Can LLMs Alleviate Catastrophic Forgetting in Graph Continual Learning? A Systematic Study Appendix

Ziyang Cheng[1,3] **, Zhixun Li**[1] **, Yuhan Li**[2]**, Yixin Song**[3]**, Kangyi Zhao**[4]**,**
Dawei Cheng[3,5]**, Jia Li**[2]**, Jeffrey Xu Yu**[1]
[1]The Chinese University of Hong Kong
[2]Hong Kong University of Science and Technology (Guangzhou) [3]Tongji University
[4]University of Pittsburgh [5]Shanghai Artificial Intelligence Laboratory

Github Repository: https://github.com/ZhixunLEE/LLM4GCL

A Additional Details on `LLM4GCL`

A.1 Discussion on Local Testing and Global Testing

In this section, we will first introduce the pipeline of TPP [16], followed by a comprehensive analysis of the two evaluation paradigms: *local testing* and *global testing*, supported by empirical results.

A.1.1 Approaches

TPP[1]. TPP follows the *local testing* manner and proposes to use a Laplacian smoothing approach to generate a prototypical embedding for each graph task for task ID prediction. As for the i-th task with graph data $\mathcal{G}_{s_i} = \mathcal{G}_{q_i} = (\mathbf{A}_{s_i}, \mathbf{X}_{s_i}, \mathcal{V}^l_{s_i}, \mathcal{V}^u_{s_i})$, TPP constructed a task prototype $\mathbf{p}_{s_i}$ based on the train set $\{\mathbf{x}_j \mid v_j \in \mathcal{V}^l_{s_i}\}$. Specifically, given training graph $\mathcal{G}_{s_i}$, the Laplacian smoothing is first applied to obtain the node embeddings $\mathbf{Z}_{s_i}$:

$$\mathbf{Z}_{s_i} = (\mathbf{I} - (\hat{\mathbf{D}}_{s_i})^{-\frac{1}{2}}\hat{\mathbf{L}}_{s_i}(\hat{\mathbf{D}}_{s_i})^{-\frac{1}{2}})^k \mathbf{X}_{s_i} \tag{1}$$

where k is the smoothing steps, $\mathbf{I}$ is an identity matrix, $\hat{\mathbf{L}}_{s_i}$ is the graph Laplacian matrix of $\hat{\mathbf{A}}_{s_i}$, and $\hat{\mathbf{D}}_{s_i}$ is the diagonal degree matrix. Consequently, the training task prototype $\mathbf{p}_{s_i}$ can be generated as:

$$\mathbf{p}_{s_i} = \frac{1}{|\mathcal{V}^l_{s_i}|} \sum_{v_j \in \mathcal{V}^l_{s_i}} \mathbf{z}_{s_i,j} (\hat{\mathbf{D}}_{s_i,jj})^{-\frac{1}{2}} \tag{2}$$

Thus, all the task prototypes can be separately constructed and stored as $\mathcal{P} = \{\mathbf{p}_{s_1}, \mathbf{p}_{s_2}, ..., \mathbf{p}_{s_n}\}$. Similarly, the task prototype of the local testing on $\mathcal{G}_{q_i} = \mathcal{G}_{s_i}$ can be denoted as:

$$\mathbf{p}_{q_i} = \frac{1}{|\mathcal{V}^u_{q_i}|} \sum_{v_j \in \mathcal{V}^u_{q_i}} \mathbf{z}_{q_i,j} (\hat{\mathbf{D}}_{q_i,jj})^{-\frac{1}{2}} \tag{3}$$

Then, during local testing, the task ID prediction is conducted between the train and test prototypes:

$$t_{q_i} = \arg\min(d(\mathbf{p}_{q_i}, \mathbf{p}_{s_1}), d(\mathbf{p}_{q_i}, \mathbf{p}_{s_2}), ..., d(\mathbf{p}_{q_i}, \mathbf{p}_{s_n})) \tag{4}$$

[1]In this work, we focus solely on formalizing the pipeline of TPP; for rigorous mathematical derivations and proofs, we direct readers to the original literature.

Table 1: Performance evaluation of TPP against two ablation models: Mean Pooling and MLP. AA and AF are the average classification accuracy and forgetting ratio across all sessions, respectively.

Methods	Cora		Citeseer		WikiCS		Photo		Products		Arxiv-23		Arxiv	
	AA	AF	AA	AF	AA	AF	AA	AF	AA	AF	AA	AF	AA	AF
TPP	95.2	-0.0	87.9	-0.0	93.6	-0.0	91.3	-0.0	89.6	-0.0	91.6	-0.0	85.7	-0.0
Mean Pooling	95.2	-0.0	87.9	-0.0	93.6	-0.0	91.3	-0.0	89.6	-0.0	91.6	-0.0	85.7	-0.0
MLP	90.3	-0.0	86.9	-0.0	89.2	-0.0	79.0	-0.0	83.1	-0.0	88.8	-0.0	81.6	-0.0

