704	0.6087
Arxiv-std + PubMed-std + Link	**0.8246**	**0.8026**
Arxiv-mix + PubMed-mix + Link	0.6451	0.5886

Table 8: Instruction template for the graph matching task.

Graph Matching
Input: Given a sequence of graph tokens `<graph>` that constitute a subgraph of a citation graph, where the first token represents the central node of the subgraph, and the remaining nodes represent the first and second order neighbors of the central node. Each graph token contains the title and abstract information of the paper at this node. Here is a list of paper titles: `{1. Paper 1. 2. Paper 2. ... n. Paper n}`, please reorder the list of papers according to the order of graph tokens (i.e., complete the matching of graph tokens and papers).
Output: Based on the given graph tokens and the list of paper titles, we obtain the matching of graph tokens and papers as follows: `{Graph token 1 corresponds to Paper Q, Graph token 2 corresponds to Paper P, ... Graph token n corresponds to Paper R}`.

A.2 Detailed Descriptions of Baselines

For a comprehensive performance comparison, our baseline set consists of the following methods which are presented below:

1) Conventional Deep Learning Methods

- **MLP**: This method employs a multilayer perceptron to independently encode different node representations.

2) Graph Neural Encoders

- **GraphSAGE** [8]: It is a framework designed for inductive representation learning on large graphs, enabling the efficient generation of node embeddings for previously unseen data.
- **GCN** [17]: This method extends the solution of convolutional neural networks to model graph-structured features.
- **GAT** [39]: This method introduces graph attention networks that leverage masked self-attentional layers to address the limitations of GCN. It achieves this by differentiating the message passing among different nodes with weighted information aggregation.
- **RevGNN** [21]: This work investigates several techniques, including reversible connections, group convolutions, weight tying, and equilibrium models, to enhance the memory and parameter efficiency of graph neural networks (GNNs).

3) Self-Supervised Graph Learning Approaches

- **DGI** [40]: It maximizes the mutual information between patch representations and corresponding high-level summaries of graphs. By doing so, DGI enables the model to capture meaningful information from local patches of the graph and leverage it to learn informative node representations in an unsupervised manner.

4) Graph Knowledge Distillation Frameworks

- **GKD** [54]: This method distills knowledge from a teacher GNN trained on a complete graph to a student GNN operating on a smaller or sparser graph. It achieves this by utilizing the Neural Heat Kernel (NHK) method, which encapsulates the geometric property of the underlying manifold.
- **GLNN** [63]: This work combines the advantages of graph neural networks and MLPs using knowledge distillation, with the aim of removing the dependency on the inference graph.

5) Graph Transformer Networks

- **NodeFormer** [50]: This work leverages a kernelized Gumbel-Softmax operator to reduce the computational complexity of the graph transformer model, while also enabling the learning of latent graph structures from large graphs.
- **DIFFormer** [49]: This work introduces an energy-constrained diffusion-based graph transformer. This transformer is designed to encode a batch of instances into evolutionary states by incorporating information from other instances.

6) Large Language Models

- **Baichuan-7B** [53]: It is an open-source, large-scale pre-trained model with 7 billion parameters. It is specifically designed to be bilingual, supporting both Chinese and English languages. This model has been trained extensively on a diverse range of data.
- **vicuna-7B-v1.1** [3]: It is an open-source chatbot that has undergone fine-tuning using user-shared conversations collected from ShareGPT. The base model used for this fine-tuning is Llama-1, a reference to a specific model architecture.
- **vicuna-7B-v1.5** [3]: This model is an enhanced iteration of vicuna-7b-v1.1, building upon Llama-2 as its base model.

Table 9: Instruction template for node classification and link prediction on different datasets.

