SERIALIZATION IS NOT ENOUGH: FROM PRESENTATION TO UTILIZATION IN LLM-BASED GRAPH PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph serialization makes structural evidence explicit to large language models (LLMs), whereas this explicit presentation does not ensure that the evidence is reliably used for prediction. We study this distinction through degree profiles, which record category-specific neighbor counts. Constructing and using a controlled diagnostic GRAPHMACC, which separates recognition of category presence from calculation and comparison of category counts, we find a consistent recognition-utilization gap across LLMs and representative presentation-alignment methods. We further examine a possible contributor to this gap. Under a local exchangeability assumption, normalized attention exhibits *marginal response compression*, where successive occurrences of the same structural category induce progressively smaller changes in the aggregated representation, reducing the separation between adjacent counts. This motivates ***utilization alignment***, which directly targets how presented structural evidence is accumulated and propagated for prediction. We instantiate this idea with SAIL (**S**tructural **A**ccumulation and **I**ntegration for **L**LMs), which preserves occurrence-level structural information through unnormalized accumulation and uses the resulting structural state to correct the original LLM candidate scores. Under mild conditions, we further show that the accumulated differences can be mapped to improved prediction margins for adjacent counts. Experiments substantiate that SAIL improves category-count utilization across LLM families and betters graph prediction across node-, edge-, and graph-level tasks. Its gains are also complementary to existing presentation-alignment methods and persist under fine-tuning and in-context learning.

1 INTRODUCTION

Serialization enables natural language inference across graph prediction tasks by representing nodes, edges, and associated attributes as text (Zhu et al., 2024; Zhao et al., 2023). As such, references to individual graph elements are made explicit (Wu et al., 2026; Perozzi et al., 2024), making the evidence for a large language model (LLM) prediction accessible (Fatemi et al., 2024). However, this accessibility does not ensure that an LLM reliably uses the presented structure for prediction. Recent methods have mainly improved how graph evidence is presented to the LLM, such as reordering serialized elements, routing graph evidence, or incorporating structural relations into positional encoding and attention (Xypolopoulos et al., 2024; Wang et al., 2026; Loveland et al., 2026; Liu et al., 2026). We refer to these methods as ***presentation alignment***. Whereas they make relevant structural evidence easier to access or distinguish, a separate question remains, *i.e.,* does an LLM retain and use that evidence when a prediction depends on the number of times a structural pattern occurs?

Degree profile concretizes this question. For each node, its degree profile counts its neighbors by category and therefore captures differences in neighborhood composition (Abbe & Sandon, 2015). Two nodes can have the same degree and the same neighboring categories but different profiles. Such category-specific neighbor counts find classical graph tasks, such as community detection and graph isomorphism testing (Abbe & Sandon, 2015; McKay et al., 1981), and have been materialized in document and web-network classification (Getoor, 2005; Sen et al., 2008), link prediction (Liben-Nowell & Kleinberg, 2003), and molecular property prediction (Shervashidze et al., 2011), as shown in Figure 1 (a). We therefore ask whether LLMs can recover degree-profile counts when adjacency

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.

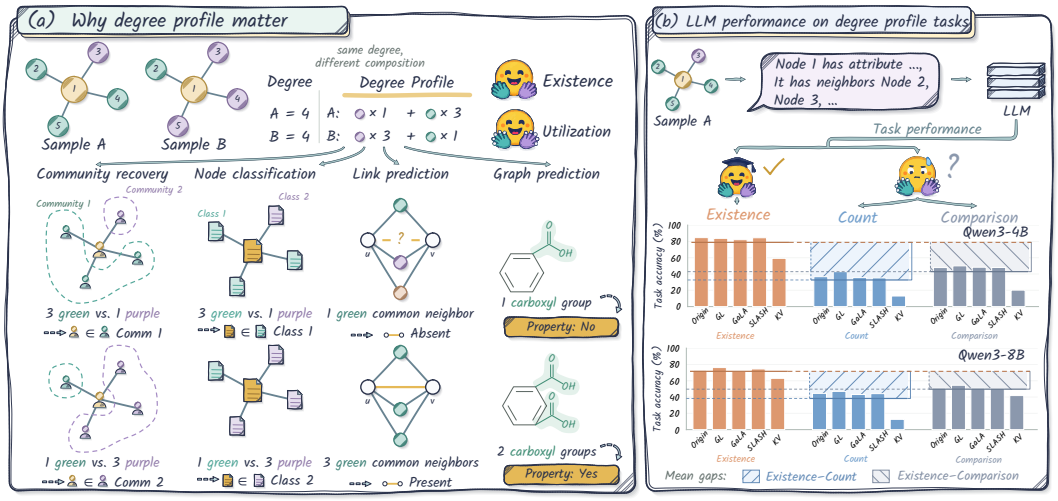


Figure 1: (a) Degree profiles capture both which categories are present in a neighborhood and how many neighbors belong to each. Differences in these profiles may lead to different outcomes across multiple graph tasks. (b) On GRAPHMACC, both LLMs recognize category presence more reliably than they calculate or compare counts, even with the evaluated presentation-alignment methods.

relations and node attributes are explicitly presented. If they cannot, it is unclear how reliably they can use the same structural information in more complex graph prediction tasks.

To answer this question, we construct GRAPHMACC, a controlled diagnostic that varies category-specific neighbor counts while holding total degree fixed. GRAPHMACC evaluates three capabilities that follow a functional progression (Cao et al., 2025), namely, (a) *existence*, which asks whether a category is represented in the neighborhood of a queried node; (b) *calculation*, which asks for the corresponding neighbor count; and (c) *comparison*, which asks which of two nodes has more neighbors from that category. Across two LLMs and four representative presentation-alignment methods, as shown in Figure 1 (b), we observe a consistent gap, where accuracy on existence is much higher than that on calculation or comparison. Thus, making structural evidence accessible to an LLM does not necessarily preserve the distinctions encoded by that evidence through subsequent computation.

We investigate one possible contributor through a simplified attention response. With fixed contributions from other neighbors and context tokens, successive additions of target-category neighbors produce progressively smaller changes under softmax normalization when the target-category and reference values differ. Consequently, representations associated with adjacent category counts can become progressively less separated even when the underlying structural pattern occurrences remain individually accessible. Presentation alignment can alter where evidence appears, which evidence is visible, or how attention is allocated, but it generally retains normalized aggregation. Improving structural presentation therefore does not necessarily eliminate this *marginal response compression*. This motivates a complementary ***utilization alignment***, which aims to directly improve how presented structural evidence is accumulated and propagated for prediction.

We instantiate this idea with **S**tructural **A**ccumulation and **I**ntegration for **LLM**s (SAIL). For a query-relative structural scope, SAIL constructs a message for each occurrence from a frozen LLM and sums these messages with query-relative role encodings without averaging them. When occurrence messages are fixed, adding a neighbor contributes an increment that does not shrink with the number of existing neighbors. SAIL then combines the accumulated state with query and candidate-answer representations to correct the original LLM candidate scores. Experiments substantiate both the effectiveness of our design and the distinction between presentation and utilization alignment. SAIL improves calculation and comparison in GraphMACC and yields gains across node-, edge-, and graph-level prediction benchmarks. Moreover, combining SAIL with presentation-alignment methods generally improves over using either component alone, and the same complementarity remains under fine-tuning and in-context learning. Ablation studies further attribute the gains to structural accumulation and its query- and candidate-conditioned integration.

Our main contributions include:

Table 1: Four types of structural presentation interventions and their representatives evaluated in this work.

Graph structure aspect	LLM-side intervention	Representative method
Placement	Token sequence reordering	Graph Linearization
Evidence visibility	Attention mask and routing	Graph-KV
Structural distance	RoPE, distance-based attention bias	GaLA
Local connectivity	Topology-related attention allocation	SLASH

- **Diagnosing structural utilization.** We use degree profiles and GRAPHMACC to distinguish presence recognition from multiplicity recovery and use, revealing a persistent recognition-utilization gap across evaluated LLMs and presentation-alignment methods.
- **Connecting the gap to normalized aggregation.** We characterize marginal response compression in softmax attention and show how successive structural occurrences can induce progressively smaller representational changes. This motivates utilization alignment as a complement to improving structural presentation.
- **Instantiating utilization alignment with SAIL.** We develop SAIL, which preserves occurrence-level structural information through unnormalized accumulation and integrates the resulting structural state through residual prediction correction. Empirical results show that SAIL improves structural utilization and provides additional gains when combined with presentation alignment, fine-tuning, and in-context learning.

2 MOTIVATION AND ANALYSIS

2.1 RELATED WORK

Graph serialization methods. A graph's structure is independent of the order in which its nodes are enumerated, provided that their attributes and connectivity remain unchanged. When the graph is serialized for an autoregressive LLM, however, the chosen order affects the causal visibility and positional relationships through which structural evidence is presented (Fatemi et al., 2024; Zhao et al., 2023; Xypolopoulos et al., 2024). Existing methods address this dependence through four broad types of presentation interventions, as shown in Table 1. One line of work adjusts serialization through graph traversal and reordering, which determine where structural elements appear in the token sequence (Xypolopoulos et al., 2024; Wang et al., 2023). Graph-aware masks and KV routing regulate evidence visibility by controlling which structural elements are accessible to a query (Wang et al., 2026; Guan et al., 2025). Positional methods incorporate structural relationships into attention through RoPE or graph-distance-dependent biases, which adjust how relative positions are represented (Chen et al., 2026; Loveland et al., 2026). While the first three families primarily regulate structural placement, visibility, and positional encoding, attention-redistribution methods use topology-related signals to guide attention reallocation to enhance the distinguishability of specific graph structures within the LLM (Liu et al., 2026; Zhan et al., 2026). The presentation alignment methods adjust how graph evidence is organized, presented, positionally encoded, or assigned attention while retaining the normalized attention-aggregation form and inter-layer propagation.

Degree profiles. Neighbor counts by category were recognized as an important local graph feature and widely used in classical graph algorithms as early as the 1970s and 1980s. Corneil & Gotlieb (1970) used these counts to subdivide vertex classes, and the graph isomorphism algorithm of McKay et al. (1981) also employed related partition refinement. In the early twenty-first century, the value of similar statistics was also demonstrated on graph benchmarks emerging at the time. For example, Getoor (2005) found that models retaining neighborhood statistics on category counts achieved better performance on Cora, CiteSeer, and WebKB than those recording only which neighbor categories were present. Similarly, Sen et al. (2008) demonstrated the importance of using counts of neighbors in each category as collective classification features for document and web network classification. Subsequently, Abbe & Sandon (2015) further formalized these statistics as *degree profiles* and used them to correct community assignments. More recently, LLM evaluations report declining accuracy on degree calculation and comparison as graphs grow and examine information presentation across sequence positions as a possible explanation (Cao et al., 2025). However, these evaluations remain insufficient to answer whether, when the relevant information is accessible, LLMs can preserve the category-count differences encoded by degree profiles during aggregation and subsequent propagation, and effectively use this information for prediction.

2.2 EMPIRICAL MOTIVATION

To evaluate LLMs' ability to recover and use degree-profile information from graph inputs for prediction, we conduct a set of controlled tests (GRAPHMACC) comprising *existence*, *calculation*, and *comparison* across different query nodes. In GRAPHMACC, we vary the target-category neighbor count while keeping the query node's total degree fixed within each degree setting, controlling for the effect of total degree on prediction. The three tasks use the same graph input and differ only in the task-specific prompt, with construction details provided in Appendix A.

For Qwen3-4B and Qwen3-8B, we evaluate the original LLM and four representative presentation-alignment methods: Graph Linearization, Graph-KV, GaLA, and SLASH, corresponding to the four intervention types introduced in Section 2.1. For each degree setting, we report the prediction accuracies of the three tasks separately, denoted by Acc_E, $\mathrm{Acc}_{\mathrm{Count}}$, and $\mathrm{Acc}_{\mathrm{compare}}$, as defined in Appendix A.5. As shown on the right of Figure 1, across both LLMs and all five evaluation settings, models recognize category presence in degree profiles substantially more reliably than they calculate or compare the corresponding neighbor counts. This gap persists across degree settings, and the evaluated presentation-alignment methods do not consistently close it. Complete GRAPHMACC results and supplementary validation on real graph topologies are provided in Appendix A.6.

These results show that recognizing category presence does not fully translate into reliable use of degree-profile counts. Beyond improving graph presentation, we therefore need to examine what limits their use during aggregation, subsequent propagation, and prediction in LLMs.

2.3 THEORETICAL ANALYSIS

To investigate a possible mechanism behind the recognition–utilization gap (Section 2.2), we examine how category counts affect the attention response. For a graph $G = (V, E)$, let $C_1, \ldots, C_K$ be a fixed partition of V into node categories and $N(v)$ the neighbors of v, excluding v itself.

Definition 2.1 (Degree Profile)
The degree profile of node v with respect to this partition is (Abbe & Sandon, 2015)

$$\mathbf{d}(v) = \big(d_1(v), \ldots, d_K(v)\big), \qquad d_j(v) = |N(v) \cap C_j|. \tag{1}$$

Here $d_j(v)$ counts the neighbors in category C_j, and $\sum_{j=1}^{K} d_j(v) = \deg(v)$ gives the total degree.

For a target category C_t, let $c = d_t(v)$ denote the target-category count. We next examine how attention combines information from these neighbors as c varies.

Following aggregation settings in GNNs (Zhang & Xie, 2020) and Transformers (Behrens et al., 2024), we propose a local *exchangeability assumption* for neighbor aggregation in LLMs.

Assumption 2.2 (Local Exchangeability)
For a fixed query, neighbors in C_t share a score s_t and value $\mathbf{v}_t$, both fixed as c varies.

Consider adding one structural occurrence from C_t with unnormalized attention weight $a_t = \exp(s_t) > 0$ to the query node's neighborhood. We hold neighbors outside C_t, context tokens, and their attention scores and value vectors fixed as c varies. Let $Z_{-t} > 0$ denote their total unnormalized attention weight and $\mathbf{v}_{-t}$ their attention-weighted average value.

Theorem 2.3 (Marginal Response Compression)
Under these conditions, for any integer $c \geq 0$, the reference attention response $\mathbf{h}_t^\star(c)$ and the increment from adding one neighbor in C_t are

$$\mathbf{h}_t^\star(c) = \frac{Z_{-t}\mathbf{v}_{-t} + ca_t\mathbf{v}_t}{Z_{-t} + ca_t}, \quad \boldsymbol{\Delta}_{t,c}^\star = \mathbf{h}_t^\star(c+1) - \mathbf{h}_t^\star(c). \tag{2}$$

If $\mathbf{v}_t \neq \mathbf{v}_{-t}$, then $\|\boldsymbol{\Delta}_{t,c+1}^\star\|_2 < \|\boldsymbol{\Delta}_{t,c}^\star\|_2$ and $\lim_{c\to\infty} \|\boldsymbol{\Delta}_{t,c}^\star\|_2 = 0$.

Each additional neighbor contributes the same unnormalized weight, but normalization reduces its effect as the aggregate approaches $\mathbf{v}_t$ (proof in Appendix B.1.2). The fixed-degree replacement case is analyzed separately in Appendix B.1.3. Although presentation-alignment methods may violate local exchangeability, any intervention that retains normalized aggregation keeps the response within the convex hull of value vectors, so some adjacent-count separation must shrink as the count range grows (Appendix B.2.3). This motivates utilization alignment, a complementary perspective that

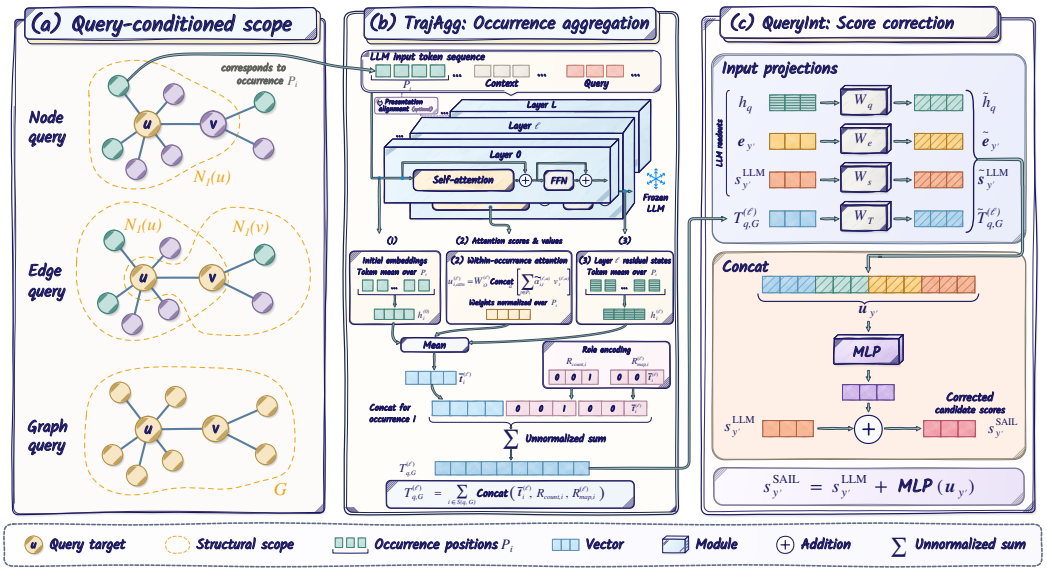


Figure 2: SAIL uses a frozen LLM. (a) The query determines the structural scope. (b) TrajAgg accumulates trajectory messages and role encodings through an unnormalized sum. (c) QueryInt combines this state with query and candidate information to correct the original candidate scores.

directly targets how presented graph evidence is aggregated and propagated through layers to support prediction. Within this perspective, we introduce SAIL as a simple correction that addresses the above limitation of normalized aggregation.

3 SAIL: Structural Accumulation for Graph Prediction

Motivated by the marginal response compression analyzed in Section 2.3, SAIL accumulates structural evidence separately from normalized attention while retaining the frozen LLM's original path for contextual processing. When combining SAIL with a presentation-alignment method, we retain that method's presentation operation and use the resulting LLM trajectories and candidate scores. The framework comprises two sequential modules (shown in Figure 2): **TrajAgg**, which accumulates occurrence-level trajectory messages through an unnormalized sum, and **QueryInt**, which performs query- and candidate-conditioned integration of the accumulated structural state through residual corrections to the LLM's candidate scores.

3.1 Structural Scope and Trajectory Extraction

For a graph $G = (V, E)$ and prediction query q, the structural scope $\mathcal{S}(q, G) \subseteq V$ comprises the target node's k-hop neighborhood for a node query, the union of both endpoint neighborhoods for an edge query, or all nodes for a graph query (Figure 2(a)). Here k is the neighborhood radius, and neighborhoods include their target nodes. Each node $i \in \mathcal{S}(q, G)$ is treated as a separate structural occurrence, so nodes in the same category remain distinct occurrences. For each occurrence i, we extract three vectors of LLM hidden width d_{LLM}: its initial embedding $h_i^{(0)}$, its contextual residual representation $h_i^{(\ell)}$ at extraction layer $\ell > 0$, and its attention-value contribution $u_{i,\text{attn}}^{(\ell)}$. The first two vectors average token embeddings and residual states, respectively, over the nonempty set $\mathcal{P}_i$ of token positions associated with node i in the serialized input.

To extract the attention channel, we renormalize attention separately within each occurrence. Suppressing head indices, let $s_{q,t}^{(\ell)}$ denote the pre-softmax attention score from the query readout position (the final prompt token) to token t. Its conditional weight within $\mathcal{P}_i$ is

$$\widetilde{\alpha}_{i,t}^{(\ell)} = \frac{\exp(s_{q,t}^{(\ell)})}{\sum_{t' \in \mathcal{P}_i} \exp(s_{q,t'}^{(\ell)})}, \qquad t \in \mathcal{P}_i. \tag{3}$$

Within each attention head, these weights pool the corresponding token value vectors (details in Appendix C.2). Concatenating the head contributions and applying the LLM output projection gives $u^{(\ell)}_{i,\mathrm{attn}}$. Compared with the original attention contribution $u^{(\ell),\mathrm{orig}}_{i,\mathrm{attn}}$ computed using the LLM's full attention denominator, $u^{(\ell)}_{i,\mathrm{attn}}$ removes the normalization-induced attenuation caused by other occurrences and context tokens by excluding them from its pooling denominator.

3.2 TrajAgg: Occurrence-Preserving Trajectory Aggregation

As a simple instantiation of utilization alignment, SAIL aims to recover sensitivity to count changes without explicitly separating contributions by category, and we use only one-hop neighborhoods ($k = 1$) for node and edge queries. TrajAgg therefore accumulates occurrence-level messages through an unnormalized sum, following the sum aggregation used in GIN (Xu et al., 2018). For each node $i \in \mathcal{S}(q, G)$, we first average the three channels into a trajectory message $\bar{\mathbf{t}}^{(\ell)}_i$:

$$\bar{\mathbf{t}}^{(\ell)}_i = \mathrm{Mean}\Big(h^{(0)}_i, h^{(\ell)}_i, u^{(\ell)}_{i,\mathrm{attn}}\Big), \quad \mathbf{T}^{(\ell)}_{q,G} = \sum_{i \in \mathcal{S}(q,G)} \mathrm{Concat}\Big(\bar{\mathbf{t}}^{(\ell)}_i, \mathbf{R}_{\mathrm{count},i}, \mathbf{R}^{(\ell)}_{\mathrm{map},i}\Big). \tag{4}$$

Here Concat concatenates vectors, and $\mathbf{T}^{(\ell)}_{q,G}$ is the accumulated structural state. With occurrence messages fixed, the unnormalized outer sum adds each new contribution without rescaling existing messages. Two relation encodings describe query-relative roles, which are distinct from node categories (details in Appendix C.3): $\mathbf{R}_{\mathrm{count},i}$ is a one-hot role indicator, while $\mathbf{R}^{(\ell)}_{\mathrm{map},i}$ places the trajectory message in the corresponding role block and zeros elsewhere. Specifically, node queries distinguish the target from its neighbors; edge queries distinguish occurrences belonging only to one endpoint's neighborhood or to both; graph queries use a single role.

