BRIDGING GRAPH AND LANGUAGE STRUCTURES: STRUCTURE-AWARE FINE-TUNING FOR GRAPH REASONING WITH LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Serializing graphs as text enables large language models (LLMs) to perform graph reasoning, but structural differences between graphs and sequences limit effective use of graph information. We propose a structure-aware fine-tuning framework for decoder LLMs that addresses three key issues: insufficient global readouts, mismatch between sequence positions and graph relations, and unequal information aggregation under causal attention. The framework introduces hierarchical graph serialization to organize local and global structural information, structural role rotary position encoding to reduce permutation-sensitive positional bias, and a structural visibility mask with learnable attention biases to control information flow across edge, node, and graph levels. A thresholded readout decorrelation loss further encourages complementary global representations while retaining necessary information sharing. The framework requires no additional graph encoder and supports parameter-efficient fine-tuning of pretrained LLMs. Under the specified evaluation protocol, it improves macro-average accuracy by 20.88 and 14.11 percentage points on GraphInstruct and MolecularNet, respectively, over a recent attention-sink-based method, while achieving competitive performance on node classification and link prediction. Ablation studies further confirm the contribution of each component. These results demonstrate that jointly adapting graph serialization, positional relations, and structural information flow can substantially improve graph reasoning with LLMs.

1 INTRODUCTION

LLM-based graph reasoning has become an emerging research direction in the field of graph learning. Traditional graph neural network (GNN) methods rely on specialized graph encoders to capture topological structure, and their capabilities are limited by the scale of training data and task-specific supervision signals. With powerful semantic understanding and generalization capabilities, LLMs offer new possibilities for graph tasks. However, LLMs are natively designed for serialized text, so how to translate graph data with two-dimensional topological structure into input that LLMs can understand constitutes a core challenge in current research. Against this backdrop, a class of methods that do not require additional graph encoders has gradually attracted attention. The core idea of such methods is to directly serialize graphs into text sequences or other special token sequences, and leverage the language understanding and reasoning capabilities of LLMs themselves to accomplish downstream tasks. Compared with hybrid architectures in which LLMs and GNNs collaborate, this strategy has clear advantages in deployment simplicity and architectural uniformity.

Existing studies have shown that graph serialization methods can effectively improve the performance of downstream graph reasoning tasks: (Dai et al., 2025) found through substructure extraction tasks that although adjacency lists and edge lists can represent the same graph, their learning efficiency differs under the settings examined; (Guo et al., 2026) combined reversible graph serialization with byte pair encoding to transform frequent substructures into reusable tokens; (Xu et al., 2026) examined the interaction among graph types, serialization formats, and prompting strategies through the GraphOmni system, finding that there is no universally optimal order for input effectiveness, as it also depends on the task and model. These studies establish the importance of graph serialization from the perspectives of mechanism analysis, representation design, and systematic

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

evaluation. In fact, the effective utilization of structural information depends not only on the graph serialization method but is also closely related to the internal information allocation of LLMs: (Liu et al., 2026b) found that when serialization is properly grouped by source nodes, the attention of some intermediate layers exhibits a "zigzag" pattern corresponding to the local graph structure; (Gu et al., 2025) showed that attention sinks already emerge during pure language pretraining, and their formation is related to optimization, data distribution, and Softmax normalization. As a result, recent studies have begun to directly modify the structural inductive bias of LLMs, such as (Wang et al., 2025a)'s UniGTE, which uses shared graph positions, structural attention biases, and aligned tokens to construct task-relevant graph representations; (Loveland et al., 2026), which analyzes the attention distortion caused by the interaction between linearization and RoPE through graph bandwidth analysis and injects graph structural biases into frozen LLMs; and (Liu et al., 2026b), which improves graph reasoning by attenuating attention sink weights. The full related work discussion is provided in Appendix A.

These observations organize prior work into three related aspects: sequence organization, positional encoding, and attention aggregation. Existing studies emphasize different subsets of these aspects, leaving their joint treatment less developed. We systematically analyze the specific differences between graphs and language along all three aspects and use them to design an integrated adaptation for decoder-only LLMs. The appendix B provides an illustration of these three differences and their corresponding design motivations.

Overall, the first difference highlights the need for multiple nonredundant tokens that can access the complete graph structure. The second concerns the sensitivity of graph tokens to positional permutations under existing positional encoding. The third motivates causal masks that support balanced aggregation scopes for graph tokens. These issues are interdependent: graph serialization affects both positional adaptation and the scope of causal masking, while positional adaptation can also influence the effect of the mask. We address the three differences through structure aware fine tuning for graph reasoning with LLMs. Our hierarchical serialization organizes edge records and edge summaries, node blocks and node summaries, and multiple global readouts, separating locally ordered descriptions from aggregation over sets. We then introduce structural role rotary position encoding (SR-RoPE), which preserves native positional rules in language regions, represents internal order with local positions, and assigns shared positional anchors to tokens with the same role. Our structural attention (SA) mechanism defines visibility relations at the edge, node, and graph levels and regulates aggregation weights through learnable biases. Finally, we augment the answer generation objective with a thresholded readout decorrelation loss to discourage excessive similarity among global readouts. Together, these components adapt input organization, positional relations, attention information flow, and representation constraints within a framework for parameter efficient fine tuning of pretrained LLMs. Our main contributions are as follows:

- We characterize the structural differences between graph serialization and decoder LLMs in terms of readout organization, positional dependencies, and prefix visibility, explicitly distinguishing permutation robustness, aggregation balance, and readout complementarity.
- We introduce hierarchical graph serialization, SR-RoPE, structural attention, and a thresholded readout decorrelation objective to explicitly organize the aggregation of graph information while preserving native attention projections and autoregressive answer generation.
- We report comparisons across backbones, sequential component ablations, and hyperparameter analyses on four different task scenarios. We also examine the scope of the method through mechanistic discussion and illustrations. The experiments support performance improvements under the settings examined.

2 METHOD

Let $G = (V, E, \mathcal{R}, X_V, X_E)$ be a directed attributed graph, and given a query Q, a serialization $S(G)$, and an answer $Y = (y_1, \ldots, y_T)$, we learn $P_\theta(Y \mid Q, S(G))$. Appendix C gives the full preliminaries. As shown in Figure 1, our structure-aware fine-tuning integrates graph hierarchies into a decoder LLM through four coordinated modules without an extra graph encoder. Hierarchical graph serialization (Section 2.1) groups each node header and its incident edges into a node block, places edge and node summaries at the ends of edge records and node blocks, and appends multiple global

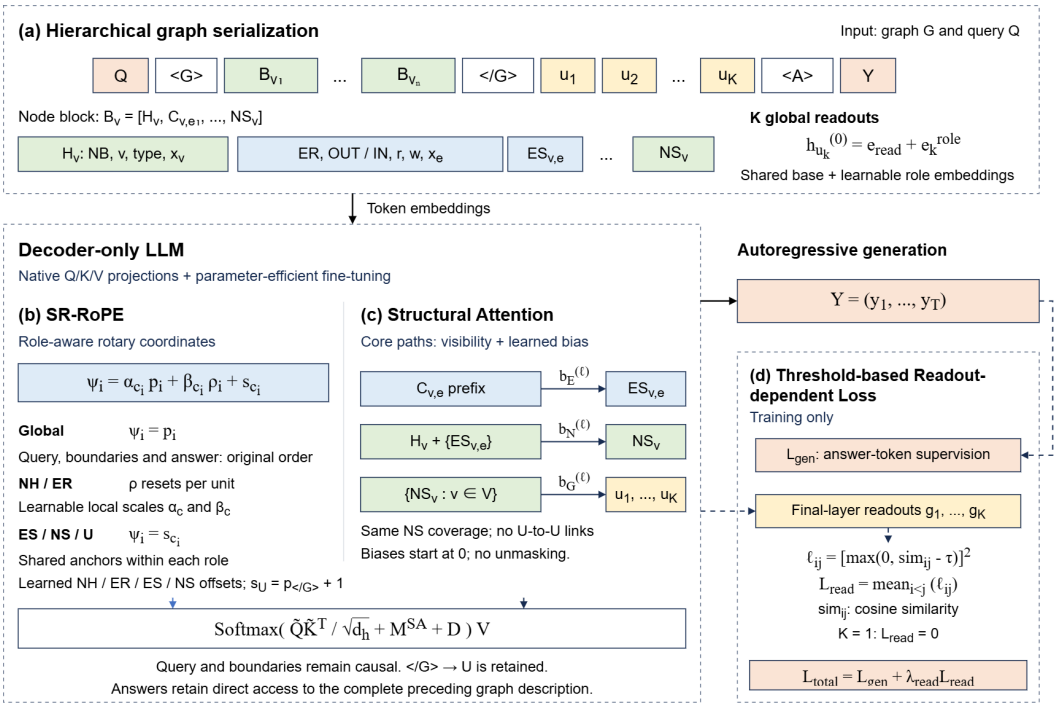

Figure 1: Overview of structure aware fine tuning. Hierarchical graph serialization (Section 2.1) organizes node headers, edge records, edge summaries, node summaries, and multiple global readouts. Structural role rotary position encoding (Section 2.2) and structural attention (Section 2.3) operate within the attention computation of the same decoder LLM, respectively. During training, the thresholded readout decorrelation loss (Section 2.4) is jointly optimized with the generation loss on the answer region.

readouts with independent role embeddings after the graph description. SR-RoPE (Section 2.2) preserves native positional rules for the query, boundaries, and answer, uses local coordinates and learnable scaling for order within node headers and edge records, and assigns shared positional anchors to same-role summaries and readouts, reducing the effect of semantically insignificant permutations. Structural attention (Section 2.3) works with SR-RoPE via a structural visibility mask and learnable biases, routing information hierarchically from edge records to edge summaries, from node headers and edge summaries to node summaries, and from node summaries to global readouts, with identical node summary coverage per readout and no direct connections between readouts. The thresholded readout decorrelation loss (Section 2.4) penalizes only readout pairs whose final hidden representations exceed a cosine similarity threshold, jointly optimizing with the autoregressive generation loss on the answer region to reduce readout redundancy while allowing necessary information sharing.

2.1 Hierarchical graph serialization

We use the following complete sequence:

$$X = [Q, S(G), \langle A \rangle, Y], \qquad S(G) = [\langle G \rangle, B, \langle /G \rangle, U]. \tag{1}$$

Here, $B = [B_{v_1}, \ldots, B_{v_n}]$ is the sequence of node blocks, $U = [u_1, \ldots, u_K]$ is the sequence of global readouts, $\langle G \rangle$ and $\langle /G \rangle$ delimit the graph region, and $\langle A \rangle$ marks the beginning of the answer. Square brackets denote concatenation in the specified order. When defining attention sets, the same symbols denote sets of the corresponding token occurrence positions, distinguishing different occurrences with identical content.

2.1.1 Node blocks and edge records

For a central node v, the node block is defined as

$$B_v = [H_v, C_{v,e_1}, \ldots, C_{v,e_{d_v}}, \mathrm{NS}_v], \tag{2}$$

where C_{v,e_1} is an edge record for node v corresponding to edge e_1, and d_v is the number of encoded incident edge records. An isolated node has $d_v = 0$ but retains its header and node summary. The header H_v contains the node identifier and any type and attribute fields; each edge record specifies its direction relative to v, relation, neighboring node, and any edge attributes, followed by an edge summary $\mathrm{ES}_{v,e}$. Appendix D.1 provides the exact field definitions and a serialization example.

Within each edge record, the relation, direction, and node identifier follow a fixed semantic order. Different incident edge records, however, generally have no natural order. Placing an edge summary at the end of its record allows it to aggregate information from that record. Placing a node summary at the end of its node block allows it to read all edge summaries. The header precedes the edge records so that each record can access information about the central node. This organization separates local text encoding from structural aggregation over sets and establishes explicit boundaries for the masking rules that follow.

2.1.2 MULTIPLE GLOBAL READOUTS

We place K global readouts after all node blocks and initialize their representations as

$$\mathbf{h}_{u_k}^{(0)} = \mathbf{e}_{\mathrm{read}} + \mathbf{e}_k^{\mathrm{role}}, \qquad k = 1, \ldots, K, \tag{3}$$

where $\mathbf{e}_{\mathrm{read}}$ is a shared base embedding and $\mathbf{e}_k^{\mathrm{role}}$ is an independently learnable role embedding. The shared embedding represents the common function of each readout token. The role embeddings break the initial symmetry among readouts. We do not assign predefined semantics, such as paths, connectivity, or motifs, to individual readouts, nor do we regard the shared embedding itself as a guarantee against collapse.

2.2 SR-ROPE

In a serialized graph, different regions have different ordering properties. If every token uses its physical position p_i, permutations without semantic significance in the graph are mapped to different RoPE relative distances. We therefore propose SR-RoPE to distinguish locally ordered text from structural summaries treated as sets while preserving the original RoPE mechanism as much as possible. Its central idea is to retain the rotation function of RoPE while jointly determining its input coordinate from the global physical sequence position, local structural position, structural role, and positional anchor of each token, formally expressed as $\psi_i = \alpha_{c_i} p_i + \beta_{c_i} \rho_i + s_{c_i}$. Here, c_i is the structural role, p_i is the global physical position, ρ_i is the local position, α_{c_i} and β_{c_i} weight the two types of positions, and s_{c_i} is the role anchor.

2.2.1 MUTUALLY EXCLUSIVE STRUCTURAL ROLES AND LOCAL COORDINATES

The six mutually exclusive roles are Global (query, graph boundaries, answer marker, and answer), Local (node header (NH) and edge record excluding its summary (ER)), and Unordered (edge summary (ES), node summary (NS), and global readout (U)). Appendix D.2.1 gives the complete assignment. For locally ordered roles, ρ_i starts at 0 at the beginning of the corresponding header or edge record and increases by one for each successive token within that unit. The count restarts independently for each unit. For all other roles, $\rho_i = 0$. This rule preserves order within fields without encoding the permutation of records in the local coordinates. The fixed reference configuration for the roles is

$$(\alpha_c, \beta_c) = \begin{cases} (1, 0), & c = \mathrm{Global}, \\ (0, 1), & c \in \{\mathrm{NH}, \mathrm{ER}\}, \\ (0, 0), & c \in \{\mathrm{ES}, \mathrm{NS}, \mathrm{U}\}. \end{cases} \tag{4}$$

However, removing global positions entirely or fully retaining local positions may be overly rigid. For example, in graphs with short textual node attributes or simple relation descriptions, a small amount of global order information may still help the model maintain contextual continuity. We therefore generalize α_c and β_c to learnable scaling factors for different roles: $\alpha_c = \sigma(a_c)$ and $\beta_c = \sigma(b_c)$. Here, $c \in \{\mathrm{NH}, \mathrm{ER}\}$, a_c, b_c are learnable scalars. With finite parameters, Sigmoid takes values in $(0, 1)$. Thus, the values 0 and 1 in Eq. equation 4 are reference endpoints approximated at initialization, rather than values attained exactly by finite Sigmoid parameters. Global and unordered roles retain their fixed configurations.

2.2.2 ROLE ANCHORS

Role anchors provide a base reference position for every token. As with local coordinates, we specify an anchor for each role. The detailed configuration is

$$\begin{aligned} s_{\mathrm{Global}} &= 0, \\ s_{\mathrm{NH}} &= p_{\langle G\rangle} + 1 + \mathrm{softplus}(\delta_{\mathrm{NH}}), \\ s_{\mathrm{ER}} &= s_{\mathrm{NH}} + \mathrm{softplus}(\delta_{\mathrm{ER}}), \\ s_{\mathrm{ES}} &= s_{\mathrm{ER}} + \mathrm{softplus}(\delta_{\mathrm{ES}}), \\ s_{\mathrm{NS}} &= s_{\mathrm{ES}} + \mathrm{softplus}(\delta_{\mathrm{NS}}), \\ s_{\mathrm{U}} &= p_{\langle /G\rangle} + 1. \end{aligned} \tag{5}$$

Here, $p_{\langle G\rangle}+1$ and $p_{\langle /G\rangle}+1$ denote the physical positions immediately following the token marking the beginning and end of the graph region, respectively. The learnable scalars δ_{NH}, δ_{ER}, δ_{ES}, and δ_{NS} determine positive increments through their softplus transforms; these increments accumulate to give structural offsets from the beginning of the graph region. Summaries of the same type share an anchor. For every token with the Global role, SR-RoPE reduces exactly to the native RoPE of the LLM. SR-RoPE ultimately applies $\widetilde{\mathbf{q}}_i = \mathbf{R}(\psi_i)\mathbf{q}_i$ and $\widetilde{\mathbf{k}}_i = \mathbf{R}(\psi_i)\mathbf{k}_i$ without changing the query, key, or value projections.

