LoReC: Rethinking Large Language Models for Graph Data Analysis

Hongyu Zhan[1], Qixin Wang[2], Yusen Tan[1], Haitao Yu[1], Jingbo Zhou[3]
Jia Li[1], Shuai Chen[4,†], Xiao Tan[4,†], Jun Xia[1,5,†]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Tongji University, [3]Westlake University, [4]Ant International
[5]The Hong Kong University of Science and Technology
{hzhan701, ytan277, hyu382}@connect.hkust-gz.edu.cn
2450850@tongji.edu.cn, zhoujingbo@westlake.edu.cn
{shuai.cs, alex.tx}@ant-intl.com
{jialee, junxia}@hkust-gz.edu.cn

Abstract

The advent of Large Language Models (LLMs) has fundamentally reshaped the way we interact with graphs, giving rise to a new paradigm called GraphLLM. As revealed in recent studies, graph learning can benefit from LLMs. However, we observe limited benefits when we directly utilize LLMs to make predictions for graph-related tasks within GraphLLM paradigm, which even yields suboptimal results compared to conventional GNN-based approaches. Through in-depth analysis, we find this failure can be attributed to LLMs' limited capability for processing graph data and their tendency to overlook graph information. To address this issue, we propose **LoReC** (**Lo**ok, **Re**member, and **C**ontrast), a novel plug-and-play method for GraphLLM paradigm, which enhances LLM's understanding of graph data through three stages: (1) **Look**: redistributing attention to graph; (2) **Remember**: re-injecting graph information into the Feed-Forward Network (FFN); (3) **Contrast**: rectifying the vanilla logits produced in the decoding process. Extensive experiments demonstrate that LoReC brings notable improvements over current GraphLLM methods and outperforms GNN-based approaches across diverse datasets. The implementation is available at https://github.com/Git-King-Zhan/LoReC.

1 Introduction

Recently, the integration of Graph Neural Networks (GNNs) with Large Language Models (LLMs) has catalyzed a new paradigm termed GraphLLM, empowering LLMs to understand complex graph data and conduct downstream graph tasks [1–3]. In many cases, existing works have practically demonstrated that LLMs can effectively facilitate graph learning. For example, GraphGPT [4] aligns graph structural knowledge with LLMs for graph-related tasks, and Dr.E [5] utilizes LLMs to comprehend graph data. Nevertheless, we observe minor improvements or even significant performance degradation when directly utilizing LLMs for predictions in graph-related tasks within GraphLLM paradigm. Previous work [6] also observed unsatisfactory performance when utilizing LLMs to make predictions on graphs, but did not delve into this issue.

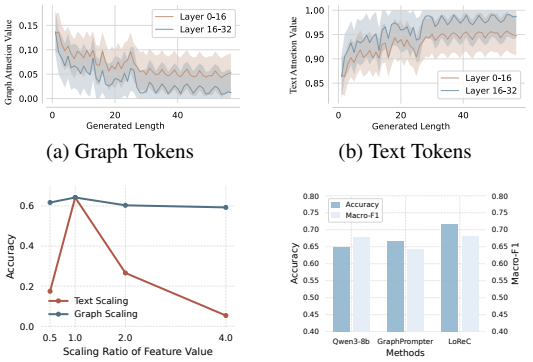
To explain these phenomena, we conduct several experiments as shown in Fig. 1. As depicted in Fig. 1a and Fig. 1b, attention to text tokens increases while attention to graph tokens decreases as more tokens are generated, impeding LLMs' understanding of graph data. This issue worsens in

[†]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

deeper layers, where over 90% of attention is allocated to text tokens. To investigate the respective contributions of graph-structured data and textual context to predictions, we proportionally scale text and graph feature values. As shown in Fig. 1c, performance degradation from scaling text features is more pronounced than from scaling graph features, indicating that GraphLLM predictions are predominantly driven by textual features. Moreover, as demonstrated in Fig. 1d, text-only LLM achieves performance comparable to GraphLLM model despite the absence of graph inputs, revealing that predictions are primarily driven by textual priors rather than graph inputs. Additional evidence across more datasets is provided in the Appendix A.

As a consequence, performance degradation in GraphLLM models stems from two interrelated factors: *diluted attention allocated to graph tokens* and *overconfidence in textual priors*. Specifically, certain text tokens with minimal predictive utility receive disproportionately high attention weights during decoding, which intensifies as decoding proceeds stepwise. Furthermore, the higher information density inherent in graph-structured data impedes GraphLLM models from effectively comprehending graph inputs, causing these models excessively rely on textual priors. This naturally motivates the following question: *Can we enhance LLMs' understanding of graph data as well as mitigate their over-reliance on textual priors?*



(a) Graph Tokens (b) Text Tokens

(c) Scaling (d) Performance Comparison

Figure 1: (a-b) Attention distribution change between graph and text tokens across decoding layers as generation length increases. (c) Model performance under different scaling ratios applied to graph and text features. (d) Performance comparison of three approaches on the Citeseer dataset: text-only LLM baseline (Qwen3-8b), GraphPrompter, and LoReC-enhanced GraphPrompter.

To answer this question, we propose **LoReC** (**Lo**ok, **Re**member, and **C**ontrast), a novel training-free decoding framework for the GraphLLM models, inspired by the natural human cognition process: *when attempting to understand complex information, people instinctively redistribute attention to overlooked evidence, calibrate their memory, and rectify conflicting information before committing to an answer*. Following this principle, we employs a three-stage pipeline: (***i***) attention redistribution to ensure the model "**looks**" at graph tokens, (***ii***) memory calibration via graph information reinjection to ensure it "**remembers**" graph evidence, and (***iii***) dual-contrastive decoding to ensure it "**contrasts**" away false priors. The main contributions can be summarized as follows:

- We conduct comprehensive and in-depth investigation demonstrating why GraphLLM models exhibit suboptimal performance.
- We propose LoReC, a three-stage training-free decoding framework that enhances GraphLLM models' understanding of graph data.
- Extensive experiments across diverse datasets demonstrate that LoReC brings consistent and notable improvements over state-of-the-art GraphLLM methods and outperforms conventional GNN-based approaches.

2 Related Work

2.1 Large Language Models

The emergence of Large Language Models (LLMs) has revolutionized deep learning across multiple domains, demonstrating remarkable capabilities in understanding and generation. Leading models, notably ChatGPT [7] and Gemini [8], have pushed the frontier of artificial intelligence with advanced capabilities in multimodality, long-context processing and agentic reasoning. Concurrently, the open-source ecosystem has thrived, with powerful foundation models such as Llama [9], Qwen [10] and Gemma [11]. A significant trend is the enhancement of complex reasoning. For instance, DeepSeek-R1 [12] demonstrates that reasoning capabilities can be effectively incentivized through reinforcement

learning, complementing prompting strategies like Chain of Thought [13] and Tree of Thoughts [14]. Underpinning these successes are foundational techniques such as In-Context Learning [15] for few-shot generalization and Reinforcement Learning from Human Feedback(RLHF) [16] for instruction alignment. Despite these remarkable strides in text processing and logical reasoning, LLMs exhibit inherent limitations when interpreting complex structural data. A recent benchmark [17] investigates how LLMs understand graph patterns and reveals that LLMs struggle to understand graph topologies and structural patterns. This critical gap has motivated the development of GraphLLM paradigm, which aims to equip LLMs with graph comprehension capabilities.

2.2 GraphLLM

Recently, the convergence of Graph Neural Networks (GNNs) and LLMs has established the nascent field of GraphLLM. This paradigm fundamentally transforms graph interaction by enabling the alignment between graph structures and natural languages. Initial efforts addressed the gap by projecting graph components into the LLM's token space via instruction tuning, as exemplified by GraphGPT [4] and LLaGA [2], soft prompting strategies like GNP [18] and GraphPrompter [19], or efficient adapter modules as seen in GraphAdapter [1]. To achieve finer granularity and broader generalization, subsequent research introduced token-level quantization techniques, such as Dr.E [5], and developed unified graph foundation models. Additionally, GOFA [20] interleaves GNN layers with LLMs, while Unigraph [3] leverages text-attributed graphs for cross-domain transfer. Furthermore, to address the context window limitations inherent in large-scale graphs, GraphChain [21] innovates by treating the LLM as an intelligent agent that performs analysis via logical tool chaining. However, benchmarks including GLBench [6] and GraCoRe [22] indicate that despite promising performance in semantic understanding and zero-shot scenarios, the misalignment between graph topology and text semantics often impedes accuracy in downstream tasks.