3.3 QueryInt: Candidate-Conditioned Residual Integration

For query q, let h_q denote its LLM representation and $\mathcal{Y}(q)$ its predefined candidate-response set. For each candidate $y' \in \mathcal{Y}(q)$, let $\mathbf{e}_{y'}$ and $s^{\mathrm{LLM}}_{y'}$ denote its LLM representation and original score. QueryInt normalizes and projects $\mathbf{T}^{(\ell)}_{q,G}$, h_q, $\mathbf{e}_{y'}$, and the candidate score standardized across $\mathcal{Y}(q)$ into a common response space, then concatenates them into $\mathbf{u}_{y'}$ (Figure 2(c), details in Appendix C.4). Then, a simple multilayer perceptron (MLP) outputs a scalar correction:

$$s^{\mathrm{SAIL}}_{y'} = s^{\mathrm{LLM}}_{y'} + \mathrm{MLP}(\mathbf{u}_{y'}). \tag{5}$$

For a fixed integer $n \geq 0$, we compare prediction margins before and after correction for graph inputs G_n and G_{n+1} with target-category counts n and $n+1$, respectively. With occurrence messages fixed, their accumulated TrajAgg states satisfy:

$$\mathbf{T}^{(\ell)}_{q,G_{n+1}} = \mathbf{T}^{(\ell)}_{q,G_n} + \mathbf{m}_t. \tag{6}$$

Here $\mathbf{m}_t = \mathrm{Concat}(\bar{\mathbf{t}}^{(\ell)}_i, \mathbf{R}_{\mathrm{count},i}, \mathbf{R}^{(\ell)}_{\mathrm{map},i})$ is the complete message of the added target-category node i. Assume that this increment remains nonzero after shared fixed structural scaling. The correct responses for the two inputs are y_- and y_+, respectively. For $X \in \{\mathrm{LLM}, \mathrm{SAIL}\}$, define the prediction margins of the correct responses over the competing candidates as

$$\begin{aligned} M^X_- &= s^X_{y_-}(n) - s^X_{y_+}(n), \\ M^X_+ &= s^X_{y_+}(n+1) - s^X_{y_-}(n+1). \end{aligned} \tag{7}$$

Proposition 3.1 (Margin Improvement for Adjacent Counts)

Under the above setting and the candidate conditions in Appendix B.3, let $\Delta_{\max} > 0$ bound the MLP output magnitude. For any target readout separation $\kappa > 0$ used for correction, there exists a set of QueryInt projection and MLP parameters shared by both inputs and candidates such that

$$M^{\mathrm{SAIL}}_{\pm} = M^{\mathrm{LLM}}_{\pm} + \Delta_{\max} \tanh(\kappa/2) > M^{\mathrm{LLM}}_{\pm}. \tag{8}$$

For sufficiently large $\Delta_{\max}$, this construction further yields $M^{\mathrm{SAIL}}_- > 0$ and $M^{\mathrm{SAIL}}_+ > 0$.

Remark. The unnormalized sum preserves an increment that does not decay with count. For the same target readout separation κ, the minimum linear readout weight norm is independent of count, whereas that of the normalized attention reference (Section 2.3) grows as $\Theta(n^2)$.

Figure 3 illustrates this correction for adjacent counts, and Appendix B.3 provides the proof and comparison.

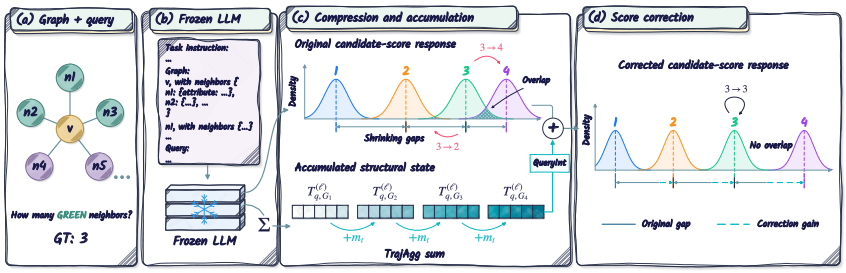


Figure 3: This example illustrates SAIL for counting green neighbors. (a,b) The graph and query are serialized for a frozen LLM. (c) TrajAgg preserves count-dependent differences through an unnormalized sum. (d) QueryInt corrects the candidate scores to recover the true count. Peaks schematically represent responses grouped by ground-truth count.

Table 2: Results on GRAPHMACC and native benchmarks. Scores are percentages. **Bold** and underlined values mark the best and second-best results. Details in Appendix E.1.

	GRAPHMACC		Native Graph Benchmarks					
Method	$G \downarrow$	$\mathrm{Acc}_E \uparrow$	Cora Acc.↑	PubMed Acc.↑	WikiCS Acc.↑	Arxiv Acc.↑	Citation2 MRR↑	MolHIV AUC↑
Qwen3-4B								
Original LLM	42.34	84.63	32.00	34.10	14.98	11.02	0.82	54.50
Graph Linearization	37.31	83.72	30.20	**38.70**	14.09	9.26	0.86	56.67
Graph-KV	42.60	59.19	18.60	**38.70**	13.99	9.97	0.59	45.33
GALA	40.17	82.11	**32.30**	35.10	14.88	10.63	0.67	53.72
SLASH	42.84	84.45	29.70	31.90	7.88	**21.16**	0.99	59.59
SAIL	**33.37**	**84.99**	30.20	35.66	**19.71**	18.58	**2.91**	**60.05**
Qwen3-8B								
Original LLM	25.16	72.78	13.90	39.50	20.35	15.26	0.63	54.86
Graph Linearization	25.45	76.15	13.80	**41.70**	19.51	15.33	0.51	**63.84**
Graph-KV	35.75	62.94	10.60	38.10	7.83	4.31	0.87	62.55
GALA	24.49	71.42	13.80	39.60	20.47	14.89	0.60	54.33
SLASH	27.08	74.45	13.60	38.90	20.23	12.72	0.66	53.35
SAIL	**19.68**	**77.61**	**15.44**	37.68	**22.65**	**18.55**	**4.06**	54.83

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate SAIL on GRAPHMACC (Appendix A) and six native graph benchmarks: Cora (Sen et al., 2008) and PubMed (Namata et al., 2012), WikiCS (Mernyei & Cangea, 2020), and ogbn-arxiv, ogbl-citation2, and ogbg-molhiv (Hu et al., 2020). For GRAPHMACC, we use the task accuracies defined in Section 2.2, with existence accuracy Acc_E, utilization score $U = (\mathrm{Acc}_{\mathrm{Count}} + \mathrm{Acc}_{\mathrm{compare}})/2$, and recognition-utilization gap $G = \mathrm{Acc}_E - U$. For native tasks, we report accuracy, mean reciprocal rank (MRR), and ROC–AUC for node, link, and graph prediction, respectively. Native inputs omit raw text to keep predictions structure-dependent, lowering absolute scores (Appendix D.2). Main comparisons use Qwen3-4B and Qwen3-8B (Yang et al., 2025), with additional evaluation on Qwen2.5 (Hui et al., 2024), Llama (Grattafiori et al., 2024) and Gemma 3 (Team et al., 2025) backbones. We compare the original LLM and SAIL with four presentation-alignment methods in Section 2.1 and evaluate their combinations, including conditions with LoRA fine-tuning (FT) (Hu et al., 2021) or in-context learning (ICL) (Brown et al., 2020). For the main experiments, we train one shared QueryInt module per backbone on GRAPHMACC and the six native benchmarks. Model and training details are in Appendix D.

4.2 DOES SAIL IMPROVE GRAPH PREDICTION AND COMPLEMENT PRESENTATION ALIGNMENT?

Table 3: Complementarity with presentation alignment: average Δ_{Best}, with the number of improved benchmarks out of seven (GRAPHMACC and six native benchmarks).

LLM	Graph Linearization	Graph-KV	GALA	SLASH
Qwen3-4B	+5.75 *(7/7)*	+6.17 *(7/7)*	+2.17 *(5/7)*	+1.08 *(6/7)*
Qwen3-8B	+4.77 *(7/7)*	+6.74 *(7/7)*	+2.42 *(5/7)*	+1.41 *(6/7)*

Table 4: Complementarity under FT and ICL: average Δ_{joint}, with the number of improved benchmarks out of seven (GRAPHMACC and six native benchmarks).

LLM	Adaptation	Graph Linearization	Graph-KV	GALA	SLASH
Qwen3-4B	FT	+2.52 *(5/7)*	+6.63 *(5/7)*	+4.20 *(5/7)*	+3.37 *(6/7)*
	ICL	+1.57 *(5/7)*	+5.26 *(5/7)*	+1.79 *(4/7)*	+2.42 *(4/7)*
Qwen3-8B	FT	+1.06 *(3/7)*	+5.00 *(4/7)*	+2.40 *(4/7)*	+3.89 *(6/7)*
	ICL	+2.49 *(4/7)*	+3.82 *(4/7)*	+2.83 *(5/7)*	+0.09 *(3/7)*

Structural utilization and graph prediction. SAIL achieves the highest existence accuracy and the smallest GRAPHMACC gap on both LLMs (Table 2), improving beyond presence detection. SAIL also ranks first or second in 9 of 12 native benchmark cells, supporting utilization alignment across node, edge, and graph prediction (Appendix E.1).

Complementarity with presentation alignment. Combining SAIL with presentation alignment outperforms the stronger standalone method in 50 of 56 cases across GRAPHMACC and six native benchmarks (Table 3). These gains support presentation and utilization alignment as complementary intervention targets for improving serialization-based graph prediction (Appendix E.2).

Complementarity under FT and ICL. Under FT and ICL, combining SAIL with presentation alignment outperforms the strongest two-component alternative in 72 of 112 cases across seven benchmarks (Table 4). These gains show that SAIL continues to improve serialization-based graph prediction when combined with widely used downstream adaptation methods (Appendix E.3).

4.3 How Does SAIL Improve Structural Utilization?

Count separation and prediction changes. Figure 4 shows projections of candidate-score representations grouped by ground-truth target-category count, before and after SAIL correction. For clarity, we display a randomly sampled 30% subset; complete visualizations appear in Appendix F.1. Before correction, adjacent-count gaps shrink as the target-category count increases, consistent with the theoretical analysis. Adjacent-count confusions account for 98.5% and 94.7% of errors on GRAPHMACC and controlled Cora counting, respectively. SAIL consistently widens these gaps and substantially reduces overall counting errors on both tasks.

Table 5: Component ablations on GRAPHMACC with Qwen3-4B. Overall is the mean accuracy across the three tasks. Results are mean ± SD. The best result in each column is shown in **bold**. Details in Appendix F.3.

Module	Variant	Overall↑	Acc_E ↑	Count↑	Compare↑
	Full SAIL	**70.41** ± *0.34*	92.61 ± *0.20*	**47.10** ± *0.41*	**71.53** ± *0.85*
TrajAgg	Mean aggregation	64.40 ± *0.22*	92.42 ± *0.12*	44.51 ± *0.44*	56.26 ± *0.62*
	No relation encoding	66.05 ± *0.29*	92.70 ± *0.19*	47.02 ± *0.28*	58.44 ± *0.63*
QueryInt	Response only	64.70 ± *0.33*	**92.85** ± *0.26*	43.30 ± *0.43*	57.97 ± *0.59*
	No candidate embedding	57.18 ± *0.88*	91.96 ± *0.26*	33.31 ± *0.38*	46.25 ± *0.21*

Component contributions. For the ablations, we train a separate SAIL model on GRAPHMACC alone. Full SAIL achieves the highest overall accuracy (Table 5). Because all variants share the same training data and budget, the drops in calculation and comparison under mean aggregation and Response only isolate the effect of unnormalized accumulation from that of training itself. Removing candidate embeddings causes the largest drop and removing relation encoding mainly hurts comparison, while existence accuracy remains stable (Appendix F.3).

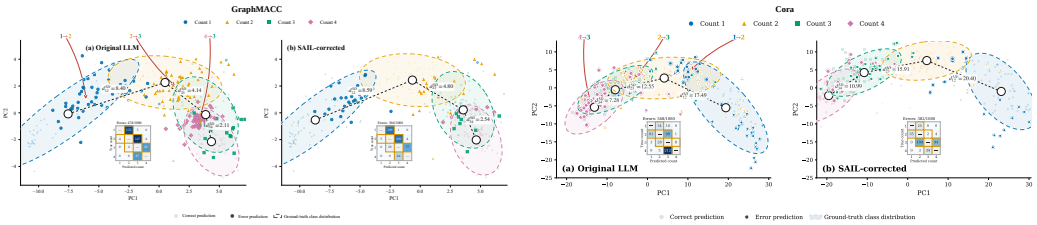


Figure 4: Candidate-score projections compare the original LLM with SAIL on GRAPHMACC (left pair) and Cora counting (right pair). Points show a random 30% subset, while centroid distances and error counts are computed from all 1,080 records per dataset (Appendix F.1).

Table 6: Cross-model applicability. Green checkmarks indicate positive mean gains in both Acc_E and U on GRAPHMACC, and Native reports the number of improved benchmarks out of six.

	Llama				Qwen2.5				Gemma 3		
	3.2-1B	3.2-3B	2-7B	2-13B	0.5B	3B	7B	14B	1B	4B	12B
GRAPHMACC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Native	5/6	5/6	5/6	4/6	5/6	6/6	5/6	3/6	6/6	3/6	2/6

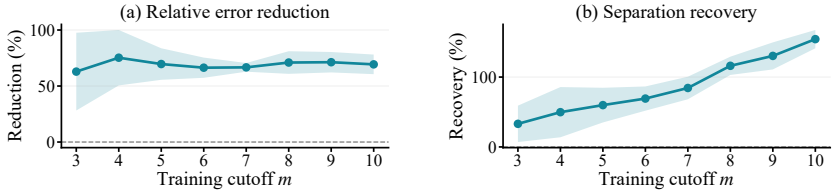


Figure 5: Panels (a) and (b) show SAIL's relative error reduction and adjacent-count separation recovery within the training range ($c \leq m$), both measured relative to the original LLM. Shading shows 95% confidence intervals computed as described in Appendix G.2.

4.4 How Well Does SAIL Generalize?

Generalization across LLMs and training count ranges. SAIL improves recognition and utilization on GRAPHMACC across all 11 additional backbones from the Llama, Qwen, and Gemma families, with native-task gains in 49 of 66 settings (Table 6, Appendix G.1).

Training on counts $c \leq m$ for $m = 3, \ldots, 10$, SAIL consistently reduces in-range prediction errors by 63–75%, while average adjacent-class separation recovery increases with the training range (Figure 5; metric definitions in Appendix G.2). The latter is consistent with our compression analysis: smaller gaps at higher target-category counts offer greater potential for relative separation recovery.

Real-world case and limitation. A representative improvement is MolHIV-2191, a molecule containing two equivalent aryl-imidazoline motifs (Zhou et al., 2023). SAIL's structural-utilization correction changes the original LLM's "inactive" prediction to the "active" label (Appendix G.3). However, improvements within the training c range do not translate into reliable extrapolation: gains are unstable for $c > m$ and become negative at $c = m + 2$ (Appendix G.2). This motivates future work on structural utilization that generalizes to unseen counts.

5 Conclusion

This paper studied the gap between structural presentation and structural utilization in LLM-based graph prediction. Diagnosing through degree profiles, we showed that making structural evidence accessible does not ensure that LLMs can reliably recover and use category-specific multiplicities. We further identified marginal response compression in normalized attention as one possible contributor. We therefore proposed *utilization alignment* as a complementary optimization target and instantiated it with SAIL, which preserves occurrence-level structural information through unnormalized accumulation and integrates it into the LLM prediction. Experiments across LLM families and node-, edge-, and graph-level tasks show improved structural utilization and consistent gains when combined with presentation alignment, fine-tuning, and in-context learning. Motivated by these results, we advocate prioritizing structural utilization in graph–LLM systems.

AI USE STATEMENT

We did not use LLM or other AI tools for the preparation of this manuscript.

CODE AND DATA AVAILABILITY

We will release the code and data at `https://anonymous.4open.science/r/SAIL-0478/` upon acceptance of this paper.

REFERENCES

Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th annual symposium on foundations of computer science*, pp. 670–688. IEEE, 2015.

Freya Behrens, Luca Biggio, and Lenka Zdeborová. Counting in small transformers: The delicate interplay between attention and feed-forward layers. *arXiv preprint arXiv:2407.11542*, 2024.

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

Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S Kevin Zhou. Graphinsight: Unlocking insights in large language models for graph structure understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12096–12134, 2025.

Jiaxuan Chen, Zixing Zhang, Ruijun Mao, Wei Sun, Zhicheng Liang, Yuhang Zhang, Yaxi Liu, and Fangxin Wang. Weaving graph over tokens: Contextualizing structured sequences for llms. In *Forty-third International Conference on Machine Learning*, 2026.

Derek G Corneil and Calvin C Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM (JACM)*, 17(1):51–64, 1970.

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.

Lise Getoor. Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pp. 189–207. Springer, 2005.

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.

Zhong Guan, Likang Wu, Hongke Zhao, Ming He, and Jianpin Fan. Attention mechanisms perspective: Exploring llm processing of graph-structured data. *arXiv preprint arXiv:2505.02130*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Yiming Liu, Bin Lu, Xinbing Wang, Chenghu Zhou, and Meng Jin. Slash the sink: Sharpening structural attention inside llms. *arXiv preprint arXiv:2605.10503*, 2026.

Donald Loveland, Puja Trivedi, Ari Weinstein, Edward W Huang, and Danai Koutra. Formalizing and mitigating structural distortion in llm attention for zero-shot graph reasoning. *arXiv preprint arXiv:2606.15633*, 2026.

Brendan D McKay et al. Practical graph isomorphism. 1981.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, pp. 1, 2012.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*, 2024.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–106, 2008.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Haoyu Wang, Peihao Wang, Mufei Li, Shikun Liu, Siqi Miao, Zhangyang Wang, Pan Li, et al. Graph-kv: Breaking sequence via injecting structural biases into large language models. *Advances in Neural Information Processing Systems*, 38:68358–68387, 2026.

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.

Jingyao Wu, Bin Lu, Zijun Di, Xiaoying Gan, Meng Jin, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. ¡ sog_k¿: One llm token for explicit graph structural understanding. *arXiv preprint arXiv:2602.01771*, 2026.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Christos Xypolopoulos, Guokan Shang, Xiao Fei, Giannis Nikolentzos, Hadi Abdine, Iakovos Evdaimon, Michail Chatzianastasis, Giorgos Stamou, and Michalis Vazirgiannis. Graph linearization methods for reasoning on graphs with large language models. *arXiv preprint arXiv:2410.19494*, 2024.

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.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Hongyu Zhan, Qixin Wang, Yusen Tan, Haitao Yu, Jingbo Zhou, Shuai Chen, Jia Li, Xiao Tan, and Jun Xia. Lorec: Rethinking large language models for graph data analysis. *arXiv preprint arXiv:2604.17897*, 2026.

Shuo Zhang and Lei Xie. Improving attention mechanism in graph neural networks via cardinality preservation. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 1395–1402. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/194. URL `https://doi.org/10.24963/ijcai.2020/194`.

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

Zhicheng Zhou, Isabelle Staropoli, Anne Brelot, Peggy Suzanne, Aurélien Lesnard, Fanny Fontaine, Serge Perato, Sylvain Rault, Olivier Helynck, Fernando Arenzana-Seisdedos, et al. Discovery of bis-imidazoline derivatives as new cxcr4 ligands. *Molecules*, 28(3):1156, 2023.

Kerui Zhu, Bo-Wei Huang, Bowen Jin, Yizhu Jiao, Ming Zhong, Kevin Chang, Shou-De Lin, and Jiawei Han. Investigating instruction tuning large language models on graphs. *arXiv preprint arXiv:2408.05457*, 2024.

APPENDIX CONTENTS

A GRAPHMACC: Construction and Evaluation **15**
- A.1 Diagnostic Scope and Matched Tasks . . . 15
- A.2 Controlled Synthetic Graph Construction . . . 16
- A.3 Supplementary Validation on Real Graph Topologies . . . 16
- A.4 Baseline Selection and Evaluation Settings . . . 17
- A.5 Metrics and Statistical Protocol . . . 17
- A.6 Diagnostic Results . . . 19

B Theoretical Analysis and Proofs **23**
- B.1 Marginal Response Compression in Softmax Attention . . . 23
- B.2 Attention Distortion under Graph Serialization . . . 25
- B.3 Proof of Proposition 3.1 . . . 29

C SAIL Method Details **31**
- C.1 Query-Conditioned Structural Scope . . . 31
- C.2 Occurrence Trajectory Extraction . . . 31
- C.3 Query-Relative Relation Encodings . . . 31
- C.4 QueryInt: Projection and Residual Integration . . . 32
- C.5 Training Objective . . . 33

D Experimental Setup **35**
- D.1 Datasets and Prediction Tasks . . . 35
- D.2 Data Splits and Graph Inputs . . . 35
- D.3 LLMs and Baseline Methods . . . 36
- D.4 SAIL Training and Statistical Reporting . . . 37
- D.5 Implementation and Computational Resources . . . 37

E Predictive Performance and Complementarity **40**
- E.1 Per-Benchmark Results . . . 40
- E.2 Combinations with Presentation Alignment . . . 42
- E.3 Combinations under Fine-Tuning and In-Context Learning . . . 45

F Multiplicity Separation and Ablation Details **52**
- F.1 Candidate-Score Visualization and Count Separation . . . 52
- F.2 Prediction Changes between Multiplicity Classes . . . 54
- F.3 Ablation Variants and Controlled Comparisons . . . 54

G Generalization and Limitations **56**
- G.1 Cross-Model Applicability . . . 56
- G.2 Multiplicity Range and Extrapolation . . . 57
- G.3 Molecular Case Study . . . 57

A GRAPHMACC: CONSTRUCTION AND EVALUATION

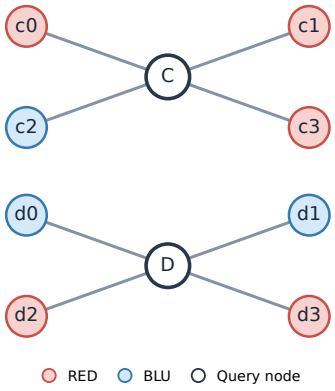
A.1 DIAGNOSTIC SCOPE AND MATCHED TASKS

GRAPHMACC is a controlled synthetic graph benchmark for evaluating LLMs' ability to recover and use the category-count information encoded by degree profiles. We consider three tasks: *Existence* asks whether a queried node has any neighbors in a specified category; *Calculation* asks how many such neighbors it has; *Comparison* asks which of two queried nodes has more neighbors in that category, with the two target-category counts constrained to differ. Existence tests category-presence recognition, while calculation and comparison require using the corresponding neighbor counts.