2.3 STRUCTURAL ATTENTION

Positional encoding only specifies how positional relationships between token pairs are represented; it does not change the visibility scope determined by the causal mask. We therefore introduce a SA mechanism. Its basic principle is to fuse graph information across structural levels only at predefined aggregation tokens while keeping distinct structural units at the same level isolated. Specifically, a structural visibility mask explicitly defines visibility between token pairs, and a structural attention bias imposes soft structural constraints on token pairs that are allowed to interact.

2.3.1 STRUCTURAL VISIBILITY MASK

Let $\mathrm{Prefix}(Z, x_i)$ denote the set of token occurrence positions in sequence Z that are no later than x_i, including x_i itself. Let $\mathcal{Q} = Q \cup \{\langle G\rangle\}$ denote the query context, and let $E(v)$ be the index set of incident edge records encoded in the block for node v. The set of positions that a structural token is allowed to access is defined as

$$\Gamma(x_i) = \begin{cases} \mathcal{Q} \cup \mathrm{Prefix}(H_v, x_i), & x_i \in H_v, \\ \mathcal{Q} \cup H_v \cup \mathrm{Prefix}(C_{v,e}, x_i), & x_i \in C_{v,e}, \\ \mathcal{Q} \cup H_v \cup \{\mathrm{ES}_{v,e} : e \in E(v)\}, & x_i = \mathrm{NS}_v, \\ \mathcal{Q} \cup \{\mathrm{NS}_v : v \in V\} \cup \{\langle /G\rangle\}, & x_i \in U. \end{cases} \tag{6}$$

A node header preserves internal causality. Each edge record reads only the query, the current node header, and its own prefix. A node summary aggregates the current header and all edge summaries, and all global readouts access the same set of node summaries. For the query, graph boundaries, answer marker, and answer region, we use the native causal visibility rule $\Gamma(x_i) = \mathrm{Prefix}(X, x_i)$. The complete mask can be written as

$$M_{ij}^{\mathrm{SA}} = \begin{cases} 0, & x_j \in \Gamma(x_i), \\ -\infty, & \text{otherwise.} \end{cases} \tag{7}$$

All the visibility connections above point to the current or an earlier position, so they do not allow an answer token to access future answer content. The graph end marker can aggregate the preceding graph description and serve as an input to the global readouts. Similarly, the answer region can directly read the graph description.

2.3.2 STRUCTURAL ATTENTION BIAS

The structural visibility mask specifies visibility between structural token pairs. For pairs that are allowed to interact, however, native Self-Attention does not explicitly encode which visible token

deserves greater attention. We therefore use the explicit structural relationships in graph serialization to define three core types of relationships between interacting token pairs. These are aggregation from an edge record into its edge summary ($\mathcal{P}_E$), from a node header and its edge summaries into the node summary ($\mathcal{P}_N$), and from node summaries into global readouts ($\mathcal{P}_G$); their complete pair-set definitions appear in Appendix D.3.2. At layer ℓ, we construct the bias using three learnable scalars:

$$D_{ij}^{(\ell)} = \sum_{r\in\{E,N,G\}} b_r^{(\ell)} \mathbb{I}[(x_i, x_j) \in \mathcal{P}_r]. \tag{8}$$

Here, $\mathbb{I}[\cdot]$ is the indicator function, and $b_r^{(\ell)}$ is initialized to 0 with no constraint on its sign. This allows the model to strengthen or suppress the corresponding aggregation relationships without assuming in advance that an edge record must be more important than the query.

Combining this mechanism with SR-RoPE gives the final attention:

$$\mathbf{A}^{(\ell,h)} = \mathrm{Softmax}_{\mathrm{row}}\left(\frac{\widetilde{\mathbf{Q}}^{(\ell,h)}(\widetilde{\mathbf{K}}^{(\ell,h)})^\top}{\sqrt{d_h}} + \mathbf{M}^{\mathrm{SA}} + \mathbf{D}^{(\ell)}\right). \tag{9}$$

The output is still computed as $\mathbf{O}^{(\ell,h)} = \mathbf{A}^{(\ell,h)}\mathbf{V}^{(\ell,h)}$. SA modifies visibility relationships and attention scores without introducing a separate graph encoding network.

2.4 Thresholded Readout Decorrelation Loss

During training, we retain the native autoregressive generation objective of the Decoder-only LLM as the primary objective:

$$\mathcal{L}_{\mathrm{gen}} = -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(y_t \mid Q, S(G), y_{<t}). \tag{10}$$

The generation loss is computed only over the answer region, with the query and graph description serving as conditioning context. However, this loss alone does not ensure that different global readout token representations capture complementary graph information. To this end, we introduce the Thresholded Readout Decorrelation Loss, which penalizes a pair of readouts only when their representation similarity exceeds a specified threshold. Denote the global readout representation at layer N_L by $\mathbf{g}_k = \mathbf{h}_{u_k}^{(N_L)}$. We use $\widehat{\mathbf{g}}_k = \mathbf{g}_k / \max(\|\mathbf{g}_k\|_2, \varepsilon)$ and $\mathrm{sim}_{ij} = \widehat{\mathbf{g}}_i^\top \widehat{\mathbf{g}}_j$, where $\varepsilon > 0$ provides numerical stability. We further introduce a similarity threshold $\tau \in [0, 1)$ and define the thresholded readout decorrelation loss as

$$\mathcal{L}_{\mathrm{read}} = \begin{cases} \dfrac{2}{K(K-1)} \displaystyle\sum_{1\le i<j\le K} \big[\max(0, \mathrm{sim}_{ij} - \tau)\big]^2, & K \ge 2, \\ 0, & K = 1. \end{cases} \tag{11}$$

When two readout representations satisfy $\mathrm{sim}_{ij} \le \tau$, their pair incurs no additional regularization loss. When $\mathrm{sim}_{ij} > \tau$, the regularization penalty increases quadratically with the amount by which their similarity exceeds the threshold, discouraging different global readout token representations from converging to highly similar states. Our overall training objective combines the autoregressive generation loss with the thresholded readout decorrelation loss:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{gen}} + \lambda_{\mathrm{read}}\mathcal{L}_{\mathrm{read}}, \qquad \lambda_{\mathrm{read}} \ge 0. \tag{12}$$

The auxiliary regularizer $\mathcal{L}_{\mathrm{read}}$ does not prescribe the structural semantics that each readout token should learn; it only discourages excessive redundancy among readouts. This allows different global readout token representations to develop distinct structural emphases while jointly serving the final generation task.

3 Experiments

3.1 Setup and evaluation protocol

We evaluate nine GraphInstruct tasks (Chen et al., 2024) and five MolecularNet datasets (Wu et al., 2018) using accuracy, plus node classification on eight datasets and link prediction on seven using

Table 1: Macro-average accuracy (0–1). Each GraphInstruct mean covers nine tasks; each MolecularNet mean covers five datasets. †: published results (Liu et al., 2026b). Full task means and standard deviations appear in Tables 4 and 6. A dash denotes an unreported combination.

	GraphInstruct			MolecularNet		
Backbone	Vanilla†	SLASH†	Ours	Vanilla†	SLASH†	Ours
Llama-3.2-3B	0.085	0.214	**0.456**	0.521	0.643	**0.718**
Llama-3.1-8B	0.282	0.352	**0.528**	0.616	0.644	**0.796**
Qwen3-4B	0.151	0.195	**0.442**	0.345	0.642	**0.727**
Qwen3-8B	0.174	0.367	**0.511**	0.063	0.544	**0.785**
Qwen3-14B	0.202	0.298	**0.532**	0.619	0.670	**0.823**
GraphWiz-Mistral-7B	0.544	0.544	**0.680**	–	–	–
GraphWiz-LLaMA2-7B	0.642	0.639	**0.706**	–	–	–
GraphWiz-LLaMA2-13B	0.637	0.642	**0.743**	–	–	–
MolecularGPT	–	–	–	0.364	0.366	**0.682**

accuracy and AUC-ROC. For transfer, the target dataset is excluded from source training (Liu et al., 2026a). Backbones include Llama and Qwen3 (Grattafiori et al., 2024; Yang et al., 2025), with GraphWiz and MolecularGPT as task-adapted comparisons (Chen et al., 2024; Liu et al., 2024b). Appendices E and J specify all datasets, baselines, splits, and graph sizes.

We freeze pretrained weights and train query/value LoRA (rank 8, scaling 16, dropout 0.05), structural parameters, and token embeddings for one epoch. Defaults are $K = 4$, $\tau = 0.6$, and $\lambda_{\text{read}} = 0.01$. Training randomly permutes node blocks and incident records. Link-prediction context uses positive training edges and excludes the predicted edge. Appendix E.4 details optimization, hardware, and input construction.

Our results use five seeds $\{42, 43, 44, 45, 46\}$; full tables report means and standard deviations, and summaries use unweighted macro averages. Published baselines (†) retain their original reporting. It should be noted that our alignment with the baseline models does not account for specific training budgets or preprocessing pipelines.

3.2 Main results

3.2.1 Graph reasoning across backbones

Table 1 summarizes every backbone from the full GraphInstruct and MolecularNet tables in Appendix F.1. Across five general-purpose backbones and nine GraphInstruct tasks, our method improves on Vanilla and SLASH by 31.51 and 20.88 percentage points on average, respectively. It exceeds SLASH in 43 of 45 comparisons (95.6%). Across all eight GraphInstruct backbones, the gain over SLASH is 16.85 points, with 69 wins in 72 comparisons. Task-adapted GraphWiz configurations also improve in macro accuracy, suggesting that structural adaptation can complement prior instruction tuning.

These gains are uneven. On GraphInstruct, general-purpose Topology accuracy remains 0.024–0.089; Llama-3.2-3B and Qwen3-14B fall below SLASH on this task. Shortest and Flow also remain difficult. Llama-3.1-8B gains 48.2 points over SLASH on Hamilton but only 5.8 on Shortest. Thus, higher averages do not establish reliable mastery of topological sorting, exact paths, or flow computation.

On MolecularNet, our method exceeds both Vanilla and SLASH in all 25 general-purpose backbone/dataset comparisons, with respective average gains of 33.67 and 14.11 points. Qwen3-14B reaches 0.823 macro accuracy. MolecularGPT reaches 0.682, a 31.60-point gain over SLASH, but its ClinTox result remains below several general-purpose backbones. For Llama-3.1-8B, the gain on BACE is 45.7 points versus 2.2 on HIV. Domain adaptation and structural adaptation therefore interact with the task and available baseline headroom.

Table 2: Transfer across datasets on standard graph tasks: macro-average node-classification accuracy (NC, %, eight datasets) and link-prediction AUC-ROC (LP, %, seven datasets). Published aggregates (†) are retained from the source studies; full comparisons include additional NC baselines.

Method	NC	LP
GCOPE† (Zhao et al., 2024)	35.38	80.80
SAMGPT† (Yu et al., 2025a)	36.89	73.44
MDGFM† (Wang et al., 2025b)	35.09	75.88
RiemannGFM† (Sun et al., 2025)	39.31	82.38
R-GFM† (Liu et al., 2026a)	46.84	86.90
Ours (Llama-3.1-8B)	46.75	87.37
Ours (Qwen3-8B)	**47.76**	**87.50**

Table 3: Sequential component additions with fixed training budgets and splits within each backbone. GI and MN denote GraphInstruct and MolecularNet accuracy (0–1); NC is accuracy (%) and LP is AUC-ROC (%). All entries are macro averages.

Configuration	GI	MN	NC	LP
Base (Llama-3.1-8B)	0.317	0.623	35.82	80.14
+ Hierarchical serialization	0.385	0.695	38.56	83.28
+ SR-RoPE	0.445	0.740	42.02	85.66
+ Structural mask	0.505	0.780	46.18	86.82
+ Structural bias	0.518	0.788	46.54	87.12
+ Readout decorrelation (Full)	0.528	0.796	46.75	87.37
Base (Qwen3-8B)	0.189	0.104	36.54	81.26
+ Hierarchical serialization	0.305	0.525	39.12	84.36
+ SR-RoPE	0.385	0.625	44.48	86.02
+ Structural mask	0.465	0.725	46.92	87.04
+ Structural bias	0.490	0.755	47.42	87.32
+ Readout decorrelation (Full)	0.511	0.785	47.76	87.50

3.2.2 Competitiveness across datasets on standard graph tasks

For standard graph tasks, Table 2 compares aggregate results with published graph foundation models; all methods and dataset-level results remain in Tables 7 and 9 in Appendix F.2. Qwen3-8B reaches 47.76% node-classification accuracy, approximately 0.92 points above R-GFM, and has higher means on all eight datasets. Llama-3.1-8B reaches 46.75%, slightly below R-GFM's 46.84%, despite improving on five datasets. On link prediction, Llama-3.1-8B and Qwen3-8B reach 87.37% and 87.50% AUC-ROC, gains of 0.47 and 0.60 points over R-GFM. Each wins on five of seven datasets. These results indicate competitive transfer performance with dataset- and backbone-dependent tradeoffs.

The dataset-level rankings further qualify the aggregate results. Qwen3-8B has the highest node-classification means on Citeseer, Cora, Pubmed, Computers, and Texas, while Llama-3.1-8B leads on Wisconsin, Cornell, and Photos. For link prediction, Qwen3-8B improves over R-GFM by 1.95 points on Cora and 0.88 on Texas, but declines by 0.56 on Pubmed and 0.04 on Citeseer. Llama-3.1-8B falls slightly below R-GFM on Pubmed and Photos. An advantage for one backbone on selected datasets therefore does not imply uniform superiority across transfer settings.

3.3 Sequential ablations

Table 3 adds components sequentially with fixed splits and training budgets within each backbone. Base is a trained control using basic text serialization, native RoPE, standard causal attention, and generation loss, distinct from the literature's Vanilla configuration.



Figure 2: Hyperparameter sensitivity on GraphInstruct (top) and MolecularNet (bottom). Curves show macro-average accuracy for Llama-3.1-8B and Qwen3-8B. Each varies one parameter while fixing the others at $K = 4$, $\tau = 0.6$, and $\lambda_{\text{read}} = 0.01$; shaded bands mark these shared defaults. Tested settings are equally spaced.

Mean performance improves at every step for both backbones in all four task categories. Hierarchical serialization, SR-RoPE, and the mask account for most gains; structural bias and decorrelation add smaller improvements. These effects depend on the addition order and interdependent components. The ablations do not isolate independent causal effects or separate extra input tokens from changes in information flow.

3.4 Hyperparameter analysis

Figure 2 summarizes the results of the univariate hyperparameter analysis conducted on two 8B backbones. Increasing the number of global readouts from $K = 1$ to 4 improves macro-average accuracy by 5.6-8.7 percentage points; additional readouts offer no further gains despite lower readout similarity. All four backbone-benchmark combinations achieve their highest accuracy at $\tau = 0.6$, favoring moderate decorrelation within the tested range. Setting $\lambda_{\text{read}} = 0.01$ improves accuracy by 0.9-1.5 points over no decorrelation loss and performs best in three combinations; Llama-3.1-8B on MolecularNet is 0.2 points higher at 0.1. We therefore retain $K = 4$, $\tau = 0.6$, and $\lambda_{\text{read}} = 0.01$ as shared defaults. Appendix G reports the full results.

3.5 Cost and mechanism illustrations

Dense attention remains $O(L^2 d)$ including structural tokens, with no verified sparse speedup (Appendix H). Appendix I contains all four case figures and analyses of readout complementarity, permutation robustness, aggregation balance, and stable but incorrect predictions.