After precisely identifying the task ID, TPP subsequently incorporates a graph prompt learning framework that assigns distinct prompt vectors to adapt a pretrained GNN for diverse tasks:

$$\overline{\mathbf{x}}_{q_i,j} = \mathbf{x}_{q_i,j} + \sum_{m}^{M} \alpha_m \phi_{i,m}, \alpha_m = \frac{e^{(\mathbf{w}_m)^T \mathbf{x}_{q_i,j}}}{\sum_l^M e^{(\mathbf{w}_l)^T \mathbf{x}_{q_i,j}}} \tag{5}$$

where $\Phi_i = [\phi_{i,1}, \phi_{i,2}, ..., \phi_{i,M}]^T \in \mathbb{R}^{M \times F}$ represents the learnable graph prompt for the i-th task, with M denoting the number of vector-based tokens ϕ_i. α_m is the importance score of token $\phi_{i,m}$ in the prompt, while $\mathbf{w}_j$ corresponds to a learnable projection weight. The final prompted graph representation $\mathcal{G}_{q_i} = \{\mathbf{A}_{q_i}, \mathbf{X}_{q_i} + \Phi_i\}$ is then processed by a frozen pre-trained GNN model $f(\cdot, \cdot)$ to generate predictions:

$$\hat{\mathcal{Y}}_{q_i} = \varphi_i(f(\mathbf{A}_{q_i}, \mathbf{X}_{q_i} + \Phi_i)) \tag{6}$$

where φ_i denotes the task-specific MLP layer for the i-th task. Crucially, **TPP effectively transforms the class-incremental learning problem into a task-incremental learning paradigm, decomposing the overall continual learning objective into multiple independent sub-tasks, thereby significantly mitigating the learning difficulty**.

Mean Pooling. TPP proposes that when applying a sufficiently large number of Laplacian smoothing iterations k, the distance between training and test prototypes from the same task approaches zero, while maintaining substantial separation between prototypes belonging to different CIL tasks. This discriminative property ensures reliable prediction accuracy. To evaluate the effectiveness of this technique, we implement an ablation model called Mean Pooling, where the Laplacian smoothing operation is replaced by a basic mean pooling aggregation, formally defined as:

$$\mathbf{p}_{s_i} = \frac{1}{|\mathcal{V}_{s_i}^l|} \sum_{v_j \in \mathcal{V}_{s_i}^l} \mathbf{x}_{s_i,j}, \;\; \mathbf{p}_{q_i} = \frac{1}{|\mathcal{V}_{q_i}^u|} \sum_{v_j \in \mathcal{V}_{q_i}^u} \mathbf{x}_{q_i,j} \tag{7}$$

Under this formulation, the distance between training and test prototypes depends solely on the feature distributions of the training and testing nodes.

MLP. By decomposing the CIL task into independent subtasks, TPP eliminates the need for session-wide classification across all accumulated classes. Instead, the model can focus exclusively on individual subtasks, significantly reducing task complexity. Building upon this observation, we propose an additional ablation model employing a basic two-layer MLP for classification:

$$\hat{\mathcal{Y}}_{q_i} = \varphi_i(\mathbf{X}_{q_i}) \tag{8}$$

A.1.2 Experimental Results

To investigate the impact of Laplacian smoothing and assess subtask complexity, we evaluate the performance of TPP against two variants, Mean Pooling and MLP, with results presented in Table 1. We employ the *local testing* paradigm, where model performance is quantified using two key metrics: Average Accuracy (AA) and Average Forgetting (AF). These metrics are defined as follows:

$$\text{AA} = \frac{1}{n} \sum_{j=1}^{n} (\mathcal{A}_{n,j}), \;\; \text{AF} = \frac{1}{n} \sum_{j=1}^{n-1} (\mathcal{A}_{n,j} - \mathcal{A}_{j,j}) \tag{9}$$

where n denotes the total number of sessions, and $\mathcal{A}_{i,j}$ represents the model's classification accuracy for task $\mathcal{T}_{q_j}$ after training on task $\mathcal{T}_{s_i}$. The experimental results yield two key conclusions:

Table 2: Statistics and experimental settings for all datasets in LLM4GCL. The symbol '*' indicates adjusted statistics that differ from the original dataset.

	Dataset	Cora	Citeseer	WikiCS	Photo	Products	Arxiv-23	Arxiv
Attributes	# Nodes	2,708	3,186	11,701	48,362	53,994*	46,196*	169,343
	# Edges	5,429	4,277	215,863	500,928	72,242*	78,542*	1,166,243
	# Features	1,433	3,703	300	768	100	300	128
	# Classes	7	6	10	12	31*	37*	40
	Avg. # Token	183.4	210.0	629.9	201.5	152.9*	237.7*	239.8
	Domain	Citation	Citation	Web link	E-Commerce	E-Commerce	Citation	Citation
NCIL	# Classes per session	2	2	3	3	4	4	4
	# Sessions	3	3	3	4	8	9	10
	# Shots per class	100	100	200	400	400	400	800
FSNCIL	# Base Classes	3	2	4	4	11	13	12
	# Novel Classes	4	4	6	8	20	24	28
	# Ways per session	2	2	3	4	4	4	4
	# Sessions	3	3	3	3	6	7	8
	# Shots per base class	100	100	200	400	400	400	800
	# Shots per novel class	5	5	5	5	5	5	5

♣ **Laplacian smoothing demonstrates limited necessity for accurate task identification.** Comparative analysis between TPP and Mean Pooling demonstrates that both approaches achieve accurate task ID prediction. This phenomenon can be explained by the *local testing* setup, where training and testing nodes are drawn from the same graph ($\mathcal{G}_{s_i} = \mathcal{G}_{q_i}$). In this configuration, the feature distribution similarity between training and testing nodes within each class provides sufficient discriminative information for reliable task identification. These results indicate that the *local testing* paradigm is **inherently susceptible to task ID leakage**.

♠ **Decomposition into subtasks significantly reduces task complexity.** As evidenced by the experimental results, the MLP baseline achieves comparable performance to TPP across Citeseer, WikiCS, Arxiv-23, and Arxiv datasets, with merely a 3.08% average performance gap. This finding demonstrates that task decomposition effectively reduces learning difficulty, enabling simpler architectures to attain competitive performance. However, this characteristic **may compromise the evaluation of models' true continual learning capabilities**, as the simplified subtasks fail to fully capture the challenges inherent in the original CIL problem.