For each graph sample in GRAPHMACC, the three tasks share the same graph, node attributes, target category, and node order, while only the task instruction changes. In the controlled construction, we vary the target-category count while keeping the queried nodes' total degree fixed within each degree setting. Ground-truth answers are computed from each graph sample before serialization. The prompts do not include ground-truth answers or precomputed count statistics. All prompts request a single numeric answer: Existence uses 1 for presence and 0 for absence, Calculation returns an integer in $\{0, \ldots, d\}$ for queried degree d, Comparison uses 0 and 1 to select the first and second queried nodes. Our unified GRAPHMACC evaluation protocol selects the highest-scoring valid candidate and compares it with the ground-truth answer. Responses without a parsable number or with an invalid task answer are counted as incorrect rather than excluded.

For the GRAPHMACC evaluations of SAIL, prediction selects the highest-scoring valid candidate (Appendix C.4). SAIL and its corresponding uncorrected baseline use the same candidate set and scoring interface. Figure 6 shows a graph sample and its three matched prompts using an adjacency-list serialization.

Graph sample



Shared graph input

```
Undirected graph.
Node attributes: c0=RED, c1=RED,
    d0=BLU, c2=BLU, c3=RED, d1=BLU,
    d3=RED, d2=RED.
D: {d1, d0, d3, d2}
c0: {C}
d0: {D}
d3: {D}
c2: {C}
c1: {C}
C: {c2, c3, c1, c0}
d2: {D}
c3: {C}
d1: {D}
```

Existence: task instruction

```
Question: Does center C have at least one RED neighbor? Answer 1 for
    yes or 0 for no.
```

Calculation: task instruction

```
Question: How many RED neighbors does center C have? Answer with one
    number.
```

Comparison: task instruction

```
Question: Which center has more RED neighbors? Answer 0 for C or 1
    for D.
```

Figure 6: The three GraphMACC tasks use the same graph input with different task instructions. Query nodes C and D both have degree 4, with three and two RED neighbors, respectively. The correct answers are 1 for existence, 3 for calculation, and 0 (selecting C) for comparison.

A.2 Controlled Synthetic Graph Construction

Each synthetic graph contains two queried nodes, C and D, with disjoint neighborhoods of the same fixed degree $d \in \{4, 8, 16\}$. We vary topology among their neighbors using three settings: no additional edges (star), independent edges with probability 0.25 (Erdős–Rényi), or two equal-sized blocks with within- and between-block edge probabilities of 0.55 and 0.05 (stochastic block model). Specifically, these additional edges leave the queried nodes' degrees unchanged. For node C, the target-category count k follows

$$\mathcal{K}_4 = \{0, 1, 2, 3, 4\},$$
$$\mathcal{K}_8 = \{0, 1, 2, 4, 7, 8\},$$
$$\mathcal{K}_{16} = \{0, 1, 2, 8, 15, 16\}.$$

These grids cover absence, low counts, intermediate counts, and high counts up to the full neighborhood size.

For each topology, degree, and count combination, we generate 30 base graphs with randomly assigned neighbor attributes, yielding $3 \times (5 + 6 + 6) \times 30 = 1{,}530$ graph samples in total. For Comparison, after fixing node C's target-category count k, we assign node D a count of $k + 1$ or $k - 1$, alternating across replicates for $0 < k < d$ and using 1 or $d - 1$ at the respective boundaries. Each base graph produces three matched tasks, two textual serializations (adjacency lists and explicit source–neighbor statements), and three seeded node-order permutations, yielding 4,590 task instances and 27,540 serialized prompts. Unless otherwise specified, for experiments related to GraphMACC, we randomly select five base graphs per topology–degree–count combination for training and another five from the remaining graphs for testing across the three matched tasks, yielding disjoint sets in a 1:1 ratio. For the main-table GraphMACC comparison, we construct three mutually disjoint test partitions using the sampling procedure above, each containing 255 graphs and 4,590 serialized prompts. Model and baseline hyperparameters are selected exclusively on the training set and fixed before test evaluation. All tasks, serializations, and node-order permutations derived from the same base graph remain in the same partition. The training set and selected configurations remain fixed across these partitions.

A.3 Supplementary Validation on Real Graph Topologies

To test whether the recognition–utilization gap extends beyond synthetic graphs, we apply the matched tasks and metrics of GraphMACC to neighborhoods sampled from real graphs. It covers Cora (Sen et al., 2008), PubMed (Namata et al., 2012), WikiCS (Mernyei & Cangea, 2020), ogbn-arxiv (Hu et al., 2020), Texas, and Wisconsin (Sen et al., 2008). We use undirected graph topology and sample d neighbors for each of two queried nodes whose original degrees are at least d. The sampled neighborhoods are disjoint, and pairs are selected so that retaining all source edges among the sampled nodes does not increase either queried node's degree beyond d. We retain the source-node mapping and verify the reconstructed neighborhoods, edges, and target-category counts before serialization.

We consider two attribute settings: The *controlled* setting assigns binary target attributes using $\mathcal{K}_d = \{0, 1, 2, \lfloor d/2 \rfloor, d-1, d\}$ with duplicates removed, and constructs unequal comparison counts following the adjacent-count and boundary rules in Appendix A.2. The *native* setting derives target-category membership from original node labels, and selects categories and node pairs with unequal comparison counts without imposing adjacency or a balanced multiplicity grid.

Practically, Cora, PubMed, and WikiCS use sampled degrees 4, 8, and 16; ogbn-arxiv additionally includes 32; Texas, and Wisconsin use 2 and 4. Each sampled graph supports the three matched

tasks, two textual serializations, and three seeded node-order permutations, following the answer and scoring rules in A.1. Across the six datasets and Qwen3-4B/8B, all 69,840 responses for these three tasks yielded parsable numeric answers.

A.4 Baseline Selection and Evaluation Settings

As described in Section 2.2, GraphMACC evaluates structural recognition and utilization under different presentation-alignment methods. In this work, we use the original LLM without additional presentation interventions as the baseline and evaluate four representative presentation-alignment methods: Graph Linearization, Graph-KV, GaLA, and SLASH. These correspond to the four intervention types discussed in Section 2.1; their descriptions and implementation details are provided in Appendix D.3.

For each backbone, all baselines use the same GraphMACC dataset, query instances, and decoding and scoring rules described in A.1, while allowing method-specific adjustments to structural presentation. Baseline configurations and hyperparameters are selected on the training set and fixed before test evaluation. During inference, ground-truth answers, precomputed target-category counts, and other label-revealing metadata are excluded from baseline inputs.

A.5 Metrics and Statistical Protocol

Task and aggregate metrics. For each degree d, we report the three task metrics introduced in Section 2.2: Acc_E, $\mathrm{Acc}_{\mathrm{Count}}$, and $\mathrm{Acc}_{\mathrm{compare}}$. Existence and Comparison use balanced accuracy, assigning equal weight to the two ground-truth answer classes. Calculation uses exact-match accuracy averaged equally over the evaluated ground-truth counts:

$$\mathrm{Acc}_{\mathrm{Count}}(d) = \frac{1}{|\mathcal{K}_d|} \sum_{k \in \mathcal{K}_d} \frac{1}{N_{d,k}} \sum_{i \in \mathcal{I}_{d,k}} \mathbf{1}[\hat{k}_i = k],$$

where $\mathcal{I}_{d,k}$ contains counting instances with degree d and true count k, $N_{d,k} = |\mathcal{I}_{d,k}|$, and $\hat{k}_i$ is the predicted count. Invalid answers are counted as incorrect, as specified in A.1.

For the experimental comparisons in Section 4.2 and Appendix E.1, we additionally use aggregate metrics, combining Calculation and Comparison into a structural-utilization score:

$$U(d) = \frac{\mathrm{Acc}_{\mathrm{Count}}(d) + \mathrm{Acc}_{\mathrm{compare}}(d)}{2}, \qquad G(d) = \mathrm{Acc}_E(d) - U(d).$$

Here, Acc_E is existence accuracy, U summarizes count-sensitive utilization, and G quantifies their observed score difference. This raw gap does not adjust for differences in task-specific chance levels, which we examine below. Detailed diagnostic results are reported in Appendix A.6.

To assess the robustness of the diagnostic conclusion, we examine counting-error magnitude and degree-specific gaps in original LLMs and Graph Linearization across both model sizes and all three degrees. For the main-table GraphMACC comparison, all methods are evaluated on the same three test partitions. We report the mean and sample standard deviation across these partitions in Table 19.

Adjacent-count error analysis. To distinguish adjacent-count errors from larger counting errors, we supplement exact-match accuracy with macro-averaged absolute error (Macro-MAE) and ± 1 accuracy. Within each degree setting, we first compute the mean absolute error over parsable numeric predictions in each ground-truth count category, then average equally across categories. Numeric predictions are not clipped to the legal range, and numeric-response coverage is reported alongside these metrics; if a category contains no parsable predictions, the corresponding metric is marked unavailable rather than computed by excluding that category. The ± 1 accuracy uses the same category weights as exact-match accuracy, accepting only legal integer answers whose absolute difference from the ground-truth count is at most one. We evaluate original LLMs and Graph Linearization using Qwen3-4B and Qwen3-8B across $d \in \{4, 8, 16\}$, reporting exact-match accuracy, Macro-MAE, ± 1 accuracy, adjacent-count error share, and numeric-response coverage in Table 7. For each degree, we compute each metric separately within the three test partitions and report their mean.

Table 7: Counting accuracy and error magnitude for original LLMs and Graph Linearization across degrees. Values are percentages except Macro-MAE, which is in count units.

Method	d	Exact	±1	Macro-MAE	Adjacent	Coverage
Qwen3-4B						
Original LLM	4	45.70	92.74	0.617	86.62	100.00
	8	36.30	78.70	0.936	66.54	100.00
	16	28.09	67.35	1.533	54.59	100.00
Graph Linearization	4	50.07	95.78	0.541	91.58	100.00
	8	39.57	79.69	0.954	66.40	100.00
	16	38.95	74.20	1.207	57.69	100.00
Qwen3-8B						
Original LLM	4	52.89	98.07	0.490	95.90	100.00
	8	43.09	91.67	0.685	85.36	100.00
	16	37.47	71.91	1.370	55.08	100.00
Graph Linearization	4	54.74	98.52	0.467	96.73	100.00
	8	45.00	93.46	0.636	88.16	100.00
	16	41.11	74.69	1.093	57.01	100.00

In most evaluated settings, adjacent-count errors account for around 55–97% of all exact-match errors, and improvements in exact counting accuracy are accompanied by reductions in error magnitude. These results indicate that LLMs recover some quantitative information with and without presentation alignment but still struggle to distinguish adjacent multiplicities precisely, consistent with the difficulty described by our theoretical analysis.

Robustness across degrees. To prevent sample sizes at different degrees from dominating the overall results and assess sensitivity to test-graph sampling, we compute metrics separately for each degree and then average equally across degrees. Accuracies and aggregate scores are reported on a 0–100 scale, raw accuracy differences in percentage points, and Macro-MAE in count units. When applying chance adjustment, we adjust scores within each degree before averaging across degrees, multiply the adjusted scores by 100, and label them accordingly. For each degree in GRAPHMACC, we sample with replacement within each of the three test partitions, drawing the same number of graphs as in that partition. Each sampled graph retains all its associated tasks, serializations, and node-order permutations. We compute the metrics within each resampled partition and average them equally across the three partitions, repeating this process 2,000 times to obtain 95% confidence intervals. When comparing methods, we use the same sampled graphs in each repetition so that sampling differences do not affect the comparison. Under the same model, method, and degree settings, we report the chance-adjusted recognition–utilization gap G_{adj} and the gap G_{binary} between existence recognition and count comparison, two binary tasks, together with their respective 95% confidence intervals in Table 8. With original accuracies expressed on the 0–1 scale, we define

$$
\begin{aligned}
b_d &= \frac{1}{d+1}, \\
G_{\mathrm{adj}}(d) &= (2\mathrm{Acc}_E(d) - 1) - \frac{1}{2}\left[\frac{\mathrm{Acc}_{\mathrm{Count}}(d) - b_d}{1 - b_d} + 2\mathrm{Acc}_{\mathrm{compare}}(d) - 1\right], \\
G_{\mathrm{binary}}(d) &= \mathrm{Acc}_E(d) - \mathrm{Acc}_{\mathrm{compare}}(d).
\end{aligned}
$$

Here, counting uses uniform guessing over $\{0, \ldots, d\}$ as its chance reference, while both binary tasks use $1/2$; negative adjusted scores are retained.

These results show that both gaps remain significant in all evaluated degree settings and are robust to evaluation-graph sampling, rather than being driven by a particular degree setting or its sample size.

Table 8: Recognition–utilization gaps across degrees with 95% bootstrap confidence intervals. Chance-adjusted gaps are multiplied by 100, and binary-task gaps are in percentage points.

Method	d	G_{adj} [95% CI]	G_{binary} [95% CI]
Qwen3-4B			
Original LLM	4	60.56 *[53.29, 67.38]*	38.11 *[33.36, 42.82]*
	8	64.69 *[57.24, 71.52]*	41.53 *[36.93, 46.03]*
	16	46.82 *[36.71, 55.64]*	30.58 *[24.10, 36.29]*
Graph Linearization	4	59.30 *[51.69, 66.60]*	39.53 *[34.14, 44.68]*
	8	56.18 *[47.40, 64.23]*	36.96 *[31.35, 42.11]*
	16	34.60 *[25.39, 43.29]*	24.80 *[18.58, 30.82]*
Qwen3-8B			
Original LLM	4	18.09 *[10.65, 26.06]*	18.23 *[14.22, 22.23]*
	8	36.84 *[28.78, 44.44]*	26.93 *[22.56, 31.06]*
	16	24.14 *[16.95, 30.93]*	20.88 *[17.14, 24.46]*
Graph Linearization	4	27.68 *[19.25, 36.28]*	23.19 *[18.82, 27.61]*
	8	38.10 *[29.77, 46.25]*	27.86 *[23.38, 32.06]*
	16	18.31 *[7.91, 28.45]*	14.06 *[8.05, 20.06]*

A.6 Diagnostic Results

We report complete and cross-model results on GraphMACC, followed by supplementary validation on real graph topologies and robustness checks across serializations and node orders.

A.6.1 Complete Diagnostic Results

Table 9 reports diagnostic results for Qwen3-4B and Qwen3-8B across three test partitions and degrees 4, 8, and 16, comparing the original LLM with four presentation-alignment methods. We report the three task accuracies together with the utilization score U and recognition–utilization gap G. The diagnostic results show a positive gap in all settings, and the evaluated presentation-alignment methods do not consistently eliminate it, providing all details for the distinction between recognizing category presence and using category-count information for prediction.

A.6.2 Results across Model Families and Sizes

We extend the diagnostic to 13 models from the Llama, Qwen, and Gemma families, covering parameter scales from 0.5B to 14B. We evaluate all models on the three test partitions defined in Appendix A.2, computing task scores within each degree according to Appendix A.5 and averaging equally over degrees 4, 8, and 16 and across the three partitions. Table 10 reports the same metrics as Table 9 for all 13 models, where the Qwen3-4B and Qwen3-8B rows average the results in Table 9 over degrees and partitions. Our results show positive raw recognition–utilization gaps across all 13 models, ranging from 11.17 to 44.70 percentage points. Additionally, increasing model size does not consistently remove the gap, motivating a distinction between model scale and effective use of structural information.

A.6.3 Results on Real Graph Topologies

Table 11 reports supplementary diagnostic results on real graph topologies, broken down by dataset, model, and attribute setting. Under both controlled and native attributes, the recognition–utilization gap remains positive in most evaluable settings, although its magnitude varies across datasets and attributes.

A.6.4 Serialization and Node-Order Robustness

We evaluate two different prompt writing methods: adjacency lists and explicit source–neighbor statements under three seeded node-order permutations. Table 12 pools predictions from GraphMACC and the supplementary real-topology evaluation for each input configuration, restricted to

Table 9: GRAPHMACC task scores and recognition–utilization gaps across degrees (mean ± SD over three test partitions). Scores are percentages and gaps are in percentage points.

Method	d	Acc_E	$\mathrm{Acc}_{\mathrm{Count}}$	$\mathrm{Acc}_{\mathrm{compare}}$	U	G
Qwen3-4B						
Original LLM	4	88.52 ±1.31	45.70 ±0.64	50.41 ±0.68	48.06 ±0.54	40.46 ±0.85
	8	87.33 ±0.51	36.30 ±1.21	45.81 ±1.33	41.05 ±0.33	46.28 ±0.29
	16	78.04 ±1.01	28.09 ±1.23	47.46 ±1.80	37.77 ±1.29	40.27 ±1.85
Graph Linearization	4	88.56 ±3.31	50.07 ±1.64	49.04 ±1.05	49.56 ±0.46	39.01 ±2.90
	8	85.22 ±0.51	39.57 ±1.72	48.26 ±3.19	43.91 ±0.79	41.31 ±0.55
	16	77.37 ±7.48	38.95 ±2.17	52.57 ±5.49	45.76 ±1.81	31.61 ±9.29
Graph-KV	4	67.59 ±0.32	29.04 ±2.65	16.46 ±2.07	22.75 ±2.33	44.84 ±2.51
	8	52.96 ±0.85	7.35 ±1.39	21.36 ±1.87	14.35 ±1.12	38.61 ±1.56
	16	57.00 ±0.84	2.53 ±0.39	22.78 ±1.38	12.66 ±0.67	44.34 ±1.25
GaLA	4	86.44 ±1.61	44.96 ±0.71	50.10 ±1.40	47.53 ±0.80	38.90 ±0.87
	8	85.22 ±1.06	34.88 ±0.93	46.55 ±0.09	40.71 ±0.45	44.51 ±0.80
	16	74.67 ±0.95	26.91 ±1.44	48.24 ±1.63	37.58 ±1.20	37.09 ±1.40
SLASH	4	88.38 ±1.45	45.11 ±0.80	50.23 ±0.70	47.67 ±0.52	40.71 ±0.95
	8	87.15 ±0.46	34.26 ±1.77	46.10 ±0.63	40.18 ±0.58	46.97 ±0.64
	16	77.81 ±2.00	26.30 ±1.30	47.67 ±2.00	36.99 ±1.18	40.83 ±2.40
Qwen3-8B						
Original LLM	4	70.42 ±3.31	52.89 ±1.92	52.19 ±0.24	52.54 ±0.99	17.88 ±3.95
	8	77.89 ±2.04	43.09 ±1.05	50.96 ±1.38	47.02 ±0.48	30.87 ±2.18
	16	70.04 ±1.74	37.47 ±0.47	49.15 ±0.12	43.31 ±0.26	26.73 ±1.79
Graph Linearization	4	76.20 ±3.57	54.74 ±1.34	53.01 ±1.62	53.88 ±1.41	22.33 ±3.52
	8	79.30 ±2.90	45.00 ±2.51	51.43 ±1.85	48.22 ±1.29	31.08 ±1.70
	16	72.96 ±6.98	41.11 ±2.59	58.90 ±3.95	50.01 ±2.15	22.96 ±5.30
Graph-KV	4	75.00 ±0.00	23.26 ±0.56	49.22 ±0.17	36.24 ±0.36	38.76 ±0.36
	8	55.56 ±0.56	10.25 ±0.75	49.89 ±0.19	30.07 ±0.31	25.49 ±0.85
	16	58.26 ±0.06	4.26 ±0.19	26.25 ±0.34	15.26 ±0.25	43.00 ±0.18
GaLA	4	68.94 ±2.05	51.63 ±1.36	51.95 ±0.64	51.79 ±0.36	17.14 ±2.36
	8	76.04 ±0.83	40.99 ±1.13	50.57 ±0.94	45.78 ±0.33	30.26 ±0.80
	16	69.30 ±1.67	36.67 ±0.98	49.77 ±0.50	43.22 ±0.74	26.08 ±1.29
SLASH	4	74.91 ±2.49	52.81 ±1.22	51.58 ±0.40	52.20 ±0.41	22.71 ±2.69
	8	76.59 ±1.80	43.02 ±0.88	50.85 ±0.90	46.94 ±0.50	29.65 ±2.17
	16	71.85 ±3.50	36.67 ±0.37	49.29 ±0.40	42.98 ±0.37	28.87 ±3.25

$d \in \{4, 8, 16\}$. These pooled results assess robustness across both graph sources and are reported separately from GRAPHMACC-only scores. According to our results, the gap remains positive in all 12 model–serialization–order settings, ranging from around 15 to 45 percentage points. Thus, the observed separation between structural recognition and utilization persists across changes in textual expression and node order.

Table 10: GRAPHMACC results across 13 LLMs from three model families. Scores are percentages and gaps are in percentage points.

Model	Acc_E	$\mathrm{Acc}_{\mathrm{Count}}$	$\mathrm{Acc}_{\mathrm{compare}}$	U	G
Llama					
Llama-3.2-1B	50.00	22.32	50.00	36.16	13.84
Llama-3.2-3B	50.00	25.52	50.00	37.76	12.24
Llama-2-7B	57.06	22.85	48.87	35.86	21.20
Llama-2-13B	69.21	22.28	49.66	35.97	33.23
Qwen					
Qwen2.5-0.5B	50.00	22.84	50.00	36.42	13.58
Qwen2.5-3B	81.92	37.36	58.19	47.78	34.14
Qwen3-4B	84.63	36.70	47.89	42.29	42.34
Qwen2.5-7B	86.42	25.86	57.57	41.72	44.70
Qwen3-8B	72.78	44.48	50.77	47.62	25.16
Qwen2.5-14B	87.50	53.59	68.66	61.13	26.38
Gemma					
Gemma-3-1B	50.06	17.86	51.18	34.52	15.54
Gemma-3-4B	53.92	23.97	49.85	36.91	17.01
Gemma-3-12B	56.76	30.43	60.76	45.60	11.17

Table 11: Recognition and utilization on real graph topologies with controlled or native attributes. Scores are percentages and gaps are in percentage points.