4 Research Contributions and Limitations

Research contributions. We connect graph structure to decoder computation through hierarchical serialization, SR-RoPE, structural attention, and thresholded readout decorrelation, preserving autoregressive generation in a pretrained LLM. Comparisons across backbones, transfer tasks, and sequential ablations support this joint adaptation under the evaluated protocols. **Limits of the structural inductive bias.** Shared summary positions and structural masks can reduce specific forms of order dependence, but local position scaling, graph boundaries, and the direct reading paths available to the answer still affect the permutation behavior of the full model. There are no explicit multiple rounds of message passing between node blocks; tasks involving longer paths or complex constraints still rely on global readouts and subsequent generation layers. The design therefore does not guarantee the ability to solve arbitrary graph algorithms, invariance under graph isomorphism, or lossless graph representations.

AI USE STATEMENT

First, we used LLMs to assist in retrieving prior research, based on which we conducted a manual analysis of the underlying motivations. Second, we employed LLMs to help generate the prototype code required for our experiments, subsequently reviewing and refining the code manually. Finally, we utilized LLMs to assist with language translation and polishing.

REPRODUCIBILITY STATEMENT

Our code is available at `https://anonymous.4open.science/r/SR-RoPE-87CC/README.md`

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (XXXXXX). We thank all anonymous commenters for their constructive comments.

REFERENCES

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

Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 3950–3957, 2021.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. GraphWiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 353–364, 2024. doi: 10.1145/3637528.3672010. URL `https://doi.org/10.1145/3637528.3672010`.

Xinnan Dai, Kai Yang, Jay Revolinsky, Kai Guo, Aoran Wang, Bohang Zhang, and Jiliang Tang. From sequence to structure: Uncovering substructure reasoning in transformers. In *Advances in Neural Information Processing Systems*, volume 38, pp. 75988–76017, 2025. doi: 10.52202/085713-2552. URL `https://proceedings.neurips.cc/paper_files/paper/2025/hash/6def705c0557104daf581c19b9e2e5e3-Abstract-Conference.html`.

Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36:52464–52489, 2023.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International conference on learning representations*, volume 2024, pp. 43909–43934, 2024.

Zhangyang Gao, Daize Dong, Cheng Tan, Jun Xia, Bozhen Hu, and Stan Z. Li. A graph is worth k words: euclideanizing graph using pure transformer. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations*, volume 2024, pp. 27896–27923, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. When attention sink emerges in language models: An empirical view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=78Nn4QJTEN.

Zeyuan Guo, Enmao Diao, Cheng Yang, and Chuan Shi. Graph tokenization for bridging graphs and transformers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jCctxI1BGF.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. Gofa: A generative one-for-all model for joint graph language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 40792–40823, 2025.

Haokun Liu, Zezhong Ding, and Xike Xie. Learning graph foundation models on riemannian graph-of-graphs. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=ZXOfmBn3Fu.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.

Yiming Liu, Bin Lu, Xinbing Wang, Chenghu Zhou, and Meng Jin. SLASH the sink: Sharpening structural attention inside LLMs. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=To2n4qTdke.

Yuyan Liu, Sirui Ding, Sheng Zhou, Wenqi Fan, and Qiaoyu Tan. Moleculargpt: Open large language model (llm) for few-shot molecular property prediction. *arXiv preprint arXiv:2406.12950*, 2024b.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pp. 417–428, 2023.

Donald Loveland, Puja Trivedi, Ari Weinstein, Edward W Huang, and Danai Koutra. Formalizing and mitigating structural distortion in llm attention for graph reasoning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3366–3377, 2026.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37:78320–78370, 2024.

Li Sun, Zhenhao Huang, Suyang Zhou, Qiqi Wan, Hao Peng, and Philip Yu. Riemanngfm: Learning a graph foundation model from riemannian geometry. In *Proceedings of the ACM on Web Conference 2025*, pp. 1154–1165, 2025.

Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 1717–1727, 2022.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. *Advances in neural information processing systems*, 36:13308–13325, 2023.

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*, pp. 491–500, 2024.

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

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rklz9iAcKQ.

Duo Wang, Yuan Zuo, Guangyue Lu, and Junjie Wu. UniGTE: Unified graph–text encoding for zero-shot generalization across graph tasks and domains. In *Advances in Neural Information Processing Systems*, volume 38, pp. 92095–92120, 2025a. doi: 10.52202/085713-3081. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/84e3cc09f56108559cd28754d94c5e06-Abstract-Conference.html.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36:30840–30861, 2023.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64806–64821. PMLR, 13–19 Jul 2025b. URL https://proceedings.mlr.press/v267/wang25dj.html.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Beyond message passing: neural graph pattern machine. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025c.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Generative graph pattern machine. *Advances in Neural Information Processing Systems*, 38:30068–30091, 2026.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Teng Xiao, Huaisheng Zhu, Zhengyu Chen, and Suhang Wang. Simple and asymmetric graph contrastive learning without augmentations. *Advances in neural information processing systems*, 36:16129–16152, 2023.

Hao Xu, Xiangru Jian, Xinjian Zhao, Wei Pang, Chao Zhang, Suyuchen Wang, Qixin Zhang, Zhengyuan Dong, Joao Monteiro, Bang Liu, Qiuzhuang Sun, and Tianshu Yu. GraphOmni: A comprehensive and extensible benchmark framework for large language models on graph-theoretic tasks. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dc8Kf2g0KC.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33: 5812–5823, 2020.

Xingtong Yu, Zhenghao Liu, Yuan Fang, Zemin Liu, Sihong Chen, and Xinming Zhang. Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6237–6250, 2024.

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*, pp. 1142–1153, 2025a.

Xingtong Yu, Jie Zhang, Yuan Fang, and Renhe Jiang. Non-homophilic graph pre-training and prompt learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 1844–1854, 2025b.

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*, pp. 4443–4454, 2024.

Haiteng Zhao, Shengchao Liu, Ma Chang, Hannan Xu, Jie Fu, Zhihong Deng, Lingpeng Kong, and Qi Liu. Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. *Advances in neural information processing systems*, 36:5850–5887, 2023.

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

A Related work

Structural adaptation between graphs and language models involves three aspects. Graph serialization defines the input units and their organization, positional encoding represents positional relations between these units, and causal masks specify the information accessible at each position. We review prior work along these three aspects.

A.1 Graph serialization

Graph serialization studies how to organize nodes, edges, and their attributes into token sequences that language models can process. In evaluations of graph representations, NLGraph (Wang et al., 2023) compares graph computation tasks of varying complexity, showing that the benefits of prompting depend on problem complexity and that models may rely on spurious cues. Fatemi et al. (2024) examine graph encoding, query formulation, and graph generation distributions through GraphQA. GraphOmni (Xu et al., 2026) further expands the combinations of graph types, serialization formats, and prompting strategies. Dai et al. (2025) analyze the differences between adjacency lists and edge lists through substructure extraction across layers, demonstrating that sequences encoding the same topology need not have the same learning efficiency. These studies establish textual graph representation as a modeling choice that requires analysis in the context of the task and model, beyond a simple format conversion.

For representation construction, WalkLM (Tan et al., 2023) converts attributed random walks into text and fine tunes a language model to obtain node representations that capture both local structure and attribute semantics. Guo et al. (2026) combine reversible graph serialization with byte pair encoding to construct a discrete vocabulary from frequent substructures, enabling standard Transformer models to process graph sequences directly. GraphWiz (Chen et al., 2024) uses explicit reasoning paths for instruction tuning on textual graph inputs and improves answer generation through preference optimization, with an emphasis on reasoning supervision for graph computation. These approaches improve the use of graph information through local traversal, structural vocabularies, and training supervision, respectively.

Graphs can also be provided to language models through learned representations. GraphGPT (Tang et al., 2024) uses a graph encoder and a projection for graph and text alignment, followed by instruction tuning in two stages. GOFA (Kong et al., 2025) interleaves GNN layers within a frozen LLM to connect graph message passing with language computation. GraphsGPT (Gao et al., 2024) aggregates a graph into multiple Graph Words through a Graph2Seq encoder and trains these representations through graph reconstruction. These interfaces show that graph structure can be organized as continuous representations or multiple summary units in addition to raw text records. Their graph encoding, message passing, or reconstruction architectures differ from direct serialization of graph descriptions.

We use hierarchical serialization that preserves node, relation, and attribute text while explicitly allocating summary positions and multiple global readouts at the levels of edge records, node blocks, and the entire graph. Relative to these representation construction methods and graph encoding interfaces, our focus is input organization within a single pretrained LLM. Explicit boundaries between locally ordered records and summaries treated as sets provide consistent structural units for the subsequent positional encoding and visibility rules.

A.2 Positional encoding

Positional encoding determines how a model incorporates positional or structural distances between input units into attention computation. In graph sequences, the physical order is produced by serialization and need not correspond to graph relations. Liu et al. (2024a) find that the position of relevant information affects question answering and retrieval over long contexts, indicating that context coverage does not guarantee effective information use. This observation provides empirical motivation for studying positional sensitivity, but it cannot be attributed directly to RoPE distortion in graph tasks.

Structural position designs primarily include graph distance biases and shared positional representations. Built on T5, GIMLET (Zhao et al., 2023) integrates shortest path distances in graphs, relative distances in text, and relations across modalities into a unified positional bias, enabling joint graph and text modeling without an additional graph encoder. UniGTE (Wang et al., 2025a) assigns shared position indices to graph tokens, constructs structural biases from shortest paths and edge information, and uses learnable alignment tokens to provide graph representations to a frozen decoder. Both methods incorporate graph information into positional modeling, with UniGTE additionally combining shared coordinates and structural biases.

For decoder LLMs, GaLA (Loveland et al., 2026) analyzes the interaction between linearization and RoPE through graph bandwidth, converts graph distances into attention biases between tokens, and performs inference with a frozen model after lightweight calibration. Our SR-RoPE directly modifies the coordinates supplied to the native rotation function: queries and answers retain their global order, node headers and edge records use local coordinates with learnable scaling, and summaries with the same role share an anchor. SR-RoPE distinguishes meaningful local order from permutations of summaries treated as sets, rather than uniformly mapping graph distances to attention biases. We study robustness to equivalent serializations with fixed node identifiers; this positional design does not imply graph isomorphism invariance under arbitrary node renaming.

A.3 Causal masks

Causal masks define the set of positions directly visible to each token. Standard decoder LLMs use prefix visibility, giving later positions access to more context. For graph units without an inherent sequential order, shared position coordinates alone do not eliminate this difference. Adapting graph structure therefore also requires examining where information is aggregated and whether structural units of the same type follow consistent access rules.

Prior work organizes information pathways through structured masks. GIMLET (Zhao et al., 2023) prevents graph tokens from reading instructions in its encoder, decoupling graph representations from task text. This provides a precedent for visibility design between graphs and text, although its encoder setting differs from the autoregressive causal constraints of decoder models. Gist Tokens (Mu et al., 2023) compress prompts into a small number of summary tokens and block subsequent positions from directly accessing the original prompt, forcing information through the summaries. These designs show that masks can specify the aggregation responsibilities of intermediate representations in addition to restricting access to future information.

Visibility control must also be distinguished from adjustments to attention weights or computational resources. StreamingLLM (Xiao et al., 2024) stabilizes inference with a sliding window by retaining the key and value caches of initial tokens. Gu et al. (2025) study the formation of attention sink and show that large attention weights need not imply proportionately large contributions from value vectors. SLASH (Liu et al., 2026b) redistributes weights during inference according to attention patterns associated with graph structure. These methods concern caching strategies or information use along allowed connections; they do not establish the hierarchical visibility sets proposed here.

Pause Tokens (Goyal et al., 2024) delay prediction by adding pause tokens, with their main benefits depending on corresponding pretraining and fine tuning. Sanford et al. (2024) further analyze the expressive capacity for graph algorithms in terms of depth, width, and the number of additional tokens. Such additional computation alone does not guarantee identical structural coverage or complementary representations across readouts.

Our structural attention preserves causal order within local records and specifies the aggregation scope of edge summaries, node summaries, and global readouts. All readouts access the same set of node summaries, without direct connections between readouts. Unlike the decoupling of graphs and text in GIMLET, summaries at every level can read the query. Unlike the mandatory summary pathway in Gist, graph boundaries and answer regions retain native causal rules, allowing answers to read the preceding graph description directly. We further adjust aggregation weights on allowed connections through learnable structural biases and constrain readout redundancy with a thresholded decorrelation objective. These designs use the hierarchical sequence to combine positional adaptation with control over visibility under causal constraints.

B MOTIVATION

Figure 3 summarizes the three principal differences and the corresponding motivations for our structural design.

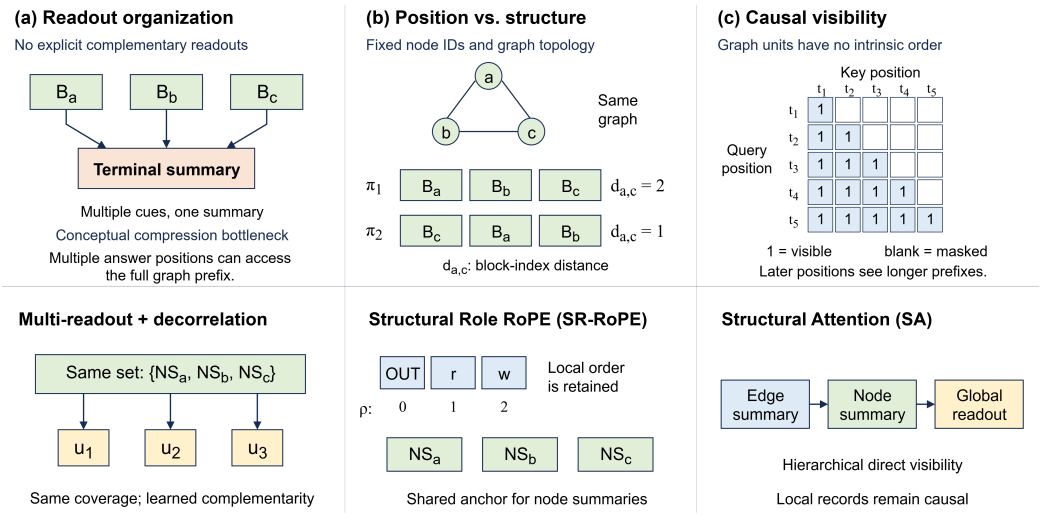

Figure 3: Three differences between graph serialization and decoder LLMs. (a) Conventional graph descriptions do not explicitly organize multiple complementary structural readouts with identical coverage. (b) Permuting node blocks while preserving node identifiers and graph topology changes physical sequence distances and positional relations under native RoPE. (c) A standard causal mask gives later positions access to larger prefixes, although graph units have no inherent linguistic order.

First, serialized graph descriptions lack explicitly organized positions for multiple global readouts. Under standard causal attention, a token near the beginning of a graph description can access only a local prefix, whereas tokens following the description can directly access the entire graph description. Although the answer region may contain several such tokens, access to the full graph does not by itself ensure complementary global representations. These positions follow the sequential semantics of natural language and are not explicitly organized to aggregate distinct aspects of global graph structure. Graph reasoning often requires retaining several types of global information in parallel, including connected components, paths and distances, cycles and local structures, node roles and centrality, and dependencies across regions. If the entire graph relies on a single token at the end of its description for a global representation, these types of information may compete within the same representational bottleneck. Conventional graph sequences do not explicitly allocate multiple readout positions that have identical structural coverage and can learn different attention priorities.

Introducing multiple complementary global readout tokens may reduce the information compression burden on an individual readout.

Second, physical sequence positions do not align with graph relations. The rotary position embedding, RoPE, incorporates linear relative distances between tokens into attention scores. When node blocks in a graph sequence change from $[B_{v_1}, B_{v_2}, B_{v_3}]$ to $[B_{v_3}, B_{v_1}, B_{v_2}]$, the topology remains unchanged, but the physical token positions and relative distances change. During reasoning, an LLM may rely on ordering cues such as "nodes closer to the readout position are more important." Meanwhile, node attribute text and individual edge records retain internal semantic order, so removing positional encoding entirely would also discard necessary information. Positional encoding must therefore distinguish locally ordered text from unordered sets of structural summaries.