3 Background

3.1 Preliminaries

Graph-Structure Data. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ represents the edges. The topological structure is encoded by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N\times N}$, where $\mathbf{A}_{ij} = 1$ indicates a connection between nodes v_i and v_j. Each node v_i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^F$, forming the feature matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$.

Graph Neural Networks (GNNs). GNNs learn node representations by recursively aggregating information from local neighborhoods. This process, referred to as the *message-passing paradigm*, distinguishes graph representation learning from architectures designed for regular modalities such as images or text. The layer-wise update for a node v's embedding $\mathbf{h}_v^{(l)}$ is formally defined as:

$$\mathbf{h}_v^{(l+1)} = \psi^{(l)}\left(\mathbf{h}_v^{(l)}, \phi^{(l)}\left(\left\{\mathbf{h}_u^{(l)} : u \in \mathcal{N}(v)\right\}\right)\right), \tag{1}$$

where $\mathcal{N}(v)$ denotes the neighbors of v. Here, $\phi(\cdot)$ serves as the aggregation function (pooling messages from neighbors), and $\psi(\cdot)$ acts as the update function to fuse the aggregated context with the node's current state. After L layers, the final representations $\mathbf{h}^{(L)}$ are utilized for downstream tasks such as classification or link prediction.

Autoregressive Language Models. LLMs are typically instantiated as autoregressive transformer decoders trained via Causal Language Modeling (CLM). Formally, given a sequence of tokens $\mathbf{s} = (s_1, s_2, \ldots, s_R)$, the model estimates the joint probability $p(\mathbf{s})$ by decomposing it into a sequence of conditional probabilities:

$$p(\mathbf{s}) = \prod_{t=1}^{R} p(s_t \mid s_{<t}), \tag{2}$$

where $s_{<t}$ denotes the preceding context. The core mechanism driving this estimation is *masked self-attention*, which allows the model to attend to historical tokens while preventing information leakage from future tokens. For a specific attention head h, the output $\mathbf{O}_h$ is computed as:

$$\mathbf{O}_h = \text{softmax}\left(\frac{\mathbf{Q}_h\mathbf{K}_h^\top}{\sqrt{d_k}}\right)\mathbf{V}_h, \tag{3}$$

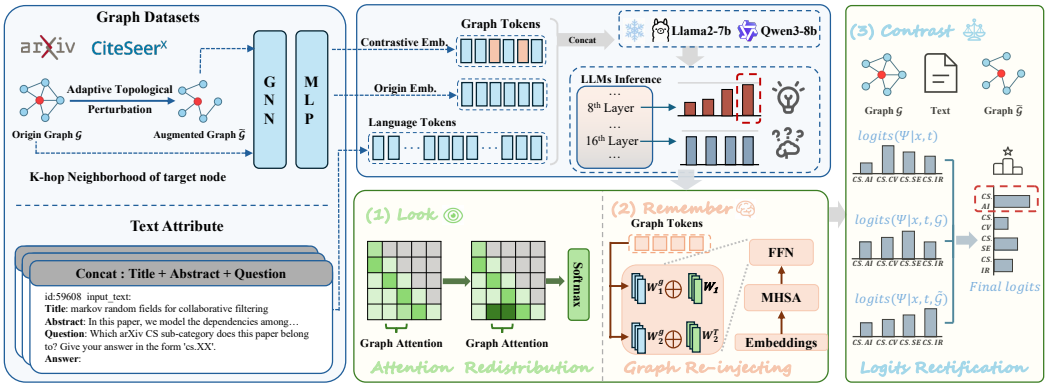


Figure 2: The overall framework of our proposed LoReC.

where $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$ serve as the query, key, and value matrices projected from the input embeddings, and d_k is the scaling factor. The final probability distribution over the vocabulary is obtained by applying a softmax function to the projected hidden states of the last layer.

3.2 Problem Formulation

Given a GraphLLM model $\mathcal{M}_\theta$ parameterized by θ, with a general architecture consisting of a text embedding layer, a graph encoder, a graph-text interface module, a text decoder with L transformer layers, and an affine layer $\varsigma(\cdot)$ that predicts the next-token distribution. For a graph-grounded task with textual query q and input graph $\mathcal{G}$, GraphLLM models first utilize the graph encoder $f_{\mathcal{G}}(\cdot)$ to transform the raw graph structure into a sequence of latent embeddings $\mathbf{Z}_{\mathcal{G}} = f_{\mathcal{G}}(\mathcal{G}) \in \mathbb{R}^{N \times d_g}$, where N is the number of nodes and d_g is the embedding dimension. The graph embeddings $\mathbf{Z}_{\mathcal{G}}$ are then projected into the LLM's word embedding space $\mathbb{R}^{d_{llm}}$ as graph context tokens $\mathbf{C}_{\mathcal{G}}$, which are concatenated with the textual instruction tokens $\mathbf{C}_{\mathcal{T}}$ and autoregressively decoded into the textual response $\mathbf{Y} = \{y_1, y_2, \ldots, y_{max}\}$:

$$p(\mathbf{Y} \mid \mathcal{G}, \mathbf{H}_{\mathcal{T}}) = \prod_{t=1}^{max} p(y_t \mid \mathbf{H}_{\mathcal{G}}, \mathbf{H}_{\mathcal{T}}, y_{<t}; \theta), \tag{4}$$

where $y_{<t}$ denotes previously generated tokens and θ represents the model parameters. This paradigm enables GraphLLM models to perform graph-related tasks.

4 Methodology

Building on the background outlined in Sec. 3, we propose **LoReC**, a framework containing three major components: (***i***) attention redistribution to ensure the model "**looks**" at graph tokens, (***ii***) Graph Re-injection via graph information reinjection to ensure it "**remembers**" graph evidence, and (***iii***) dual-contrastive decoding to ensure it "**contrasts**" away false priors. We present the framework overview in Fig. 2 and all details are listed below.

4.1 *Look*: Attention Redistribution

As demonstrated in Fig. 1, GraphLLM models allocate progressively less attention to graph tokens in deeper layers as the generation process proceeds. This phenomenon causes models to overlook the graph tokens $\mathbf{C}_{\mathcal{G}}$, preventing them from effectively "looking" at the graph data during inference. To counteract this graph attention dilution problem, we propose a mechanism dubbed **Attention Redistribution** that dynamically amplifies attention to graph tokens based on predictive uncertainty. The overall procedure is outlined in the Appendix E.

Uncertainty-Aware Trigger. We employ the Shannon entropy of the next-token distribution as a proxy for model uncertainty, which is computed by a vocabulary head $\varsigma(\cdot)$ on each layer during decoding [23]. Formally, let $P_\theta(i \mid \mathbf{x}_{<t}, \mathcal{G})$ denote the probability distribution over the top N

candidates at time step t, conditioned on the preceding context $\mathbf{x}_{<t}$ and the graph input $\mathcal{G}$. The predictive uncertainty $\mathcal{H}_t^{(l)}$ at layer l is calculated as:

$$\mathcal{H}_t^{(l)} = -\frac{1}{\log N} \sum_{i=1}^{N} P_\theta(i \mid \mathbf{x}_{<t}, \mathcal{G}) \log P_\theta(i \mid \mathbf{x}_{<t}, \mathcal{G}). \tag{5}$$

A low $\mathcal{H}_l$ suggests a sharply peaked distribution, indicating that the model is confident in its prediction. Conversely, a high $\mathcal{H}_l$ signifies a flat distribution, which suggests that the model exhibits high uncertainty regarding its prediction. In practice, we consider that uncertainty exceeding the pre-defined threshold γ at a given layer warrants intervention to recalibrate attention allocation.