Dataset	Model	Acc_E	$\mathrm{Acc}_{\mathrm{Count}}$	$\mathrm{Acc}_{\mathrm{compare}}$	U	G
Controlled attributes						
Cora	Qwen3-4B	87.77	39.44	46.68	43.06	44.72
	Qwen3-8B	59.64	39.42	50.37	44.89	14.75
PubMed	Qwen3-4B	87.01	38.16	46.55	42.35	44.66
	Qwen3-8B	59.67	40.06	49.50	44.78	14.89
WikiCS	Qwen3-4B	82.13	35.19	42.52	38.86	43.28
	Qwen3-8B	64.25	41.31	46.31	43.81	20.44
ogbn-arxiv	Qwen3-4B	81.15	29.59	45.31	37.45	43.69
	Qwen3-8B	63.15	32.17	49.74	40.96	22.19
Texas	Qwen3-4B	88.65	58.44	45.56	52.00	36.65
	Qwen3-8B	61.67	59.83	52.67	56.25	5.42
Wisconsin	Qwen3-4B	85.62	53.44	49.94	51.69	33.93
	Qwen3-8B	57.81	64.78	50.94	57.86	-0.05
Native attributes						
Cora	Qwen3-4B	69.87	33.74	62.51	48.13	21.74
	Qwen3-8B	57.98	35.34	52.28	43.81	14.17
PubMed	Qwen3-4B	74.65	25.37	56.14	40.76	33.89
	Qwen3-8B	53.06	33.83	50.45	42.14	10.92
WikiCS	Qwen3-4B	63.80	26.94	52.81	39.88	23.93
	Qwen3-8B	58.88	36.23	53.46	44.85	14.04
ogbn-arxiv	Qwen3-4B	60.92	28.45	50.48	39.47	21.45
	Qwen3-8B	51.32	33.51	48.98	41.24	10.08
Texas	Qwen3-4B	41.15	39.34	47.60	43.47	-2.32
	Qwen3-8B	52.60	47.07	52.71	49.89	2.71
Wisconsin	Qwen3-4B	78.24	49.31	48.02	48.66	29.58
	Qwen3-8B	59.38	68.17	57.49	62.83	-3.46

Table 12: Original LLM results across serializations and node orders on synthetic and real graph topologies. Scores are percentages and gaps are in percentage points.

Serialization	Node order	Acc_E	$\mathrm{Acc}_{\mathrm{Count}}$	$\mathrm{Acc}_{\mathrm{compare}}$	U	G
Qwen3-4B						
Adjacency list	1	82.67	31.58	47.53	39.56	43.11
	2	79.94	35.72	46.38	41.05	38.89
	3	80.48	35.97	47.39	41.68	38.79
Source–neighbor	1	81.96	36.53	46.78	41.66	40.30
	2	81.33	29.52	45.76	37.64	43.69
	3	82.22	34.73	47.61	41.17	41.05
Qwen3-8B						
Adjacency list	1	63.27	36.62	50.02	43.32	19.96
	2	63.28	36.89	49.80	43.34	19.94
	3	64.27	39.32	50.01	44.66	19.60
Source–neighbor	1	58.38	36.80	50.02	43.41	14.97
	2	59.77	39.16	49.11	44.14	15.63
	3	59.63	35.77	49.26	42.52	17.11

B Theoretical Analysis and Proofs

B.1 Marginal Response Compression in Softmax Attention

We analyze aggregation at a fixed attention head, layer, and query. The attention scores and value vectors are held fixed as multiplicities change, so the analysis isolates the effect of normalization. Residual connections and feed-forward layers can also carry or transform count information Behrens et al. (2024). Our results only describe count sensitivity along the specified aggregation path and identify a possible contributor to the observed gap, rather than an expressivity limit for the full LLM.

B.1.1 Uniform Replication and Fixed Background

For a queried node with a nonempty neighborhood, we represent its neighbor categories as a multiset $\mathcal{M} = \{x_1^{(m_1)}, \ldots, x_r^{(m_r)}\}$, where m_j is the number of neighbors in category x_j. The multiplicities $(m_1, \ldots, m_r)$ are the corresponding degree-profile entries. Distinct neighbors in the same category are separate occurrences of that category. We consider occurrences visible to the fixed query q and assume that those in category x_j share an unnormalized attention weight $a_j = \exp(s(q, x_j)) > 0$ and a value vector $\mathbf{v}_j \in \mathbb{R}^p$. These quantities remain fixed when occurrences are replicated, extending the local exchangeability condition in Assumption 2.2 to all replicated categories. The aggregate over these neighbor occurrences alone is

$$A(q, \mathcal{M}) = \frac{\sum_{j=1}^r m_j a_j \mathbf{v}_j}{\sum_{j=1}^r m_j a_j}. \tag{9}$$

Lemma B.1 (Replication Cancellation)**.** *Under these conditions, let $c\mathcal{M} = \{x_1^{(cm_1)}, \ldots, x_r^{(cm_r)}\}$ for an integer $c \geq 2$. Then*

$$A(q, c\mathcal{M}) = A(q, \mathcal{M}). \tag{10}$$

Proof. Replication multiplies both the numerator and denominator by c, giving

$$A(q, c\mathcal{M}) = \frac{c\sum_{j=1}^r m_j a_j \mathbf{v}_j}{c\sum_{j=1}^r m_j a_j} = A(q, \mathcal{M}). \qquad \square$$

Thus, this structure-only aggregate cannot distinguish a multiset from its uniformly replicated version. A target that changes under such replication cannot be recovered from this aggregate alone.

A serialized graph prompt also contains instructions, query tokens, and delimiters. Let $Z_0 > 0$ be their total unnormalized attention weight and $\mathbf{v}_0 \in \mathbb{R}^p$ their attention-weighted mean value. With this background held fixed, the response becomes

$$A_0(q, \mathcal{M}) = \frac{Z_0 \mathbf{v}_0 + \sum_{j=1}^r m_j a_j \mathbf{v}_j}{Z_0 + \sum_{j=1}^r m_j a_j}. \tag{11}$$

Writing $Z_{\mathcal{M}} = \sum_{j=1}^r m_j a_j$, replication gives

$$A_0(q, c\mathcal{M}) = \frac{Z_0}{Z_0 + cZ_{\mathcal{M}}} \mathbf{v}_0 + \frac{cZ_{\mathcal{M}}}{Z_0 + cZ_{\mathcal{M}}} A(q, \mathcal{M}). \tag{12}$$

Replication now changes the mixing weights between the background and the structural aggregate, so the complete response need not remain unchanged. Theorem 2.3 examines this count dependence when one target category varies and all remaining inputs stay fixed.

B.1.2 Proof of Theorem 2.3

Fix a_t, $\mathbf{v}_t$, Z_{-t}, and $\mathbf{v}_{-t}$ as specified in the theorem. Here, increasing c adds a target-category occurrence without removing any other occurrence.

Proof. Subtracting the responses at $c+1$ and c gives

$$\begin{aligned} \boldsymbol{\Delta}_{t,c}^{\star} &= \frac{Z_{-t}\mathbf{v}_{-t} + (c+1)a_t\mathbf{v}_t}{Z_{-t} + (c+1)a_t} - \frac{Z_{-t}\mathbf{v}_{-t} + ca_t\mathbf{v}_t}{Z_{-t} + ca_t} \\ &= \frac{Z_{-t}a_t}{(Z_{-t} + ca_t)(Z_{-t} + (c+1)a_t)}(\mathbf{v}_t - \mathbf{v}_{-t}). \end{aligned} \tag{13}$$

Let $\alpha_{t,c}$ denote the positive scalar coefficient in this expression. Since $Z_{-t} > 0$ and $a_t > 0$,

$$\frac{\alpha_{t,c+1}}{\alpha_{t,c}} = \frac{Z_{-t} + ca_t}{Z_{-t} + (c+2)a_t} < 1.$$

For $\mathbf{v}_t \neq \mathbf{v}_{-t}$, it follows that

$$\|\mathbf{\Delta}^\star_{t,c+1}\|_2 < \|\mathbf{\Delta}^\star_{t,c}\|_2, \qquad c \geq 0.$$

The denominator in equation 13 grows quadratically with c, while its numerator remains fixed. Hence $\|\mathbf{\Delta}^\star_{t,c}\|_2 \to 0$ as $c \to \infty$. □

Limiting response. The same response can be written as

$$\mathbf{h}^\star_t(c) - \mathbf{v}_t = \frac{Z_{-t}}{Z_{-t} + ca_t}(\mathbf{v}_{-t} - \mathbf{v}_t). \tag{14}$$

Thus $\mathbf{h}^\star_t(c) \to \mathbf{v}_t$, and

$$\|\mathbf{\Delta}^\star_{t,c}\|_2 = \frac{Z_{-t}a_t\|\mathbf{v}_t - \mathbf{v}_{-t}\|_2}{(Z_{-t} + ca_t)(Z_{-t} + (c+1)a_t)} = O(c^{-2}). \tag{15}$$

Under the theorem's nondegeneracy condition, distinct finite counts still produce distinct responses. The result concerns decreasing separation, not an exact loss of all count information.

B.1.3 Attention Response under Fixed-Degree Replacement

GraphMACC varies the target-category count while keeping the queried node's degree fixed. Increasing the target count therefore replaces a neighbor from another category, whereas Theorem 2.3 adds a neighbor and leaves the rest unchanged. We analyze this replacement operation separately.

Lemma B.2 (Fixed-Degree Composition Response)**.** *Let $d \geq 1$ be a fixed integer and*

$$\mathcal{M}_k = \{R^{(k)}, B^{(d-k)}\}, \qquad k \in \{0, \ldots, d\},$$

where R is the target category and B is the category being replaced. Fix the query, background attention mass $Z_0 > 0$, and background weighted-value sum $N_0 \in \mathbb{R}^p$. Suppose all occurrences of each category share fixed weight–value pairs $(a_R, \mathbf{v}_R)$ and $(a_B, \mathbf{v}_B)$, with $a_R, a_B > 0$. The response is

$$A(k) = \frac{N_0 + ka_R\mathbf{v}_R + (d-k)a_B\mathbf{v}_B}{Z_0 + ka_R + (d-k)a_B}. \tag{16}$$

Define

$$C = N_0 + da_B\mathbf{v}_B, \qquad \mathbf{v}_\Delta = a_R\mathbf{v}_R - a_B\mathbf{v}_B,$$
$$E = Z_0 + da_B, \qquad \delta = a_R - a_B.$$

For $k \in \{0, \ldots, d-1\}$,

$$A(k+1) - A(k) = \frac{E\mathbf{v}_\Delta - \delta C}{(E + k\delta)(E + (k+1)\delta)}. \tag{17}$$

If $E\mathbf{v}_\Delta - \delta C \neq \mathbf{0}$, adjacent response distances strictly decrease with k when $a_R > a_B$, remain constant when $a_R = a_B$, and strictly increase when $a_R < a_B$. If $E\mathbf{v}_\Delta - \delta C = \mathbf{0}$, the response is constant for all k.

Proof. Writing $A(k) = (C + k\mathbf{v}_\Delta)/(E + k\delta)$ and subtracting yields

$$A(k+1) - A(k) = \frac{(C + (k+1)\mathbf{v}_\Delta)(E + k\delta) - (C + k\mathbf{v}_\Delta)(E + (k+1)\delta)}{(E + k\delta)(E + (k+1)\delta)}$$
$$= \frac{E\mathbf{v}_\Delta - \delta C}{(E + k\delta)(E + (k+1)\delta)}.$$

Both denominator factors are positive because

$$E + k\delta = Z_0 + ka_R + (d-k)a_B > 0.$$

The numerator is fixed. As k increases, the denominator increases if $\delta > 0$, stays constant if $\delta = 0$, and decreases if $\delta < 0$. These three cases give the stated behavior of the response distances. If the numerator is zero, every increment is zero. □

For the fixed-degree construction in Appendix A.2, marginal compression occurs when target-category occurrences receive larger attention weights than the occurrences they replace, provided the response increment is nonzero. Fixed degree alone does not guarantee this condition.

B.2 Attention Distortion under Graph Serialization

We examine how serialization changes the attention response and what correcting these changes implies for marginal compression in Section 2.3. We first compare the serialized response with a fixed reference, then analyze the contribution of unequal attention weights.

B.2.1 Attention Response under Serialization

Fix a layer ℓ, an attention head h, and a query q. For the target category C_t, let c count the occurrences visible to this query and let i index these occurrences. Let σ specify the serialization, including visibility, token positions, and positional encodings. At count c, occurrence i has unnormalized attention weight and value

$$a_{t,i}^{\ell,h,\sigma}(c) = \exp\Big(s_{t,i}^{\ell,h,\sigma}(c)\Big) > 0, \qquad \mathbf{v}_{t,i}^{\ell,h,\sigma}(c) \in \mathbb{R}^p. \tag{18}$$

We omit (ℓ, h, σ) below. Let $Z_{-t}(c) > 0$ and $\mathbf{N}_{-t}(c)$ be the total unnormalized attention weight and weighted-value sum of the remaining input components. The response is

$$\mathbf{h}_t^{\mathrm{ser}}(c) = \frac{\mathbf{N}_{-t}(c) + \sum_{i=1}^{c} a_{t,i}(c)\mathbf{v}_{t,i}(c)}{Z_{-t}(c) + \sum_{i=1}^{c} a_{t,i}(c)}. \tag{19}$$

Unlike the reference in Theorem 2.3, scores and values can vary across occurrences and counts. For $c \geq 1$, define the target-category attention weight, its share of the total weight, and the background mean value as

$$\begin{aligned} S_t(c) &= \sum_{i=1}^{c} a_{t,i}(c), \quad \lambda_t(c) = \frac{S_t(c)}{Z_{-t}(c) + S_t(c)}, \\ \mathbf{v}_{-t}^{\mathrm{ser}}(c) &= \frac{\mathbf{N}_{-t}(c)}{Z_{-t}(c)}. \end{aligned} \tag{20}$$

B.2.2 Reference Response and Accumulation Residual

Fix one reference $\mathbf{h}_t^\star(c)$ satisfying Theorem 2.3. Its accumulated change after the first occurrence is

$$\mathbf{h}_t^{\mathrm{acc},\star}(c) = \sum_{u=1}^{c-1} \mathbf{\Delta}_{t,u}^\star = \mathbf{h}_t^\star(c) - \mathbf{h}_t^\star(1), \qquad c \geq 1. \tag{21}$$

The sum is zero at $c = 1$. We measure the serialized response's deviation from this accumulated change by

$$\begin{aligned} \boldsymbol{\epsilon}_t^{\mathrm{acc,ser}}(c) &= \mathbf{h}_t^{\mathrm{ser}}(c) - \mathbf{h}_t^{\mathrm{ser}}(1) - \mathbf{h}_t^{\mathrm{acc},\star}(c), \\ \mathbf{h}_t^{\mathrm{ser}}(c) &= \mathbf{h}_t^{\mathrm{ser}}(1) + \mathbf{h}_t^{\mathrm{acc},\star}(c) + \boldsymbol{\epsilon}_t^{\mathrm{acc,ser}}(c). \end{aligned} \tag{22}$$

All responses lie in the same head space. This residual measures deviation from the reference; a nonzero residual need not imply a prediction error. Appendix B.2.6 specifies the matched reference used in the diagnostics, while Appendix B.2.7 treats the first-occurrence response separately.

B.2.3 Marginal Compression after Deviation Correction

Corollary B.3 (Marginal Compression under Deviation Correction)**.** *Fix the reference in Theorem 2.3, with $\mathbf{v}_t \neq \mathbf{v}_{-t}$. Let $\widetilde{\mathbf{h}}_t(c)$ be the response after a presentation intervention. If, for every integer $c \geq 1$,*

$$\widetilde{\mathbf{h}}_t(c) - \widetilde{\mathbf{h}}_t(1) = \mathbf{h}_t^\star(c) - \mathbf{h}_t^\star(1),$$

then $\widetilde{\mathbf{h}}_t(c) = \mathbf{h}_t^\star(c) + \mathbf{b}$ for a vector $\mathbf{b}$ independent of c. Consequently,

$$\widetilde{\mathbf{h}}_t(c+1) - \widetilde{\mathbf{h}}_t(c) = \mathbf{\Delta}_{t,c}^\star, \qquad c \geq 1,$$

and these increment norms strictly decrease to zero.

Proof. Define the corrected accumulation residual as

$$\widetilde{\boldsymbol{\epsilon}}_t(c) = \widetilde{\mathbf{h}}_t(c) - \widetilde{\mathbf{h}}_t(1) - \mathbf{h}_t^{\mathrm{acc},\star}(c). \tag{23}$$

Subtracting at adjacent counts gives

$$\begin{aligned}\widetilde{\mathbf{h}}_t(c+1) - \widetilde{\mathbf{h}}_t(c) = \boldsymbol{\Delta}_{t,c}^{\star} \\ + \widetilde{\boldsymbol{\epsilon}}_t(c+1) - \widetilde{\boldsymbol{\epsilon}}_t(c).\end{aligned} \tag{24}$$

Under complete correction, $\widetilde{\boldsymbol{\epsilon}}_t(c) = \mathbf{0}$ for every $c \geq 1$. Thus $\mathbf{b} = \widetilde{\mathbf{h}}_t(1) - \mathbf{h}_t^{\star}(1)$ is constant, and the increments equal those of the reference. Theorem 2.3 gives their strict decrease and zero limit. □

Correcting deviations from this reference therefore does not remove its marginal compression. For partial correction, the increments also depend on the residual difference between adjacent counts in Eq. (24). Reducing residual norms alone does not determine whether adjacent responses become more separated. Corollary B.3 concerns interventions that reproduce the reference increments. The following result applies to arbitrary, possibly count-dependent, score redistribution.

Proposition B.4 (Bounded Count Separation under Normalized Aggregation)**.** *Fix a layer, head, and query. Let a presentation intervention change visibility, token positions, positional encodings, and attention scores arbitrarily, including in a count-dependent manner, while retaining softmax-normalized aggregation. Suppose all value vectors satisfy $\|v\|_2 \leq R$. Let f be any L-Lipschitz readout such that $f(\tilde{h}_t(c))$ is increasing in c on $\{1, \ldots, C\}$. Then*

$$\min_{1 \leq c < C} \big[f(\tilde{h}_t(c+1)) - f(\tilde{h}_t(c)) \big] \leq \frac{2LR}{C-1}. \tag{25}$$

Proof. Each $\tilde{h}_t(c)$ is a convex combination of value vectors, so $\|\tilde{h}_t(c)\|_2 \leq R$. The adjacent gaps are positive and telescope to $f(\tilde{h}_t(C)) - f(\tilde{h}_t(1)) \leq L\|\tilde{h}_t(C) - \tilde{h}_t(1)\|_2 \leq 2LR$. The minimum of $C-1$ positive terms is at most their mean. □

In contrast, with fixed occurrence messages, TrajAgg satisfies $T^{(\ell)}_{q,G_{c+1}} - T^{(\ell)}_{q,G_c} = m_t$ for every c, so a linear readout w with $w^\top m_t = \kappa$ separates every pair of adjacent counts by κ with $\|w\|$ independent of C. The bound extends to multiple heads and layers whenever each contribution to the residual stream is bounded. It does not cover interventions that insert explicit count tokens, which supply new features rather than re-present the same evidence.

B.2.4 Attention-Weight Dispersion and Effective Cardinality

For $c \geq 1$, let $\bar{a}_t(c) = c^{-1} \sum_{i=1}^{c} a_{t,i}(c)$. Define the coefficient of variation of the occurrence weights and their effective cardinality by

$$\begin{aligned}\mathrm{CV}_t(c)^2 &= \frac{c^{-1} \sum_{i=1}^{c} (a_{t,i}(c) - \bar{a}_t(c))^2}{\bar{a}_t(c)^2}, \\ c_{\mathrm{eff},t}(c) &= \frac{\left(\sum_{i=1}^{c} a_{t,i}(c)\right)^2}{\sum_{i=1}^{c} a_{t,i}(c)^2}.\end{aligned} \tag{26}$$

Effective cardinality expresses attention concentration as an equivalent number of equally weighted occurrences. It does not measure how many occurrences the model uses correctly for prediction.

Proposition B.5 (Attention Dispersion and Effective Cardinality)**.** *For $c \geq 1$ occurrences with positive attention weights,*

$$\frac{c_{\mathrm{eff},t}(c)}{c} = \frac{1}{1 + \mathrm{CV}_t(c)^2}. \tag{27}$$

Greater weight dispersion therefore gives a smaller effective-cardinality ratio.

B.2.5 Proof and Response Bound

Define the normalized weights within the target category as

$$\pi_{t,i}(c) = \frac{a_{t,i}(c)}{S_t(c)}, \qquad \sum_{i=1}^{c} \pi_{t,i}(c) = 1. \tag{28}$$

Then

$$c_{\mathrm{eff},t}(c) = \frac{1}{\sum_{i=1}^{c} \pi_{t,i}(c)^2}. \tag{29}$$

Proof. Expanding the variance gives

$$\mathrm{CV}_t(c)^2 = \frac{\sum_{i=1}^{c} a_{t,i}(c)^2}{c\bar{a}_t(c)^2} - 1. \tag{30}$$

Since $\sum_i a_{t,i}(c) = c\bar{a}_t(c)$,

$$\frac{c_{\mathrm{eff},t}(c)}{c} = \frac{c\bar{a}_t(c)^2}{\sum_{i=1}^{c} a_{t,i}(c)^2} = \frac{1}{1 + \mathrm{CV}_t(c)^2}. \tag{31}$$

□

Response change due to unequal weights. Define the weighted response, the mean value, and the value dispersion within the target category as

$$\begin{aligned} \mathbf{R}_t^{\mathrm{ser}}(c) &= \sum_{i=1}^{c} \pi_{t,i}(c)\mathbf{v}_{t,i}(c), \\ \bar{\mathbf{v}}_t(c) &= \frac{1}{c}\sum_{i=1}^{c} \mathbf{v}_{t,i}(c), \\ \sigma_{v,t}(c)^2 &= \frac{1}{c}\sum_{i=1}^{c} \|\mathbf{v}_{t,i}(c) - \bar{\mathbf{v}}_t(c)\|_2^2. \end{aligned} \tag{32}$$

The complete response is

$$\mathbf{h}_t^{\mathrm{ser}}(c) = [1 - \lambda_t(c)]\mathbf{v}_{-t}^{\mathrm{ser}}(c) + \lambda_t(c)\mathbf{R}_t^{\mathrm{ser}}(c). \tag{33}$$

To isolate unequal weighting, replace the within-category weights by $1/c$ while preserving their total attention share, all value vectors, and the background:

$$\mathbf{h}_t^{\mathrm{eq}}(c) = [1 - \lambda_t(c)]\mathbf{v}_{-t}^{\mathrm{ser}}(c) + \lambda_t(c)\bar{\mathbf{v}}_t(c). \tag{34}$$

The resulting difference is

$$\begin{aligned} \boldsymbol{\epsilon}_t^{\mathrm{wt,ser}}(c) &= \mathbf{h}_t^{\mathrm{ser}}(c) - \mathbf{h}_t^{\mathrm{eq}}(c) \\ &= \lambda_t(c)[\mathbf{R}_t^{\mathrm{ser}}(c) - \bar{\mathbf{v}}_t(c)]. \end{aligned} \tag{35}$$

Because $\sum_i(\pi_{t,i}(c) - 1/c) = 0$,

$$\mathbf{R}_t^{\mathrm{ser}}(c) - \bar{\mathbf{v}}_t(c) = \sum_{i=1}^{c} \left(\pi_{t,i}(c) - \frac{1}{c}\right)[\mathbf{v}_{t,i}(c) - \bar{\mathbf{v}}_t(c)]. \tag{36}$$

Equation (31) gives

$$\sum_{i=1}^{c} \left(\pi_{t,i}(c) - \frac{1}{c}\right)^2 = \frac{1}{c_{\mathrm{eff},t}(c)} - \frac{1}{c} = \frac{\mathrm{CV}_t(c)^2}{c}.$$

Applying Cauchy–Schwarz to Eq. (36) therefore yields

$$\|\boldsymbol{\epsilon}_t^{\mathrm{wt,ser}}(c)\|_2 \le \underbrace{\lambda_t(c)\sigma_{v,t}(c)\,\mathrm{CV}_t(c)}_{S_t^{\mathrm{CV}}(c)}. \tag{37}$$

The score $S_t^{\mathrm{CV}}(c)$ combines attention-weight dispersion with the target category's attention share and value dispersion. Holding the latter two quantities fixed, a larger CV gives a larger upper bound on the response change caused by unequal weights.