Third, causal masks induce unequal visibility of structural information. A standard Decoder-only LLM uses a lower triangular causal mask, creating systematic differences in the structural information visible at different Token positions. Earlier Token positions update from smaller local prefixes, while later Token positions can aggregate information from larger regions. Such asymmetry is appropriate for natural language, where semantics often emerge incrementally from the left context. Graphs, however, have no inherent distinction between "earlier" and "later" nodes: a token appearing later in a graph description need not have a linguistic precedence relation with an earlier token. Under a standard causal mask, two sequentially placed nodes can therefore aggregate overlapping yet unequal contexts even when their ordering has no structural meaning. This may encourage the LLM to learn dependencies induced by serialization order and obscure the actual graph relations.

C PRELIMINARIES

C.1 GRAPH REASONING WITH LLMS

Consider a directed attributed graph $G = (V, E, \mathcal{R}, X_V, X_E)$, where $V = \{v_1, \ldots, v_n\}$ is the node set, $\mathcal{R}$ is the set of relation types, and $E \subseteq V \times \mathcal{R} \times V$ is the edge set. An edge $e = (v_a, r, v_b)$ denotes relation r directed from v_a to v_b. Node and edge attributes are denoted by $X_V = \{\mathbf{x}_v \mid v \in V\}$ and $X_E = \{\mathbf{x}_e \mid e \in E\}$, respectively. An undirected edge can be converted into two directed relations with opposite orientations. Serialization must keep this conversion consistent with the counting convention for incoming and outgoing edge records.

Given a natural language query Q and a graph G, a serialization function S converts the graph into a token sequence. The LLM generates an answer $Y = (y_1, \ldots, y_T)$ by learning the conditional distribution $P_\theta(Y \mid Q, S(G))$. Tasks may include graph classification, node classification, relation prediction between node pairs, and path generation. Our notion of *equivalent serializations* permits only permutations of node blocks and edge records within each block, while preserving node identifiers, field contents, queries, and labels. This differs from graph isomorphism invariance under arbitrary node renaming.

C.2 SELF ATTENTION AND POSITIONAL ENCODING

Let the complete sequence be $X = (x_1, \ldots, x_L)$, with input hidden states $\mathbf{H}^{(\ell)} \in \mathbb{R}^{L \times d}$ at layer ℓ. The queries, keys, and values of attention head h are

$$\begin{aligned} \mathbf{Q}^{(\ell,h)} &= \mathbf{H}^{(\ell)} \mathbf{W}_Q^{(\ell,h)}, \\ \mathbf{K}^{(\ell,h)} &= \mathbf{H}^{(\ell)} \mathbf{W}_K^{(\ell,h)}, \\ \mathbf{V}^{(\ell,h)} &= \mathbf{H}^{(\ell)} \mathbf{W}_V^{(\ell,h)}. \end{aligned} \tag{13}$$

where the projection matrices have dimensions $d \times d_h$ and d_h is the dimension of each head. The standard causal mask is

$$M_{ij}^{\text{causal}} = \begin{cases} 0, & j \le i, \\ -\infty, & j > i. \end{cases} \tag{14}$$

The attention weights and outputs are

$$\begin{aligned}\mathbf{A}^{(\ell,h)} &= \mathrm{Softmax}_{\mathrm{row}}\left(\frac{\mathbf{Q}^{(\ell,h)}(\mathbf{K}^{(\ell,h)})^{\top}}{\sqrt{d_h}} + \mathbf{M}^{\mathrm{causal}}\right), \\ \mathbf{O}^{(\ell,h)} &= \mathbf{A}^{(\ell,h)}\mathbf{V}^{(\ell,h)}.\end{aligned} \tag{15}$$

To simplify the description of positional encoding, we treat the query and key of an individual token as column vectors. RoPE rotates queries and keys at physical position $p_i = i - 1$:

$$\widetilde{\mathbf{q}}_i = \mathbf{R}(p_i)\mathbf{q}_i, \qquad \widetilde{\mathbf{k}}_j = \mathbf{R}(p_j)\mathbf{k}_j. \tag{16}$$

By the relative position property of the rotation matrices, the attention score satisfies

$$\widetilde{\mathbf{q}}_i^{\top}\widetilde{\mathbf{k}}_j = \mathbf{q}_i^{\top}\mathbf{R}(p_j - p_i)\mathbf{k}_j. \tag{17}$$

Thus, identical token content can receive different attention scores under different serialization orders. In general, graph adjacency also does not correspond uniquely to relative distance in the sequence.

C.3 Graph serialization and hierarchical aggregation

A simple edge list can be written as $S_{\mathrm{edge}}(G) = [(v_a, r_1, v_b), (v_a, r_2, v_c), \ldots]$. Adjacency serialization groups records associated with the same central node, bringing textual locality into closer alignment with local graph relations. The grouping by source node used in SLASH (Liu et al., 2026b) provides a reference for this direction. We retain this organization around nodes while explicitly introducing edge summaries, node summaries, and multiple global readouts, together with adaptations to their positional rules and attention visibility scopes.

D Complete method specification

D.1 Hierarchical graph serialization

We use the following complete sequence:

$$X = [Q, S(G), \langle A\rangle, Y], \qquad S(G) = [\langle G\rangle, B, \langle /G\rangle, U]. \tag{18}$$

Here, $B = [B_{v_1}, \ldots, B_{v_n}]$ is the sequence of node blocks, $U = [u_1, \ldots, u_K]$ is the sequence of global readouts, $\langle G\rangle$ and $\langle /G\rangle$ delimit the graph region, and $\langle A\rangle$ marks the beginning of the answer. Square brackets denote concatenation in the specified order. When defining attention sets, the same symbols denote sets of the corresponding token occurrence positions, distinguishing different occurrences with identical content.

D.1.1 Node blocks and edge records

For a central node v, the node block is defined as

$$B_v = [H_v, C_{v,e_1}, \ldots, C_{v,e_{d_v}}, \mathrm{NS}_v], \tag{19}$$

where C_{v,e_1} is an edge record for node v corresponding to edge e_1, and d_v is the number of encoded incident edge records. An isolated node has $d_v = 0$ but retains its header and node summary. The node header is

$$H_v = [\mathrm{NB}, \mathrm{tok}(v), \mathrm{TYPE}(v), \mathrm{tok}(\mathbf{x}_v)]. \tag{20}$$

Here, NB marks the beginning of a node block, and tok denotes a tokenized field sequence. Node type and attribute fields may be omitted when they are absent from the data.

For an outgoing edge $e = (v, r, w)$ and an incoming edge $e' = (w, r, v)$, we construct

$$\begin{aligned}C_{v,e}^{\mathrm{out}} &= [\mathrm{ER}, \mathrm{OUT}, \mathrm{tok}(r), \mathrm{tok}(w), \mathrm{tok}(\mathbf{x}_e), \mathrm{ES}_{v,e}], \\ C_{v,e'}^{\mathrm{in}} &= [\mathrm{ER}, \mathrm{IN}, \mathrm{tok}(r), \mathrm{tok}(w), \mathrm{tok}(\mathbf{x}_{e'}), \mathrm{ES}_{v,e'}].\end{aligned} \tag{21}$$

respectively. Here, ER marks the beginning of an edge record, OUT and IN indicate direction relative to the central node, and $\mathrm{ES}_{v,e}$ and NS_v are the edge summary and node summary tokens,

respectively. Direction is always specified relative to the current node v, so v is the target node in an incoming edge record. For undirected graphs, relations in opposite directions encode the same undirected adjacency and should not be interpreted as additional, independently observed edges.

The following example illustrates a node block containing both outgoing and incoming edges.

```
<NODE_BLOCK>
NB node_3 person age=35
ER OUT follows node_8 ES
ER IN  follows node_1 ES
ER OUT owns    node_2 ES
NS
</NODE_BLOCK>
```

Within each edge record, the relation, direction, and node identifier follow a fixed semantic order. Different incident edge records, however, generally have no natural order. Placing an edge summary at the end of its record allows it to aggregate information from that record. Placing a node summary at the end of its node block allows it to read all edge summaries. The header precedes the edge records so that each record can access information about the central node. This organization separates local text encoding from structural aggregation over sets and establishes explicit boundaries for the masking rules that follow.

D.1.2 MULTIPLE GLOBAL READOUTS

We place K global readouts after all node blocks and initialize their representations as

$$\mathbf{h}_{u_k}^{(0)} = \mathbf{e}_{\mathrm{read}} + \mathbf{e}_k^{\mathrm{role}}, \qquad k = 1, \ldots, K. \tag{22}$$

where $\mathbf{e}_{\mathrm{read}}$ is a shared base embedding and $\mathbf{e}_k^{\mathrm{role}}$ is an independently learnable role embedding. The shared embedding represents the common function of each readout token. The role embeddings break the initial symmetry among readouts. We do not assign predefined semantics, such as paths, connectivity, or motifs, to individual readouts, nor do we regard the shared embedding itself as a guarantee against collapse. In Sections D.3 and D.4, we encourage different readouts to learn complementary information by giving them the same visibility scope over node summaries and imposing a decorrelation constraint on their final hidden representations.

D.2 SR-ROPE

Standard RoPE directly uses global physical sequence positions. This mechanism is naturally suited to language sequences because word order and relative distance usually carry semantic information. In a serialized graph, however, different regions have different ordering properties. Specifically, natural language queries and answers follow a strict semantic order, as do the fields within an individual edge record. In contrast, the incident edges of a node, summaries of different nodes, and multiple global graph readout tokens are more naturally treated as unordered sets. If every token uses its physical position p_i, permutations without semantic significance in the graph are mapped to different RoPE relative distances. For example, the three edge summary tokens of node v may occur at positions $40,\ 67,\ 104$, depending on the order of the edge records in its node block. Similarly, moving a node block from the beginning to the end of the sequence shifts the RoPE coordinates of every token within the block, causing the same local graph structure to receive different positional encodings under different serializations. We therefore extend the coordinate generation rule of native RoPE to distinguish locally ordered text from structural summaries treated as sets while preserving the original RoPE mechanism as much as possible. We call this extension Structural Role RoPE (SR-RoPE). Its central idea is to retain the rotation function of RoPE while jointly determining its input coordinate from the global physical sequence position, local structural position, structural role, and positional anchor of each token:

$$\psi_i = \alpha_{c_i} p_i + \beta_{c_i} \rho_i + s_{c_i}. \tag{23}$$

Here, c_i is the structural role, p_i is the global physical position, ρ_i is the local position, α_{c_i} and β_{c_i} weight the two types of positions, and s_{c_i} is the role anchor.

D.2.1 MUTUALLY EXCLUSIVE STRUCTURAL ROLES AND LOCAL COORDINATES

The role set consists of six mutually exclusive categories:

$$\mathcal{C} = \{\text{Global}, \text{NH}, \text{ER}, \text{ES}, \text{NS}, \text{U}\}. \tag{24}$$

Here, NH and ER are collectively termed locally ordered roles (Local), whereas ES, NS, and U are collectively termed unordered roles (Unordered). Roles are assigned as follows:

$$c_i = \begin{cases} \text{Global}, & x_i \in Q \cup \{\langle G\rangle, \langle /G\rangle, \langle A\rangle\} \cup Y, \\ \text{NH}, & x_i \in H_v, \\ \text{ER}, & x_i \in C_{v,e} \setminus \{\text{ES}_{v,e}\}, \\ \text{ES}, & x_i = \text{ES}_{v,e}, \\ \text{NS}, & x_i = \text{NS}_v, \\ \text{U}, & x_i \in U. \end{cases} \tag{25}$$

For locally ordered roles, ρ_i starts at 0 at the beginning of the corresponding header or edge record and increases by one for each successive token within that unit. The count restarts independently for each unit. For all other roles, $\rho_i = 0$. This rule preserves order within fields without encoding the permutation of records in the local coordinates.

The fixed reference configuration for the roles is

$$(\alpha_c, \beta_c) = \begin{cases} (1, 0), & c = \text{Global}, \\ (0, 1), & c \in \{\text{NH}, \text{ER}\}, \\ (0, 0), & c \in \{\text{ES}, \text{NS}, \text{U}\}. \end{cases} \tag{26}$$

This configuration has an intuitive interpretation. For a token with the Global role, linear order has explicit semantic significance, so we fully preserve the original positional behavior of the LLM. For a token with a locally ordered role, semantic order is confined to its own unit, so we preserve the original positional behavior of the LLM only within that unit. For a token with an unordered role, the order of elements generally carries no structural semantics, so we seek to remove the influence of position on attention scores among such elements. However, removing global positions entirely or fully retaining local positions may be overly rigid. For example, in graphs with short textual node attributes or simple relation descriptions, a small amount of global order information may still help the model maintain contextual continuity. We therefore generalize α_c and β_c to learnable scaling factors for different roles:

$$\alpha_c = \sigma(a_c), \qquad \beta_c = \sigma(b_c), \qquad c \in \{\text{NH}, \text{ER}\}, \tag{27}$$

where a_c, b_c are learnable scalars. With finite parameters, Sigmoid takes values in $(0, 1)$. Thus, the values 0 and 1 in Eq. equation 26 are reference endpoints approximated at initialization, rather than values attained exactly by finite Sigmoid parameters. We introduce the two learnable scaling parameters a_{c_i} and b_{c_i} only for locally ordered roles; Global and unordered roles retain their fixed configurations.

D.2.2 ROLE ANCHORS

Role anchors provide a base reference position for every token. As with local coordinates, we specify an anchor for each role. Every token in Global has an anchor of 0. In Local, δ_{NH} parameterizes the anchor for each node header token, and δ_{ER} parameterizes the anchor for each token within an edge record except its edge summary token. In Unordered, δ_{ES} and δ_{NS} parameterize the anchors for each edge summary token and node summary token, respectively. The detailed configuration is

$$\begin{aligned} s_{\text{Global}} &= 0, \\ s_{\text{NH}} &= p_{\langle G\rangle} + 1 + \text{softplus}(\delta_{\text{NH}}), \\ s_{\text{ER}} &= s_{\text{NH}} + \text{softplus}(\delta_{\text{ER}}), \\ s_{\text{ES}} &= s_{\text{ER}} + \text{softplus}(\delta_{\text{ES}}), \\ s_{\text{NS}} &= s_{\text{ES}} + \text{softplus}(\delta_{\text{NS}}), \\ s_{\text{U}} &= p_{\langle /G\rangle} + 1. \end{aligned} \tag{28}$$

where $p_{\langle G\rangle}+1$ and $p_{\langle /G\rangle}+1$ denote the physical positions immediately following the token marking the beginning and end of the graph region, respectively. The learnable scalars δ_{NH}, δ_{ER}, δ_{ES}, and δ_{NS} determine positive increments through their softplus transforms; these increments accumulate to give structural offsets from the beginning of the graph region. Summaries of the same type share an anchor. For every token with the Global role, SR-RoPE reduces exactly to the native RoPE of the LLM. For the other roles, the model can learn the relative arrangement of structural levels in the geometric space of RoPE during fine tuning. SR-RoPE ultimately applies $\widetilde{\mathbf{q}}_i = \mathbf{R}(\psi_i)\mathbf{q}_i$ and $\widetilde{\mathbf{k}}_i = \mathbf{R}(\psi_i)\mathbf{k}_i$ without changing the query, key, or value projections.

D.3 Structural Attention

SR-RoPE mitigates the influence of artificial linear positions introduced by serialization on graph representations through positional encoding. However, positional encoding only specifies how positional relationships between token pairs are represented; it does not change the visibility scope determined by the causal mask in a Decoder-only LLM. Consider two node summary tokens with no linguistic precedence relation. Even if they have exactly the same SR-RoPE coordinate, the token at the later physical position can still access the earlier one under the standard causal mask. As discussed in Section 1, differences in the information coverage available to each token under the standard causal mask cause a pronounced imbalance in information aggregation for graph representations. We therefore introduce a structural attention mechanism (Structural Attention, SA). Its basic principle is to fuse graph information across structural levels only at predefined aggregation tokens while keeping distinct structural units at the same level isolated. Specifically, a structural visibility mask explicitly defines visibility between token pairs, and a structural attention bias imposes soft structural constraints on token pairs that are allowed to interact.