A.2 Details of Datasets

All of the public datasets used in `LLM4GCL` were previously published, covering a multitude of domains. For each dataset, we store the graph-type data in the .pt format using PyTorch. This includes shallow embeddings, raw text of nodes, edge indices, node labels, and label names. The statistics of these datasets are presented in Table 2, and the detailed descriptions are listed in the following:

- **Cora** [3]. The Cora dataset is a citation network comprising research papers and their citation relationships within the computer science domain. The raw text data for the Cora dataset was sourced from the GitHub repository provided in Chen et al.[2]. In this dataset, each node represents a research paper, and the raw text feature associated with each node includes the title and abstract of the respective paper. An edge in the Cora dataset signifies a citation relationship between two papers. The label assigned to each node corresponds to the category of the paper.
- **Citeseer** [3]. The Citeseer dataset is a citation network comprising research papers and their citation relationships within the computer science domain. The TAG version of the dataset contains text attributes for 3,186 nodes, and the raw text data for the Citeseer dataset was sourced from the GitHub repository provided in Chen et al.. Each node represents a research paper, and each edge signifies a citation relationship between two papers.
- **WikiCS** [11]. The WikiCS dataset is an internet link network where each node represents a Wikipedia page, and each edge represents a reference link between pages. The raw text data for

[2] `https://github.com/CurryTang/Graph-LLM`

the WikiCS dataset was collected from OFA[3]. The raw text associated with each node includes the name and content of a Wikipedia entry. Each node's label corresponds to the category of the entry.

- **Ele-Photo (Photo)** [23]. The Ele-Photo dataset[4] is derived from the Amazon Electronics dataset [15]. The nodes in the dataset represent electronics-related products, and edges between two products indicate frequent co-purchases or co-views. The label for each dataset corresponds to the three-level label of the electronics products. User reviews on the item serve as its text attribute. In cases where items have multiple reviews, the review with the highest number of votes is utilized. For items lacking highly-voted reviews, a user review is randomly chosen as the text attribute.
- **Ogbn-products (Products)** [5]. The OGBN-Products dataset is characterized by its large scale, originally containing approximately 2 million nodes and 61 million edges. Following the node sampling strategy proposed in TAPE[5], we utilize a subset consisting of about 54k nodes and 72k edges for experimental use. In this dataset, each node represents a product sold on Amazon, and edges between two products indicate frequent co-purchasing relationships. The raw text data for each node includes product titles, descriptions, and/or reviews.
- **Arxiv-23** [5]. The Arxiv-23 dataset is proposed in TAPE, representing a directed citation network comprising computer science arXiv papers published in 2023 or later. Similar to Ogbn-arxiv, each node in this dataset corresponds to an arXiv paper, and directed edges indicate citation relationships between papers. The raw text data for each node includes the title and abstract of the respective paper. The label assigned to each node represents one of the 40 subject areas of arXiv CS papers, such as `cs.AI`, `cs.LG`, and `cs.OS`. These subject areas are manually annotated by the paper's authors and arXiv moderators.
- **Ogbn-arxiv (Arxiv)** [11]. The Ogbn-arxiv dataset is a citation network comprising papers and their citation relationships, collected from the arXiv platform. The raw text data for the dataset was sourced from the GitHub repository provided in OFA. The original raw texts are available here[6]. Each node represents a research paper, and each edge denotes a citation relationship.

A.3 Details of Baseline Models

In `LLM4GCL`, we integrate 15 baseline methods for NCIL and FSNCIL tasks, including 6 GNN-based methods, 4 LLM-based methods, and 5 GLM-based methods. We provide detailed descriptions of these methods used in our benchmark as follows.

- **GNN-based methods**.
  - **GCN** [8]. The first deep learning model that leverages graph convolutional layers. In `LLM4GCL`, we employ a two-layer GCN followed by one MLP layer. We use the code available at `https://github.com/pyg-team/pytorch_geometric`.
  - **EWC** [9]. A regularization-based CL method that prevents catastrophic forgetting through quadratic penalties on key parameter deviations. Parameter importance is quantified via the Fisher information matrix, maintaining performance on learned tasks during new task acquisition. We use the code available at `https://github.com/QueuQ/CGLB`.
  - **LwF** [10]. LwF minimizes the discrepancy between the logits of the old model and the new model through knowledge distillation to preserve knowledge from the old tasks. We use the code available at `https://github.com/QueuQ/CGLB`.
  - **Cosine**. Cosine first trains a GNN on the initial task session, subsequently freezes the network parameters, and leverages a training-free prototype mechanism to generate task-specific prototypes, ultimately employing cosine similarity for classification.
  - **TPP** [16]. A replay-and-forget-free GCL approach that employs Laplacian smoothing-based task identification to precisely predict task IDs and incorporates graph prompt engineering to adaptively transform the GNN into a series of task-specific classification modules. We use the code available at `https://github.com/mala-lab/TPP`.
  - **TEEN** [20]. TEEN is a training-free prototype calibration strategy that addresses the issue of new classes being misclassified into base classes by leveraging the semantic similarity between

[3] `https://github.com/LechengKong/OneForAll`
[4] `https://github.com/sktsherlock/TAG-Benchmark`
[5] `https://github.com/XiaoxinHe/TAPE`
[6] `https://snap.stanford.edu/ogb/data/misc/ogbn_arxiv`

base and new classes, which is captured by a feature extractor pre-trained on base classes. We use the code available at `https://github.com/wangkiw/TEEN`.