B.2.6 Relation to the Complete Accumulation Residual

For each replicate, serialization, node order, layer, and head, we construct the reference from the matched $c = 1$ input:

$$\begin{aligned} a_t^\star &= a_{t,1}(1), \qquad \mathbf{v}_t^\star = \mathbf{v}_{t,1}(1), \\ Z_{-t}^\star &= Z_{-t}(1), \quad \mathbf{v}_{-t}^\star = \frac{\mathbf{N}_{-t}(1)}{Z_{-t}(1)}. \end{aligned} \tag{38}$$

Holding these quantities fixed gives

$$\mathbf{h}_t^\star(c) = \frac{Z_{-t}^\star \mathbf{v}_{-t}^\star + c a_t^\star \mathbf{v}_t^\star}{Z_{-t}^\star + c a_t^\star}. \tag{39}$$

Since $\mathbf{h}_t^\star(1) = \mathbf{h}_t^{\mathrm{ser}}(1)$, the accumulation residual reduces to

$$\boldsymbol{\epsilon}_t^{\mathrm{acc,ser}}(c) = \mathbf{h}_t^{\mathrm{ser}}(c) - \mathbf{h}_t^\star(c). \tag{40}$$

Writing the reference attention share as

$$\lambda_t^\star(c) = \frac{c a_t^\star}{Z_{-t}^\star + c a_t^\star}, \tag{41}$$

we obtain

$$\begin{aligned} \boldsymbol{\epsilon}_t^{\mathrm{acc,ser}}(c) = {} & [1 - \lambda_t(c)][\mathbf{v}_{-t}^{\mathrm{ser}}(c) - \mathbf{v}_{-t}^\star] \\ & + \lambda_t(c)[\mathbf{R}_t^{\mathrm{ser}}(c) - \bar{\mathbf{v}}_t(c)] \\ & + \lambda_t(c)[\bar{\mathbf{v}}_t(c) - \mathbf{v}_t^\star] \\ & + [\lambda_t(c) - \lambda_t^\star(c)][\mathbf{v}_t^\star - \mathbf{v}_{-t}^\star]. \end{aligned} \tag{42}$$

The four terms describe changes in the background, unequal occurrence weights, changes in occurrence values, and changes in the target category's total attention share. By the triangle inequality and Eq. (37),

$$\begin{aligned} \|\boldsymbol{\epsilon}_t^{\mathrm{acc,ser}}(c)\|_2 \le {} & [1 - \lambda_t(c)]\|\mathbf{v}_{-t}^{\mathrm{ser}}(c) - \mathbf{v}_{-t}^\star\|_2 \\ & + S_t^{\mathrm{CV}}(c) + \lambda_t(c)\|\bar{\mathbf{v}}_t(c) - \mathbf{v}_t^\star\|_2 \\ & + |\lambda_t(c) - \lambda_t^\star(c)|\|\mathbf{v}_t^\star - \mathbf{v}_{-t}^\star\|_2. \end{aligned} \tag{43}$$

Thus, the CV-weighted score bounds the unequal-weight component; the other terms account for changes in the background, values, and attention share.

B.2.7 Layerwise Comparison of Onset and Accumulation

The accumulation residual excludes the change from zero to one occurrence. We measure deviation of this first-occurrence increment separately:

$$\boldsymbol{\epsilon}_t^{\mathrm{onset,ser}} = [\mathbf{h}_t^{\mathrm{ser}}(1) - \mathbf{h}_t^{\mathrm{ser}}(0)] - \boldsymbol{\Delta}_{t,0}^\star. \tag{44}$$

Its size depends on the layer, head, and serialization. For the layerwise diagnostic, we measure variation across node orders in the attention-output changes from 0 to 1 occurrence (onset) and from 1 to c occurrences (accumulation). For either component, relative dispersion is the mean Euclidean distance of these change vectors from their across-order mean, divided by the norm of that mean, with a denominator floor of 10^{-12}. We obtain D_{onset} and D_{acc} by averaging over serialization schemes and replicates, and additionally over the evaluated counts for accumulation. In the matched layerwise diagnostics, the difference in relative dispersion between accumulation and first occurrence, $D_{\mathrm{acc}} - D_{\mathrm{onset}}$, is -0.040 for Qwen3-4B and -0.014 for Qwen3-8B at layer 21. At layer 12, the differences are 0.311 and 0.295, respectively, with $p = 2.44 \times 10^{-4}$ for both models. We test whether accumulation exceeds onset using a one-sided Wilcoxon signed-rank test on the 12 replicate-level differences. The sign change shows that accumulation does not have higher relative dispersion than first occurrence at every layer.

Table 13: Attention dispersion and the complete accumulation residual. Correlations are reported as mean Spearman ρ across replicates, with within-count correlations computed at fixed counts.

Measurement	**Qwen3-4B**	**Qwen3-8B**
Correlation with residual		
CV-weighted score	0.823	0.728
CV-weighted score (within count)	0.806	0.708
One-sided Wilcoxon p	2.44×10^{-4}	2.44×10^{-4}
CV alone	0.208	-0.180
Complete bound	0.990	0.990
Weight equalization		
Mean residual reduction	0.053	0.071
One-sided Wilcoxon p	2.44×10^{-4}	2.44×10^{-4}
Number of observations	2,880	2,880

B.2.8 Accumulation Residual Diagnostics

For each model, we evaluate 12 independent replicates, two serializations, three node orders, five counts $c \in \{2, \ldots, 6\}$, and eight selected heads, giving 2,880 observations. The matched $c = 1$ input defines the reference for each observation. The correlation and equalization p-values use one-sided Wilcoxon signed-rank tests across 12 replicates, testing whether replicate-level Spearman correlations and mean residual reductions, respectively, are greater than zero. Table 13 reports the resulting diagnostics.

Raw CV has a weak positive correlation with the complete residual in Qwen3-4B and a negative correlation in Qwen3-8B. The CV-weighted score is strongly correlated with the residual in both models, including comparisons at fixed counts. Equalizing occurrence weights while retaining their total attention share also reduces the complete residual in both models. These results support unequal attention weights as one source of deviation from the reference, alongside changes in background responses, value vectors, and total attention share.

B.3 Proof of Proposition 3.1

Fix the two inputs G_n and G_{n+1} in Section 3.3, where $n \geq 0$, and let $\kappa > 0$. Both candidates y_- and y_+ are available on both inputs and are correct at counts n and $n+1$, respectively. We use the pairwise margins in Eq. (7). Assume that the normalized candidate representations $\mathbf{f}_\pm = \mathrm{LN}(\mathbf{e}_{y_\pm})$ are fixed across the two inputs and satisfy $\mathbf{f}_+ \neq \mathbf{f}_-$. The construction uses a response width of at least one and a GELU hidden width of at least two in the head defined in Appendix C.4. All projection and head parameters below are shared by the two inputs and candidates.

Proof.

Reading the structural increment. Write $\mathbf{T}(c) = \mathbf{T}^{(\ell)}_{q,G_c}$ for $c \in \{n, n+1\}$. The fixed-message condition gives

$$\mathbf{T}(n+1) - \mathbf{T}(n) = \mathbf{m}_t.$$

Under the shared affine scaling in Eq. (51), let

$$\mathbf{X}(c) = \mathrm{Scale}(\mathbf{T}(c)), \qquad \mathbf{b}_t = \mathbf{X}(n+1) - \mathbf{X}(n) = \frac{\boldsymbol{\gamma}_T \odot \mathbf{m}_t}{a_T} \neq \mathbf{0}.$$

Choose the first row and bias of the structural projection as

$$\mathbf{w}_T = \frac{\kappa \mathbf{b}_t}{\|\mathbf{b}_t\|_2^2}, \qquad \beta_n = -\mathbf{w}_T^\top \mathbf{X}(n) - \frac{\kappa}{2}.$$

The resulting scalar $z_T(c) = \mathbf{w}_T^\top \mathbf{X}(c) + \beta_n$ satisfies

$$z_T(n) = -\frac{\kappa}{2}, \qquad z_T(n+1) = \frac{\kappa}{2}.$$

Set the remaining structural projection coordinates to zero. The readout uses the accumulated state; the count is not an additional input.

Conditioning on the candidate. Let $\mathbf{d}_e = \mathbf{f}_+ - \mathbf{f}_- \neq \mathbf{0}$. Choose the first row and bias of the candidate projection as

$$\mathbf{w}_e = \frac{2\mathbf{d}_e}{\|\mathbf{d}_e\|_2^2}, \qquad \beta_e = -\frac{1}{2}\mathbf{w}_e^\top(\mathbf{f}_+ + \mathbf{f}_-).$$

Then $\xi_{y_\pm} = \mathbf{w}_e^\top \mathbf{f}_\pm + \beta_e = \pm 1$. Set the remaining candidate projection coordinates to zero. The structural–candidate interaction in Eq. (54) now contains $x_{y'}(c) = \xi_{y'} z_T(c)$ for $y' \in \{y_-, y_+\}$.

Constructing the correction. Set two hidden preactivations to $x_{y'}(c)$ and $-x_{y'}(c)$. For the standard normal cumulative distribution function Φ,

$$\mathrm{GELU}(x) - \mathrm{GELU}(-x) = x\Phi(x) + x\Phi(-x) = x.$$

The output layer can therefore produce

$$r_{y'}(c) = x_{y'}(c), \qquad g_{y'}(c) = 0.$$

The gate output is obtained by setting its output weights and bias to zero, giving $\sigma(g_{y'}(c)) = 1/2$ exactly. All unused inputs and hidden units have zero output contribution. The bounded head in Eq. (55) then gives

$$\delta_{y'}(c) = \frac{\Delta_{\max}}{2}\tanh(\xi_{y'} z_T(c))\,.$$

This is the scalar correction denoted by $\mathrm{MLP}(\mathbf{u}_{y'})$ in the main text.

Improving the two margins. Let $\eta = (\Delta_{\max}/2)\tanh(\kappa/2) > 0$. At the two inputs,

$$\delta_{y_-}(n) = \delta_{y_+}(n+1) = \eta, \qquad \delta_{y_+}(n) = \delta_{y_-}(n+1) = -\eta.$$

Adding these corrections to the original scores increases both margins by 2η:

$$M_\pm^{\mathrm{SAIL}} = M_\pm^{\mathrm{LLM}} + \Delta_{\max}\tanh(\kappa/2).$$

This is Eq. (8). For any finite original scores, both corrected margins are strictly positive whenever

$$\Delta_{\max} > \frac{\max\{0, -M_-^{\mathrm{LLM}}, -M_+^{\mathrm{LLM}}\}}{\tanh(\kappa/2)}.$$

Each input then ranks its correct response above the competing candidate, proving the proposition. □

Readout norm and count dependence. For the fixed scaled increment $\mathbf{b}_t$, Cauchy–Schwarz gives

$$\min_{\mathbf{w}^\top \mathbf{b}_t = \kappa} \|\mathbf{w}\|_2 = \frac{\kappa}{\|\mathbf{b}_t\|_2},$$

with equality for the constructed $\mathbf{w}_T$. When the occurrence message and scaling are fixed across counts, this minimum does not depend on n, although the centering bias β_n does. For the normalized attention reference in Theorem 2.3,

$$\boldsymbol{\Delta}_{t,n}^\star = \frac{Z_{-t}a_t(\mathbf{v}_t - \mathbf{v}_{-t})}{(Z_{-t} + na_t)(Z_{-t} + (n+1)a_t)}.$$

With $Z_{-t} > 0$, $a_t > 0$, and $\mathbf{v}_t \neq \mathbf{v}_{-t}$ fixed,

$$\begin{aligned}\min_{\mathbf{w}^\top \boldsymbol{\Delta}_{t,n}^\star = \kappa} \|\mathbf{w}\|_2 &= \frac{\kappa}{\|\boldsymbol{\Delta}_{t,n}^\star\|_2} \\ &= \frac{\kappa(Z_{-t} + na_t)(Z_{-t} + (n+1)a_t)}{Z_{-t}a_t\|\mathbf{v}_t - \mathbf{v}_{-t}\|_2} = \Theta(n^2).\end{aligned}$$

Thus, with each representation's scale held fixed, the sum avoids the growing readout norm needed to amplify the shrinking attention increment. This comparison concerns scalar linear readouts, not full-network norms.

The construction applies to a fixed adjacent pair and its two candidates. It establishes the existence of shared parameters, rather than a guarantee that training finds them or that they rank the correct answer above every other candidate. The centering bias depends on the chosen pair, so this proof does not establish a single set of parameters that gives the same margin gain at every count or guarantees extrapolation to unseen counts.

C SAIL METHOD DETAILS

This appendix describes the implementation of SAIL introduced in Section 3, including structural scope selection, trajectory extraction, relation encodings, residual score correction, and training.

C.1 QUERY-CONDITIONED STRUCTURAL SCOPE

For a graph $G = (V, E)$ and prediction query q, we select a node set $\mathcal{S}(q, G) \subseteq V$ for structural accumulation, as described in Section 3.1. Let $\mathcal{N}_k(v)$ denote the k-hop neighborhood of node v, including v itself. The structural scope is

$$\mathcal{S}(q, G) = \begin{cases} \mathcal{N}_k(v), & q \text{ targets node } v, \\ \mathcal{N}_k(u) \cup \mathcal{N}_k(v), & q \text{ targets edge } (u, v), \\ V, & q \text{ targets the graph.} \end{cases} \tag{45}$$

We use $k = 1$ for node and edge queries and ignore edge direction when constructing neighborhoods. Scope selection uses the input graph and query targets, without using answer labels. Each selected node is a separate structural occurrence, even when multiple nodes share the same category.

C.2 OCCURRENCE TRAJECTORY EXTRACTION

For each node $i \in \mathcal{S}(q, G)$, we average token embeddings and residual states over $\mathcal{P}_i$ to obtain $h_i^{(0)}$ and $h_i^{(\ell)}$, respectively. Here $\mathcal{P}_i$ contains the token positions associated with node i, and $\ell > 0$ is the extraction layer, with both representations having width d_{LLM}.

For attention head a, let $s_{q,t}^{(\ell,a)}$ denote the pre-softmax attention score from the final prompt position to token t, and let $v_t^{(\ell,a)} \in \mathbb{R}^{d_h}$ be its value vector. We pool these values using attention weights normalized within each occurrence:

$$\begin{aligned} \widetilde{\alpha}_{i,t}^{(\ell,a)} &= \frac{\exp(s_{q,t}^{(\ell,a)})}{\sum_{t' \in \mathcal{P}_i} \exp(s_{q,t'}^{(\ell,a)})}, \qquad t \in \mathcal{P}_i, \\ z_i^{(\ell,a)} &= \sum_{t \in \mathcal{P}_i} \widetilde{\alpha}_{i,t}^{(\ell,a)} v_t^{(\ell,a)}. \end{aligned} \tag{46}$$

Concatenating the pooled values from H heads and applying the LLM output projection gives

$$u_{i,\text{attn}}^{(\ell)} = W_O^{(\ell)} \operatorname{Concat}\big(z_i^{(\ell,1)}, \ldots, z_i^{(\ell,H)}\big), \tag{47}$$

where $W_O^{(\ell)}$ maps width Hd_h to d_{LLM}, with unused head slots set to zero when extraction uses only a subset of heads.

The normalization controls token pooling within each occurrence, not its attention strength relative to other nodes. For a single-token occurrence $\mathcal{P}_i = \{t\}$, $\widetilde{\alpha}_{i,t}^{(\ell,a)} = 1$ and $z_i^{(\ell,a)} = v_t^{(\ell,a)}$, so pooling no longer depends on the attention score. For multi-token occurrences, pooling retains the relative query attention weights within $\mathcal{P}_i$, while both cases preserve any contextual information already encoded in the value vectors.

This design removes competition for attention mass across occurrences, allowing each message to enter TrajAgg's unnormalized sum without being scaled by its share of the full attention distribution.

C.3 QUERY-RELATIVE RELATION ENCODINGS

Each node $i \in \mathcal{S}(q, G)$ receives a role index $\kappa_i \in \{1, 2, 3\}$, with three slots providing a common feature width across tasks. For node queries, the target uses slot 1 and its neighbors use slot 3, leaving slot 2 inactive. For edge queries, slots 1, 2, and 3 represent membership only in the first endpoint's neighborhood, only in the second endpoint's neighborhood, or in both, respectively. A node in the intersection is included once with role 3, which is also the role assigned to all nodes in graph queries.

For $s \in \{1,2,3\}$, the role-count vector $\mathbf{R}_{\mathrm{count},i} \in \mathbb{R}^3$ and the mapped trajectory message $\mathbf{R}^{(\ell)}_{\mathrm{map},i} \in \mathbb{R}^{3d_{\mathrm{LLM}}}$ are defined by

$$\begin{aligned} \mathbf{R}_{\mathrm{count},i}[s] &= \mathbf{1}\{s = \kappa_i\}, \\ \mathbf{R}^{(\ell)}_{\mathrm{map},i}[(s-1)d_{\mathrm{LLM}} + 1 : sd_{\mathrm{LLM}}] &= \mathbf{1}\{s = \kappa_i\}\, \bar{\mathbf{t}}^{(\ell)}_i. \end{aligned} \tag{48}$$

Here $\mathbf{1}\{\cdot\}$ is the indicator function, and $[a : b]$ selects coordinates a through b. After accumulation in Eq. (4), the count coordinates record the number of nodes in each role, while the mapped blocks store the sums of trajectory messages within each role, giving $\mathbf{T}^{(\ell)}_{q,G}$ width $4d_{\mathrm{LLM}} + 3$.

Since each node has exactly one role, summing its three mapped blocks recovers $\bar{\mathbf{t}}^{(\ell)}_i$. The total trajectory block in $\mathbf{T}^{(\ell)}_{q,G}$ is therefore the sum of its three role blocks and adds no independent information. We retain it to give QueryInt direct access to both the overall sum of trajectory messages and the separate role sums.

C.4 QueryInt: Projection and Residual Integration

For query q, h_q is the residual representation at the final prompt position in extraction layer ℓ, and $\mathcal{Y}(q)$ contains the valid candidate responses. In GraphMACC, these are $\{0,1\}$ for existence and comparison and $\{0,\dots,d\}$ for counting. Other tasks use the candidate options specified by the task input, with the same options used by SAIL and its uncorrected baseline.

For candidate y' with token sequence $(a_1,\dots,a_m)$, we obtain its representation from the frozen LLM input embedding table E_{in}:

$$\mathbf{e}_{y'} = \frac{1}{m}\sum_{j=1}^{m} E_{\mathrm{in}}[a_j]. \tag{49}$$

The original candidate score uses the first token's raw logit and the conditional log-probabilities of any remaining tokens:

$$\begin{aligned} s^{\mathrm{LLM}}_{y'} = \mathrm{logit}(a_1 \mid q, G) \\ + \sum_{j=2}^{m} \log p(a_j \mid q, G, a_1, \dots, a_{j-1}). \end{aligned} \tag{50}$$

Here conditioning on (q, G) denotes the serialized graph and query prompt. For a single-token candidate, this reduces to its next-token logit. For multiple tokens, the score equals the sequence log-probability up to a constant shared by all candidates, without length normalization. Invalid and padded candidates are excluded from score standardization, training loss, and prediction.

Fixed structural scaling. We scale the accumulated state using one denominator estimated from the training examples used to fit the response model. For their states $\{\mathbf{T}_n\}_{n=1}^{N_{\mathrm{fit}}}$ of width $d_T = 4d_{\mathrm{LLM}} + 3$, define

$$\begin{aligned} a_T &= \frac{1}{N_{\mathrm{fit}}}\sum_{n=1}^{N_{\mathrm{fit}}} \frac{\|\mathbf{T}_n\|_2}{\sqrt{d_T}} > 0, \\ \mathrm{Scale}(\mathbf{T}) &= \boldsymbol{\gamma}_T \odot \frac{\mathbf{T}}{a_T} + \boldsymbol{\beta}_T. \end{aligned} \tag{51}$$

The denominator a_T remains fixed during training and inference, while $\boldsymbol{\gamma}_T, \boldsymbol{\beta}_T \in \mathbb{R}^{d_T}$ are learned affine parameters initialized to ones and zeros, respectively. Using a shared denominator avoids normalizing away each state's multiplicity-dependent amplitude, and validation and test examples do not enter its estimate.