D.3.1 Structural visibility mask

Let $\mathrm{Prefix}(Z, x_i)$ denote the set of token occurrence positions in sequence Z that are no later than x_i, including x_i itself. Let $\mathcal{Q} = Q \cup \{\langle G\rangle\}$ denote the query context, and let $E(v)$ be the index set of incident edge records encoded in the block for node v. The set of positions that a structural token is allowed to access is defined as

$$\Gamma(x_i) = \begin{cases} \mathcal{Q} \cup \mathrm{Prefix}(H_v, x_i), & x_i \in H_v, \\ \mathcal{Q} \cup H_v \cup \mathrm{Prefix}(C_{v,e}, x_i), & x_i \in C_{v,e}, \\ \mathcal{Q} \cup H_v \cup \{\mathrm{ES}_{v,e} : e \in E(v)\}, & x_i = \mathrm{NS}_v, \\ \mathcal{Q} \cup \{\mathrm{NS}_v : v \in V\} \cup \{\langle /G\rangle\}, & x_i \in U. \end{cases} \tag{29}$$

A node header preserves internal causality. Each edge record reads only the query, the current node header, and its own prefix. A node summary aggregates the current header and all edge summaries, and all global readouts access the same set of node summaries. No additional direct attention connections are established between distinct edge records, between distinct node blocks, or between different global readouts. The states of node summaries and global readouts are retained through the Transformer residual paths.

For the query, graph boundaries, and answer region, we use the native causal visibility rule:

$$\Gamma(x_i) = \mathrm{Prefix}(X, x_i), \quad x_i \in Q \cup \{\langle G\rangle, \langle /G\rangle, \langle A\rangle\} \cup Y. \tag{30}$$

The complete mask can be written as

$$M_{ij}^{\mathrm{SA}} = \begin{cases} 0, & x_j \in \Gamma(x_i), \\ -\infty, & \text{otherwise.} \end{cases} \tag{31}$$

All the visibility connections above point to the current or an earlier position, so they do not allow an answer token to access future answer content. Each row also contains at least one allowed position, such as the query, a boundary marker, or the position itself, avoiding an undefined Softmax when every entry is $-\infty$. Equation equation 29 specifies direct connections between structural units. Under Eq. equation 30, the graph end marker can aggregate the preceding graph description and serve as an input to the global readouts. Similarly, the answer region can directly read the graph description. Our hierarchical readout therefore provides an explicit aggregation path, but it is not the only path by which graph information reaches the answer.

D.3.2 STRUCTURAL ATTENTION BIAS

The structural visibility mask specifies visibility between structural token pairs. For pairs that are allowed to interact, however, native Self-Attention does not explicitly encode which visible token deserves greater attention. For example, when forming an edge summary token representation, the corresponding edge record may warrant greater attention than the query itself. When forming a node summary token representation, the associated edge summary token representations may be as relevant as the query. We therefore use the explicit structural relationships in graph serialization to define three core types of relationships between interacting token pairs. We impose soft constraints on pairs belonging to these types to guide the learning of weight allocation in native Self-Attention. The three relationship types are

$$\begin{aligned}
\mathcal{P}_E &= \{(\mathrm{ES}_{v,e}, x_j) : x_j \in C_{v,e}\}, \\
\mathcal{P}_N &= \{(\mathrm{NS}_v, x_j) : x_j \in H_v \cup \{\mathrm{ES}_{v,e} : e \in E(v)\}\}, \\
\mathcal{P}_G &= \{(u_k, \mathrm{NS}_v) : 1 \le k \le K,\ v \in V\}.
\end{aligned} \tag{32}$$

At layer ℓ, we construct the bias using three learnable scalars:

$$D_{ij}^{(\ell)} = \sum_{r \in \{E,N,G\}} b_r^{(\ell)} \mathbb{I}[(x_i, x_j) \in \mathcal{P}_r]. \tag{33}$$

Here, $\mathbb{I}[\cdot]$ is the indicator function, and $b_r^{(\ell)}$ is initialized to 0 with no constraint on its sign. This allows the model to strengthen or suppress the corresponding aggregation relationships without assuming in advance that an edge record must be more important than the query. The bias applies only to connections permitted by the mask and does not reopen masked connections. At the start of fine tuning, SA does not introduce a change to the attention logit for any token pair that is already allowed to interact. As training proceeds, the model can learn the additional attention strength needed at the Edge, Node, and Graph structural levels. For example, a larger $b_E^{(\ell)}$ indicates a stronger preference at the current layer for aggregating information from an edge record into $ES_{v,e}$; a larger $b_N^{(\ell)}$ places greater emphasis on aggregating the node header and different edge summaries into NS_v; and $b_G^{(\ell)}$ controls the structural preference of each global readout token for acquiring information from node summaries.

Combining this mechanism with SR-RoPE gives the final attention:

$$\mathbf{A}^{(\ell,h)} = \mathrm{Softmax}_{\mathrm{row}}\left(\frac{\widetilde{\mathbf{Q}}^{(\ell,h)}(\widetilde{\mathbf{K}}^{(\ell,h)})^\top}{\sqrt{d_h}} + \mathbf{M}^{\mathrm{SA}} + \mathbf{D}^{(\ell)}\right). \tag{34}$$

The output is still computed as $\mathbf{O}^{(\ell,h)} = \mathbf{A}^{(\ell,h)}\mathbf{V}^{(\ell,h)}$. SA modifies visibility relationships and attention scores without introducing a separate graph encoding network.

D.4 THRESHOLD-BASED READOUT-DEPENDENT LOSS

During training, we retain the native autoregressive generation objective of the Decoder-only LLM as the primary objective:

$$\begin{aligned}
P_\theta(Y \mid Q, S(G)) &= \prod_{t=1}^{T} P_\theta(y_t \mid Q, S(G), y_{<t}), \\
\mathcal{L}_{\mathrm{gen}} &= -\frac{1}{T}\sum_{t=1}^{T} \log P_\theta(y_t \mid Q, S(G), y_{<t}).
\end{aligned} \tag{35}$$

The generation loss is computed only over the answer region, with the query and graph description serving as conditioning context. Training uses the standard shifted targets: the hidden state at the current position predicts the next answer token.

The generation loss $\mathcal{L}_{\mathrm{gen}}$ trains the model to perform graph reasoning and generate the final answer from the query and serialized graph. However, this loss alone does not ensure that different global readout token representations capture complementary graph information. Although each readout

token has a distinct role embedding and structural attention prohibits direct information exchange between different readout token positions, all readouts aggregate information about the entire graph from the same set of node summaries $\{NS_v \mid v \in V\}$. Their hidden representations may therefore become highly similar during training, leading to readout collapse. One simple approach is to enforce strict orthogonality among graph readouts. We instead allow readouts to share necessary global graph information while retaining distinct structural emphases. To this end, we introduce the Thresholded Readout Decorrelation Loss, which penalizes a pair of readouts only when their representation similarity exceeds a specified threshold. Denote the global readout representation at layer N_L by $\mathbf{g}_k = \mathbf{h}_{u_k}^{(N_L)}$. Let

$$\widehat{\mathbf{g}}_k = \frac{\mathbf{g}_k}{\max(\|\mathbf{g}_k\|_2, \varepsilon)}, \qquad \mathrm{sim}_{ij} = \widehat{\mathbf{g}}_i^\top \widehat{\mathbf{g}}_j, \tag{36}$$

where $\varepsilon > 0$ provides numerical stability and sim_{ij} is the numerically stabilized cosine similarity. We further introduce a similarity threshold $\tau \in [0, 1)$ and define the thresholded readout decorrelation loss as

$$\mathcal{L}_{\mathrm{read}} = \begin{cases} \dfrac{2}{K(K-1)} \displaystyle\sum_{1 \le i < j \le K} \big[\max(0, \mathrm{sim}_{ij} - \tau)\big]^2, & K \ge 2, \\ 0, & K = 1. \end{cases} \tag{37}$$

When two readout representations satisfy $\mathrm{sim}_{ij} \le \tau$, their pair incurs no additional regularization loss, i.e., $\mathcal{L}_{\mathrm{read}}^{(i,j)} = 0$. In this case, we consider the two global readouts sufficiently distinct in representation, even though they may share some graph information. When $\mathrm{sim}_{ij} > \tau$, the regularization penalty increases quadratically with the amount by which their similarity exceeds the threshold, discouraging different global readout token representations from converging to highly similar states. Unlike strict orthogonality constraints, the thresholded decorrelation loss only limits excessive representation similarity, primarily aiming to avoid $\mathrm{sim}_{ij} \to 1$. Thus, each global readout token can still share information useful for the current graph while being encouraged to learn complementary structural features. Our overall training objective combines the autoregressive generation loss with the thresholded readout decorrelation loss:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{gen}} + \lambda_{\mathrm{read}} \mathcal{L}_{\mathrm{read}}, \qquad \lambda_{\mathrm{read}} \ge 0. \tag{38}$$

where $\lambda_{\mathrm{read}} \ge 0$ is a hyperparameter controlling the weight of the readout decorrelation constraint relative to the primary generation task. The principal optimization objective remains $\mathcal{L}_{\mathrm{gen}}$, which directly trains the model to generate the correct answer given query Q and graph G. The auxiliary regularizer $\mathcal{L}_{\mathrm{read}}$ does not prescribe the structural semantics that each readout token should learn; it only discourages excessive redundancy among readouts. This allows different global readout token representations to develop distinct structural emphases while jointly serving the final generation task.

E Experimental setup and evaluation protocol

We evaluate our method in five aspects: performance across backbones, transfer to standard graph tasks, component contributions, hyperparameter sensitivity, and computational overhead. We also use case studies to explain the design rationale and the conditions under which the method applies. The experiments address the following questions:

- **RQ1 (Effectiveness across backbones):** How does our method perform with decoder LLMs across different model families and parameter scales?
- **RQ2 (Competitiveness across datasets):** How does our method compare with representative graph learning methods on node classification and link prediction when the target dataset is held out from training?
- **RQ3 (Component contributions):** How much does each component contribute: hierarchical graph serialization, SR-RoPE, the structural visibility mask, structural attention biases, and the readout decorrelation loss?
- **RQ4 (Hyperparameter sensitivity):** How do the number of global readouts, the decorrelation threshold, and the loss weight affect task performance and representation similarity?

- **RQ5 (Overhead and limits of the mechanisms):** What parameter and computational overheads do the structural modules introduce? What behaviors correspond to complementarity among multiple readouts, permutation robustness, and aggregation balance, and when are they still insufficient to ensure correct predictions?

E.1 Datasets and evaluation metrics

Graph reasoning across backbones. The first evaluation uses GraphInstruct (Chen et al., 2024) and MolecularNet (Wu et al., 2018). GraphInstruct covers nine graph computation tasks: Cycle, Connect, Bipartite, Topology, Shortest, Triangle, Flow, Hamilton, and Subgraph. For MolecularNet, we use five datasets for molecular property prediction, namely BACE, BBBP, ClinTox, HIV, and Tox21, and convert molecular topology and node and edge attributes into graph descriptions. Both groups of tasks follow our experimental protocol and evaluate predictions using Accuracy. Because molecular tasks may exhibit class imbalance, accuracy in this evaluation serves to compare model performance under the current protocol; it does not replace a dedicated assessment of the ability to identify minority classes.

Standard graph tasks. The second evaluation uses 10 standard graph datasets (Zhao et al., 2024; Wang et al., 2025b; Yu et al., 2025b) with node features and class labels retained (Chameleon and Squirrel are included as pre-training data sources but are excluded from downstream evaluation when reporting the main transfer learning results). For node classification, we report accuracy (%) on Wisconsin, Cornell, Citeseer, Cora, Pubmed, Amazon-Computers, Amazon-Photos, and Texas. For link prediction, we report AUC-ROC (%) on Wisconsin, Cornell, Citeseer, Pubmed, Cora, Amazon-Photos, and Texas. Computers and Photos in the tables abbreviate Amazon-Computers and Amazon-Photos, respectively. We follow the evaluation setting across datasets of Liu et al. (2026a): the target dataset is held out, and the model is trained on the remaining source datasets before evaluation on the target dataset. The appendix J provides the sample counts, splits, and graph sizes for the reported datasets.

E.2 Backbones and comparison methods

Large language models. Following the setup of Liu et al. (2026b), we use Llama-3.2-3B, Llama-3.1-8B, and Qwen3-4B/8B/14B as general purpose backbones, spanning two model families (Grattafiori et al., 2024; Yang et al., 2025) and parameter scales of 3B-14B. *Vanilla* denotes the original configuration of each backbone, *SLASH* denotes its configuration with structural attention sharpening, and *Ours* denotes our method, which fine tunes the model after introducing hierarchical graph serialization, SR-RoPE, structural attention, and the readout decorrelation loss. GraphWiz and MolecularGPT (Liu et al., 2024b) provide additional comparisons with backbones adapted to specific tasks, allowing us to examine whether structural modifications can complement existing domain adaptation.

Graph learning methods. Following the setup of Liu et al. (2026a), the node classification comparisons include GNNs trained with task supervision, namely GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2017), H2GCN (Zhu et al., 2020), and FAGCN (Bo et al., 2021); methods grouped under pretraining and fine tuning in the original experiments, namely DGI (Veličković et al., 2019), GraphCL (You et al., 2020), GraphACL (Xiao et al., 2023), and MixHop (Abu-El-Haija et al., 2019); prompt learning methods, namely GPPT (Sun et al., 2022), GPF (Fang et al., 2023), GraphPrompt (Liu et al., 2023), and GraphPrompt+ (Yu et al., 2024); and graph foundation models, namely GCOPE (Zhao et al., 2024), SAMGPT (Yu et al., 2025a), MDGFM (Wang et al., 2025b), RiemannGFM (Sun et al., 2025), GPM (Wang et al., 2025c), G2PM (Wang et al., 2026), and R-GFM (Liu et al., 2026a). The link prediction comparisons include GCOPE (Zhao et al., 2024), SAMGPT (Yu et al., 2025a), MDGFM (Wang et al., 2025b), RiemannGFM (Sun et al., 2025), and R-GFM (Liu et al., 2026a).

E.3 Evaluation protocol and statistical reporting

In Tables 4–9, results marked with † are taken directly from published studies; we do not supply variances or metrics that those studies do not report. We run our method independently five times for each task and data split, using random seeds $\{42, 43, 44, 45, 46\}$. The main tables report the

mean and standard deviation for each task, and Avg. denotes the unweighted arithmetic mean of the task means. Gains across tasks reported in the text are calculated from the individual task means in the tables. Percentage points always denote absolute differences in accuracy or AUC-ROC, and the proportion of wins counts only tasks with strictly higher means.

Parameter update schemes, training budgets, and input processing details are not fully controlled through a common implementation for the published baselines and our fine tuned models. The main tables therefore compare results with existing methods; the full differences cannot be attributed to the structural modules, and the comparisons do not constitute strict controls with identical training budgets. Our discussion of component contributions relies primarily on the ablation results in Section 3.3.

During training, we randomly permute the order of node blocks and the order of edge records within each node block to reduce dependence on fixed arrangements. Input length is constrained by a common token budget for each task; the rules for local graph construction and structural truncation are described below. These rules apply only to our runs, and the baseline comparisons do not use exactly the same filtering or truncation procedures. The permutation perturbation figures in Section I illustrate the mechanisms and are distinct from the task evaluation results in the main tables.

E.4 Implementation details

We directly modify the internal position coordinates and information flow in self attention within decoder LLMs, without introducing an additional GNN encoder. SR-RoPE retains the original rotation operator. The structural attention biases $b_E^{(\ell)}$, $b_N^{(\ell)}$, and $b_G^{(\ell)}$ are all initialized to 0, and the global readouts use a shared base embedding and separate role embeddings.

Computing environment. Qwen3-14B and GraphWiz-LLaMA2-13B use two NVIDIA GeForce RTX 5090 GPUs, each with 32 GB of memory; all other listed backbones use one GPU of the same type. Experiments with two GPUs use the `FULL_SHARD` mode of FSDP to shard model parameters, gradients, and optimizer states, with `use_orig_params=True` when frozen and trainable parameters are mixed.