Node Classification (Arxiv in the COT manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Which arXiv CS sub-category does this paper belong to? Give 5 likely arXiv CS sub-categories as a comma-separated list ordered from most to least likely, in the form "cs.XX". Please think about the categorization in a step by step manner and avoid making false associations. Then provide your reasoning.
Output: Based on the information, `{reasoning process and answers}`.
Arxiv in the standard manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, and other nodes are its one-hop or multi-hop neighbors, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Which arXiv CS sub-category does this paper belong to? Give the most likely arXiv CS sub-categories of this paper directly, in the form "cs.XX" with full name of the category.
Output: `{the ground-truth answer}`.
Node Classification (PubMed in the COT manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Does the paper involve any cases of Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes? Please give one or more answers of either Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes; if multiple options apply, provide a comma-separated list ordered from most to least related. Please think about the categorization in a step by step manner and avoid making false associations. Then provide your reasoning for each choice.
Output: Based on the information, `{reasoning process and answers}`.
Node Classification (PubMed in the standard manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, and other nodes are its one-hop or multi-hop neighbors, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Which case of Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes does this paper involve? Please give one answer of either Type 1 diabetes, Type 2 diabetes, or Experimentally induced diabetes directly.
Output: `{the ground-truth answer}`.
Node Classification (Cora in the COT manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Which of the following subcategories of computer science does this paper belong to: `{1. Categories 1. 2. Categories 2. ... n. Categories n}`? Give 5 likely categories as a comma-separated list ordered from most to least likely. Please think about the categorization in a step by step manner and avoid making false associations. Then provide your reasoning for each choice.
Output: Based on the information, `{reasoning process and answers}`.
Node Classification (Cora in the standard manner)
Input: Given a citation graph: `<graph>` where the `0`th node is the target paper, with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. Question: Which of the following subcategories of computer science does this paper belong to: `{1. Categories 1. 2. Categories 2. ... n. Categories n}`? Directly give the full name of the most likely category of this paper.
Output: `{the ground-truth answer}`.
Link Prediction (PubMed)
Input: Given a sequence of graph tokens: `<graph>` that constitute a subgraph of a citation graph, where the first token represents the central node of the subgraph, and the remaining nodes represent the first and second order neighbors of the central node. The information of the central node is as follow: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. The other sequence of graph tokens: `<graph>`, where the first token (the central node) with the following information: Abstract: `{abstract of the central node}`. Title: `{title of the central node}`. If the connections between nodes represent the citation relationships between papers, are these two central nodes connected? Give me a direct answer of "yes" or "no".
Output: `{the ground-truth answer}`.

A.3 Details of Text-Graph Grounding

For the implementation details of Text-Graph Grounding 3.1, we follow previous work [48], aligning text features and graph features across three dimensions. Mathematically, these dimensions can be represented by three sets of transformation equations:

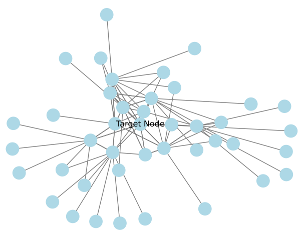
$$g_1^{(1)}(\hat{\mathrm{H}}) = \{\hat{\mathrm{H}}_i, 1 \le i \le N\}, g_1^{(2)}(\hat{\mathrm{T}}) = \{\hat{\mathrm{T}}_i, 1 \le i \le N\}$$
$$g_2^{(1)}(\hat{\mathrm{H}}) = \{\hat{\mathrm{H}}_i, 1 \le i \le N\}, g_2^{(2)}(\hat{\mathrm{T}}) = \{\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\hat{\mathrm{T}}_j, 1 \le i \le N\}$$
$$g_3^{(1)}(\hat{\mathrm{H}}) = \{\hat{\mathrm{T}}_i, 1 \le i \le N\}, g_3^{(2)}(\hat{\mathrm{T}}) = \{\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\hat{\mathrm{T}}_j, 1 \le i \le N\} \tag{5}$$

where $\hat{\mathrm{H}} \in \mathbb{R}^{N\times d}$ denotes graph representations and $\hat{\mathrm{T}} \in \mathbb{R}^{N\times d}$ indicates text representations, and N is the number of nodes.

A.4 Instruction Templates across Tasks/Data

We provide instruction templates for different tasks (graph matching, node classification, and link prediction) on various datasets in our experiment. The templates are shown in Table 8 and 9. While the same template is used for paper citation graphs, we design specific templates for each data in the downstream tasks, including both

Table 10: Comparison of our GraphGPT's predictions with ChatGPT on a arxiv paper (Continued).