Input projections. We project the structural state, query representation, candidate embedding, and standardized candidate score to a common width d_R:

$$\begin{aligned}\widetilde{\mathbf{T}}_{q,G}^{(\ell)} &= W_T\,\mathrm{Scale}(\mathbf{T}_{q,G}^{(\ell)}) + \mathbf{b}_T,\\ \widetilde{h}_q &= W_q\,\mathrm{LN}(h_q) + \mathbf{b}_q,\\ \widetilde{\mathbf{e}}_{y'} &= W_e\,\mathrm{LN}(\mathbf{e}_{y'}) + \mathbf{b}_e,\\ \widetilde{\mathbf{s}}_{y'}^{\mathrm{LLM}} &= W_s[\mathrm{Std}(\mathbf{s}_{\mathcal{Y}(q)}^{\mathrm{LLM}})]_{y'} + \mathbf{b}_s.\end{aligned} \tag{52}$$

The projection matrices have dimensions $W_T \in \mathbb{R}^{d_R \times d_T}$, $W_q, W_e \in \mathbb{R}^{d_R \times d_{\mathrm{LLM}}}$, and $W_s \in \mathbb{R}^{d_R \times 1}$, with biases in $\mathbb{R}^{d_R}$. LN is standard LayerNorm with learned affine parameters, applied only to the query and candidate representations. Std subtracts the mean over valid candidate scores and divides by their population standard deviation with a floor of 10^{-6}, so equal scores map to zero. The standardized score is an input to the correction head, while the residual is added to the original score $s_{y'}^{\mathrm{LLM}}$.

Interaction features and residual head. The four projected inputs form the vector used in Section 3.3:

$$\mathbf{u}_{y'} = \mathrm{Concat}\big(\widetilde{\mathbf{T}}_{q,G}^{(\ell)}, \widetilde{h}_q, \widetilde{\mathbf{e}}_{y'}, \widetilde{\mathbf{s}}_{y'}^{\mathrm{LLM}}\big) \in \mathbb{R}^{4d_R}. \tag{53}$$

We include element-wise products of the candidate embedding with the structural and query representations to form the network input:

$$\begin{aligned}\mathbf{z}_{y'} = \mathrm{Concat}\big(&\widetilde{\mathbf{T}}_{q,G}^{(\ell)}, \widetilde{h}_q, \widetilde{\mathbf{e}}_{y'},\\ &\widetilde{\mathbf{T}}_{q,G}^{(\ell)} \odot \widetilde{\mathbf{e}}_{y'}, \widetilde{h}_q \odot \widetilde{\mathbf{e}}_{y'}, \widetilde{\mathbf{s}}_{y'}^{\mathrm{LLM}}\big) \in \mathbb{R}^{6d_R}.\end{aligned} \tag{54}$$

Here $\odot$ denotes element-wise multiplication. A network $F_\theta : \mathbb{R}^{6d_R} \to \mathbb{R}^2$ with two affine layers separated by GELU produces a raw correction $r_{y'}$ and a gate logit $g_{y'}$:

$$\begin{aligned}(r_{y'}, g_{y'}) &= F_\theta(\mathbf{z}_{y'}),\\ \mathrm{MLP}(\mathbf{u}_{y'}) &= \Delta_{\max}\tanh(r_{y'})\,\sigma(g_{y'}).\end{aligned} \tag{55}$$

The notation $\mathrm{MLP}(\mathbf{u}_{y'})$ in the main text includes the interaction construction and this bounded output, where σ is the logistic sigmoid and $\Delta_{\max} > 0$ controls the correction bound. The projections and network parameters are shared across candidates, and the final affine layer is initialized to zero so that SAIL initially reproduces the original scores. We add the correction as in Eq. (5) and predict $\hat{y}_{\mathrm{SAIL}} = \arg\max_{y' \in \mathcal{Y}(q)} s_{y'}^{\mathrm{SAIL}}$.

C.5 Training Objective

We train QueryInt with the LLM frozen, updating its normalization affine parameters, input projections, and residual head while keeping a_T fixed. For a training batch $\mathcal{B}$, let y_q be the correct answer to query q and let $\mathcal{B}_+$ contain the queries that the original LLM answers correctly. We write $\delta_{q,y'} = \mathrm{MLP}(\mathbf{u}_{y'})$ for the complete gated correction to candidate y' on query q, as defined in Eq. (55).

For $X \in \{\mathrm{LLM}, \mathrm{SAIL}\}$, $p_X^{(q)} = \mathrm{softmax}(\mathbf{s}_{\mathcal{Y}(q)}^X)$ is the distribution over valid candidates. The training objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{task}} &= \frac{1}{|\mathcal{B}|}\sum_{q\in\mathcal{B}} \mathrm{CE}(\mathbf{s}_{\mathcal{Y}(q)}^{\mathrm{SAIL}}, y_q),\\ \mathcal{L}_{\Delta} &= \frac{\sum_{q\in\mathcal{B}}\sum_{y'\in\mathcal{Y}(q)} \delta_{q,y'}^2}{\sum_{q\in\mathcal{B}}|\mathcal{Y}(q)|},\\ \mathcal{L}_{\mathrm{pres}} &= \begin{cases}\dfrac{1}{|\mathcal{B}_+|}\displaystyle\sum_{q\in\mathcal{B}_+} D_{\mathrm{KL}}(p_{\mathrm{LLM}}^{(q)} \| p_{\mathrm{SAIL}}^{(q)}), & |\mathcal{B}_+| > 0,\\ 0, & \text{otherwise},\end{cases}\\ \mathcal{L} &= \mathcal{L}_{\mathrm{task}} + \lambda_\Delta \mathcal{L}_\Delta + \lambda_{\mathrm{pres}}\mathcal{L}_{\mathrm{pres}}.\end{aligned} \tag{56}$$

Here CE is cross-entropy on the corrected candidate scores, and $\lambda_\Delta, \lambda_{\text{pres}} \geq 0$ control the two regularization terms. The task loss favors the correct answer, while $\mathcal{L}_\Delta$ penalizes large score corrections. The preservation term uses the original LLM distribution as the reference and discourages changing it when the original prediction is correct. All terms use valid candidates only, with training settings specified in Appendix D.4.

Table 14: Native graph benchmarks and metrics. Node and edge counts are per-graph averages.

Dataset	Graphs	Nodes	Edges	Metric
Node classification				
Cora (Sen et al., 2008)	1	2,708	10,556	Accuracy
PubMed (Namata et al., 2012)	1	19,717	88,648	Accuracy
WikiCS (Mernyei & Cangea, 2020)	1	11,701	431,726	Accuracy
ogbn-arxiv (Hu et al., 2020)	1	169,343	1,166,243	Accuracy
Link prediction				
ogbl-citation2 (Hu et al., 2020)	1	2,927,963	30,387,995	MRR
Graph classification				
ogbg-molhiv (Hu et al., 2020)	41,127	25.51	54.94	ROC-AUC

Table 15: Full test sets for the native benchmarks.

Dataset	Split	Test targets
Node classification		
Cora	Planetoid public	1,000 nodes
PubMed	Planetoid public	1,000 nodes
WikiCS	Official (split 0)	5,847 nodes
ogbn-arxiv	OGB temporal	48,603 nodes
Link prediction		
ogbl-citation2	OGB temporal	86,596 queries
Graph classification		
ogbg-molhiv	OGB scaffold	4,113 graphs

D Experimental Setup

For the experiments in Section 4.1, paired comparisons use the same evaluation queries and valid candidate sets.

D.1 Datasets and Prediction Tasks

We evaluate SAIL on GraphMACC (Appendix A) and six native graph benchmarks, whose sizes, prediction tasks, and evaluation metrics are listed in Table 14. For native tasks evaluated on sampled neighborhoods, we follow the neighborhood-sampling procedure in Appendix D.2 and predict the datasets' original labels, extending the evaluation beyond counting and comparison.

D.2 Data Splits and Graph Inputs

Data splits. Standalone methods and presentation combinations are evaluated on the complete test sets listed in Table 15. We use the Planetoid public splits (Yang et al., 2016) for Cora and PubMed. For WikiCS (Mernyei & Cangea, 2020), we use the first of the 20 provided training/validation splits and the shared test set. For the OGB datasets (Hu et al., 2020), we use the official temporal splits for ogbn-arxiv and ogbl-citation2 and the scaffold split for ogbg-molhiv. Training and model selection use the designated training and validation partitions, without access to test labels. GraphMACC follows the partitioning and aggregation rules in Appendices A.2 and A.5.

Node classification. Each input contains the target node and up to 32 one-hop neighbors, together with all source edges among the selected nodes. Neighborhood selection ignores edge direction, while serialization preserves direction for directed graphs. When more than 32 neighbors are available, we select them by a fixed hash ordering of the dataset, target, and neighbor identifiers with a fixed random seed, without using node labels. Each node is represented by its identifier and up to 16 nonzero feature entries with the largest absolute values, with ties resolved by feature index. Ground-

truth node labels are excluded from the graph input, and each class is assigned a fixed answer alias shared across methods.

Link prediction. For ogbl-citation2, each official validation or test query ranks one positive target against the 1,000 negative targets provided by OGB. For each source–target pair, the input contains both endpoints and their sampled one-hop neighbors. Each endpoint is described by its identifier, observed degree, and the six feature entries with the largest absolute values. SAIL and all presentation-alignment methods operate on this sampled input graph. Observed degrees are computed from the provided training graph, without adding held-out links. The model scores `Yes` and `No` for whether the source cites the candidate target, and their score difference determines the candidate ranking. SAIL uses the corresponding corrected scores. All methods share the official candidate groups and order, and MRR is computed with the OGB evaluator, including its treatment of tied scores.

Graph classification. For ogbg-molhiv, each input contains the complete molecular graph, with atoms represented by their identifiers and the first four categorical atom features supplied by OGB. The answer candidates are `No` and `Yes`, corresponding to the inactive and active labels. Input size is controlled through the neighborhood and feature limits above, without additional edge removal or token-level truncation. Paired methods receive the same underlying graph information and answer candidates, while presentation-alignment methods may change their ordering or encoding.

D.3 LLMs and Baseline Methods

We use Qwen3-4B and Qwen3-8B (Yang et al., 2025) in the main results. Cross-model evaluation additionally uses Llama-3.2-1B/3B (Grattafiori et al., 2024), Llama-2-7B/13B, Qwen2.5-0.5B/3B/7B/14B, and Gemma-3-1B/4B/12B (Team et al., 2025) on GraphMACC, with all also evaluated on the native benchmarks (Appendix G.1). Each backbone has its own trained SAIL module, and all baselines evaluated on that backbone use the same LLM checkpoint and tokenizer.

Our baselines represent the four families of presentation-alignment methods in Table 1, using the original LLM as the reference.

Graph Linearization. Graph Linearization (Xypolopoulos et al., 2024) orders graph descriptions using node importance, measured by degree, core number, or PageRank. The original method visits nodes in descending score order and emits their incident edges, with optional node relabeling to align identifiers with this order. We apply this ordering principle to the node blocks in our serializations, preserving their attributes, connectivity, and query targets. The selected variants are PageRank for Qwen3-4B and degree for Qwen3-8B, without node relabeling. GraphMACC resolves ties using its seeded ordering, while native-task inputs use node identifiers to break ties.

Graph-KV. Graph-KV (Wang et al., 2026) controls cross-block visibility through KV-cache routing: source blocks are encoded independently, and each target block reads selected source caches before the query reads the resulting representations. Source blocks share positional ranges, reducing the dependence on their order in a concatenated sequence. We adapt these cache operations to the common backbone, with GraphMACC routing groups defined by the queried nodes and their one-hop neighborhoods. For native node and graph tasks, all other visible node blocks feed an anchor block, chosen as the target node or the node with the lexicographically smallest identifier, respectively. For Citation2, the encoded blocks cover both endpoints and their sampled neighbors, and the query reads the resulting caches.

GaLA. GaLA (Loveland et al., 2026) adds a graph-distance bias to attention scores before softmax while retaining the backbone's original RoPE computation. Tokens are mapped to graph nodes, and pairs belonging to distinct connected nodes receive a bias inversely proportional to their shortest-path distance, with zero bias otherwise. We use the entropy-based variant, which scales each head's bias by the square root of its attention entropy normalized by the maximum entropy within that layer. The bias is applied in early decoder layers, with head weights estimated on GraphMACC training inputs and the overall strength selected before test evaluation.

SLASH. SLASH (Liu et al., 2026) reduces attention concentrated on the initial sink token at intervention locations identified from graph inputs. Its offline selection combines attention entropy with

structural concentration in token regions describing edges from the same source node. At inference, the sink weight is multiplied by a retained fraction γ, and the released attention is distributed proportionally across the remaining visible tokens, preserving their relative weights. We use the official Qwen attention update, with intervention locations and γ calibrated on the GRAPHMACC training partition and fixed for evaluation.

Supervised comparisons. The presentation-alignment methods above modify graph inputs or attention and use the frozen LLM's existing prediction head, without training an additional prediction module. Compared with these methods, SAIL provides a simple instantiation of utilization alignment by accumulating occurrence-level trajectory messages through TrajAgg, which introduces no trainable parameters. QueryInt then learns to integrate this accumulated state with query and candidate information to correct the original candidate scores. To examine complementarity, we combine SAIL with each presentation-alignment method by keeping its presentation operation fixed and training QueryInt on the resulting representations. Additionally, we examine the contribution of structural information alongside learned prediction through the *Response only* ablation, which retains a trained prediction module but removes the accumulated structural input. Its lower overall accuracy on GRAPHMACC supports the contribution of that input (Table 5, Appendix F.3).

D.4 SAIL TRAINING AND STATISTICAL REPORTING

Training data. During the training of SAIL, we train a shared QueryInt module on the training data from node-, edge-, and graph-level tasks for each backbone, assigning equal sampling weight to each task–dataset group. The LLM remains frozen, and QueryInt uses the architecture and objective in Appendices C.4 and C.5. Within the five training graphs per topology–degree–count combination in GRAPHMACC, three are used for fitting and two for model selection, with all inputs derived from the same graph kept together. Native tasks use the training and validation partitions specified in Appendix D.2, and test examples are excluded from fitting and model selection. For ogbl-citation2, considering the limitation of computation resource, we randomly select 1,000 positive edges from the official training partition and pair each with 1,000 fixed negative targets sampled from other nodes not linked to the source in the training graph.

Configuration and optimization. Table 16 lists the configuration for Qwen3-4B and Qwen3-8B. The structural scale a_T is estimated from fitting examples and held fixed, while the affine normalization parameters, input projections, and residual head are trained. We evaluate the checkpoint after each epoch using the mean validation score across task–dataset groups, with accuracy for classification and MRR for link ranking. If the mean score on native tasks falls below that of the original LLM, the selection score is reduced by four times this mean decrease, with all scores expressed on the 0–1 scale. We retain the checkpoint with the highest selection score and stop when it has not improved for the number of epochs listed in the table.

Repeated evaluation. For GRAPHMACC, the training set, selected configuration, and trained checkpoint remain fixed across the three disjoint test partitions defined in Appendix A.2. The same protocol applies to the combinations with presentation alignment and the ablation variants. For native benchmarks, we repeat QueryInt training using five fixed random seeds on the same data partitions and report the mean and sample standard deviation across these runs, and combinations with presentation alignment follow the same protocol. Training-free baselines on native benchmarks are deterministic under the fixed evaluation setup, so we report their scores on the complete test set without standard deviations.

D.5 IMPLEMENTATION AND COMPUTATIONAL RESOURCES

Implementation. We implement SAIL with Python 3.11, PyTorch 2.7.1 (CUDA 12.8), and Transformers 4.51.3, using eight NVIDIA A40 GPUs. The frozen backbone uses bfloat16, while QueryInt and its cached inputs use float32. For each fixed backbone and presentation setting, we cache the QueryInt inputs and reuse them across training epochs and random seeds.

Trainable parameters. With the configuration in Appendix D.4, SAIL adds the QueryInt parameters listed in Table 17, while TrajAgg introduces no trainable parameters.

Table 16: SAIL configuration for Qwen3-4B and Qwen3-8B. Layer and head indices are zero-based.

Setting	Value
Trajectory extraction	
Extraction layer index	21
Qwen3-4B head indices	2, 15, 25, 11, 13, 26, 3, 21
Qwen3-8B head indices	3, 31, 1, 13, 11, 26, 2, 0
QueryInt	
Projection width d_R	64
MLP hidden width	128
Correction bound $\Delta_{\max}$	32
Training	
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	10^{-4}
Batch size	128
Maximum epochs	120
Early-stopping patience	15 epochs
Correction penalty λ_{Δ}	10^{-4}
Preservation penalty λ_{pres}	0.5

Table 17: Trainable parameters added by SAIL.

Backbone	Trainable parameters
Qwen3-4B	1,063,816
Qwen3-8B	1,672,072

Training and evaluation time. Table 18 reports the approximate time required for each native dataset from feature extraction through training and complete test evaluation on eight A40 GPUs. The calculation includes feature extraction for the training, validation, and test sets, QueryInt optimization, validation after each epoch, and final test scoring for one seed under the 120-epoch budget in Appendix D.4. These times are extrapolated from measured GPU throughput, assuming the workload is evenly divided across the eight GPUs. Features are computed once and reused across runs, while shared QueryInt optimization time is allocated according to the task–dataset sampling weights. On the same basis, the global training on GRAPHMACC and the native datasets followed by complete evaluation on the six native test sets requires approximately 169 hours for Qwen3-4B and 247 hours for Qwen3-8B.

GPU memory. Peak GPU memory during feature extraction is approximately 21.8 GiB for Qwen3-4B and 26.6 GiB for Qwen3-8B, measured on A40 GPUs.

Table 18: Estimated runtime (hours) for SAIL training and complete test evaluation on eight A40 GPUs, extrapolated from measured throughput.

Dataset	Qwen3-4B	Qwen3-8B
Node classification		
Cora	0.17	0.23
PubMed	0.17	0.23
WikiCS	0.49	0.66
ogbn-arxiv	3.4	5.0
Link prediction		
ogbl-citation2	164	239
Graph classification		
ogbg-molhiv	0.42	0.56

E PREDICTIVE PERFORMANCE AND COMPLEMENTARITY

The main experiments and combinations with presentation-alignment methods follow the protocols in Appendix D, while FT and ICL use smaller training and evaluation budgets as described in Appendix E.3.

E.1 PER-BENCHMARK RESULTS

We report per-benchmark scores and, where applicable, sample standard deviations in Tables 19–25, following the protocol in Appendix D.4. All scores use a 0–100 scale, and the Δ_{orig} columns in the native result tables report gains over the original LLM in metric points. Gains are computed before rounding as the method score minus the original LLM score, except for G, where we use $G_{\text{orig}} - G_{\text{Method}}$ so that positive values indicate improvement.

Table 19: GRAPHMACC scores are reported as mean ± SD.

Method	$\text{Acc}_E \uparrow$	$U \uparrow$	$G \downarrow$
Qwen3-4B			
Original LLM	<u>84.63</u> ± *0.52*	42.29 ± *0.40*	42.34 ± *0.90*
Graph Linearization	83.72 ± *3.34*	<u>46.41</u> ± *0.58*	<u>37.31</u> ± *3.89*
Graph-KV	59.19 ± *0.45*	16.59 ± *1.26*	42.60 ± *1.69*
GALA	82.11 ± *0.40*	41.94 ± *0.45*	40.17 ± *0.62*
SLASH	84.45 ± *0.64*	41.61 ± *0.20*	42.84 ± *0.80*
SAIL	**84.99** ± *0.60*	**51.62** ± *0.45*	**33.37** ± *0.78*
Qwen3-8B			
Original LLM	72.78 ± *0.35*	47.62 ± *0.17*	25.16 ± *0.25*
Graph Linearization	<u>76.15</u> ± *0.39*	<u>50.70</u> ± *0.72*	25.45 ± *0.38*
Graph-KV	62.94 ± *1.17*	27.19 ± *0.29*	35.75 ± *0.45*
GALA	71.42 ± *0.28*	46.93 ± *0.21*	<u>24.49</u> ± *0.16*
SLASH	74.45 ± *1.50*	47.37 ± *1.21*	27.08 ± *1.42*
SAIL	**77.61** ± *1.19*	**57.92** ± *0.30*	**19.68** ± *0.96*

‘‘‘

Table 20: We report Cora scores and gains. Dashes mark deterministic baselines without SD.

Method	Acc.↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	32.00 —	*0.00*
Graph Linearization	30.20 —	*-1.80*
Graph-KV	18.60 —	*-13.40*
GALA	**32.30** —	*+0.30*
SLASH	29.70 —	*-2.30*
SAIL	30.20 ± *2.57*	*-1.80*
Qwen3-8B		
Original LLM	13.90 —	*0.00*
Graph Linearization	13.80 —	*-0.10*
Graph-KV	10.60 —	*-3.30*
GALA	13.80 —	*-0.10*
SLASH	13.60 —	*-0.30*
SAIL	**15.44** ± *0.32*	*+1.54*

Table 21: We report PubMed scores and gains. Dashes mark deterministic baselines without SD.

Method	Acc.↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	34.10 —	*0.00*
Graph Linearization	**38.70** —	*+4.60*
Graph-KV	**38.70** —	*+4.60*
GALA	35.10 —	*+1.00*
SLASH	31.90 —	*-2.20*
SAIL	35.66 ± *3.76*	*+1.56*
Qwen3-8B		
Original LLM	39.50 —	*0.00*
Graph Linearization	**41.70** —	*+2.20*
Graph-KV	38.10 —	*-1.40*
GALA	39.60 —	*+0.10*
SLASH	38.90 —	*-0.60*
SAIL	37.68 ± *2.82*	*-1.82*

Table 22: We report WikiCS scores and gains. Dashes mark deterministic baselines without SD.

Method	Acc.↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	14.98 —	*0.00*
Graph Linearization	14.09 —	*-0.89*
Graph-KV	13.99 —	*-0.99*
GALA	14.88 —	*-0.10*
SLASH	7.88 —	*-7.10*
SAIL	**19.71** ± *2.04*	*+4.72*
Qwen3-8B		
Original LLM	20.35 —	*0.00*
Graph Linearization	19.51 —	*-0.84*
Graph-KV	7.83 —	*-12.52*
GALA	20.47 —	*+0.12*
SLASH	20.23 —	*-0.12*
SAIL	**22.65** ± *1.51*	*+2.30*

Table 23: We report Arxiv scores and gains. Dashes mark deterministic baselines without SD.