Parameter efficient fine tuning. We freeze the original pretrained weights of the backbone and update the LoRA parameters and the newly introduced structural module parameters, including structural position parameters, structural attention biases, and structural token embeddings. LoRA is applied to `q_proj` and `v_proj` in each Transformer layer, with rank $r = 8$, scaling factor $\alpha = 16$, and dropout of 0.05. Qwen3-4B and Llama-3.2-3B use 72 and 56 LoRA modules, respectively; Qwen3-8B, GraphWiz-LLaMA2-7B, GraphWiz-Mistral-7B, and Llama-3.1-8B each use 64; Qwen3-14B and GraphWiz-LLaMA2-13B each use 80. Module counts are reported per model and do not double with the number of GPUs. Unless otherwise specified in the dedicated hyperparameter experiments, we set $K = 4$, $\tau = 0.6$, and $\lambda_{\text{read}} = 0.01$.

Input length and local graphs. The `max_length` values for GraphInstruct and MolecularNet are 6144 and 4096, respectively, and both node classification and link prediction use 4096. For standard graph tasks, we extract local graphs within 2 hops, cap the number of candidate nodes at 80, and then apply structural truncation according to the token budget. We retain the target node for node classification and both endpoints of the edge to be predicted for link prediction. The context for link prediction uses only positive training edges and excludes the current edge to be predicted, preventing direct exposure of the prediction target to the model.

Optimization and training budget. All four task categories use AdamW with a peak learning rate of 5×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay of 0, and a gradient clipping threshold of 1.0. We apply linear warmup over the first 3% of optimizer updates and keep the learning rate constant thereafter. Computation uses BF16 without weight quantization; gradient checkpointing is enabled, and KV cache is disabled during training. The global effective batch size is 8: with one GPU, we process 1 example per step and accumulate gradients over 8 steps; with two GPUs, each GPU processes 1 example per step and accumulates gradients over 4 steps. Each task category uses an initial training budget of 1 epoch. We pool and randomly shuffle the source training examples permitted for each run, without additional oversampling or dropping the final incomplete batch. In evaluations across datasets, the held out target dataset is excluded from the source training set.

F FULL EXPERIMENTAL RESULTS

F.1 GRAPH REASONING ACROSS BACKBONES

For RQ1, Table 4 shows that our method achieves higher average accuracy than Vanilla and SLASH on all five general purpose backbones. Across the 45 paired comparisons formed by five backbones and nine tasks, our method yields average absolute gains of 31.51 and 20.88 percentage points over Vanilla and SLASH, respectively. It outperforms SLASH in 43 comparisons, corresponding to a win rate of 95.6%. This trend holds for both the Llama and Qwen model families, indicating consistent gains across backbones under the current training configuration.

The gains also vary substantially by task. For Llama-3.1-8B, our method improves on SLASH by 48.2 and 27.2 percentage points on Hamilton and Bipartite, respectively, but by only 5.8 percentage points on Shortest. Our Topology results for the five general purpose backbones remain in the range 0.024–0.089, with Llama-3.2-3B and Qwen3-14B falling below their corresponding SLASH results by 0.3 and 5.0 percentage points, respectively. Absolute accuracy on Shortest and Flow is also generally low. Higher average performance therefore does not imply reliable mastery of all algorithmic graph tasks; substantial room for improvement remains on tasks involving topological sorting, exact path computation, or flow computation.

Backbones adapted to specific tasks exhibit a similar trend. With our method, GraphWiz-Mistral-7B, GraphWiz-LLaMA2-7B, and GraphWiz-LLaMA2-13B achieve average accuracy over the nine tasks of 0.680, 0.706, and 0.743, respectively. Across all eight backbones, our method improves on SLASH by an average of 16.85 percentage points and wins 69/72 task comparisons. Connect performance on GraphWiz-Mistral-7B remains slightly below its Vanilla and SLASH configurations, indicating that structural modifications do not provide additional gains over existing task adaptation for every task.

Table 6 presents the results for molecular property prediction. Across the 25 comparisons involving five general purpose backbones, our method achieves average gains of 33.67 and 14.11 percentage points over Vanilla and SLASH, respectively, and exceeds both in all 25 comparisons. Qwen3-14B achieves the highest average accuracy over the five tasks, at 0.823. MolecularGPT with our configuration achieves an average accuracy of 0.682, with a macro average gain of 31.60 percentage points over SLASH across tasks. This suggests that existing domain adaptation still leaves room for further adjustments to graph inputs and the flow of structural information.

The gains on molecular tasks are uneven. For example, Llama-3.1-8B improves on SLASH by 45.7 percentage points on BACE but by only 2.2 percentage points on HIV, where baseline accuracy is already high. The MolecularGPT configuration achieves 0.512 on ClinTox, still below the results of several general purpose backbones, indicating that a backbone's domain training history does not directly determine its ranking on a task under the current serialization protocol. Overall, these comparisons support the practical effectiveness of our approach across backbones. Because the training budgets of Vanilla, SLASH, and our method are not strictly identical, the contribution of the structural design itself must be assessed together with the ablation results.

F.2 COMPETITIVENESS ACROSS DATASETS ON STANDARD GRAPH TASKS

For RQ2, Table 7 summarizes the node classification results. The Qwen3-8B configuration achieves an average accuracy of 47.76%, approximately 0.92 percentage points above R-GFM, the strongest method among the published results, and obtains a higher mean on all eight datasets. Comparing all methods in the table by column, this configuration ranks first on five datasets: Citeseer, Cora, Pubmed, Computers, and Texas. The Llama-3.1-8B configuration achieves the highest means on Wisconsin, Cornell, and Photos.

The Llama-3.1-8B configuration achieves an average accuracy of 46.75%, slightly below the 46.84% of R-GFM. Although it outperforms R-GFM on five datasets, decreases on Citeseer, Cora, and Pubmed offset some of these gains. Transfer performance across datasets thus depends on backbone choice; advantages observed for one backbone on selected datasets do not imply that every configuration outperforms existing graph foundation models.

Table 4: Accuracy on GraphInstruct (range: 0-1). Avg. for our method is the arithmetic mean of the means for the nine tasks. † denotes values taken directly from published results in the cited studies. The best mean for each backbone is shown in bold. We report the mean and standard deviation over five runs for each task. Additional columns are reported in Table 5.

Backbone	Method	Cycle	Connect	Bipartite	Topology	Shortest
Llama-3.2-3B	Vanilla†	0.180	0.340	0.005	0.045	0.030
	SLASH†	0.573	0.340	0.235	**0.048**	0.030
	Ours	**0.848±0.003**	**0.700±0.003**	**0.683±0.004**	0.045±0.004	**0.045±0.005**
Llama-3.1-8B	Vanilla†	0.660	0.730	0.235	0.060	0.018
	SLASH†	0.680	0.797	0.547	0.022	0.015
	Ours	**0.969±0.003**	**0.836±0.004**	**0.819±0.004**	**0.089±0.007**	**0.073±0.002**
Qwen3-4B	Vanilla†	0.645	0.420	0.048	0.003	0.018
	SLASH†	0.690	0.388	0.195	0.003	0.025
	Ours	**0.864±0.004**	**0.693±0.004**	**0.797±0.003**	**0.024±0.003**	**0.044±0.002**
Qwen3-8B	Vanilla†	0.395	0.537	0.177	0.000	0.000
	SLASH†	0.905	0.858	0.570	0.000	0.000
	Ours	**0.964±0.004**	**0.895±0.003**	**0.894±0.008**	**0.035±0.002**	**0.073±0.003**
Qwen3-14B	Vanilla†	0.545	0.217	0.155	0.018	0.025
	SLASH†	0.740	0.733	0.155	**0.100**	0.037
	Ours	**0.976±0.007**	**0.916±0.004**	**0.899±0.004**	0.050±0.004	**0.080±0.003**
GraphWiz-Mistral-7B	Vanilla†	0.923	**0.897**	0.726	0.200	0.306
	SLASH†	0.914	0.891	0.726	0.211	0.306
	Ours	**0.933±0.008**	0.888±0.005	**0.896±0.004**	**0.306±0.004**	**0.472±0.003**
GraphWiz-LLaMA2-7B	Vanilla†	0.874	0.814	0.854	0.471	0.231
	SLASH†	0.891	0.806	0.854	0.446	0.231
	Ours	**0.943±0.004**	**0.854±0.003**	**0.872±0.004**	**0.510±0.004**	**0.356±0.003**
GraphWiz-LLaMA2-13B	Vanilla†	0.891	0.891	0.883	0.554	0.246
	SLASH†	0.931	0.894	0.883	0.554	0.249
	Ours	**0.953±0.002**	**0.918±0.003**	**0.905±0.003**	**0.6330±0.005**	**0.407±0.003**

Table 5: *Continued from Table 4.* Accuracy on GraphInstruct (range: 0-1). Avg. for our method is the arithmetic mean of the means for the nine tasks. † denotes values taken directly from published results in the cited studies. The best mean for each backbone is shown in bold. We report the mean and standard deviation over five runs for each task.

Backbone	Method	Triangle	Flow	Hamilton	Subgraph	Avg.
Llama-3.2-3B	Vanilla†	0.000	0.075	0.092	0.000	0.085
	SLASH†	0.048	0.113	0.278	0.268	0.214
	Ours	**0.103±0.002**	**0.166±0.004**	**0.720±0.003**	**0.797±0.003**	**0.456**
Llama-3.1-8B	Vanilla†	0.018	0.128	0.305	0.385	0.282
	SLASH†	0.052	0.175	0.315	0.560	0.352
	Ours	**0.120±0.003**	**0.214±0.005**	**0.797±0.005**	**0.835±0.003**	**0.528**
Qwen3-4B	Vanilla†	0.000	0.000	0.223	0.005	0.151
	SLASH†	0.003	0.015	0.290	0.145	0.195
	Ours	**0.016±0.001**	**0.045±0.005**	**0.683±0.006**	**0.813±0.004**	**0.442**
Qwen3-8B	Vanilla†	0.000	0.062	0.177	0.215	0.174
	SLASH†	0.000	0.062	0.318	0.588	0.367
	Ours	**0.028±0.002**	**0.085±0.004**	**0.785±0.005**	**0.844±0.003**	**0.511**
Qwen3-14B	Vanilla†	0.013	0.068	0.190	0.588	0.202
	SLASH†	0.013	0.070	0.247	0.588	0.298
	Ours	**0.053±0.003**	**0.126±0.004**	**0.823±0.007**	**0.865±0.002**	**0.532**
GraphWiz-Mistral-7B	Vanilla†	0.389	0.306	0.291	0.863	0.544
	SLASH†	0.369	0.311	0.300	0.866	0.544
	Ours	**0.491±0.003**	**0.440±0.004**	**0.824±0.005**	**0.874±0.003**	**0.680**
GraphWiz-LLaMA2-7B	Vanilla†	0.523	0.437	0.814	0.754	0.642
	SLASH†	0.517	0.437	0.809	0.757	0.639
	Ours	**0.618±0.002**	**0.505±0.004**	**0.833±0.003**	**0.861±0.006**	**0.706**
GraphWiz-LLaMA2-13B	Vanilla†	0.546	0.423	0.494	0.809	0.637
	SLASH†	0.450	0.443	0.620	0.754	0.642
	Ours	**0.613±0.004**	**0.540±0.004**	**0.844±0.003**	**0.87±0.0001**	**0.743**

Table 6: Accuracy on MolecularNet (range: 0-1). Avg. is the arithmetic mean across the five datasets. † denotes published results from the original studies; the best mean for each backbone is shown in bold. We report the mean and standard deviation over five runs for each task.

Backbone	Method	BACE	BBBP	ClinTox	HIV	Tox21	Avg.
Llama-3.2-3B	Vanilla†	0.085	0.495	0.675	0.932	0.418	0.521
	SLASH†	0.493	0.495	0.793	0.932	0.500	0.643
	Ours	**0.648±0.031**	**0.564±0.032**	**0.854±0.032**	**0.940±0.027**	**0.584±0.030**	**0.718**
Llama-3.1-8B	Vanilla†	0.335	0.430	0.875	0.932	0.510	0.616
	SLASH†	0.335	0.560	0.875	0.932	0.518	0.644
	Ours	**0.792±0.028**	**0.698±0.035**	**0.892±0.029**	**0.954±0.031**	**0.642±0.027**	**0.796**
Qwen3-4B	Vanilla†	0.230	0.315	0.578	0.113	0.488	0.345
	SLASH†	0.525	0.420	0.843	0.935	0.488	0.642
	Ours	**0.686±0.028**	**0.542±0.033**	**0.900±0.033**	**0.952±0.031**	**0.554±0.030**	**0.727**
Qwen3-8B	Vanilla†	0.233	0.060	0.018	0.003	0.003	0.063
	SLASH†	0.280	0.430	0.855	0.670	0.485	0.544
	Ours	**0.760±0.030**	**0.694±0.043**	**0.922±0.029**	**0.954±0.024**	**0.594±0.031**	**0.785**
Qwen3-14B	Vanilla†	0.345	0.430	0.875	0.932	0.513	0.619
	SLASH†	0.615	0.430	0.875	0.932	0.497	0.670
	Ours	**0.878±0.030**	**0.744±0.031**	**0.912±0.028**	**0.960±0.026**	**0.620±0.031**	**0.823**
MolecularGPT	Vanilla†	0.650	0.430	0.200	0.068	0.470	0.364
	SLASH†	0.650	0.430	0.200	0.068	0.480	0.366
	Ours	**0.916±0.024**	**0.640±0.031**	**0.512±0.028**	**0.832±0.026**	**0.508±0.029**	**0.682**

Table 9 presents the link prediction results. Among the published comparison methods, R-GFM achieves the highest average AUC-ROC, at 86.90%. The Llama-3.1-8B and Qwen3-8B configurations achieve 87.37% and 87.50%, corresponding to average gains of 0.47 and 0.60 percentage points, respectively, and each obtains higher means than R-GFM on five of the seven datasets. Qwen3-8B improves by 1.95 and 0.88 percentage points on Cora and Texas, respectively, but declines by 0.56 and 0.04 percentage points on Pubmed and Citeseer. Llama-3.1-8B falls slightly below R-GFM on Pubmed and Photos. The main finding for this task is therefore competitive average performance, with remaining tradeoffs that depend on the dataset and backbone.

RQ1 and RQ2 provide complementary evidence: the former examines the applicability of the overall training approach across backbones on algorithmic graph tasks and molecular graph tasks, whereas the latter examines its performance across datasets on standard tasks involving nodes and edges. Both support further investigation of directly modifying the flow of structural information in large language models. They do not establish comprehensive superiority over existing methods under a common budget or guarantee improvements on every task.

G HYPERPARAMETER ANALYSIS

To address RQ4, we examine the number of global readouts K, the decorrelation threshold τ, and the loss weight λ_{read} on GraphInstruct and MolecularNet, reporting macro averages across tasks for Llama-3.1-8B and Qwen3-8B separately. We vary one parameter at a time while fixing the others. The mean cosine similarity between readouts describes the extent of representational overlap; a decrease in similarity does not by itself imply improved task performance.

Number of global readouts K. Table 10 examines $K \in \{1, 2, 4, 8, 16\}$. When $K = 1$, no comparison between distinct readouts is possible, so the readout decorrelation term is 0 and the similarity column is marked as not applicable. Increasing the readout count from $K = 1$ to 4 improves accuracy on GraphInstruct and MolecularNet by 5.6 and 8.4 percentage points for Llama-3.1-8B and by 6.3 and 8.7 percentage points for Qwen3-8B, respectively. Further increases to 8 or 16 yield no additional accuracy gains, even though mean cosine similarity continues to decrease. Thus, increasing the number and dissimilarity of readouts does not automatically produce more useful graph representations; $K = 4$ offers a favorable balance between performance and overhead within the search range considered.

Table 7: Node classification accuracy (%). † denotes values taken directly from published results in the cited studies. Avg. for our method is the arithmetic mean of the means for the eight datasets; aggregate values for published baselines are retained from the original studies. The best mean in each column is shown in bold, and the second best is underlined. Additional columns are reported in Table 8.