Pay Attention to Graph. To mitigate GraphLLM models' tendency to overlook graph evidence, we intervene directly in the attention layers to enforce a "*Look*" operation. Unlike rigid masking or post-hoc probability scaling, our Attention Redistribution mechanism explicitly reallocates attention weights to graph tokens.

We first extract the graph token attention weights from the pre-softmax attention logits $\tilde{\mathbf{e}}_t$ for the currently generated token. When the model exhibits high uncertainty—as indicated by the layer-wise entropy $\mathcal{H}^{(l)}$ (defined in Eq. (5))—we adaptively rectify the attention distribution to allocate more weight to graph tokens. The rectified attention logits are formulated as follows:

$$\begin{aligned} \tilde{e}_{t,j} &= e_{t,j} + \Gamma(\mathcal{H}^{(l)} > \gamma) \cdot \eta \cdot |e_{t,j}|, \\ &\textit{subject to} \quad j \in \Omega_{\mathcal{G}}, \end{aligned} \tag{6}$$

where η is a tunable hyperparameter controlling the rectification strength, and $\Gamma(\cdot)$ denotes the gating function. The amplification is applied uniformly to all graph tokens $j \in \Omega_{\mathcal{G}}$. Since the model's final vocabulary probability distribution is derived directly from the last token's hidden state projection, we extract the last token's attention weights over the graph tokens by indexing $\tilde{e}_{t,j}$.

Following the attention intervention, we apply softmax function to redistribute attention weights across tokens when reassigning encoded hidden states. This procedure repeats autoregressively for each subsequent token prediction. Furthermore, this approach is mathematically principled as it is topology-preserving by amplifying attention proportionally to graph's intrinsic relevance.

4.2 ***Remember*: Graph Re-injection**

Simply directing the model's attention to graph tokens through the "*Look*" mechanism is not enough to ensure the model's understanding of graph data. To fully enhance graph comprehension, we must extend our intervention from the attentional surface to the parametric depth, ensuring the model can "*Remember*" graph information. Specifically, we propose **Graph Re-injection**, a mechanism that dynamically injects graph information into the FFN layers based on predictive uncertainty. The overall procedure is outlined in the Appendix E.

Memory Retrieval Formulation. The Feed-Forward Network (FFN) layers constitute approximately two-thirds of the parameters in Transformer-based architectures, serving as the primary repository for factual knowledge [24]. Consider the input hidden state $x \in \mathbb{R}^d$ for a given layer, the vanilla FFN can be formulated as:

$$\text{FFN}(x) = \phi(xW_1)W_2, \tag{7}$$

where $\phi(\cdot)$ denotes the activation function (e.g., ReLU or SiLU), $W_1 \in \mathbb{R}^{d \times d_m}$ and $W_2 \in \mathbb{R}^{d_m \times d}$ represent the up-projection and down-projection weight matrices, respectively, with d_m being the intermediate dimension. Specifically, the weight matrices W_1 and W_2 can be explicitly decomposed into sets of vectors as:

$$\mathbf{W_1} = [\mathbf{k}_1, \ldots, \mathbf{k}_{d_m}], \mathbf{W_2} = [\mathbf{v}_1, \ldots, \mathbf{v}_{d_m}]^\top, \tag{8}$$

where $k_i \in \mathbb{R}^d$ and $v_i \in \mathbb{R}^d$ are entries of key and value, respectively. As a consequence, FFN can be reformulated as a weighted aggregation of memory slots [25]:

$$\text{FFN}(x) = \sum_{i=1}^{d_m} \phi(\langle x, k_i \rangle) \cdot v_i, \tag{9}$$

where $m_i = \phi(\langle x, k_i \rangle)$ serves as the *memory coefficient*, quantifying the relevance of the i-th memory slot to the current input. Considering the formulation, FFN essentially performs a "soft retrieval" operation: the input x queries the parameter space to activate relevant patterns (k_i) and retrieves the associated knowledge (v_i).

Re-inject Graph Information. Motivated by the findings above, we intervene in the FFN layers to implement a "*Remember*" operation following the "*Look*" stage. Specifically, we propose a *Graph Re-injection* mechanism that treats graph tokens as keys and values (in Eq. (9)) and re-injects them into specific FFN layers based on uncertainty triggers.

Let $\mathbf{C}_{\mathcal{G}} = (\mathbf{g}_1, \ldots, \mathbf{g}_J) \in \mathbb{R}^{d \times J}$ denote the set of graph token embeddings, and let $x \in \mathbb{R}^d$ represent the current hidden state of the language model. We reinterpret the FFN as a key-value memory retrieval mechanism, where x serves as a query vector and graph tokens act as auxiliary memory slots encoding rich graph structural context. Once the model exhibits high unertainty (i.e., $\mathcal{H}^{(l)} > \gamma$, as defined in Eq. (5)), graph information is explicitly re-injected into selected FFN layers. The operation can be expressed as:

$$\Delta(\mathcal{G} \mid x) = \sum_{j=1}^{J} \phi\big(\langle x, \mathbf{g}_j \rangle\big)\, \mathbf{g}_j. \tag{10}$$

To integrate this information, we treat the retrieved graph information as supplementary evidence that operate concurrently with the original FFN output. The calibrated output, $\widehat{\mathrm{FFN}}(x)$, is derived by fusing the vanilla FFN output with the graph-based correction term:

$$\widehat{\mathrm{FFN}}(x) = \underbrace{(1-\alpha) \cdot \phi(x\mathbf{W}_1)\mathbf{W}_2}_{\text{Original Memory}} + \underbrace{\alpha \cdot \phi(x\mathbf{W}_1^g)\mathbf{W}_2^g}_{\text{Auxiliary Graph Memory}}. \tag{11}$$

Here, Original Memory represents the vanilla FFN output, while the Auxiliary Graph Memory represents the output of our Graph Re-injection Module (defined in Eq. (10)). The parameters $\mathbf{W}_1^g$ and $\mathbf{W}_2^g$ are derived from graph tokens and encode supplementary graph-related information in the hidden states. The scalar $\alpha \in [0, 1]$ controls the magnitude of the graph intervention.

Through *Attention Redistribution* that enables the model to "*Look*" at graph information and *Graph Re-injection* that enables it to "*Remember*", LoReC enhances the model's perception and understanding of the input graph structure.

4.3 *Contrast*: Logit Rectification

While the "*Look*" and "*Remember*" modules successfully enhance GraphLLM models' perception and understanding of graph data in latent space, the predictions may still inherit false priors. Standard decoding algorithms greedily select tokens with maximum probability, often polluted by the model's pre-training distribution rather than the specific context. In GraphLLM paradigm, this issue is further compounded by graph-specific prior biases originating from message-passing mechanisms and inductive biases inherent in graph encoders.

To eliminate these biases and enable models to yield less biased predictions, we rectify the models' vanilla logits, ensuring they can "*Contrast*" away false priors. Specifically, we introduce **Logit Rectification**, a mechanism that directly rectifies the final next-token distribution using the divergence between the vanilla output logits and two types of negative logits: text-only logit and augmented graph logit, thereby producing more faithful predictions.

Adaptive Topological Perturbation. To mitigate the negative prior that stems primarily from the recursive aggregation of high-degree "hub nodes", which dominate the message-passing trajectory and enforce a rigid homophily prior, we require the augmented graph view $\tilde{\mathcal{G}}$ that preserves dominant hub structures while stochastically pruning sparse, peripheral connections that often contain fine-grained evidence. Naive uniform edge dropout is insufficient as it fails to account for the heterogeneous importance of nodes. Following [26], we employ an adaptive augmentation strategy based on degree centrality to generate $\tilde{\mathcal{G}}$. Formally, let $\varphi(v) = \log(\deg(v) + \epsilon)$ denote the degree centrality of node v. The connectivity strength of edge $(u, v) \in \mathcal{E}$ is defined as s_{uv} and normalized to $\tilde{s}_{uv}$:

$$s_{uv} = \frac{\varphi(u) + \varphi(v)}{2}, \quad \tilde{s}_{uv} = \frac{s_{\max} - s_{uv}}{s_{\max} - s_{avg}}, \tag{12}$$

where $s_{\max}$ and s_{avg} denote the maximum and average strength values in the graph. We further introduce an overall edge drop rate μ and a truncation threshold τ (effectively preventing structural collapse). The final edge dropout distribution is formulated as:

$$w_{uv} = \min\left(\tau,\ \mu \cdot (1 - \tilde{s}_{uv})\right). \tag{13}$$

Note that the augmented graph logits (Ψ_{aug}) will be subtracted from the original logits (Eq. (14)). Therefore, the augmented graph view $\tilde{\mathcal{G}}$ should maximally preserve and amplify the graph inductive bias, so that the contrastive subtraction can effectively remove this bias.