Title: Greening Cloud-Enabled Big Data Storage Forensics: Syncany as a Case Study. (with **41** first-order and second-order neighbors)
Ground-Truth Category: cs.CR, Cryptography and Security
ChatGPT (Node Content Only): Based on the title and abstract of the paper, we can make an educated guess about the likely arXiv CS sub-categories that this paper belongs to. Here is a step-by-step breakdown of the reasoning: 1. The paper mentions "cloud-enabled big data storage solutions" as the focus of investigation. This suggests that the paper is related to cloud computing and big data. 2. The paper also mentions "cyber forensics" and "investigation process" in the context of cloud-enabled big data platforms. This indicates that the paper is specifically about the forensic analysis of data stored in cloud-enabled big data platforms. Based on these observations, here are five likely arXiv CS sub-categories for this paper: 1. cs.DC (Distributed, Parallel, and Cluster Computing) - This sub-category covers topics related to distributed computing systems, which includes cloud computing. 2. cs.DB (Databases) - This sub-category covers topics related to database systems, which are often used in big data storage solutions. 3. cs.CR (Cryptography and Security) - This sub-category covers topics related to cyber security, which includes forensic analysis of digital evidence. 4. cs.SE (Software Engineering) - This sub-category covers topics related to software development and engineering, which may include the development of cloud-enabled big data storage solutions. 5. cs.CY (Computers and Society) - This sub-category covers topics related to the impact of computing on society, which may include the ethical and legal implications of forensic analysis in cloud-enabled big data storage.
ChatGPT (Node Content and Text-based Graph Structure): Based on the title of the paper "Greening Cloud Enabled Big Data Storage Forensics: Syncany as a Case Study" we can make an initial assumption that the paper belongs to the "Information Retrieval" (cs.IR) sub-category. However, to determine the most likely arXiv CS sub-categories, we need to analyze the content and context of the paper. 1. cs.DB (Databases): The paper focuses on cloud-enabled big data storage, which is closely related to databases. It discusses the identification, collection, analysis, preservation, and archiving of digital evidences in the context of Syncany private cloud storage service. This suggests that the paper may belong to the cs.DB sub-category. 2. cs.CR (Cryptography and Security): The paper addresses the forensic investigation of cloud-enabled big data storage platforms, which involves analyzing and recovering residual artifacts of criminal activities. This indicates a strong connection to the field of cybersecurity and digital forensics, which falls under the cs.CR sub-category. 3. cs.SE (Software Engineering): The paper may also belong to the cs.SE sub-category as it discusses the investigation process and the challenges involved in analyzing cloud-enabled big data storage platforms. It explores techniques and methodologies for locating and documenting residual artifacts of forensic value, which are relevant to the field of software engineering. 4. cs.CY (Computers and Society): The paper touches upon the societal impact of cloud-enabled big data storage and the need for efficient forensic investigation techniques. It highlights the potential reduction in time and resources spent on real-world investigations, which aligns with the concerns and issues addressed in the cs.CY sub-category. 5. cs.DC (Distributed, Parallel, and Cluster Computing): The paper mentions the decentralization of digital evidence acquisition in the context of BitTorrent Sync. This suggests a connection to distributed computing and parallel processing, which are topics covered in the cs.DC sub-category. Based on the above analysis, the likely arXiv CS sub-categories for the paper, ordered from most to least likely, are: cs.DB, cs.CR, cs.SE, cs.CY, cs.DC.
GraphGPT: Based on the title and abstract of the paper, we can infer that the paper is related to cyber forensics and cloud-enabled big data storage.  We can then proceed to categorize it based on the sub-categories of arXiv CS. 1. cs.CR: This category is for cryptography research, which involves the study of cryptographic algorithms, protocols, and systems. While the paper does not explicitly mention cryptography, it does involve the analysis of data remnants on a cloud-enabled big data platform, which could potentially involve cryptographic techniques. 2. cs.DS: This category is for database systems, which involves the design, implementation, and management of databases. The paper mentions the use of a popular storage engine for big data platforms, which suggests that the paper could be related to database systems. 3. cs.IR: This category is for information retrieval, which involves the study of how to effectively retrieve and organize information from large collections of data. The paper's focus on identifying and documenting residual artifacts of forensic value on a cloud-enabled big data platform suggests that it could be related to information retrieval. 4. cs.IT: This category is for information technology, which involves the study of the design, development, and use of information technology systems. The paper's focus on cloud-enabled big data storage and forensics suggests that it could be related to information technology. 5. cs.SY: This category is for systems and networking, which involves the study of computer systems and networks. The paper's focus on the analysis of data remnants on a cloud-enabled big data platform suggests that it could be related to systems and networking.

standard and COT (Chain-of-Thought) instructions. The `<graph>` token serves as an indicator, and `...` denotes fillable blanks.

A.5 Supplementary Experimental Results

In this subsection, we analyze the link prediction performance using different instruction mixtures. We then compare the prediction results of our GraphGPT and ChatGPT under different instruction formats, including node content only, node content with text-based graph structure, and our custom-designed graph instruction.

A.5.1 **Performance of Link Prediction.** Four different instruction mixtures are compared: i) COT instruction data for node classification only (w/o Link) ii) Instruction data for link prediction only (only Link) iii) A mix of standard instruction data for node classification and instruction data for link prediction (Arxiv-std + PubMed-std + Link) iv) A mix of 50% standard instruction for node classification, 50% COT instruction data for node classification, and instruction for link prediction (Arxiv-mix + PubMed-mix + Link). When compared to representative baselines, as presented in Table 7, the combination of standard instruction data for node classification and instruction for link prediction significantly outperformed the baselines. Furthermore, as mentioned earlier in Table 2, this particular variant outperformed state-of-the-art approaches in node classification. This improvement can be attributed to the task of link prediction, which enhances our GraphGPT's understanding of graph structural information and facilitates node classification tuning. Importantly, the inclusion of node classification instructions does not impede the model's comprehension of link prediction.

A.5.2 **Comparison Between ChatGPT and GraphGPT.** Table 10 shows more comparison cases between ChatGPT and GraphGPT.