Method	Wisconsin	Cornell	Citeseer	Cora
Task-Supervised GNNs				
GCN†	17.46±3.35	19.53±5.77	26.89±3.01	31.98±4.11
GAT†	16.86±3.16	16.51±8.41	25.27±2.10	26.81±4.42
H2GCN†	20.34±5.77	29.30±10.40	24.47±1.35	23.61±3.92
FAGCN†	26.19±7.93	28.72±8.12	20.56±1.02	20.75±3.85
Self-Supervised Pretraining + Fine-tuning				
DGI†	28.24±5.30	29.73±10.11	34.52±8.81	39.88±5.53
GraphCL†	32.16±8.83	31.35±12.47	27.56±5.01	35.26±8.62
GraphACL†	29.41±11.09	25.41±9.86	28.34±4.19	36.96±8.71
MixHop†	29.82±9.32	17.14±5.54	21.35±3.67	21.46±5.95
Prompt-based Adaptation				
GPPT†	23.53±15.93	25.41±5.27	33.24±8.73	40.60±6.00
GPF†	23.53±9.36	36.22±9.46	27.16±4.10	34.74±3.48
GraphPrompt†	20.39±7.92	28.65±5.92	19.80±2.79	33.68±9.83
GraphPrompt+†	16.86±5.63	29.19±5.24	28.28±3.55	29.76±6.06
Graph Foundation Models				
GCOPE†	28.69±7.00	25.00±6.58	38.62±2.58	39.05±2.53
SAMGPT†	21.57±12.09	36.22±5.30	28.78±8.99	43.08±10.40
MDGFM†	17.65±7.34	26.49±10.73	23.32±15.32	42.88±9.12
RiemannGFM†	31.20±9.35	31.35±7.76	31.86±6.45	43.48±5.28
GPM†	29.19±4.69	27.83±11.00	29.88±6.36	38.96±8.18
G2PM†	34.70±8.73	32.96±5.44	31.04±2.90	40.04±3.13
R-GFM†	35.41±7.29	36.71±9.92	<u>57.54±9.49</u>	<u>49.50±3.97</u>
Ours(Llama-3.1-8B)	**36.02±6.15**	**37.28±7.54**	56.12±7.21	47.18±5.86
Ours(Qwen3-8B)	<u>35.56±5.03</u>	<u>36.94±4.68</u>	**57.68±6.35**	**51.62±4.91**

Table 8: *Continued from Table 7.* Node classification accuracy (%). † denotes values taken directly from published results in the cited studies. Avg. for our method is the arithmetic mean of the means for the eight datasets; aggregate values for published baselines are retained from the original studies. The best mean in each column is shown in bold, and the second best is underlined.

Method	Pubmed	Computers	Photos	Texas	Avg.
Task-Supervised GNNs					
GCN†	44.29±2.22	39.43±5.68	50.39±1.19	18.48±4.08	31.06
GAT†	45.11±5.52	38.05±4.47	56.51±2.35	18.36±4.63	30.44
H2GCN†	40.97±1.26	19.29±6.87	42.95±4.32	29.94±16.38	28.86
FAGCN†	41.65±1.46	27.02±2.97	47.91±2.17	31.35±14.17	30.52
Self-Supervised Pretraining + Fine-tuning					
DGI†	47.10±4.95	34.59±5.94	50.66±6.93	29.73±6.34	36.81
GraphCL†	43.66±3.12	34.40±6.93	48.86±5.53	31.35±22.66	35.58
GraphACL†	38.20±10.11	44.42±7.06	56.60±8.90	24.86±9.44	35.53
MixHop†	43.57±4.05	29.77±15.57	32.77±9.26	26.09±15.93	27.75
Prompt-based Adaptation					
GPPT†	41.64±9.29	44.57±13.38	53.94±11.93	18.38±12.00	35.16
GPF†	26.62±10.18	42.15±8.75	59.13±6.05	28.65±17.31	34.78
GraphPrompt†	42.94±1.98	33.34±6.09	42.60±6.75	24.32±9.17	30.71
GraphPrompt+†	38.74±1.70	43.01±6.72	58.23±4.61	19.46±14.84	32.94
Graph Foundation Models					
GCOPE†	44.52±2.51	28.50±4.91	59.17±2.45	19.50±12.18	35.38
SAMGPT†	37.95±9.58	45.80±4.54	49.80±11.89	31.89±11.26	36.89
MDGFM†	38.42±7.92	42.68±2.53	57.78±9.07	31.54±11.78	35.09
RiemannGFM†	44.10±5.05	46.20±8.79	57.20±16.84	29.12±7.44	39.31
GPM†	35.80±8.20	48.58±10.66	37.85±8.17	27.70±12.09	34.47
G2PM†	49.71±8.66	51.92±10.10	59.90±5.02	30.37±9.36	41.33
R-GFM†	<u>49.80±5.38</u>	52.30±3.33	61.08±5.26	32.36±12.10	<u>46.84</u>
Ours(Llama-3.1-8B)	48.52±7.24	<u>54.06±5.28</u>	**61.85±6.18**	<u>32.98±8.92</u>	46.75
Ours(Qwen3-8B)	**50.88±7.31**	**54.54±6.35**	<u>61.32±5.22</u>	**33.54±9.05**	**47.76**

Table 9: Link prediction AUC-ROC (%). † denotes values taken from published results in the cited studies. Avg. is the arithmetic mean across the seven datasets. The best mean in each column is shown in bold, and the second best is underlined.

Method	Wisconsin	Cornell	Citeseer	Pubmed
GCOPE†	83.12±1.51	64.72±4.99	85.09±0.34	84.99±0.06
SAMGPT†	65.32±7.27	53.53±4.57	89.91±2.70	84.92±0.40
MDGFM†	67.46±6.82	54.92±21.10	89.12±4.68	84.78±2.09
RiemannGFM†	78.19±2.08	79.06±2.96	90.63±0.58	86.37±0.33
R-GFM†	84.15±0.71	85.90±0.69	<u>90.88±0.70</u>	**88.62±0.41**
Ours(Llama-3.1-8B)	<u>84.86±0.73</u>	<u>86.62±0.71</u>	**91.72±0.72**	<u>88.48±0.43</u>
Ours(Qwen3-8B)	**85.02±0.70**	**86.78±0.68**	90.84±0.69	88.06±0.41

Method	Cora	Photos	Texas	Avg.
GCOPE†	88.71±0.57	78.45±5.51	80.51±1.75	80.80
SAMGPT†	88.62±3.51	80.59±0.33	51.20±4.42	73.44
MDGFM†	88.41±2.12	81.24±5.80	65.22±4.05	75.88
RiemannGFM†	88.54±0.11	81.44±0.95	72.41±5.96	82.38
R-GFM†	89.27±0.64	<u>81.53±0.83</u>	87.94±0.96	86.90
Ours(Llama-3.1-8B)	<u>90.44±0.66</u>	81.38±0.85	<u>88.08±0.98</u>	<u>87.37</u>
Ours(Qwen3-8B)	**91.22±0.63**	**81.76±0.82**	**88.82±0.95**	**87.50**

Table 10: Effect of the global readout token count K, with $\tau = 0.6$ and $\lambda_{\text{read}} = 0.01$ fixed.

Backbone	K	GraphInstruct Mean accuracy	MolecularNet Mean accuracy	Mean readout cosine similarity
Llama-3.1-8B	1	0.472	0.712	–
	2	0.508	0.768	0.81
	4	**0.528**	**0.796**	0.62
	8	0.526	0.793	0.58
	16	0.519	0.786	0.56
Qwen3-8B	1	0.448	0.698	–
	2	0.486	0.754	0.83
	4	**0.511**	**0.785**	0.64
	8	0.508	0.781	0.60
	16	0.501	0.774	0.58

Decorrelation threshold τ. Table 11 examines $\tau \in \{0.0, 0.2, 0.4, 0.6, 0.8, 0.95\}$. Under the unilateral penalty, a lower threshold penalizes a wider range of positive cosine similarities without enforcing strict orthogonality between readouts. For both backbones, macro average accuracy on each benchmark is highest at $\tau = 0.6$. Relative to $\tau = 0$, Llama-3.1-8B improves by 2.7 and 3.8 percentage points, and Qwen3-8B by 2.9 and 4.3 percentage points, respectively. Meanwhile, mean similarity increases with the threshold. This trend is consistent with the motivation to retain a moderate amount of shared graph information, although similarity alone cannot establish whether individual readouts encode specific structural semantics. Performance declines when τ is further increased to 0.95, indicating that neither an overly strong nor an overly weak constraint is optimal in the current setting.

Table 11: Effect of the decorrelation threshold τ, with $K = 4$ and $\lambda_{\text{read}} = 0.01$ fixed.

Backbone	τ	GraphInstruct Mean accuracy	MolecularNet Mean accuracy	Mean readout cosine similarity
Llama-3.1-8B	0.0	0.501	0.758	0.18
	0.2	0.512	0.774	0.31
	0.4	0.521	0.786	0.47
	0.6	**0.528**	**0.796**	0.62
	0.8	0.524	0.791	0.74
	0.95	0.516	0.782	0.86
Qwen3-8B	0.0	0.482	0.742	0.20
	0.2	0.494	0.758	0.33
	0.4	0.503	0.772	0.49
	0.6	**0.511**	**0.785**	0.64
	0.8	0.507	0.780	0.76
	0.95	0.498	0.770	0.88

Decorrelation loss weight λ_{read}. Table 12 examines $\lambda_{\text{read}} \in \{0, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Increasing the weight from 0 to 10^{-2} improves accuracy on the two benchmarks by 0.9 and 1.2 percentage points for Llama-3.1-8B and by 1.3 and 1.5 percentage points for Qwen3-8B, respectively. At a weight of 1, both backbones perform substantially worse than at 10^{-2}, suggesting that excessive weight on the auxiliary objective may impair optimization of the primary task. Llama-3.1-8B achieves its highest mean accuracy on MolecularNet at 10^{-1}, exceeding the result at 10^{-2} by 0.2 percentage points. This small difference indicates that a shared weight need not be individually optimal for every backbone and dataset.

Table 12: Effect of the decorrelation loss weight λ_{read}, with $K = 4$ and $\tau = 0.6$ fixed.

Backbone	λ_{read}	GraphInstruct Mean accuracy	MolecularNet Mean accuracy
Llama-3.1-8B	0	0.519	0.784
	10^{-3}	0.524	0.790
	10^{-2}	**0.528**	0.796
	10^{-1}	0.521	**0.798**
	1	0.503	0.762
Qwen3-8B	0	0.498	0.770
	10^{-3}	0.504	0.777
	10^{-2}	**0.511**	**0.785**
	10^{-1}	0.502	0.774
	1	0.481	0.746

Considering the results for both backbones, we use $K = 4$, $\tau = 0.6$, and $\lambda_{\text{read}} = 0.01$ throughout the remaining experiments. These results characterize sensitivity within the specified search ranges.

H COMPUTATIONAL COMPLEXITY AND RUNTIME OVERHEAD

To address RQ5, we analyze the additional overhead of our method, which arises primarily from structural token additions, visibility masks, structural parameters, and the readout decorrelation loss. Let the base sequence length be L_0 and the number of added structural token instances be $\Delta L = O(|E|+|V|+K)$. The actual input length is then $L = L_0 + \Delta L$. All complexity expressions below use this actual sequence length, including the structural token additions.

Parameter count. For a fixed role set, SR-RoPE introduces a constant number of role position parameters. Structural attention adds three scalar biases per layer, giving $O(N_L)$ parameters across N_L layers. Global readout role embeddings require $O(Kd)$ parameters, while embeddings for the remaining fixed set of structural token types require $O(d)$. The structural module therefore has a parameter complexity of $O(N_L + Kd)$, which is small relative to the backbone parameter count. LoRA parameters are part of the fine tuning overhead and should be counted separately from the added structural module.

Attention time and GPU memory. With a dense attention implementation, the structural visibility mask changes only which token pairs are permitted; it does not automatically reduce matrix computation. The time complexity of self attention in a single layer remains $O(L^2 d)$, and the space complexity is $O(L^2)$ when attention scores are stored explicitly. Relative to the base sequence length, the increase in the dominant attention term is proportional to $(L_0 + \Delta L)^2 - L_0^2$. A future implementation with block sparsity that supports the current visibility structure could exploit connections within local structures and readout connections on the order of $K|V|$ to reduce some of the overhead in the graph region. However, the visibility scopes of the query and answer, kernel implementation, and truncation strategy also affect overall cost. We do not treat this potential as a verified speedup.

Auxiliary loss. Computing pairwise cosine similarities among K readouts requires $O(K^2 d)$ time and $O(K^2)$ storage for the similarities. With $K = 4$, this computation is much smaller than attention over long sequences, although its exact share of the overall cost depends on the implementation.

Runtime records. Table 13 summarizes the mean wall clock times by group retained from the initial draft. The group of small and medium backbones comprises Llama-3.2-3B, Llama-3.1-8B, Qwen3-4B, Qwen3-8B, GraphWiz-Mistral-7B, and GraphWiz-LLaMA2-7B; the group of large backbones comprises Qwen3-14B and GraphWiz-LLaMA2-13B. Hardware configurations are specified in Section E.4.

Table 13: Mean runtime by group recorded during actual runs, in seconds. Total, training, and evaluation times are retained separately as originally recorded.

Task	Backbone group	Total time	Training time	Evaluation time
GraphInstruct	Small and medium	35534.675	25369.231	10052.703
GraphInstruct	Large	48682.505	35111.016	13591.254
MolecularNet	Small and medium	59989.347	59370.494	395.595
MolecularNet	Large	83325.203	81765.044	541.490
Node classification	Two 8B backbones	24516.054	20069.748	3843.154
Link prediction	Two 8B backbones	446482.851	425325.606	19751.068

I CASE STUDIES AND MECHANISM ILLUSTRATIONS

We use four case figures to illustrate readout complementarity, permutation robustness, aggregation balance, and their boundary conditions. The individual cases state their empirical or illustrative scope; these figures are not used to estimate occurrence frequencies in real-world tasks.

I.1 READOUT COMPLEMENTARITY

To examine whether multiple global readout token representations capture complementary information about a graph, we select representative examples with correct predictions, visualize the attention distribution of each readout token $u_1, \ldots, u_K$ over node summaries NS_v, and compute the cosine

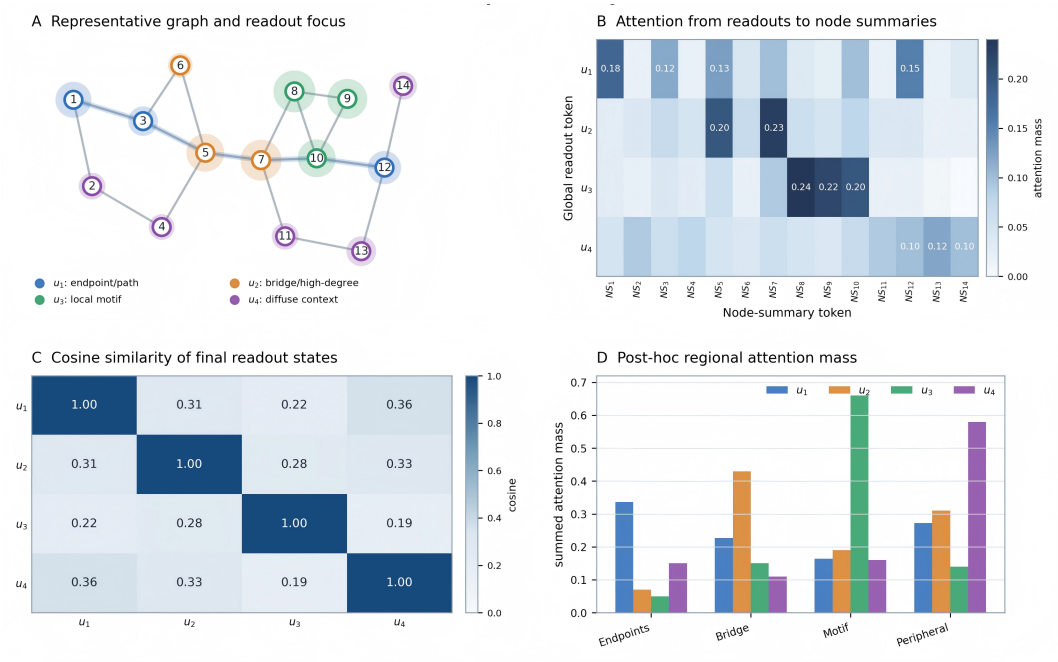

Figure 4: Illustration of readout complementarity. A: Graph structure and regions defined for interpretation. B: Attention from four global readouts to node summaries. C: Cosine similarity between final readout representations. D: Attention mass aggregated over endpoints, bridging regions, motifs, and peripheral regions.

similarity between their final hidden representations. As shown in Figure 4, the readouts exhibit distinct attention peaks across node summaries: u_1 focuses mainly on path endpoints and key nodes along the main path, u_2 tends to cover bridging nodes and regions with high node degrees, u_3 concentrates attention around local motifs, and u_4 provides more diffuse coverage of peripheral context. Regional attention statistics also reveal substantial differences in how the readouts distribute attention mass across endpoints, bridging regions, local motifs, and peripheral nodes, suggesting that they do not simply replicate the same global information.