Dual-Contrastive Decoding. To rectify text and graph biases in the original logits Ψ_{orig}, we obtain text-only logits Ψ_{text} and perturbed-graph logits Ψ_{aug} that amplify these biases, respectively. Specifically, we calculate Ψ_{text} by masking all graph tokens before decoding, and calculate Ψ_{aug} use the adaptively augmented graph $\tilde{\mathcal{G}}$ as input. In practice, we use a gating function $\mathbb{I}_{\text{gate}}$ that prevents invalid perturbations (when no edges are dropped or when cutting edges would remove vital information from sparse graphs). The dual-contrastive decoding process is formulated as:

$$\Psi_{\text{final}} = \Psi_{\text{orig}} + \omega \underbrace{(\Psi_{\text{orig}} - \Psi_{\text{text}})}_{\Delta_{\text{Text de-biasing}}} + \beta \cdot \mathbb{I}_{\text{gate}} \underbrace{(\Psi_{\text{orig}} - \Psi_{\text{aug}})}_{\Delta_{\text{Graph de-biasing}}}, \tag{14}$$

where ω and β are hyperparameters controlling the suppression magnitude. $\Delta_{\text{Text de-biasing}}$ penalizes text bias in Ψ_{text} deviating from ground truth, while $\Delta_{\text{Graph de-biasing}}$ penalizes graph bias in Ψ_{aug}.

In practice, we define a dynamic candidate set $\mathcal{Z}_{\text{head}}$ that selectively contains only tokens with high probability under the unperturbed prediction distribution, thereby ensuring the coherence and semantic validity of the generated sequence. The complete formulation is as follows:

$$\begin{aligned} y_t \sim \text{Softmax}\left[\Psi_{\text{orig}} + \omega \cdot (\Psi_{\text{orig}} - \Psi_{\text{text}}) + \beta \cdot (\Psi_{\text{orig}} - \Psi_{\text{aug}})\right], \\ \text{subject to} \quad y_t \in \mathcal{Z}_{\text{head}}(y_{<t}). \end{aligned} \tag{15}$$

Overall, the three-stage LoReC framework, comprising "*Look*", "*Remember*" and "*Contrast*", empowers GraphLLMs with enhanced graph understanding and debiased prediction capabilities, similar to human cognitive processes. The computational costs analysis is provided in Appendix B, and the complete LoReC pseudo code is outlined in Appendix E.

5 Experiments

5.1 Experimental Setup

Datasets and Evaluation Protocols. To evaluate the performance of our proposed LoReC, we conduct experiments on five widely recognized graph datasets: PubMed [27], Cora [28, 29], Citeseer [28], Ogbn-arxiv and Ogbn-products [30]. The statistics of these datasets are summarized in Table 1. We randomly partition all datasets into training, validation, and testing sets with a ratio of 3:1:1. Note that for experiments in GraphGPT, we adhere to the its vanilla split setting with a training, validation, and testing ratio of 6:2:3. Performance is reported using Accuracy for balanced tasks and Micro-F1 for multi-label settings.

Table 1: Statistics of datasets.

DATASET	# NODES	# EDGES	# CLASSES
CORA	2,708	10,556	7
CITESEER	3,327	9,228	6
PUBMED	19,717	44,338	3
ARXIV	169,343	1,166,243	40
PRODUCTS	2,449,029	61,859,140	47

Baseline Models. We compare our proposed method against three distinct categories of baselines to ensure a comprehensive evaluation: (1) GNN-based models, specifically GCN [28], GAT [31], GKD [32], and GLNN [33]; (2) LLM-only approaches, including Baichuan-7b, Llama2-7b [34], Vicuna-7b [35] and Qwen3-8b [10] , which serve as semantic-centric baselines that rely solely on node text while ignoring graph topology; and (3) GraphLLM models, such as GraphGPT [4] and GraphPrompter [19] , representing state-of-the-art hybrid paradigm against which we benchmark our specific improvements.

Implementation Details. We utilize GraphGPT and GraphPrompter as base models. Following their original setups, we employ Vicuna-7B and Llama2-7B as backbones, respectively. Additionally,

Table 2: Performance comparison based on GraphGPT. We highlight the performance of LoReC with gray background. The highest performances across different datasets are highlighted in **bold**. The results of GNN-based models and general LLMs are taken from published reports in GraphGPT.

Model	Arxiv-Arxiv		Arxiv-Pubmed		(Arxiv+Pubmed)-Arxiv		(Arxiv+Pubmed)-Cora	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
GNN-based Models								
GCN	0.5267	0.3202	0.3940	0.1884	0.0122	0.0008	0.0187	0.0032
GAT	0.5332	0.3118	0.3940	0.1884	0.1707	0.0285	0.0161	0.0057
GKD	0.5570	0.1595	0.3645	0.2561	0.2089	0.0179	0.0406	0.0037
GLNN	0.6088	0.3757	0.4298	0.3182	0.3373	0.1115	0.0182	0.0092
General LLMs								
Baichuan-7b	0.0946	0.0363	0.4642	0.3876	0.0946	0.0363	0.0405	0.0469
Vicuna-7b-v1.5	0.4962	0.1853	0.6351	0.5231	0.4962	0.1853	0.1489	0.1213
GraphLLM Models								
GraphGPT-std	0.6134	0.2607	0.6770	0.6283	0.6173	0.2594	0.1259	0.0773
GraphGPT-cot	0.5630	0.2483	0.5213	0.4816	0.6476	0.2854	0.1459	0.1287
Ours								
LoReC(GraphGPT-std)	**0.6312**	**0.2831**	**0.7018**	**0.6523**	0.6428	0.2753	0.1278	0.0785
LoReC(GraphGPT-cot)	0.5826	0.2720	0.5437	0.5175	**0.6734**	**0.3278**	**0.1643**	**0.1394**

Table 3: Performance comparison based on GraphPrompter. We highlight the performance of LoReC with gray background. The highest performances across different datasets are highlighted in **bold**.

Model	Cora		Citeseer		Arxiv		Products	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
General LLMs								
Llama2-7b	0.3764	0.3002	0.3846	0.4731	0.0909	0.0304	0.1117	0.0474
Qwen3-8b	0.6568	0.5647	0.6591	0.6786	0.6204	0.2162	0.1761	0.1461
GraphLLM Models								
GraphPrompter (Llama2-7b)	0.7915	0.7134	0.7029	0.6555	0.7008	0.4153	0.7720	0.3833
GraphPrompter (Qwen3-8b)	0.8026	0.7727	0.6652	0.6430	0.7331	0.3891	0.7422	0.3675
Ours								
LoReC (GraphPrompter-Llama2)	**0.8137**	**0.7931**	0.7119	0.6418	0.7094	0.4348	**0.7767**	**0.3854**
LoReC (GraphPrompter-Qwen3)	0.8026	0.7762	**0.7164**	**0.6804**	**0.7437**	**0.4674**	0.7459	0.3772

we evaluate both GraphPrompter and LoReC-GraphPrompter using a more recent and powerful backbone, Qwen3-8B. Experiments are conducted on 4 $\times$ H100 GPUs (80GB) for GraphGPT and 2 $\times$ A800 GPUs (80GB) for GraphPrompter.