Method	Acc.↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	11.02 —	*0.00*
Graph Linearization	9.26 —	*-1.76*
Graph-KV	9.97 —	*-1.05*
GALA	10.63 —	*-0.39*
SLASH	**21.16** —	*+10.14*
SAIL	18.58 ± *1.75*	*+7.56*
Qwen3-8B		
Original LLM	15.26 —	*0.00*
Graph Linearization	15.33 —	*+0.07*
Graph-KV	4.31 —	*-10.95*
GALA	14.89 —	*-0.37*
SLASH	12.72 —	*-2.54*
SAIL	**18.55** ± *0.99*	*+3.28*

Table 24: We report Citation2 scores and gains. Dashes mark deterministic baselines without SD.

Method	MRR↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	0.82 —	*0.00*
Graph Linearization	0.86 —	*+0.04*
Graph-KV	0.59 —	*-0.23*
GALA	0.67 —	*-0.15*
SLASH	0.99 —	*+0.17*
SAIL	**2.91** ± *0.72*	*+2.09*
Qwen3-8B		
Original LLM	0.63 —	*0.00*
Graph Linearization	0.51 —	*-0.12*
Graph-KV	0.87 —	*+0.24*
GALA	0.60 —	*-0.03*
SLASH	0.66 —	*+0.02*
SAIL	**4.06** ± *0.54*	*+3.43*

Table 25: We report MolHIV scores and gains. Dashes mark deterministic baselines without SD.

Method	AUC↑	Δ_{orig} ↑
Qwen3-4B		
Original LLM	54.50 —	*0.00*
Graph Linearization	56.67 —	*+2.17*
Graph-KV	45.33 —	*-9.17*
GALA	53.72 —	*-0.78*
SLASH	59.59 —	*+5.09*
SAIL	**60.05** ± *1.30*	*+5.55*
Qwen3-8B		
Original LLM	54.86 —	*0.00*
Graph Linearization	**63.84** —	*+8.97*
Graph-KV	62.55 —	*+7.69*
GALA	54.33 —	*-0.53*
SLASH	53.35 —	*-1.51*
SAIL	54.83 ± *0.25*	*-0.03*

“‘

Within each backbone and benchmark, bold and underlined values mark the best and second-best distinct scores, with ties sharing the same rank. Under this convention, SAIL ranks first or second in nine of the twelve native benchmark settings. The gains span node, edge, and graph prediction, extending the benefits of utilization alignment beyond the controlled diagnostic. On GRAPHMACC, utilization improves more than recognition on both backbones, showing that SAIL strengthens the use of structural information beyond presence detection. Together, these results support structural utilization as an effective optimization target for serialization-based graph prediction.

E.2 COMBINATIONS WITH PRESENTATION ALIGNMENT

For each combination, we apply the presentation-alignment method from Appendix D.3 before extracting structural trajectories from the frozen LLM. TrajAgg accumulates occurrence-level trajectory messages, and we train QueryInt on the resulting representations with the presentation operation held fixed, following the protocol in Appendix D.4. For each backbone, the combined and stan-

dalone methods use the same evaluation queries and candidate sets, with standalone scores reported in Appendix E.1.

Let B denote a presentation-alignment method and M the evaluation score, using U for GRAPHMACC and the corresponding native metric for each real dataset. On the 0–100 score scale, the gain over the stronger standalone method is

$$\Delta_{\text{Best}} = M(B + \text{SAIL}) - \max\{M(B), M(\text{SAIL})\}, \tag{57}$$

where $B + \text{SAIL}$ denotes the combined method. We also report gains over each standalone component,

$$\begin{aligned} \Delta_B &= M(B + \text{SAIL}) - M(B), \\ \Delta_{\text{SAIL}} &= M(B + \text{SAIL}) - M(\text{SAIL}). \end{aligned} \tag{58}$$

For each backbone and presentation-alignment method, Table 3 averages Δ_{Best} equally over GRAPHMACC and the six native benchmarks, including zero and negative gains, and counts the benchmarks with strictly positive gains. The metric-point average gives each benchmark the same weight regardless of its sample size. Table 30 groups these gains by task, averaging the four node-classification datasets and reporting GRAPHMACC, link prediction, and graph classification separately. Win, tie, and loss counts record positive, zero, and negative Δ_{Best}, respectively. Tables 26–29 report the standalone and combined scores and the three gains for each benchmark. The additional GRAPHMACC Acc_E and G rows describe changes in recognition and the gap without entering the seven-benchmark average, and gains for G use the reverse difference so that reductions are positive.

Table 26: Graph Linearization (B) combined with SAIL. Gains are in metric points. A dash indicates that native baseline SDs are not reported.

Dataset	Metric	B	SAIL	$B + \text{SAIL}$	Δ_B	Δ_{SAIL}	Δ_{Best}
Qwen3-4B							
GRAPHMACC	$G \downarrow$	37.31 ± *3.89*	33.37 ± *0.78*	**25.76** ± *2.66*	*+11.55*	*+7.61*	*+7.61*
	$\text{Acc}_E \uparrow$	83.72 ± *3.34*	84.99 ± *0.60*	**86.04** ± *3.01*	*+2.32*	*+1.05*	*+1.05*
	$U \uparrow$	46.41 ± *0.58*	51.62 ± *0.45*	**60.29** ± *0.47*	*+13.88*	*+8.67*	*+8.67*
Cora	Acc.↑	30.20 —	30.20 ± *2.57*	**32.14** ± *2.80*	*+1.94*	*+1.94*	*+1.94*
PubMed	Acc.↑	38.70 —	35.66 ± *3.76*	**50.62** ± *2.71*	*+11.92*	*+14.95*	*+11.92*
WikiCS	Acc.↑	14.09 —	19.71 ± *2.04*	**25.01** ± *1.72*	*+10.92*	*+5.30*	*+5.30*
Arxiv	Acc.↑	9.26 —	18.58 ± *1.75*	**20.38** ± *1.97*	*+11.12*	*+1.80*	*+1.80*
Citation2	MRR↑	0.86 —	2.91 ± *0.72*	**9.73** ± *0.51*	*+8.87*	*+6.82*	*+6.82*
MolHIV	AUC↑	56.67 —	60.05 ± *1.30*	**63.85** ± *1.93*	*+7.18*	*+3.80*	*+3.80*
Qwen3-8B							
GRAPHMACC	$G \downarrow$	25.45 ± *0.38*	**19.68** ± *0.96*	25.58 ± *0.96*	*−0.13*	*−5.90*	*−5.90*
	$\text{Acc}_E \uparrow$	76.15 ± *0.39*	77.61 ± *1.19*	**84.38** ± *0.59*	*+8.23*	*+6.77*	*+6.77*
	$U \uparrow$	50.70 ± *0.72*	57.92 ± *0.30*	**58.80** ± *0.91*	*+8.10*	*+0.88*	*+0.88*
Cora	Acc.↑	13.80 —	15.44 ± *0.32*	**20.14** ± *0.61*	*+6.34*	*+4.70*	*+4.70*
PubMed	Acc.↑	41.70 —	37.68 ± *2.82*	**52.78** ± *1.85*	*+11.08*	*+15.10*	*+11.08*
WikiCS	Acc.↑	19.51 —	22.65 ± *1.51*	**26.68** ± *1.87*	*+7.17*	*+4.03*	*+4.03*
Arxiv	Acc.↑	15.33 —	18.55 ± *0.99*	**21.95** ± *0.99*	*+6.62*	*+3.40*	*+3.40*
Citation2	MRR↑	0.51 —	4.06 ± *0.54*	**9.09** ± *0.63*	*+8.58*	*+5.03*	*+5.03*
MolHIV	AUC↑	63.84 —	54.83 ± *0.25*	**68.10** ± *0.44*	*+4.26*	*+13.27*	*+4.26*

Table 27: Graph-KV (B) combined with SAIL. Gains are in metric points. A dash indicates that native baseline SDs are not reported.

Dataset	Metric	B	SAIL	$B + \text{SAIL}$	Δ_B	Δ_{SAIL}	Δ_{Best}
Qwen3-4B							
	$G \downarrow$	42.60 ± *1.69*	33.37 ± *0.78*	**32.20** ± *1.69*	*+10.40*	*+1.17*	*+1.17*
GRAPHMACC	$\text{Acc}_E \uparrow$	59.19 ± *0.45*	84.99 ± *0.60*	**87.62** ± *0.88*	*+28.43*	*+2.63*	*+2.63*
	$U \uparrow$	16.59 ± *1.26*	51.62 ± *0.45*	**55.42** ± *1.61*	*+38.83*	*+3.80*	*+3.80*
Cora	Acc.↑	18.60 —	30.20 ± *2.57*	**35.96** ± *3.35*	*+17.36*	*+5.76*	*+5.76*
PubMed	Acc.↑	38.70 —	35.66 ± *3.76*	**50.70** ± *1.98*	*+12.00*	*+15.03*	*+12.00*
WikiCS	Acc.↑	13.99 —	19.71 ± *2.04*	**25.44** ± *2.03*	*+11.45*	*+5.73*	*+5.73*
Arxiv	Acc.↑	9.97 —	18.58 ± *1.75*	**24.23** ± *4.20*	*+14.26*	*+5.65*	*+5.65*
Citation2	MRR↑	0.59 —	2.91 ± *0.72*	**9.34** ± *0.59*	*+8.75*	*+6.43*	*+6.43*
MolHIV	AUC↑	45.33 —	60.05 ± *1.30*	**63.90** ± *7.10*	*+18.57*	*+3.85*	*+3.85*
Qwen3-8B							
	$G \downarrow$	35.75 ± *0.45*	**19.68** ± *0.96*	21.52 ± *1.11*	*+14.23*	*−1.84*	*−1.84*
GRAPHMACC	$\text{Acc}_E \uparrow$	62.94 ± *1.17*	77.61 ± *1.19*	**85.69** ± *1.42*	*+22.75*	*+8.08*	*+8.08*
	$U \uparrow$	27.19 ± *0.29*	57.92 ± *0.30*	**64.17** ± *1.23*	*+36.98*	*+6.25*	*+6.25*
Cora	Acc.↑	10.60 —	15.44 ± *0.32*	**25.44** ± *0.37*	*+14.84*	*+10.00*	*+10.00*
PubMed	Acc.↑	38.10 —	37.68 ± *2.82*	**52.86** ± *4.45*	*+14.76*	*+15.18*	*+14.76*
WikiCS	Acc.↑	7.83 —	22.65 ± *1.51*	**29.45** ± *5.38*	*+21.62*	*+6.80*	*+6.80*
Arxiv	Acc.↑	4.31 —	18.55 ± *0.99*	**20.55** ± *1.70*	*+16.24*	*+2.00*	*+2.00*
Citation2	MRR↑	0.87 —	4.06 ± *0.54*	**9.17** ± *0.16*	*+8.30*	*+5.11*	*+5.11*
MolHIV	AUC↑	62.55 —	54.83 ± *0.25*	**64.80** ± *3.82*	*+2.25*	*+9.97*	*+2.25*

Table 28: GaLA (B) combined with SAIL. Gains are in metric points. A dash indicates that native baseline SDs are not reported.

Dataset	Metric	B	SAIL	$B + \text{SAIL}$	Δ_B	Δ_{SAIL}	Δ_{Best}
Qwen3-4B							
	$G \downarrow$	40.17 ± *0.62*	33.37 ± *0.78*	**27.53** ± *2.27*	*+12.64*	*+5.84*	*+5.84*
GRAPHMACC	$\text{Acc}_E \uparrow$	82.11 ± *0.40*	**84.99** ± *0.60*	84.59 ± *3.51*	*+2.48*	*−0.40*	*−0.40*
	$U \uparrow$	41.94 ± *0.45*	51.62 ± *0.45*	**57.07** ± *1.30*	*+15.13*	*+5.45*	*+5.45*
Cora	Acc.↑	**32.30** —	30.20 ± *2.57*	28.12 ± *2.20*	*−4.18*	*−2.08*	*−4.18*
PubMed	Acc.↑	35.10 —	35.67 ± *3.76*	**46.38** ± *3.30*	*+11.28*	*+10.71*	*+10.71*
WikiCS	Acc.↑	14.88 —	19.71 ± *2.04*	**22.20** ± *1.87*	*+7.32*	*+2.49*	*+2.49*
Arxiv	Acc.↑	10.63 —	18.58 ± *1.75*	**19.12** ± *1.86*	*+8.49*	*+0.54*	*+0.54*
Citation2	MRR↑	0.67 —	**2.91** ± *0.72*	2.74 ± *0.37*	*+2.07*	*−0.17*	*−0.17*
MolHIV	AUC↑	53.72 —	60.05 ± *1.30*	**60.43** ± *1.41*	*+6.71*	*+0.38*	*+0.38*
Qwen3-8B							
	$G \downarrow$	24.49 ± *0.16*	19.68 ± *0.96*	**12.40** ± *0.35*	*+12.09*	*+7.28*	*+7.28*
GRAPHMACC	$\text{Acc}_E \uparrow$	71.42 ± *0.28*	**77.61** ± *1.19*	77.51 ± *1.28*	*+6.09*	*−0.10*	*−0.10*
	$U \uparrow$	46.93 ± *0.21*	57.92 ± *0.30*	**65.11** ± *0.90*	*+18.18*	*+7.19*	*+7.19*
Cora	Acc.↑	13.80 —	15.44 ± *0.32*	**18.94** ± *0.75*	*+5.14*	*+3.50*	*+3.50*
PubMed	Acc.↑	39.60 —	37.68 ± *2.82*	**43.78** ± *2.36*	*+4.18*	*+6.10*	*+4.18*
WikiCS	Acc.↑	20.47 —	**22.65** ± *1.51*	22.42 ± *1.69*	*+1.95*	*−0.23*	*−0.23*
Arxiv	Acc.↑	14.89 —	18.55 ± *0.99*	**19.34** ± *1.02*	*+4.45*	*+0.79*	*+0.79*
Citation2	MRR↑	0.60 —	**4.06** ± *0.54*	0.96 ± *0.13*	*+0.36*	*−3.10*	*−3.10*
MolHIV	AUC↑	54.33 —	54.83 ± *0.25*	**59.42** ± *0.26*	*+5.09*	*+4.59*	*+4.59*

Table 29: SLASH (B) combined with SAIL. Gains are in metric points. A dash indicates that native baseline SDs are not reported.

Dataset	Metric	B	SAIL	$B + \text{SAIL}$	Δ_B	Δ_{SAIL}	Δ_{Best}
Qwen3-4B							
GRAPHMACC	$G \downarrow$	42.84 ± *0.80*	**33.37** ± *0.78*	34.93 ± *1.74*	*+7.91*	*−1.56*	*−1.56*
	$\text{Acc}_E \uparrow$	84.45 ± *0.64*	**84.99** ± *0.60*	84.74 ± *1.63*	*+0.29*	*−0.25*	*−0.25*
	$U \uparrow$	41.61 ± *0.20*	**51.62** ± *0.45*	49.81 ± *2.30*	*+8.20*	*−1.81*	*−1.81*
Cora	Acc.↑	29.70 —	30.20 ± *2.57*	**30.44** ± *1.71*	*+0.74*	*+0.24*	*+0.24*
PubMed	Acc.↑	31.90 —	35.66 ± *3.76*	**35.90** ± *6.07*	*+4.00*	*+0.23*	*+0.23*
WikiCS	Acc.↑	7.88 —	19.71 ± *2.04*	**21.01** ± *1.20*	*+13.13*	*+1.30*	*+1.30*
Arxiv	Acc.↑	21.16 —	18.58 ± *1.75*	**22.16** ± *1.42*	*+1.00*	*+3.58*	*+1.00*
Citation2	MRR↑	0.99 —	2.91 ± *0.72*	**9.13** ± *1.03*	*+8.14*	*+6.22*	*+6.22*
MolHIV	AUC↑	59.59 —	60.05 ± *1.30*	**60.40** ± *0.63*	*+0.81*	*+0.35*	*+0.35*
Qwen3-8B							
GRAPHMACC	$G \downarrow$	27.08 ± *1.42*	**19.68** ± *0.96*	24.02 ± *1.68*	*+3.06*	*−4.34*	*−4.34*
	$\text{Acc}_E \uparrow$	74.45 ± *1.50*	77.61 ± *1.19*	**78.10** ± *0.81*	*+3.65*	*+0.49*	*+0.49*
	$U \uparrow$	47.37 ± *1.21*	**57.92** ± *0.30*	54.08 ± *1.99*	*+6.71*	*−3.84*	*−3.84*
Cora	Acc.↑	13.60 —	15.44 ± *0.32*	**16.02** ± *0.83*	*+2.42*	*+0.58*	*+0.58*
PubMed	Acc.↑	38.90 —	37.68 ± *2.82*	**44.04** ± *3.57*	*+5.14*	*+6.36*	*+5.14*
WikiCS	Acc.↑	20.23 —	22.65 ± *1.51*	**24.08** ± *4.87*	*+3.85*	*+1.43*	*+1.43*
Arxiv	Acc.↑	12.72 —	18.55 ± *0.99*	**19.60** ± *0.87*	*+6.88*	*+1.05*	*+1.05*
Citation2	MRR↑	0.66 —	4.06 ± *0.54*	**9.12** ± *0.92*	*+8.46*	*+5.06*	*+5.06*
MolHIV	AUC↑	53.35 —	54.83 ± *0.25*	**55.25** ± *1.36*	*+1.90*	*+0.42*	*+0.42*

Table 30: We report average gains from combining SAIL with presentation alignment, by task group and across all seven benchmarks.

Method	GRAPHMACC	Node	Link	Graph	All	W/T/L
Qwen3-4B						
Graph Linearization	*+8.67*	*+5.24*	*+6.82*	*+3.80*	*+5.75*	7/0/0
Graph-KV	*+3.80*	*+7.29*	*+6.43*	*+3.85*	*+6.17*	7/0/0
GaLA	*+5.45*	*+2.39*	*-0.17*	*+0.38*	*+2.17*	5/0/2
SLASH	*-1.81*	*+0.69*	*+6.22*	*+0.35*	*+1.08*	6/0/1
Qwen3-8B						
Graph Linearization	*+0.88*	*+5.81*	*+5.03*	*+4.26*	*+4.77*	7/0/0
Graph-KV	*+6.25*	*+8.39*	*+5.11*	*+2.25*	*+6.74*	7/0/0
GaLA	*+7.19*	*+2.06*	*-3.10*	*+4.59*	*+2.42*	5/0/2
SLASH	*-3.84*	*+2.06*	*+5.06*	*+0.42*	*+1.41*	6/0/1

Within each row, bold and underlined scores mark the best and second-best distinct values. Combining SAIL with presentation alignment outperforms the stronger standalone method in most settings on both backbones, with positive average gains across all four presentation-alignment methods. The additional gains after graph inputs or attention have been adjusted indicate that improving structural presentation still leaves room to improve how the resulting information is used. These results support presentation and utilization alignment as complementary intervention targets for improving serialization-based graph prediction.

E.3 COMBINATIONS UNDER FINE-TUNING AND IN-CONTEXT LEARNING

Evaluation sets. We evaluate FT and ICL with Graph Linearization, Graph-KV, GaLA, and SLASH on Qwen3-4B and Qwen3-8B, following the repeated-evaluation protocol in Appendix D.4 with the smaller sample budgets in Table 31. For GRAPHMACC, we keep the training set and trained checkpoint fixed across three mutually disjoint test partitions, compute metrics separately within each degree, and average equally across degrees. For native benchmarks, trainable methods

use five training seeds on fixed data partitions, while training-free baselines are evaluated once on the fixed lightweight evaluation set. We report single scores for these baselines and means with sample standard deviations across the five training runs or three GRAPHMACC test partitions. Within each comparison, all methods share the evaluation pool and candidate sets, with evaluation queries excluded from training, model selection, and demonstration retrieval. Each GRAPHMACC graph contributes all three matched tasks and both serializations, while each Citation2 query retains one positive and 1,000 negative candidates.

Table 31: Evaluation set sizes for FT and ICL.

Dataset	Evaluation size
GRAPHMACC	198 graphs, 1,188 serialized prompts
Cora	340 nodes
PubMed	260 nodes
WikiCS	600 nodes
ogbn-arxiv	400 nodes
ogbl-citation2	100 queries
ogbg-molhiv	600 graphs

Fine-tuning. For each backbone and run, we train one shared LoRA adapter on the pooled native training data and reuse it across the four presentation-alignment methods. The training pool contains up to 1,000 examples per dataset, and each of three epochs samples 1,000 examples by balancing task scopes, datasets, and labels. LoRA updates the query, key, value, and output projections with rank 8, scaling parameter $\alpha = 16$, and dropout 0.05, while the original LLM weights remain frozen. We use AdamW with learning rate 2×10^{-4}, zero weight decay, a microbatch size of 1, and 32 gradient accumulation steps, with training sequences capped at 16,384 tokens for Qwen3-4B and 8,192 for Qwen3-8B. After fine-tuning, we freeze the adapter and extract the structural trajectories, query representations, candidate representations, and original candidate scores under each fixed presentation-alignment method. We then train QueryInt using the objective and validation rule in Appendix D.4.

In-context learning. ICL prepends four labeled demonstrations retrieved from the corresponding task's training examples using the dataset, task, structural scope, degree, and candidate count, with deterministic tie-breaking. Retrieval excludes the query itself and examples sharing its target entities or graph records, and all ICL configurations for the same query use the same demonstrations. We keep the LLM frozen, extract SAIL inputs from the resulting prompts under each presentation-alignment method, and fit QueryInt using the same procedure as above.

Joint gains. For adaptation method $A \in \{\mathrm{FT}, \mathrm{ICL}\}$ and presentation-alignment method B, we compare the three-component system with all three two-component alternatives:

$$\begin{aligned}\Delta_{\mathrm{joint}} = {} & M(A + B + \mathrm{SAIL}) \\ & - \max\{M(A + B), M(A + \mathrm{SAIL}), M(B + \mathrm{SAIL})\}.\end{aligned} \tag{59}$$

Here, M follows the metric definitions in Appendix E.2, and a positive gain means that the three-component system outperforms every two-component alternative on the same evaluation set. Table 4 averages these gains equally across seven benchmarks for each backbone, presentation-alignment method, and adaptation condition, including zero and negative gains. Tables 32–35 report all three pair scores and the triple score for each presentation-alignment method and benchmark under FT and ICL, together with joint gains and win, tie, or loss outcomes determined from unrounded scores. Table 36 summarizes the gains by task group and across all seven benchmarks.