The cosine similarity matrix of the readout representations further shows that similarities between distinct readouts are generally low, although the representations remain far from strictly orthogonal. This pattern is consistent with sharing global information needed for prediction while retaining distinct preferences for structural features. We do not assign predefined semantics such as "path," "connectivity," or "motif" to individual readouts before training; these interpretations are based solely on post hoc analysis of attention distributions and representation similarities after training. The result therefore suggests that multiple readouts can develop complementary graph representations with some division of roles in the absence of explicit role supervision.

I.2 Permutation robustness

To examine whether the model relies on positional shortcuts introduced by serialization, we randomly generate multiple topologically equivalent serializations of the same graph. Specifically, we randomize only the order of node blocks and the order of edge records within each node, keeping the query, graph structure, and ground truth label unchanged. We then compare prediction agreement and logit fluctuations across permutations. For M permutations, pairwise prediction agreement is defined as

$$C_{\text{pair}} = \frac{2}{M(M-1)} \sum_{1 \le a < b \le M} \mathbb{I}\left[\widehat{y}^{(a)} = \widehat{y}^{(b)}\right], \tag{39}$$

where $\widehat{y}^{(a)}$ is the prediction under permutation a. Figure 5 presents an illustrative case: across 24 equivalent serializations, the logit margin $\Delta = z_{\text{Yes}} - z_{\text{No}}$ for Vanilla crosses 0 repeatedly, giving

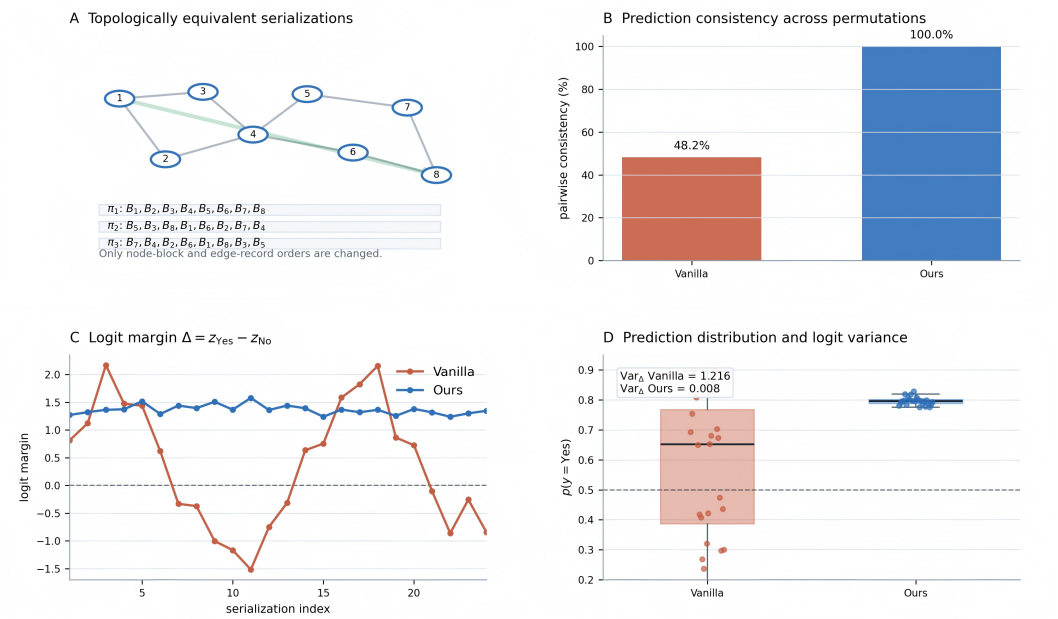

Figure 5: Illustration of permutation robustness under topologically equivalent serializations. A: Different node block orders for a fixed graph. B: Pairwise prediction agreement across 24 permutations. C: Binary classification logit margins and the decision boundary at zero. D: Distributions of predicted probabilities and margin variances.

a pairwise prediction agreement of 48.2%; the margin for Ours remains positive, with an agreement of 100.0%. The margin variances are 1.216 and 0.008, respectively. This figure illustrates how output stability under permutation perturbations should be measured. It does not provide experimental evidence that SR-RoPE alone produces an improvement of this magnitude, nor does consistent prediction necessarily imply correctness. The distribution of predicted probabilities for Ours is also more concentrated in this illustration, indicating less sensitivity to topologically equivalent permutations within the simulated example. This pattern illustrates the intended reduction in positional shortcuts induced by serialization order. It does not, by itself, establish an improvement in final Accuracy or demonstrate that the model relies more strongly on graph structure when reasoning.

I.3 Aggregation Balance

To examine whether the aggregation of structural information is systematically affected by sequence position, we select an example with pronounced dependencies of higher order and change only the positions of the critical node blocks, keeping the graph structure, query, and ground truth label fixed. As shown in Figure 6, we place the critical node set $\{1, 3, 5, 8, 10, 12\}$ near the beginning (π_F) or the end (π_B) of the node block sequence. We compare the coverage of node summaries by answer attention, the final predictions, and the distances between intermediate representations of matched node summaries across the two permutations. In the figure, Vanilla exhibits a pronounced positional bias: mean attention coverage of the critical nodes is only 0.292 when they occur near the beginning, increasing to 0.792 near the end, a difference of 0.500. This imbalance in coverage is accompanied by a change in the final prediction: the probability of the correct class increases from 0.46 under π_F to 0.84 under π_B. Substantial drift also occurs in the intermediate representations, with a mean representation distance of 0.373, increasing to 0.505 for the critical nodes.

Ours exhibits more balanced structural aggregation across the two permutations: mean coverage of the critical nodes is 0.824 and 0.817, respectively, with a reported difference of only 0.008. The probability of the correct class remains near 0.81 and 0.82, and the mean representation distance decreases to 0.064. This pattern is consistent with the intended role of the structural visibility mask: reducing positional bias due to differences in visible prefixes under standard causal attention, pro-

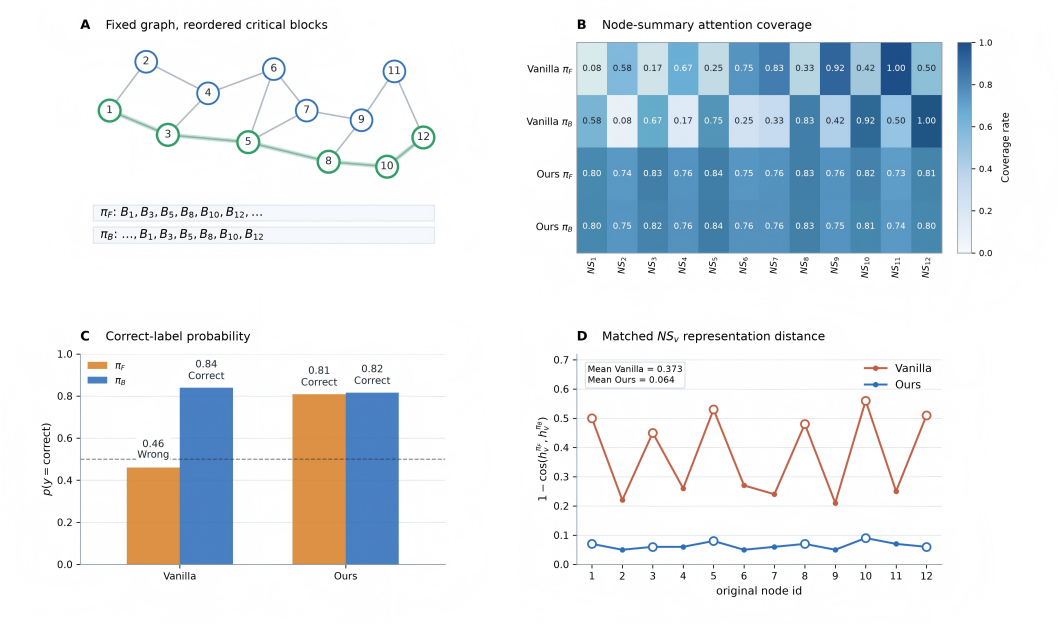

Figure 6: Illustration of aggregation balance under reordering of critical node blocks. A: A fixed graph with critical node blocks placed near the beginning or the end of the sequence. B: Summary coverage aligned by original node identity. C: Probability of the correct class. D: Cosine distance between matched node summaries under the two permutations.

viding node summaries with more consistent access to their local structures, and enabling global readouts to predict from the same set of stable node summaries. The illustration alone does not isolate the causal contribution of the mask.

I.4 Boundary conditions: stable predictions can still be incorrect

We illustrate a potential limitation of our method using a synthetic case in which continuous attributes dominate the prediction task. Figure 7 presents a synthetic example whose label is determined primarily by continuous attributes. Keeping all edges and other attributes fixed, we change only attribute x_7 of v_7 from 0.50 to 0.54. The constructed nonlinear decision score changes from -0.0311 to $+0.0114$, and the label changes from 0 to 1. For the positive example G^+, the configurations labeled Vanilla and Ours assign probabilities of 0.391 and 0.380 to the positive class, respectively, both below the threshold of 0.5. Their simulated predictions are incorrect under all 24 equivalent serializations, although the latter has a smaller logit margin variance.

This case distinguishes robustness to structural permutations from task correctness. When the main difficulty lies in precise encoding of continuous attributes and their nonlinear interactions, a model may consistently produce an incorrect answer even after unnecessary order perturbations are removed. Our structural design aims to improve the alignment between graphs and sequence computation; it cannot replace attribute modeling or precise numerical reasoning. The illustration also does not establish how frequently this synthetic situation occurs in real data.

J Dataset and sample statistics

Table 14: GraphInstruct and MolecularNet.

Dataset/task	Original training source	Original test	Preprocessed training	Preprocessed test
GraphInstruct				
Cycle	2,717	400	2,717	400
Connect	2,687	400	2,687	400
Bipartite	2,013	400	2,013	400
Topology	872	400	872	400
Shortest	1,392	400	1,392	400
Triangle	2,756	400	2,756	400
Flow	405	400	405	400
Hamilton	2,097	400	2,097	400
Subgraph	3,186	400	3,186	400
Total	18,125	3,600	18,125	3,600
MolecularNet				
BACE	1,513	100	1,513	100
BBBP	2,050	100	2,050	100
ClinTox	1,484	100	1,484	100
HIV	41,127	100	41,127	100
Tox21	8,014	100	8,014	100
Total	54,188	500	54,188	500

Table 15: Standard node classification datasets: full graph sizes and node splits.

Dataset	$\|V\|$	$\|E\|$	Classes	Training	Validation	Test
Wisconsin	251	450	5	120	80	51
Cornell	183	277	5	87	59	37
Citeseer	3,327	4,552	6	120	500	1,000
Cora	2,708	5,278	7	140	500	1,000
Pubmed	19,717	44,324	3	60	500	1,000
Amazon-Computers	13,752	245,861	10	8,252	2,750	2,750
Amazon-Photos	7,650	119,081	8	4,591	1,530	1,529
Texas	183	279	5	87	59	37
Total task samples	–	–	–	13,457	5,978	7,404

Table 16: Standard link prediction datasets: numbers of node pair instances.

Dataset	Training pairs	Validation pairs	Test pairs
Wisconsin	720	90	90
Cornell	442	54	58
Citeseer	7,282	910	912
Pubmed	70,918	8,864	8,866
Cora	8,444	1,054	1,058
Amazon-Photos	190,528	23,816	23,818
Texas	446	54	58
Total	278,780	34,842	34,860

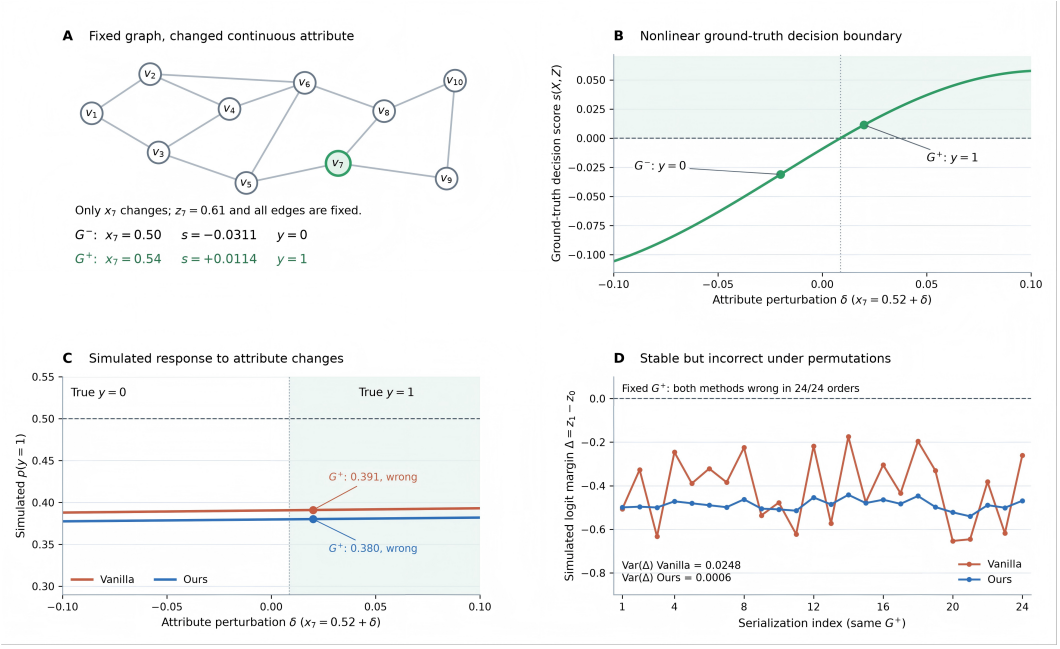

Figure 7: Failure mechanism when continuous attributes dominate the label. A: A small change in a single node attribute on a fixed topology. B: A manually specified nonlinear decision boundary. C: Simulated prediction probabilities that respond insufficiently to the attribute change. D: Incorrect predictions for the same positive example under all 24 equivalent permutations, with different fluctuation magnitudes.

Table 17: Mean graph sizes after preprocessing, measured as the mean numbers of nodes and edges in the graph associated with each sample.

Dataset/task	Training $\overline{\lvert V \rvert}$	Training $\overline{\lvert E \rvert}$	Test $\overline{\lvert V \rvert}$	Test $\overline{\lvert E \rvert}$
GraphInstruct				
Cycle	32.28	88.79	49.75	80.86
Connect	30.72	44.92	49.92	53.88
Bipartite	32.24	74.30	51.79	179.62
Topology	14.64	80.94	26.62	169.30
Shortest	17.55	38.86	49.62	107.73
Triangle	6.34	3.45	13.47	15.99
Flow	17.95	36.10	26.06	45.59
Hamilton	16.87	105.66	25.33	158.83
Subgraph	19.36	29.91	28.53	71.40
MolecularNet				
BACE	34.09	36.86	33.35	36.08
BBBP	24.03	25.92	22.05	23.52
ClinTox	26.23	27.97	25.84	27.73
HIV	25.51	27.47	29.07	31.40
Tox21	18.51	19.23	19.34	20.09