- **LLM-based methods**.
  - **BERT** [1, 4]. A Transformer-based model family that employs masked language modeling and next sentence prediction pretraining to learn bidirectional contextual representations. It achieves state-of-the-art performance across multiple NLP tasks with minimal task-specific modifications. We use the code available at `https://github.com/google-research/bert` and `https://github.com/prajjwal1/generalize_lm_nli`.
  - **RoBERTa** [12]. An optimized BERT variant family that enhances pretraining through dynamic masking, larger batches, and extended data. It removes the next sentence prediction objective while maintaining MLM, achieving superior performance across NLP benchmarks with more efficient training. We use the code available at `https://github.com/facebookresearch/fairseq/tree/main/examples/xlmr`.
  - **LLaMA** [19]. A foundational large language model family featuring architectural optimizations like RMSNorm and SwiGLU activations. Trained on trillions of tokens from diverse public corpora, it achieves remarkable few-shot performance while maintaining efficient inference compared to similarly-sized models. We use the code available at `https://github.com/meta-llama/llama-cookbook`.
  - **SimpleCIL** [26]. A computationally efficient approach that freezes the PTM's embedding function and derives class prototypes by averaging embeddings for classification, which harnesses the model's inherent generalizability for effective knowledge transfer without fine-tuning. We use the code available at `https://github.com/LAMDA-CL/RevisitingCIL`.
- **GLM-based methods**.
  - **$\mathbf{GCN_{Emb}}$**. A simple method enhances GNNs by replacing their shallow embeddings with deep LLM-generated embeddings, where node text descriptions are encoded using LLaMA-8B to improve representation quality.
  - **ENGINE** [27]. A framework adds a lightweight and tunable G-Ladder module to each layer of the LLM, which uses a message-passing mechanism to integrate structural information. This enables the output of each LLM layer (i.e., token-level representations) to be passed to the corresponding G-Ladder, where the node representations are enhanced and then used for node classification. We use the code available at `https://github.com/ZhuYun97/ENGINE`.
  - **GraphPropmter** [13]. A framework integrating graph structures and LLMs via adaptive prompts, projecting GNN-based structural embeddings through MLP to align with the LLM's semantic space for joint topology-text processing. We use the code available at `https://github.com/franciscoliu/graphprompter`.
  - **GraphGPT** [17]. A framework that initially aligns the graph encoder with natural language semantics through text-graph grounding, and then combines the trained graph encoder with the LLM using a projector, through which the model can directly complete graph tasks with natural language, thus performing zero-shot transferability. We use the code available at `https://github.com/WxxShirley/LLMNodeBed`.
  - **LLaGA** [2]. LLaGA utilizes node-level templates to transform graph data into structured sequences, then maps them into token embeddings, enabling LLMs to process graph data with improved versatility and generalizability. We use the code available at `https://github.com/WxxShirley/LLMNodeBed`.

B Details on Implementations

B.1 Hyper-parameters

- For GCN, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256]
layer_num = 2, dropout = 0.5
```

- For EWC, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256],
strength in [1, 100, 10000]
layer_num = 2, dropout = 0.5
```

- For LwF, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256],
strength in [1, 100, 10000], lambda in [0.1, 1.0], T in [0.2, 2.0]
layer_num = 2, dropout = 0.5
```

- For Cosine, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256]
layer_num = 2, dropout = 0.5, T = 1.0, sample_num = 100
```

- For TPP, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256],
pe in [0.2, 0.3], pf in [0.2, 0.3]
layer_num = 2, dropout = 0.5,
pretrain_batch_size = 500, pretrain_lr = 1e-3
```

- For TEEN, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256]
layer_num = 2, dropout = 0.5
T = 1.0, sample_num = 100, softmax_T = 16, shift_weight = 0.5
```

- For BERT and RoBERTa, we grid-search the hyperparameters

```
lr in [1e-4, 2e-4, 5e-4], batch_size in [5, 10, 20]
min_lr = 5e-6, weight_decay = 5e-2,
dropout = 0.1, att_drouput = 0.1, max_length = 256
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05
```

- For LLaMA, we grid-search the hyperparameters

```
lr in [1e-4, 2e-4, 5e-4], batch_size in [5, 10, 20]
min_lr = 5e-6, weight_decay = 5e-2,
dropout = 0.1, att_drouput = 0.1, max_length = 512
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05
```

- For SimpleCIL, we grid-search the hyperparameters

```
lr in [1e-4, 2e-4, 5e-4], batch_size in [5, 10, 20]
min_lr = 5e-6, weight_decay = 5e-2,
dropout = 0.1, att_drouput = 0.1, max_length = 256
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05
T = 1.0, sample_num = 20
```

- For $\mathrm{GCN}_{\mathrm{Emb}}$, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256]
layer_num = 2, dropout = 0.5
```

- For ENGINE, we grid-search the hyperparameters

```
lr in [1e-5, 1e-4, 1e-3], hidden_dim in [64, 128, 256], r in [1, 32]
layer_num = 1, dropout = 0.5, T = 0.1,
layer_select = [0, 6, 12, 18, 24, -1]
```

- For GraphPrompter, we follow the instructions from the paper [13], use

```
lr = 2e-4, batch_size = 10, min_lr = 5e-6, weight_decay = 5e-2,
dropout = 0.1, att_drouput = 0.1, max_length = 512
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05
gnn = 'GCN', layer_num = 4, hidden_dim = 1024, output_dim = 1024,
dropout = 0.5, proj_hidden_dim = 1024
```

- For GraphGPT, we follow the instructions from the paper [22], use

```
dropout = 0.1, att_drouput = 0.1, max_length = 512
s1_k_hop = 2, s1_num_neighbors = 5, s1_max_txt_length = 512,
s1_epoch = 2, s1_batch_size = 5, s1_lr = 1e-4
s2_num_neighbors = 4, max_txt_length = 512,
s2_epoch = 10, s2_batch_size = 5, s2_lr = 1e-4
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05
```

- For LLaGA, we follow the instructions from the paper [2], use