Configuration. For our "*Look*" and "*Remember*" modules, we employ the Shannon entropy of the next-token distribution to trigger attention amplification and graph re-injection, with a shared entropy threshold of γ. Specifically, we apply attention amplification across layers 15-22 with an amplification factor of η. Graph re-injection is applied to layers 8-16 with a fusion ratio of α. For our "*Contrast*" module, we set the text-only contrast weight ω to 0.5 and the augmented graph contrast weight β to 1.0 by default. The adaptive plausibility constraint parameter is set to 1.0 to filter out unlikely tokens. For graph view generation, we employ a degree-based edge dropout strategy with a default rate of $\mu = 0.2$, which varies across datasets and selectively removes edges connected to low-degree nodes to create structurally valid contrastive views. The detailed parameter configurations can be found in the Appendix C.1.

5.2 Comparison with State-of-the-art Results

We compare our proposed method against three categories of baselines: (1) GNN-based models (GCN, GAT, GKD, GLNN), (2) General LLMs (Llama2-7b, Vicuna-7b, Baichuan-7b, Qwen3-8b), and (3) GraphLLM models (GraphGPT, GraphPrompter). Note that LoReC is not trained from scratch but rather applied as a plug-and-play decoding method; thus, its performance depends on the underlying GraphLLM models. We present performance comparisons on the benchmarks utilized by GraphGPT and GraphPrompter, with results summarized in Tables 2 and 3. More experiment results can be found in the Appendix D.

Overall Performance. As shown in Tables 2 and 3, our method consistently outperforms all baselines across all datasets. Note that following GraphGPT's setup, we evaluate our method in two settings: (1) supervised learning, where models are trained and tested on the same dataset (e.g., Arxiv), and (2) zero-shot transfer, where models trained on one dataset (e.g., Arxiv) are tested on another (e.g., PubMed). In Table 2, "-v1.5-" represents version of the base Vicuna model. "-std" and "cot" denote the use of the standard and generated COT instruction datasets, respectively. While state-of-the-art GraphLLM models (GraphGPT and GraphPrompter) integrate graph features, their standard decoding mechanism overlooks graph information and inherits false textual priors. In contrast, our method achieves accuracy improvements of up to 2.58% and 5.12% over GraphGPT and GraphPrompter, respectively, with macro-F1 gains of up to 4.24% and 7.97%, demonstrating its effectiveness in enhancing graph understanding.

5.3 Ablation Study

To validate the effectiveness of three modules and the hyper-parameters we introduced in LoReC, we conducted in-depth ablation experiments as detailed below. All ablation experiments are conducted based on GraphGPT. More details can be found in the Appendix C.

Effect of Individual Components. To validate our three-stage design, we conduct ablation studies by enabling "*Look*", "*Remember*", and "*Contrast*" modules individually and in pairs. As illustrated in Figure 3, each individual module provides performance gains over the baseline GraphLLM model, demonstrating its independent contribution. Pairwise combinations consistently outperform individual modules, indicating synergistic effects. However, only the complete LoReC framework—integrating all three stages—achieves optimal performance, surpassing all partial configurations. These results validate our core design philosophy: "*Look*" strengthens model's perception to graph data at the attention layer, "*Remember*" extends the intervention from the attentional surface to the parametric depth by re-injecting graph evidence into FFN layers, and "*Contrast*" finally rectifies false priors through dual-contrastive decoding. The complementary nature of these three stages is essential for comprehensively enhancing GraphLLM models' understanding of graph data.

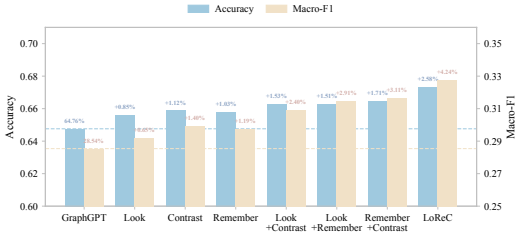


Figure 3: Impact of LoReC's components on the Arxiv dataset.

Effect of Amplification Factor and Injection Ratio. As shown in Figure 4, we analyze the impact of amplification factors η and injection ratios α on both accuracy and macro-F1 scores. For η, values between 0.05 and 0.45 consistently improve model performance, with the optimal setting observed at approximately 0.20. Similarly, injection ratio α in the moderate range of 15% to 35% yields positive effects, peaking at around 25%. Notably, performance degrades when α exceeds 35%, indicating that excessive graph information injection can be detrimental.

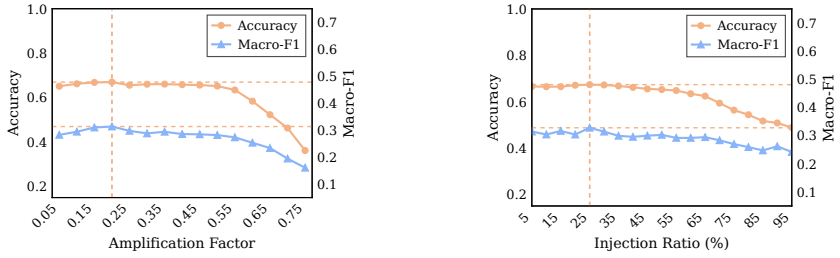


Figure 4: Ablation studies on the Arxiv dataset: (Left) Results under different amplification factors η; (Right) Results under different injection ratios α. Both use GraphGPT as the base model.

6 Conclusion

In this paper, we explain why LLMs struggle in GraphLLM paradigm and propose LoReC, a novel decoding method that comprehensively enhances LLMs' understanding of graph data through three stages: "*Look*", "*Remember*", and "*Contrast*". To our knowledge, this represents the first work that systematically investigates the underlying causes of suboptimal performance in GraphLLM models and addresses them through a training-free decoding framework, simultaneously enhancing graph comprehension and prediction faithfulness without requiring any additional fine-tuning. Extensive experiments across multiple datasets affirm that LoReC delivers consistent and notable improvements over state-of-the-art GraphLLM methods.

7 Acknowledgements

We sincerely thank the anonymous reviewers for their insightful comments and constructive suggestions. This research is supported by the National Natural Science Foundation of China Project (No.62606430, No. 623B2086), Sponsored by CCF-GHFund (No. OF 2026005), Sponsored by CIPS-SMP-Zhipu Large Model Fund, Ant Group and Shanghai AI Lab. This work is partially supported by the Research Travel Grant of the Base of Red Bird MPhil (RBM) at The Hong Kong University of Science and Technology (Guangzhou). We would like to thank our RBM Project Supervisor Ke Xue and Yue Wang for the academic support.

References

[1] X. Huang, K. Han, Y. Yang, D. Bao, Q. Tao, Z. Chai, and Q. Zhu, "Can gnn be good adapter for llms?" in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 893–904.

[2] R. Chen, T. Zhao, A. K. JAISWAL, N. Shah, and Z. Wang, "LLaGA: Large language and graph assistant," in *Forty-first International Conference on Machine Learning*, 2024.

[3] Y. He, Y. Sui, X. He, and B. Hooi, "Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 2025, pp. 448–459.

[4] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. 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*, 2024, pp. 491–500.

[5] Z. Liu, L. Wu, M. He, Z. Guan, H. Zhao, and N. Feng, "Multi-view empowered structural graph wordification for language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 23, 2025, pp. 24 714–24 722.

[6] Y. Li, P. Wang, X. Zhu, A. Chen, H. Jiang, D. Cai, V. W. Chan, and J. Li, "Glbench: A comprehensive benchmark for graph with large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 42 349–42 368, 2024.

[7] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[8] G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen *et al.*, "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[9] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[10] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv *et al.*, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.

[11] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière *et al.*, "Gemma 3 technical report," *arXiv preprint arXiv:2503.19786*, 2025.

[12] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi *et al.*, "Deepseek-r1 incentivizes reasoning in llms through reinforcement learning," *Nature*, vol. 645, no. 8081, pp. 633–638, 2025.

[13] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[14] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *Advances in neural information processing systems*, vol. 36, pp. 11 809–11 822, 2023.

[15] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang *et al.*, "A survey on in-context learning," in *Proceedings of the 2024 conference on empirical methods in natural language processing*, 2024, pp. 1107–1128.

[16] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[17] X. Dai, H. Qu, Y. Shen, B. Zhang, Q. Wen, W. Fan, D. Li, J. Tang, and C. Shan, "How do large language models understand graph patterns? a benchmark for graph pattern comprehension," in *The Thirteenth International Conference on Learning Representations*, 2025.