```
lr = 2e-4, batch_size = 10, min_lr = 5e-6, weight_decay = 5e-2,
dropout = 0.1, att_drouput = 0.1, max_length = 512
llm_freeze = 'True', neighbor_template = 'ND', nd_mean = True,
k_hop = 2, sample_size = 10, hop_field = 4, proj_layer = 2
```

- For SimGCL, we follow the instructions from the paper [21], use

```
lr = 2e-4, batch_size = 10, min_lr = 5e-6, weight_decay = 5e-2,
lora_r = 5, lora_alpha = 16, lora_dropout = 0.05,
T = 1.0, sample_num = 50, hop = [20, 20],
include_label = False, max_node_text_len = 128
```

B.2 Backbone Selection

Table 3: Selection of GNN and LLM Backbones in `LLM4GCL`, $\mathbf{A}, \mathbf{X}, \mathcal{R}$ refers to the adjacent matrix, feature matrix, and raw text attributes of the input TAG, respectively.

Type	Method	Predictor	Input	Output Format	GNN Backbone	LLM Backbone
GNN-based	GCN [8]	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	GCN	-
	EWC [9]	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	GCN	-
	LwF [10]	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	GCN	-
	Cosine	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	GCN	-
	TPP [16]	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	SGC, GCN	-
	TEEN [20]	GNN	$(\mathbf{A}, \mathbf{X})$	Logits	GCN	-
LLM-based	BERT [4]	LLM	$\mathcal{R}$	Logits	-	BERT-base (110M)
	RoBERTa [12]	LLM	$\mathcal{R}$	Logits	-	RoBERTa-large (355M)
	LLaMA [19]	LLM	$\mathcal{R}$	Prediction Texts	-	LLaMA3-8B
	SimpleCIL [26]	LLM	$\mathcal{R}$	Logits	-	RoBERTa-large (355M)
GLM-based	GCN_{Emb}	GNN	$(\mathbf{A}, \mathcal{R})$	Logits	GCN	LLaMA3-8B
	ENGINE [27]	GNN	$(\mathbf{A}, \mathcal{R})$	Logits	GCN	LLaMA3-8B
	GraphPrompter [7]	LLM	$(\mathbf{A}, \mathbf{X}, \mathcal{R})$	Prediction Texts	GCN	LLaMA3-8B
	GraphGPT [17]	LLM	$(\mathbf{A}, \mathbf{X}, \mathcal{R})$	Prediction Texts	-	LLaMA3-8B
	LLaGA [2]	LLM	$(\mathbf{A}, \mathbf{X}, \mathcal{R})$	Prediction Texts	-	LLaMA3-8B

To compare the various baseline methods in `LLM4GCL` as fairly as possible, we try to utilize the same GNN and LLM backbones in our implementations. Table 3 shows the GNN and LLM backbones used in the original implementations, as well as those implemented in `LLM4GCL`.

C Text Prompt Design

For GraphPrompter [13], GraphGPT [17], and LLaGA [2], we utilize the prompt templates provided in their original papers for several datasets, including Cora and arXiv. For datasets not originally addressed, such as Photo, we adapt their prompt designs to create similarly formatted prompts. For SimGCL, we adopt the graph prompt template established by Wang et al. [21]. Below is a summary of these prompt templates using Cora as an example, where ⟨**labels**⟩ denotes the dataset-specific label space, ⟨**graph**⟩ represents the tokenized graph context, ⟨**raw_text**⟩ and ⟨**paper_titles**⟩ refer to the node's original raw text, and ⟨**target node**⟩ is the node's id within the graph. It is important to emphasize that during the continual learning process, as new classes emerge sequentially, the content of ⟨**labels**⟩ dynamically expands to incorporate these newly introduced class labels.

Illustration of Prompts Utilized by LLaMA and GraphPrompter on the Cora Dataset

Cora: ⟨**raw_text**⟩. Which of the following sub-categories of AI does this paper belong to? Here are the |⟨**labels**⟩| categories: ⟨**labels**⟩. Reply with only one category that you think this paper might belong to. Only reply to the category phrase without any other explanatory words.

Illustration of Prompts Utilized by GraphGPT on the Cora Dataset

Matching: 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 or 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: ⟨**paper_titles**⟩. Please reorder the list of papers according to the order of graph tokens (i.e., complete the matching of graph tokens and papers).

Instruction: Given a citation graph: ⟨**graph**⟩ where the 0th node is the target paper, with the following information: ⟨**raw_text**⟩. Question: Which of the following specific research does this paper belong to: ⟨**labels**⟩. Directly give the full name of the most likely category of this paper.

Illustration of Prompts Utilized by LLaGA on the Cora Dataset

System: You are a helpful language and graph assistant. You can understand the graph content provided by the user and assist with the node classification task by outputting the most likely label.

Instruction: Given a node-centered graph: ⟨**graph**⟩, each node represents a paper, we need to classify the center node into |⟨**labels**⟩| classes: ⟨**labels**⟩, please tell me which class the center node belongs to?

Illustration of Prompts Utilized by SimGCL on the Cora Dataset

System: You are a good graph reasoner. Given a graph description from the Cora dataset, understand the structure and answer the question.

Instruction: ⟨**target node**⟩ ⟨**raw_text**⟩, known neighbor papers at hop 1: ⟨**1-hop neighbors**⟩ ⟨**raw_texts**⟩, known neighbor papers at hop 2: ⟨**2-hop neighbors**⟩ ⟨**raw_texts**⟩. **Question**: Please predict which of the following sub-categories of AI this paper belongs to. Choose from the following categories: ⟨**labels**⟩.

Table 4: Performance comparison of `LLM4GCL` baselines in NCIL scenario across different class sizes (W) and session numbers (S) on Arxiv. The **best**, second, and third results are highlighted.

Methods	8W5S		5W8S		4W10S		2W20S		Avg.
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	Rank
GCN	24.5	2.0	20.5	1.1	21.2	1.5	13.4	0.7	12.5
EWC	31.6	19.6	26.2	13.8	24.9	13.8	15.4	4.2	7.6
LwF	29.2	8.6	22.6	2.7	18.9	1.6	14.2	1.0	10.2
Cosine	35.6	23.2	38.3	27.9	37.4	27.8	41.6	31.4	3.1
BERT	36.7	15.0	29.7	10.2	26.0	9.7	16.6	5.0	5.1
RoBERTa	33.9	3.9	26.7	2.1	24.4	1.4	15.6	0.3	10.1
LLaMA	35.0	3.8	27.6	1.5	24.9	1.4	15.9	0.3	9.6
SimpleCIL	46.1	**31.4**	49.7	**35.9**	50.6	**36.5**	52.6	**39.1**	**1.5**
$\text{GCN}_{\text{LLMEmb}}$	33.7	3.8	26.6	2.0	24.3	1.4	15.6	0.3	10.9
ENGINE	34.1	3.7	26.5	2.0	24.6	1.3	15.7	0.2	11.2
GraphPrompter	35.1	2.9	27.8	2.0	24.8	1.4	16.8	0.4	8.9
GraphGPT	38.9	7.4	35.0	6.2	32.2	3.9	23.5	3.1	5.1
LLaGA	34.8	8.6	27.9	5.3	26.1	1.3	16.6	0.9	7.6
SimGCL (Ours)	**51.6**	28.7	**53.0**	30.4	**59.9**	33.8	**57.4**	17.5	1.6

Table 5: Performance comparison of GNN-, LLM-, and GLM-based methods in the NCIL scenario of `LLM4GCL`. The **best**, second, and third results are highlighted.

Methods	Cora		Citeseer		WikiCS		Photo		Products		Arxiv-23		Arxiv		Avg.
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	Rank
GCN	57.0	38.2	52.4	30.2	54.9	34.9	46.5	19.9	25.5	5.4	19.9	2.9	21.2	1.5	10.5
EWC	56.0	31.0	45.9	28.5	55.3	33.0	46.9	20.5	29.4	15.9	25.7	15.0	24.9	13.8	8.7
LwF	55.7	30.8	47.8	28.2	55.0	34.0	45.8	20.4	26.0	7.5	21.1	6.8	18.9	1.6	11.1
Cosine	65.4	45.2	50.7	31.0	66.5	53.5	63.6	49.6	36.1	16.1	36.1	24.2	37.4	27.8	4.3
TPP	45.7	13.7	45.4	9.6	35.1	9.8	36.3	5.7	15.0	0.0	12.2	0.3	19.6	3.4	15.3
BERT	56.0	29.9	53.8	28.7	58.4	30.0	43.4	18.4	26.9	4.1	27.1	5.1	26.0	9.7	9.9
RoBERTa	54.6	29.6	54.1	28.6	55.1	30.8	43.5	19.1	27.6	3.2	23.1	0.8	24.4	1.4	11.7
LLaMA	65.6	53.8	55.7	31.7	55.5	30.9	44.6	19.1	29.8	0.4	24.3	1.0	24.9	1.4	8.7
SimpleCIL	70.8	58.3	66.4	49.5	71.4	57.3	62.1	52.5	66.8	52.6	**52.4**	**38.8**	50.6	**36.5**	2.2
$\text{GCN}_{\text{LLMEmb}}$	59.1	31.1	53.6	30.4	53.4	27.5	47.7	21.0	26.9	0.1	22.9	0.8	24.3	1.4	11.0
ENGINE	59.2	31.3	53.5	29.8	56.4	30.1	47.9	21.0	27.2	1.1	22.5	0.7	24.6	1.3	10.2
GraphPrompter	61.9	46.8	60.2	30.6	59.6	38.3	51.5	31.0	29.0	0.8	23.4	0.9	24.8	1.4	7.5
GraphGPT	55.5	31.6	60.0	30.1	62.0	49.2	50.8	30.2	35.5	3.2	30.7	5.8	32.2	3.9	6.6
LLaGA	58.2	30.2	51.3	27.8	53.7	27.6	47.2	20.7	25.7	0.2	26.9	4.1	26.1	1.3	11.3
SimGCL-1 Hops	**89.4**	**83.9**	76.4	64.2	32.4	20.3	44.7	33.5	70.9	43.7	28.2	6.1	38.2	10.7	5.5
SimGCL-2 Hops	84.6	80.0	**77.1**	**66.3**	**73.5**	**61.9**	**82.1**	**72.6**	**71.1**	**60.2**	38.7	13.6	**59.9**	33.8	**1.5**

D Additional Experimental Results

D.1 Session Numbers

Table 4 presents the performance of baseline models and SimGCL on the Arxiv dataset under different session-class configurations. Consistent with **Obs. ❸**, as session numbers increase, the performance advantages of BERT, GraphGPT, and LLaGA over GNN-based approaches diminish significantly, demonstrating pronounced catastrophic forgetting in long-session scenarios.

D.2 Extended Analysis of SimGCL

To comprehensively assess SimGCL's performance, we conducted an evaluation using 1-hop neighborhood aggregation, with detailed results presented in Table 5 and Table 6.

Table 6: Performance comparison of GNN-, LLM-, and GLM-based methods in the FSNCIL scenario of `LLM4GCL`. The **best**, second, and third results are highlighted.

Methods	Cora		Citeseer		WikiCS		Photo		Products		Arxiv-23		Arxiv		Avg.
	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	$\bar{\mathcal{A}}$	$\mathcal{A}_N$	Rank
GCN	68.0	38.1	39.5	17.4	62.4	47.4	58.5	32.3	36.0	14.1	22.2	5.4	25.3	2.0	10.4