[18] Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, and P. Xu, "Graph neural prompting with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 080–19 088.

[19] Z. Liu, X. He, Y. Tian, and N. V. Chawla, "Can we soft prompt llms for graph learning tasks?" in *Companion Proceedings of the ACM Web Conference 2024*, 2024, pp. 481–484.

[20] L. Kong, J. Feng, H. Liu, C. Huang, J. Huang, Y. Chen, and M. Zhang, "Gofa: A generative one-for-all model for joint graph language modeling," *arXiv preprint arXiv:2407.09709*, 2024.

[21] C. Wei, W. Hu, X. Hao, X. Wang, Y. Yang, Y. Wang, Y. Tian, and Y. Chen, "Graphchain: Large language models for large-scale graph analysis via tool chaining," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[22] Z. Yuan, M. Liu, H. Wang, and B. Qin, "Gracore: Benchmarking graph comprehension and complex reasoning in large language models," in *Proceedings of the 31st International Conference on Computational Linguistics*, 2025, pp. 7925–7948.

[23] Y.-S. Chuang, Y. Xie, H. Luo, Y. Kim, J. R. Glass, and P. He, "Dola: Decoding by contrasting layers improves factuality in large language models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=Th6NyL07na

[24] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[25] S. Jie, Y. Tang, N. Ding, Z.-H. Deng, K. Han, and Y. Wang, "Memory-space visual prompting for efficient vision-language fine-tuning," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 22 062–22 074.

[26] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.

[27] X. He, X. Bresson, T. Laurent, A. Perold, Y. LeCun, and B. Hooi, "Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning," *arXiv preprint arXiv:2305.19523*, 2023.

[28] T. Kipf, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[29] Z. Wen and Y. Fang, "Augmenting low-resource text classification with graph-grounded pre-training and prompting," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 506–516.

[30] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[32] C. Yang, Q. Wu, and J. Yan, "Geometric knowledge distillation: Topology compression for graph neural networks," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[33] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less neural networks: Teaching old MLPs new tricks via distillation," in *International Conference on Learning Representations*, 2022.

[34] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[35] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-bench and chatbot arena," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the main contributions and scope of our work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section C.5, we discuss the limitation of our model.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper is based on empirical results no theoretical proof is required.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The three core components of the model are thoroughly introduced in Section 4, while the complete implementation details are provided in Section C.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our data and code are available at (`https://anonymous.4open.science/r/LoReC-63F4`)

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the details, including data splits and hyperparameters. Additionally, implementation details are included in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: To ensure the fairness and stability of our results, we conduct 5 independent experiments and report the average results across these runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have reported the GPU configurations used in our experiments in Section 5.1, along with detailed memory consumption and execution time analyses in Section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research fully complies with all requirements of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper does not explicitly show the positive/negative societal impacts. But we still discuss its limitation in Section C.5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All core data and models are properly referred and have been available in public domain.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: In the abstract, we provide an anonymous GitHub link to offer open access to our code and data.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.

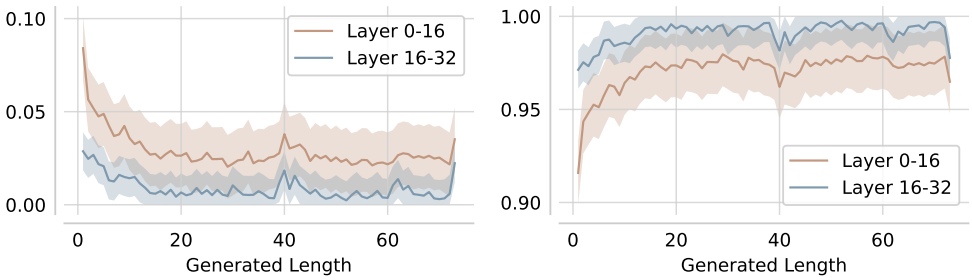
Technical appendices and supplementary material

In the Appendix, we provide supplementary material to the main paper. The structure is as follows:

1. Section A presents the attention distribution across graph and text tokens on Pubmed dataset and extended attention diagnostics.
2. Section B presents computational costs analysis of LoReC.
3. Section C outlines the experimental settings, hyper-parameters analysis and limitation in detail.
4. Section D presents additional experimental results.
5. Section E presents pseudo codes of LoReC.

A Extended Attention Diagnostics

A.1 Attention Distribution Across Graph and Text Tokens on PubMed Dataset.



(a) Graph Tokens (b) Text Tokens

Figure 5: Visualization of the attention distribution between graph and text tokens across decoding layers on the PubMed dataset. The results illustrate the dynamic evolution of attention patterns as the generation proceeds.

As illustrated in Fig. 5, our experiments using GraphGPT on the PubMed dataset reveal that attention progressively shifts from graph tokens to text tokens as generation proceeds. This phenomenon explains the suboptimal performance of GraphLLM models: due to their autoregressive nature, LLMs tend to over-rely on textual features while neglecting graph information. Notably, this attention imbalance becomes more pronounced in deeper layers, confirming our hypothesis.

A.2 Attention Decay across Datasets and Models

We evaluate GraphPrompter, using a different LLM backbone from the GraphGPT configuration in Fig. 1, on Cora, Citeseer, and ogbn-products. Table 4 reports the graph-token attention ratio over generated token length.

Table 4: Graph-token attention ratio (%) across generated token length.

Dataset	Generated token length			
	Length 1–10	Length 11–20	Length 21–30	Length 31–40
Cora	1.41	1.17	1.10	1.02
Citeseer	1.68	1.29	1.07	0.93
ogbn-products	1.37	1.24	1.13	1.08

Graph-token attention decreases monotonically across the reported intervals on all three datasets. This pattern agrees with the GraphGPT observations in Fig. 1.

A.3 Attention Dynamics after Applying LoReC

Following the configuration conducted in Fig. 1, we compare GraphGPT with and without LoReC. Table 5 reports graph-token attention over generation, together with the change from the first interval to the last.

Table 5: Graph-token attention ratio (%) and change across generated token length.

Model	Generated token length				Decay (first→last)
	Length 1–10	Length 11–20	Length 21–30	Length 31–40	
GraphGPT	8.7	5.4	3.6	1.6	−7.1
GraphGPT + LoReC	11.2	9.8	6.9	4.7	−6.5

LoReC increases graph-token attention in every reported generation interval. In particular, attention in the final interval rises from 1.6% to 4.7%. The absolute first-to-last decline decreases modestly, from 7.1 to 6.5 percentage points. Relative to the attention in the first interval, GraphGPT retains 18.4% in the final interval, whereas GraphGPT with LoReC retains 42.0%. Thus, although attention still declines during generation, LoReC maintains a higher level of graph-token attention and improves its proportional retention over the reported intervals.

These observations are consistent with LoReC's objective of sustaining the influence of graph information during decoding through attention redistribution and the reintroduction of graph evidence under high uncertainty.

A.4 Layer-Wise Attention Changes

We further examine attention allocation across layer groups. LoReC directly intervenes in the layer ranges used by Remember (8–16) and Look (15–22). In Table 6, a dagger marks groups containing directly intervened layers; it does not indicate that every layer within a marked group is modified.

Table 6: Graph-token attention ratio (%) by layer group.

Model	Layer group			
	Layer 0–8	Layer 8–16†	Layer 16–24†	Layer 24–32
GraphGPT	18.4	10.3	6.2	3.1
GraphGPT + LoReC	18.5	13.6	9.4	6.8

The shallowest layer group changes only slightly, from 18.4% to 18.5%, while attention increases in both groups containing directly intervened layers. More notably, graph-token attention in the deepest group increases from 3.1% to 6.8%, despite this group receiving no direct intervention. This downstream change is consistent with the effects of LoReC propagating beyond the layers it directly modifies.

Together, the temporal and layer-wise analyses show attention changes aligned with the motivation of LoReC: graph-token attention remains higher during generation, and the increase extends to deeper layers outside the intervention ranges.