EWC	59.0	36.3	49.2	21.2	58.4	40.4	62.0	28.4	45.7	31.5	36.4	29.5	31.5	18.3	8.7
LwF	63.3	43.5	45.1	20.7	59.4	41.0	60.1	29.4	50.3	38.7	30.8	15.1	27.8	3.8	8.4
Cosine	72.6	57.8	49.1	25.7	68.0	50.7	67.9	50.5	50.9	33.6	40.1	27.9	27.2	17.2	5.2
TEEN	60.9	40.3	59.0	39.5	59.3	42.4	59.3	35.5	49.6	28.4	39.2	27.1	31.8	18.6	6.8
TPP	39.0	9.1	37.3	12.6	37.3	12.4	40.0	14.0	14.0	0.0	7.3	0.0	11.1	6.0	16.2
BERT	56.4	34.7	61.1	38.8	61.7	33.0	47.7	25.8	22.4	3.8	15.2	3.4	14.9	6.0	12.1
RoBERTa	59.6	41.9	54.0	29.2	67.2	42.3	58.2	29.6	38.8	6.9	22.6	1.4	25.7	1.3	10.4
LLaMA	72.6	55.6	75.9	55.5	65.2	43.9	61.4	32.3	43.5	13.7	23.5	1.5	22.7	1.4	7.8
SimpleCIL	69.6	53.6	64.1	49.9	**73.2**	63.1	66.3	53.0	65.6	53.6	**49.8**	**40.0**	**46.4**	**36.6**	2.6
$\text{GCN}_{\text{LLMEmb}}$	68.2	40.1	54.3	28.7	54.7	31.0	66.0	34.2	30.6	0.2	21.4	1.0	22.1	0.9	11.2
ENGINE	52.2	28.3	47.6	25.5	46.8	23.7	48.0	21.5	20.9	0.1	17.4	0.5	17.3	1.4	15.0
GraphPrompter	63.2	37.3	65.3	34.5	69.5	51.6	69.7	51.0	37.9	2.2	27.4	2.3	27.3	1.0	7.8
GraphGPT	62.4	39.6	65.0	41.7	71.2	61.6	62.2	38.9	43.2	16.2	25.4	1.7	24.3	2.0	7.5
LLaGA	62.0	39.6	52.2	28.0	49.4	30.4	48.8	18.0	28.0	0.2	18.2	0.9	16.6	1.6	13.3
SimGCL-1 Hop	**85.5**	**77.9**	77.3	63.5	30.4	23.2	50.6	44.8	57.2	30.1	19.9	4.2	25.9	3.9	7.4
SimGCL-2 Hops	78.0	67.6	**78.0**	**63.8**	68.8	**64.1**	**81.2**	**71.3**	**69.7**	**62.7**	31.8	10.3	36.3	6.8	**2.4**

Observation ❾ Rich structural information better enhances task comprehension. In Table 5 and Table 6, SimGCL exhibits performance degradation when limited to 1-hop neighborhood information, particularly on densely-connected datasets, WikiCS and Photo, where it underperforms even the vanilla GCN baseline. We hypothesize this stems from an inadequate structural context, misleading the model's representation learning. Notably, modest improvements occur on sparser graph datasets, Cora and Citeseer, we attribute this to their smaller scale and lower connectivity makes 1-hop information sufficient for effective prediction.

D.3 Visualization

To have deeper insights into the comparative learning dynamics, we present performance evolution curves in Figure 1. The visualization demonstrates SimGCL's consistent superiority over baseline models across multiple training sessions, underscoring its robust capability to maintain previously acquired knowledge while effectively assimilating new information.

E Broader Discussion

E.1 Limitations

Restricted to node-level tasks. Currently, `LLM4GCL` is limited to GCL for node classification tasks, as most GLM-based methods are designed for single-task scenarios, and node classification remains the predominant task in GML. However, GML encompasses other critical tasks such as link prediction and graph classification. Moreover, in practical applications, GCL has significant value in these task domains, like predicting interactions between existing and new users (edge-level class incremental learning, ECIL) and identifying novel protein categories (graph-level class incremental learning, GCIL). Notably, recent advances in Graph Foundation Models [14] have enabled Graph-enhanced LLMs like OFA [11], UniGraph [6], and UniAug [18] to support edge-level or graph-level tasks. Therefore, a comprehensive evaluation of GLM-based methods in ECIL and GCIL scenarios is imperative, and we designate this as one of the key directions for our future research.

Restricted to TAGs. In `LLM4GCL`, all seven benchmark datasets are TAGs, as the evaluated GLM-based methods inherently depend on textual node attributes. However, this requirement limits applicability to real-world non-textual graphs (e.g., transportation networks or flight routes). Currently, emerging solutions like GCOPE [25] and SAMGPT [24] demonstrate promising approaches for

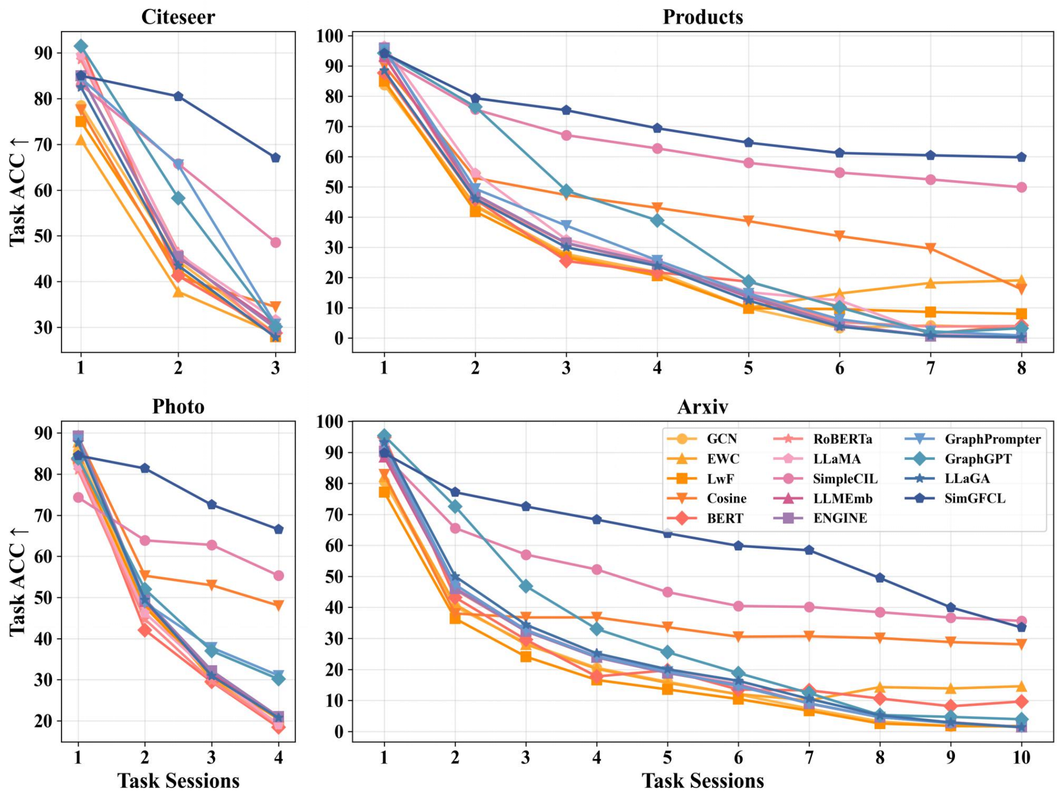

Figure 1: Performance evolution of all models on Citeseer, Photo, Products, and Arxiv datasets.

text-free graph foundation models. Consequently, extending GLM-based methods to such non-textual graph scenarios remains a critical direction for future research.

E.2 Potential Impacts