B Computational Costs Analysis

LoReC operates "Look" and "Remember" dynamically based on uncertainty. Specifically, when uncertainty remains low across all layers, indicating high model confidence, these operations are not triggered. This mechanism efficiently enhances model's comprehension of graph data without extra computation. Similarly, for the "Contrast" operation, we employ a gate function $\mathbb{I}_{\text{gate}}$ to dynamically trigger graph-contrast logits calculation only when needed, thereby optimizing both computational time and memory consumption. Overall, while LoReC incurs a modest increase in time and memory costs, it achieves substantial performance improvements. The comparisons on latency, time cost, and memory are shown in Table 7.

Table 7: Performance comparison of different methods and LoReC in latency, time cost, memory usage, and accuracy.

Method	Latency (ms/token)	Memory (GB)	Time Cost(s)	Accuracy(%)
LoReC	82.69	27.93	280.65	71.64
Qwen3-8b	13.67	18.22	259.15	65.91
GraphPrompter	45.23	18.43	201.11	66.52

To further evaluate the computational overhead of LoReC, we extend the inference cost comparison with GraphPrompter to two additional datasets, Cora and Arxiv. Table 8 reports the latency per generated token, memory usage, and accuracy.

Table 8: Inference cost comparison between GraphPrompter and GraphPrompter with LoReC on Cora and Arxiv.

Dataset	Method	Latency (ms/token)	Memory (GB)	Accuracy (%)
Cora	GraphPrompter	37.87	16.72	79.15
	+ LoReC	45.86	20.29	81.37
Arxiv	GraphPrompter	50.48	16.87	73.31
	+ LoReC	58.64	20.21	74.37

C Hyper-parameters Analysis

C.1 Complete Configurations

Following GraphGPT [4] and GraphPrompter [19], we adopt their reported parameter initializations and only optimize the hyperparameters specific to LoReC. For GraphGPT, the edge dropout probability μ ranges from 0.2 to 0.3 to maintain essential graph structural integrity. Additionally, we adaptively change the edge threshold across different datasets to avoid removing vital information from sparse graphs. All detailed dataset-specific hyperparameter configurations are summarized in Table 9.

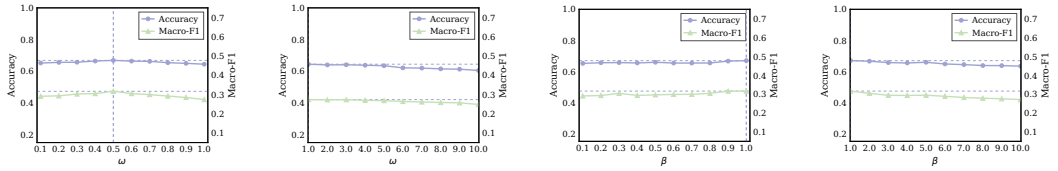
Table 9: Parameter settings for LoReC across different base models and datasets.

Base Model	Dataset	μ	τ	ω	β	η	α	Entropy Threshold	Edge Threshold
GraphGPT	Arxiv	0.2	0.7	0.5	1.0	0.2	0.25	0.75	10
	PubMed	0.2	0.7	1.0	1.0	0.1	0.1	0.75	10
	Cora	0.3	0.7	0.8	1.0	0.15	0.2	0.75	20
GraphPrompter	Cora	0.2	0.7	1.0	1.0	0.1	0.1	0.75	10
	Citeseer	0.2	0.7	1.0	1.0	0.1	0.1	0.75	10
	Arxiv	0.2	0.7	1.0	1.0	0.1	0.1	0.75	10
	Products	0.2	0.7	1.0	1.0	0.1	0.1	0.75	10

C.2 Effect of Dual-contrastive Decoding Strength.

To identify optimal settings for the contrastive weights ω and β, we perform a grid search over the range [0, 10.0]. As illustrated in Fig. 6b and Fig. 6d, we observe that while moderate contrastive strength enhances performance, excessive values (> 1.0) can compromise the model's original inference capability by over-correcting the logits. Specifically, the contrastive divergence terms tend

to overwhelm the final logit distribution, causing severe over-correction that distorts the model's original inference capability. Therefore, we conduct a refined grid search over the range [0.1, 1.0] to determine optimal values. As demonstrated in Fig. 6a and Fig. 6c, the optimal text-contrastive magnitude is approximately 0.5, while the optimal graph-contrastive magnitude is around 1.0.



(a) (b) (c) (d)

Figure 6: (a-b) Results under different text-contrastive magnitudes ω; (c-d) Results under different graph-contrastive magnitudes β.

C.3 Effect of Entropy Threshold and Edge Threshold.

Figure 7 demonstrates the impact of entropy threshold and edge threshold. For the entropy threshold, which determines when to activate the "*Look*" and "*Remember*" modules, exhibits optimal performance around 0.75 within an effective range of [0.65, 0.90]. The edge threshold serves as the gating function $\mathbb{I}_{\text{gate}}$ in Eq. (14) to determine whether graph-contrastive decoding should be applied, which achieves peak performance at approximately 10 edges. Understandably, a high edge threshold makes graph-contrastive decoding hard to trigger, while a low edge threshold would remove vital information from sparse graphs.

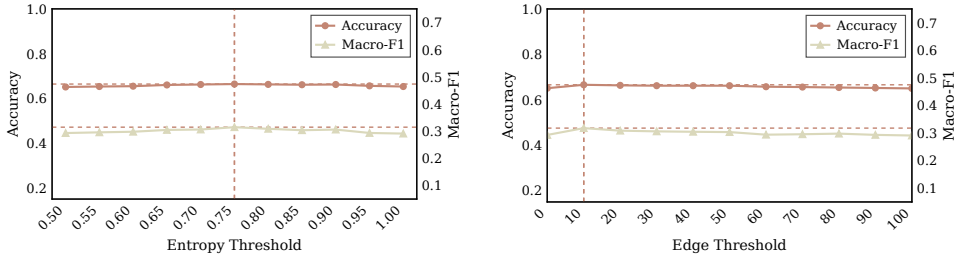


Figure 7: (Left) Results under different entropy threshold. (Right) Results under different edge threshold. Both evaluated on the Arxiv dataset with GraphGPT as the base model.

C.4 Ablation of the Two Contrastive Branches

To assess the contribution of each branch in Dual-Contrastive Decoding, we conduct additional ablation experiments on Arxiv. Specifically, we evaluate text-contrastive decoding alone by setting $\beta = 0$, and graph-contrastive decoding alone by setting $\omega = 0$. All other experimental settings remain unchanged.

Table 10: Ablation of the two contrastive branches on Arxiv.

Method	Accuracy	Macro-F1
GraphGPT	0.6476	0.2854
+ Text-Contrastive only ($\beta = 0$)	0.6562	0.2956
+ Graph-Contrastive only ($\omega = 0$)	0.6533	0.2924
+ Dual-Contrastive (full)	**0.6588**	**0.2994**

As shown in Table 10, both branches individually improve Accuracy and Macro-F1 over GraphGPT. Text-contrastive decoding yields gains of 0.86 and 1.02 percentage points, respectively, while graph-contrastive decoding yields gains of 0.57 and 0.70 percentage points. Combining both branches

achieves the strongest performance, improving Accuracy by 1.12 percentage points and Macro-F1 by 1.40 percentage points over the baseline. These results provide empirical evidence that each branch contributes to predictive performance and that their combination offers additional benefits.

C.5 Limitation

Although LoReC brings consistent and notable improvements over current GraphLLM methods, it currently focuses on node classification tasks and has not been extended to other important graph tasks such as link prediction and graph classification. Exploring the applicability of LoReC to these tasks could further broaden its impact within the GraphLLM paradigm. Moreover, it introduces several critical hyperparameters that influence the final performance. Determining the optimal settings for these enhancements remains a challenging task. This indicates that there is still room for improvement in our model's performance due to the impact of hyperparameters.

D Additional Experimental Results

D.1 Evaluation with a Smaller Language Model

To examine whether LoReC remains effective with reduced language model capacity, we replace the Qwen3-8b backbone of GraphPrompter with Qwen3-4b and evaluate both the base model and its LoReC-enhanced counterpart on Arxiv and Citeseer.

Table 11: Performance with the smaller Qwen3-4b backbone.