The integration of LLMs with graph tasks represents an emerging and highly promising research direction, demonstrating substantial potential across diverse applications and particularly in GCL scenarios. This paper proposes `LLM4GCL` to advance research focus on this novel paradigm that leverages graph-enhanced LLMs for improved GCL performance. Through `LLM4GCL`, we conduct a systematic investigation into graph-enhanced LLM methodologies and establish comprehensive benchmarking across various graph domains. We believe this work will serve as the catalyst and pioneer for accelerated progress in this developing research community.

Moreover, as `LLM4GCL` demonstrates several efficient LLM- and GLM-based GCL methods, we argue that these advancements hold significant societal value. Specifically, the proposed methods facilitate efficient knowledge updates in dynamic environments, such as social networks and recommendation systems, through seamless integration with LLMs, thereby substantially reducing computational overhead compared to conventional training-from-scratch paradigms. Furthermore, the cross-domain adaptability of these models offers distinct advantages in mission-critical applications, including healthcare diagnostics and financial forecasting. In such domains, continuous model adaptation to emergent knowledge, ranging from novel therapeutic interventions to evolving market trends, can be achieved while effectively mitigating catastrophic forgetting. However, practical deployment necessitates careful consideration of several key challenges, including data privacy implications, the potential for bias propagation in automated decision-making processes, and ensuring robustness against adversarial data injection, particularly in open-domain applications.

In the future, we will keep track of newly emerged techniques in the GCL field and continuously update `LLM4GCL` with more solid experimental results and detailed analyses.

References

[1] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.

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

[3] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[5] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*, 2023.

[6] Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. *arXiv preprint arXiv:2402.13630*, 2024.

[7] Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. Can gnn be good adapter for llms? In *Proceedings of the ACM Web Conference 2024*, pages 893–904, 2024.

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

[9] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[10] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[11] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[13] Zheyuan Liu, Xiaoxin He, Yijun Tian, and Nitesh V Chawla. Can we soft prompt llms for graph learning tasks? In *Companion Proceedings of the ACM Web Conference 2024*, pages 481–484, 2024.

[14] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Mikhail Galkin, and Jiliang Tang. Graph foundation models. *CoRR*, 2024.

[15] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

[16] Chaoxi Niu, Guansong Pang, Ling Chen, and Bing Liu. Replay-and-forget-free graph class-incremental learning: A task profiling and prompting approach. *arXiv preprint arXiv:2410.10341*, 2024.

[17] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500, 2024.

[18] Wenzhuo Tang, Haitao Mao, Danial Dervovic, Ivan Brugere, Saumitra Mishra, Yuying Xie, and Jiliang Tang. Cross-domain graph data scaling: A showcase with diffusion models. *arXiv preprint arXiv:2406.01899*, 2024.

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

[20] Qi-Wei Wang, Da-Wei Zhou, Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Few-shot class-incremental learning via training-free prototype calibration. *Advances in Neural Information Processing Systems*, 36:15060–15076, 2023.

[21] Yuxiang Wang, Xinnan Dai, Wenqi Fan, and Yao Ma. Exploring graph tasks with pure llms: A comprehensive benchmark and investigation. *arXiv preprint arXiv:2502.18771*, 2025.

[22] Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. A comprehensive analysis on llm-based node classification algorithms. *arXiv preprint arXiv:2502.00829*, 2025.

[23] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

[24] Xingtong Yu, Zechuan Gong, Chang Zhou, Yuan Fang, and Hui Zhang. Samgpt: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *Proceedings of the ACM on Web Conference 2025*, pages 1142–1153, 2025.

[25] Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4443–4454, 2024.

[26] Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133(3):1012–1032, 2025.

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