Model	Arxiv		Citeseer	
	Accuracy	Macro-F1	Accuracy	Macro-F1
GraphPrompter (Qwen3-4b)	0.7029	0.3783	0.6525	0.6272
LoReC (GraphPrompter-Qwen3-4b)	**0.7194**	**0.4387**	**0.6961**	**0.6559**

As shown in Table 11, LoReC improves both metrics on both datasets. On Arxiv, Accuracy and Macro-F1 increase by 1.65 and 6.04 percentage points, respectively; on Citeseer, the corresponding gains are 4.36 and 2.87 percentage points. Together with the Qwen3-8b results in Table 3, these findings support the effectiveness of LoReC across the two tested backbone sizes.

D.2 Evaluation on Real-World Social Networks

To further evaluate applicability beyond citation networks, we conduct additional experiments on two real-world social network datasets, Instagram and Reddit.

Table 12: Performance on the real-world social network datasets.

Model	Instagram		Reddit	
	Accuracy	Macro-F1	Accuracy	Macro-F1
GraphPrompter	0.5950	0.5098	0.6203	0.6168
GraphPrompter + LoReC	**0.6291**	**0.5162**	**0.6630**	**0.6614**

Table 12 shows improvements on both datasets. LoReC increases Accuracy by 3.41 percentage points on Instagram and 4.27 percentage points on Reddit, while improving Macro-F1 by 0.64 and 4.46 percentage points, respectively. These results extend the empirical evidence for LoReC from citation and e-commerce graphs to social networks.

D.3 Comparison with Recent Methods

We further compare LoReC with three recent methods, LLaGA, GraphAdapter, and GOFA, on Arxiv and Cora. LLaGA and GraphAdapter are retrained using Qwen3-8b, matching the backbone of the LoReC configuration evaluated here. For GOFA, we report results from its original paper, which uses Llama2-13b and does not report Macro-F1.

Table 13: Comparison with recent methods.

Model	Arxiv		Cora	
	Accuracy	Macro-F1	Accuracy	Macro-F1
LLaGA	0.7261	0.4513	0.7563	0.7348
GraphAdapter	0.7358	0.4584	0.7248	0.7023
GOFA	**0.7477**	–	0.7650	–
GraphPrompter + LoReC	0.7437	**0.4674**	**0.8026**	**0.7762**

As shown in Table 13, GraphPrompter enhanced with LoReC outperforms LLaGA and GraphAdapter in both Accuracy and Macro-F1 on both datasets under the same Qwen3-8b backbone. Compared with the reported GOFA results, it achieves 0.40 percentage points lower Accuracy on Arxiv and 3.76 percentage points higher Accuracy on Cora, while using an 8B rather than a 13B language model. Overall, these results demonstrate the competitiveness of the LoReC-enhanced model against the evaluated recent methods.

D.4 Extension to Link Prediction

To evaluate LoReC beyond node classification, we conduct link prediction experiments on Cora, Citeseer, and PubMed. We apply the same LoReC decoding procedure without task-specific modifications to the method.

Table 14: Link prediction accuracy.

Model	Cora	Citeseer	PubMed
	Accuracy	Accuracy	Accuracy
GraphPrompter	0.7682	0.8413	0.9041
GraphPrompter + LoReC	**0.8019**	**0.8732**	**0.9244**

Table 14 shows that LoReC improves link prediction Accuracy on all three datasets, with gains of 3.37, 3.19, and 2.03 percentage points on Cora, Citeseer, and PubMed, respectively. These results demonstrate that its empirical benefits extend beyond node classification to the evaluated link prediction settings. They are consistent with the motivation that improving the use of graph information during decoding can benefit multiple graph reasoning tasks.

E Pseudo Codes of LoReC

We present the algorithm pipeline as follows. The equations mentioned in the algorithm can be seen in the main text.

Algorithm 1 *Look*: Attention Redistribution

Input: Graph tokens $\mathbf{C}_{\mathcal{G}}$, Text tokens $\mathbf{C}_{\mathcal{T}}$, Model $\mathcal{M}$.
At every decoding step t:
Initial set *trigger* = False.
for $l = 0$ **to** $L - 1$ **do**
  $\mathcal{H}_t^{(l)} = -\frac{1}{\log N} \sum_{i=1}^{N} p_\theta^{(l)} \log p_\theta^{(l)}$ (Eq. (5)).
  if $\mathcal{H}_t^{(l)} > \gamma$ **then**
    trigger = True.
  end if
end for
for $l = l + 1$ **to** $L - 1$ **do**
  Select graph attention logits $\mathbf{e}_j^{(l)}$ at layer l.
  Execute $\tilde{e}_j^{(l)} = e_j^{(l)} + \eta \, |e_j^{(l)}|$ (Eq. (6)).
  Recalculate attention weights $\tilde{\mathcal{A}}^{(l)} = \text{Softmax}(\tilde{\mathbf{e}}^{(l)})$.
end for
Output: Redistributed attention weights $\tilde{\mathcal{A}}$.

Algorithm 2 *Remember*: Graph Re-injection

Input: Input embedding $x^{(0)}$, Graph Tokens $\mathbf{C}_{\mathcal{G}}$.
Initial set *trigger* = False.
for $l = 0$ **to** $L - 1$ **do**
  $\mathbf{FFN}_{\text{base}} = \phi(\mathbf{x}^{(l)} \mathbf{W}_1^{(l)}) \mathbf{W}_2^{(l)}$;
  $\mathcal{H}_t^{(l)} = -\frac{1}{\log N} \sum_{i=1}^{N} p_{\theta,i}^{(l)} \log p_{\theta,i}^{(l)}$ (Eq. (5)).
  if $\mathcal{H}_t^{(l)} > \gamma$ and *trigger* == False **then**
    trigger = True.
    $\mathbf{W}_1^g \leftarrow \mathbf{C}_{\mathcal{G}}, \quad \mathbf{W}_2^g \leftarrow \mathbf{C}_{\mathcal{G}}^\top$;
    $\mathbf{Mem}_{\text{graph}} = \phi(\mathbf{x}^{(l)} \mathbf{W}_1^g) \mathbf{W}_2^g$;
    $\mathbf{FFN}_{\text{final}} = (1 - \alpha) \cdot \mathbf{FFN}_{\text{base}} + \alpha \cdot \mathbf{Mem}_{\text{graph}}$.
  else
    $\mathbf{FFN}_{\text{final}} = \mathbf{FFN}_{\text{base}}$.
  end if
end for
Output: Final hidden state $x^{(L)}$.

Algorithm 3 LoReC Strategy

Input: Graph tokens $\mathbf{C}_{\mathcal{G}}$, Text tokens $\mathbf{C}_{\mathcal{T}}$, Model $\mathcal{M}$.
Construct perturbed graph $\tilde{\mathcal{G}}$ via Adaptive Augmentation (Eq. (13)).
At every decoding step t:
for $l = 0$ **to** $L - 1$ **do**
  Compute Uncertainty $\mathcal{H}_t^{(l)}$ (Eq. (5)).
  if $\mathcal{H}_t^{(l)} > \gamma$ **then**
    Apply **Look**: Reallocate Attention (Eq. (6)).
    Apply **Remember**: Inject Graph (Eq. (11)).
  end if
  Calculate logits $\mathbf{v}_{\text{orig}} = \mathcal{M}(\mathcal{G}, \mathbf{x}_{<t})$.
  Calculate logits $\mathbf{v}_{\text{text}} = \mathcal{M}(\emptyset, \mathbf{x}_{<t})$.
  Calculate logits $\mathbf{v}_{\text{aug}} = \mathcal{M}(\tilde{\mathcal{G}}, \mathbf{x}_{<t})$.
  Apply **Contrast**: $\mathbf{v}_{\text{final}} \leftarrow \mathbf{v}_{\text{orig}}, \mathbf{v}_{\text{text}}, \mathbf{v}_{\text{aug}}$ (Eq. (14)).
  Sample $y_t \sim \text{Softmax}(\mathbf{v}_{\text{final}})$.
  $\mathbf{x}_{<t+1} \leftarrow \text{Concat}(\mathbf{x}_{<t}, y_t)$.
end for
Output: Model prediction y_t