Table 32: Combinations of FT or ICL (A), Graph Linearization (B), and SAIL. Scores are percentages and gains are in metric points. A dash indicates that native ICL baseline SDs are not reported.

Dataset	A	Metric	$A + B$	$A + \text{SAIL}$	$B + \text{SAIL}$	$A + B + \text{SAIL}$	Δ_{joint}	W/T/L
Qwen3-4B								
GRAPHMACC	FT	$U \uparrow$	47.49 ± *2.45*	51.76 ± *2.85*	48.29 ± *2.03*	**53.95** ± *2.54*	*+2.19*	W
	ICL	$U \uparrow$	49.28 ± *1.98*	49.56 ± *0.40*	48.29 ± *2.03*	**70.81** ± *3.98*	*+21.25*	W
Cora	FT	Acc.↑	11.47 ± *1.43*	12.06 ± *0.42*	24.41 ± *1.14*	**26.47** ± *0.75*	*+2.06*	W
	ICL	Acc.↑	15.88 —	17.65 ± *0.36*	24.41 ± *1.14*	**27.06** ± *3.21*	*+2.65*	W
PubMed	FT	Acc.↑	34.23 ± *0.38*	29.62 ± *1.02*	33.85 ± *3.68*	**40.38** ± *0.54*	*+6.15*	W
	ICL	Acc.↑	**45.00** —	43.46 ± *1.76*	33.85 ± *3.68*	39.62 ± *3.39*	*−5.38*	L
WikiCS	FT	Acc.↑	18.50 ± *1.88*	18.83 ± *0.24*	16.33 ± *2.85*	**27.83** ± *0.93*	*+9.00*	W
	ICL	Acc.↑	5.83 —	18.33 ± *0.97*	16.33 ± *2.85*	**23.67** ± *0.91*	*+5.33*	W
Arxiv	FT	Acc.↑	**11.50** ± *0.18*	4.75 ± *0.61*	9.00 ± *6.35*	**11.50** ± *1.90*	*0.00*	T
	ICL	Acc.↑	5.50 —	10.75 ± *1.76*	9.00 ± *6.35*	**11.00** ± *3.38*	*+0.25*	W
Citation2	FT	MRR↑	0.76 ± *0.53*	0.65 ± *0.28*	1.12 ± *0.82*	**2.02** ± *0.76*	*+0.91*	W
	ICL	MRR↑	3.49 —	4.80 ± *0.15*	1.12 ± *0.82*	**5.70** ± *2.83*	*+0.91*	W
MolHIV	FT	AUC↑	68.03 ± *7.92*	**73.79** ± *0.32*	44.29 ± *4.29*	71.12 ± *4.57*	*−2.67*	L
	ICL	AUC↑	**53.26** —	45.17 ± *1.68*	44.29 ± *4.29*	39.25 ± *1.65*	*−14.01*	L
Qwen3-8B								
GRAPHMACC	FT	$U \uparrow$	53.53 ± *1.15*	56.83 ± *1.74*	**63.43** ± *3.19*	58.60 ± *1.94*	*−4.83*	L
	ICL	$U \uparrow$	56.64 ± *1.65*	56.16 ± *3.16*	63.43 ± *3.19*	**71.19** ± *1.39*	*+7.76*	W
Cora	FT	Acc.↑	14.41 ± *1.19*	12.35 ± *0.59*	14.41 ± *0.81*	**27.35** ± *1.33*	*+12.94*	W
	ICL	Acc.↑	17.06 —	17.65 ± *1.38*	14.41 ± *0.81*	**27.35** ± *1.33*	*+9.71*	W
PubMed	FT	Acc.↑	**43.08** ± *0.38*	39.23 ± *2.49*	31.92 ± *2.65*	37.31 ± *1.02*	*−5.77*	L
	ICL	Acc.↑	38.08 —	**41.54** ± *1.41*	31.92 ± *2.65*	38.46 ± *5.15*	*−3.08*	L
WikiCS	FT	Acc.↑	9.83 ± *2.45*	19.67 ± *2.95*	20.00 ± *1.53*	**28.00** ± *1.29*	*+8.00*	W
	ICL	Acc.↑	15.00 —	16.50 ± *1.05*	20.00 ± *1.53*	**27.50** ± *0.93*	*+7.50*	W
Arxiv	FT	Acc.↑	1.25 ± *0.53*	1.50 ± *1.09*	12.75 ± *0.75*	**15.25** ± *0.95*	*+2.50*	W
	ICL	Acc.↑	12.75 —	9.00 ± *0.47*	12.75 ± *0.75*	**16.25** ± *2.77*	*+3.50*	W
Citation2	FT	MRR↑	0.28 ± *0.09*	**0.33** ± *0.23*	**0.33** ± *0.16*	0.27 ± *0.14*	*−0.06*	L
	ICL	MRR↑	**3.65** —	3.33 ± *1.36*	0.33 ± *0.16*	3.06 ± *2.62*	*−0.59*	L
MolHIV	FT	AUC↑	**65.50** ± *5.17*	60.10 ± *2.84*	45.64 ± *0.95*	60.11 ± *0.05*	*−5.39*	L
	ICL	AUC↑	64.79 —	**66.42** ± *1.38*	45.64 ± *0.95*	59.03 ± *3.34*	*−7.40*	L

Table 33: Combinations of FT or ICL (A), Graph-KV (B), and SAIL. Scores are percentages and gains are in metric points. A dash indicates that native ICL baseline SDs are not reported.

Dataset	A	Metric	$A + B$	$A + \text{SAIL}$	$B + \text{SAIL}$	$A + B + \text{SAIL}$	Δ_{joint}	W/T/L
Qwen3-4B								
GRAPHMACC	FT	$U \uparrow$	32.25 ± *0.91*	51.76 ± *2.85*	58.37 ± *1.54*	**70.93** ± *0.78*	*+12.56*	W
	ICL	$U \uparrow$	33.98 ± *2.11*	49.56 ± *0.40*	58.37 ± *1.54*	**73.94** ± *1.20*	*+15.57*	W
Cora	FT	Acc.↑	11.47 ± *1.30*	12.06 ± *0.42*	19.71 ± *3.83*	**27.06** ± *4.95*	*+7.35*	W
	ICL	Acc.↑	20.00 —	17.65 ± *0.36*	19.71 ± *3.83*	**27.06** ± *1.21*	*+7.06*	W
PubMed	FT	Acc.↑	26.92 ± *0.38*	29.62 ± *1.02*	31.15 ± *1.19*	**37.69** ± *3.36*	*+6.54*	W
	ICL	Acc.↑	42.31 —	**43.46** ± *1.76*	31.15 ± *1.19*	35.00 ± *0.47*	*−8.46*	L
WikiCS	FT	Acc.↑	5.17 ± *0.12*	18.83 ± *0.24*	11.50 ± *1.39*	**28.33** ± *2.08*	*+9.50*	W
	ICL	Acc.↑	11.17 —	18.33 ± *0.97*	11.50 ± *1.39*	**28.33** ± *2.81*	*+10.00*	W
Arxiv	FT	Acc.↑	1.75 ± *2.05*	4.75 ± *0.61*	6.75 ± *0.50*	**18.00** ± *2.27*	*+11.25*	W
	ICL	Acc.↑	5.50 —	10.75 ± *1.76*	6.75 ± *0.50*	**15.00** ± *2.13*	*+4.25*	W
Citation2	FT	MRR↑	**1.30** ± *1.50*	0.65 ± *0.28*	0.78 ± *0.69*	0.51 ± *0.23*	*−0.78*	L
	ICL	MRR↑	0.51 —	**4.80** ± *0.15*	0.78 ± *0.69*	4.27 ± *2.50*	*−0.53*	L
MolHIV	FT	AUC↑	38.02 ± *1.95*	**73.79** ± *0.32*	53.53 ± *2.77*	**73.79** ± *1.47*	*0.00*	T
	ICL	AUC↑	63.65 —	45.17 ± *1.68*	53.53 ± *2.77*	**72.56** ± *2.07*	*+8.91*	W
Qwen3-8B								
GRAPHMACC	FT	$U \uparrow$	32.82 ± *2.03*	56.83 ± *1.74*	66.21 ± *1.41*	**71.13** ± *2.90*	*+4.92*	W
	ICL	$U \uparrow$	33.24 ± *3.56*	56.16 ± *3.16*	66.21 ± *1.41*	**76.67** ± *2.11*	*+10.46*	W
Cora	FT	Acc.↑	13.82 ± *1.00*	12.35 ± *0.59*	13.82 ± *3.21*	**27.35** ± *1.25*	*+13.53*	W
	ICL	Acc.↑	19.71 —	17.65 ± *1.38*	13.82 ± *3.21*	**27.06** ± *0.21*	*+7.35*	W
PubMed	FT	Acc.↑	35.77 ± *0.27*	**39.23** ± *2.49*	32.69 ± *2.83*	35.00 ± *1.19*	*−4.23*	L
	ICL	Acc.↑	35.77 —	**41.54** ± *1.41*	32.69 ± *2.83*	38.46 ± *0.72*	*−3.08*	L
WikiCS	FT	Acc.↑	3.67 ± *1.71*	19.67 ± *2.95*	6.50 ± *5.10*	**29.67** ± *1.23*	*+10.00*	W
	ICL	Acc.↑	16.00 —	16.50 ± *1.05*	6.50 ± *5.10*	**26.50** ± *0.67*	*+10.00*	W
Arxiv	FT	Acc.↑	0.75 ± *0.56*	1.50 ± *1.09*	1.50 ± *0.64*	**16.50** ± *1.19*	*+15.00*	W
	ICL	Acc.↑	8.50 —	9.00 ± *0.47*	1.50 ± *0.64*	**16.50** ± *2.58*	*+7.50*	W
Citation2	FT	MRR↑	**1.77** ± *0.49*	0.33 ± *0.23*	1.39 ± *0.23*	0.95 ± *0.13*	*−0.82*	L
	ICL	MRR↑	1.02 —	**3.33** ± *1.36*	1.39 ± *0.23*	2.87 ± *2.91*	*−0.46*	L
MolHIV	FT	AUC↑	29.50 ± *2.99*	**60.10** ± *2.84*	47.78 ± *5.02*	56.71 ± *0.03*	*−3.39*	L
	ICL	AUC↑	**67.72** —	66.42 ± *1.38*	47.78 ± *5.02*	62.66 ± *3.26*	*−5.06*	L

Table 34: Combinations of FT or ICL (A), GaLA (B), and SAIL. Scores are percentages and gains are in metric points. A dash indicates that native ICL baseline SDs are not reported.

Dataset	A	Metric	$A + B$	$A +$ SAIL	$B +$ SAIL	$A + B +$ SAIL	Δ_{joint}	W/T/L
Qwen3-4B								
GRAPHMACC	FT	$U \uparrow$	40.30 ± *1.96*	51.76 ± *2.85*	59.11 ± *1.38*	**68.66** ± *2.76*	*+9.55*	W
	ICL	$U \uparrow$	48.33 ± *0.30*	49.56 ± *0.40*	59.11 ± *1.38*	**70.25** ± *1.96*	*+11.14*	W
Cora	FT	Acc.↑	11.76 ± *1.53*	12.06 ± *0.42*	25.88 ± *1.53*	**27.94** ± *0.36*	*+2.06*	W
	ICL	Acc.↑	17.06 —	17.65 ± *0.36*	25.88 ± *1.53*	**27.35** ± *1.50*	*+1.47*	W
PubMed	FT	Acc.↑	33.85 ± *0.98*	29.62 ± *1.02*	31.15 ± *0.38*	**39.23** ± *1.68*	*+5.38*	W
	ICL	Acc.↑	43.08 —	**43.46** ± *1.76*	31.15 ± *0.38*	**43.46** ± *0.67*	*0.00*	T
WikiCS	FT	Acc.↑	19.33 ± *0.42*	18.83 ± *0.24*	14.83 ± *1.22*	**28.17** ± *2.49*	*+8.83*	W
	ICL	Acc.↑	6.17 —	18.33 ± *0.97*	14.83 ± *1.22*	**28.83** ± *0.60*	*+10.50*	W
Arxiv	FT	Acc.↑	10.25 ± *0.88*	4.75 ± *0.61*	9.00 ± *1.64*	**17.00** ± *0.61*	*+6.75*	W
	ICL	Acc.↑	3.75 —	10.75 ± *1.76*	9.00 ± *1.64*	**15.00** ± *0.83*	*+4.25*	W
Citation2	FT	MRR↑	1.09 ± *0.57*	0.65 ± *0.28*	**2.74** ± *0.33*	0.89 ± *0.92*	*−1.85*	L
	ICL	MRR↑	**5.97** —	4.80 ± *0.15*	2.74 ± *0.33*	4.93 ± *3.12*	*−1.05*	L
MolHIV	FT	AUC↑	63.03 ± *0.40*	**73.79** ± *0.32*	58.25 ± *0.40*	72.46 ± *3.67*	*−1.33*	L
	ICL	AUC↑	44.04 —	45.17 ± *1.68*	**58.25** ± *0.40*	44.49 ± *1.71*	*−13.76*	L
Qwen3-8B								
GRAPHMACC	FT	$U \uparrow$	49.05 ± *3.45*	56.83 ± *1.74*	**69.20** ± *3.05*	62.57 ± *0.68*	*−6.63*	L
	ICL	$U \uparrow$	50.12 ± *1.66*	56.16 ± *3.16*	69.20 ± *3.05*	**71.67** ± *1.33*	*+2.47*	W
Cora	FT	Acc.↑	12.35 ± *1.30*	12.35 ± *0.59*	15.88 ± *0.83*	**27.35** ± *0.69*	*+11.47*	W
	ICL	Acc.↑	18.82 —	17.65 ± *1.38*	15.88 ± *0.83*	**27.35** ± *2.30*	*+8.53*	W
PubMed	FT	Acc.↑	**42.69** ± *1.61*	39.23 ± *2.49*	31.54 ± *1.30*	42.31 ± *2.59*	*−0.38*	L
	ICL	Acc.↑	39.62 —	**41.54** ± *1.41*	31.54 ± *1.30*	41.15 ± *0.61*	*−0.38*	L
WikiCS	FT	Acc.↑	9.67 ± *2.53*	19.67 ± *2.95*	21.50 ± *1.03*	**28.67** ± *2.93*	*+7.17*	W
	ICL	Acc.↑	16.00 —	16.50 ± *1.05*	21.50 ± *1.03*	**28.67** ± *0.33*	*+7.17*	W
Arxiv	FT	Acc.↑	0.75 ± *0.25*	1.50 ± *1.09*	14.75 ± *2.27*	**18.25** ± *1.77*	*+3.50*	W
	ICL	Acc.↑	12.75 —	9.00 ± *0.47*	14.75 ± *2.27*	**16.50** ± *1.30*	*+1.75*	W
Citation2	FT	MRR↑	0.35 ± *0.23*	0.33 ± *0.23*	**0.96** ± *0.33*	0.35 ± *0.18*	*−0.61*	L
	ICL	MRR↑	2.96 —	**3.33** ± *1.36*	0.96 ± *0.33*	3.29 ± *0.41*	*−0.04*	L
MolHIV	FT	AUC↑	61.08 ± *3.44*	60.10 ± *2.84*	39.49 ± *1.29*	**63.32** ± *0.13*	*+2.25*	W
	ICL	AUC↑	67.44 —	66.42 ± *1.38*	39.49 ± *1.29*	**67.78** ± *3.09*	*+0.34*	W

Table 35: Combinations of FT or ICL (A), SLASH (B), and SAIL. Scores are percentages and gains are in metric points. A dash indicates that native ICL baseline SDs are not reported.

Dataset	A	Metric	$A + B$	$A + \text{SAIL}$	$B + \text{SAIL}$	$A + B + \text{SAIL}$	Δ_{joint}	W/T/L
Qwen3-4B								
GRAPHMACC	FT	$U \uparrow$	40.86 ± *0.40*	51.76 ± *2.85*	52.02 ± *2.01*	**53.06** ± *0.42*	*+1.04*	W
	ICL	$U \uparrow$	42.25 ± *2.03*	49.56 ± *0.40*	**52.02** ± *2.01*	49.77 ± *3.99*	*−2.25*	L
Cora	FT	Acc.↑	17.65 ± *1.40*	12.06 ± *0.42*	**27.65** ± *1.57*	27.35 ± *1.10*	*−0.29*	L
	ICL	Acc.↑	18.24 —	17.65 ± *0.36*	**27.65** ± *1.57*	27.06 ± *0.62*	*−0.59*	L
PubMed	FT	Acc.↑	40.00 ± *0.61*	29.62 ± *1.02*	28.85 ± *3.22*	**41.92** ± *0.61*	*+1.92*	W
	ICL	Acc.↑	41.92 —	**43.46** ± *1.76*	28.85 ± *3.22*	36.92 ± *6.26*	*−6.54*	L
WikiCS	FT	Acc.↑	17.17 ± *0.44*	18.83 ± *0.24*	7.50 ± *2.16*	**28.50** ± *2.44*	*+9.67*	W
	ICL	Acc.↑	9.33 —	18.33 ± *0.97*	7.50 ± *2.16*	**29.67** ± *0.35*	*+11.33*	W
Arxiv	FT	Acc.↑	9.25 ± *0.18*	4.75 ± *0.61*	11.25 ± *0.43*	**15.75** ± *0.31*	*+4.50*	W
	ICL	Acc.↑	5.00 —	10.75 ± *1.76*	11.25 ± *0.43*	**15.00** ± *4.21*	*+3.75*	W
Citation2	FT	MRR↑	0.53 ± *0.21*	0.65 ± *0.28*	1.03 ± *0.44*	**1.75** ± *0.09*	*+0.72*	W
	ICL	MRR↑	4.22 —	4.80 ± *0.15*	1.03 ± *0.44*	**5.29** ± *0.89*	*+0.50*	W
MolHIV	FT	AUC↑	58.22 ± *2.84*	73.79 ± *0.32*	52.83 ± *1.56*	**79.85** ± *0.26*	*+6.06*	W
	ICL	AUC↑	34.94 —	45.17 ± *1.68*	52.83 ± *1.56*	**63.56** ± *1.23*	*+10.73*	W
Qwen3-8B								
GRAPHMACC	FT	$U \uparrow$	45.83 ± *1.12*	56.83 ± *1.74*	57.98 ± *1.70*	**58.17** ± *1.13*	*+0.19*	W
	ICL	$U \uparrow$	44.84 ± *4.93*	56.16 ± *3.16*	**57.98** ± *1.70*	57.36 ± *2.41*	*−0.62*	L
Cora	FT	Acc.↑	13.24 ± *0.75*	12.35 ± *0.59*	15.59 ± *8.24*	**27.06** ± *1.91*	*+11.47*	W
	ICL	Acc.↑	15.88 —	17.65 ± *1.38*	15.59 ± *8.24*	**27.35** ± *2.33*	*+9.71*	W
PubMed	FT	Acc.↑	**41.15** ± *2.84*	39.23 ± *2.49*	30.77 ± *1.90*	40.77 ± *7.11*	*−0.38*	L
	ICL	Acc.↑	39.23 —	**41.54** ± *1.41*	30.77 ± *1.90*	38.08 ± *1.86*	*−3.46*	L
WikiCS	FT	Acc.↑	9.83 ± *0.79*	19.67 ± *2.95*	23.83 ± *1.50*	**27.83** ± *0.31*	*+4.00*	W
	ICL	Acc.↑	17.17 —	16.50 ± *1.05*	23.83 ± *1.50*	**26.50** ± *0.62*	*+2.67*	W
Arxiv	FT	Acc.↑	1.00 ± *0.94*	1.50 ± *1.09*	13.50 ± *2.09*	**19.00** ± *2.80*	*+5.50*	W
	ICL	Acc.↑	1.00 —	9.00 ± *0.47*	**13.50** ± *2.09*	0.25 ± *0.25*	*−13.25*	L
Citation2	FT	MRR↑	0.38 ± *0.34*	0.33 ± *0.23*	0.34 ± *0.11*	**0.92** ± *0.69*	*+0.53*	W
	ICL	MRR↑	**4.05** —	3.33 ± *1.36*	0.34 ± *0.11*	3.92 ± *4.66*	*−0.13*	L
MolHIV	FT	AUC↑	61.52 ± *0.89*	60.10 ± *2.84*	51.64 ± *0.01*	**67.43** ± *1.97*	*+5.92*	W
	ICL	AUC↑	62.29 —	66.42 ± *1.38*	51.64 ± *0.01*	**72.12** ± *2.57*	*+5.69*	W

Table 36: We report average joint gains from combining SAIL with presentation alignment under FT and ICL, by task group and across all seven benchmarks.

Method	A	GRAPHMACC	Node	Link	Graph	All	W/T/L
Qwen3-4B							
Graph Linearization	FT	*+2.19*	*+4.30*	*+0.91*	*-2.67*	*+2.52*	5/1/1
	ICL	*+21.25*	*+0.71*	*+0.91*	*-14.01*	*+1.57*	5/0/2
Graph-KV	FT	*+12.56*	*+8.66*	*-0.78*	*0.00*	*+6.63*	5/1/1
	ICL	*+15.57*	*+3.21*	*-0.53*	*+8.91*	*+5.26*	5/0/2
GaLA	FT	*+9.55*	*+5.76*	*-1.85*	*-1.33*	*+4.20*	5/0/2
	ICL	*+11.14*	*+4.06*	*-1.05*	*-13.76*	*+1.79*	4/1/2
SLASH	FT	*+1.04*	*+3.95*	*+0.72*	*+6.06*	*+3.37*	6/0/1
	ICL	*-2.25*	*+1.99*	*+0.50*	*+10.73*	*+2.42*	4/0/3
Qwen3-8B							
Graph Linearization	FT	*-4.83*	*+4.42*	*-0.06*	*-5.39*	*+1.06*	3/0/4
	ICL	*+7.76*	*+4.41*	*-0.59*	*-7.40*	*+2.49*	4/0/3
Graph-KV	FT	*+4.92*	*+8.57*	*-0.82*	*-3.39*	*+5.00*	4/0/3
	ICL	*+10.46*	*+5.44*	*-0.46*	*-5.06*	*+3.82*	4/0/3
GaLA	FT	*-6.63*	*+5.44*	*-0.61*	*+2.25*	*+2.40*	4/0/3
	ICL	*+2.47*	*+4.27*	*-0.04*	*+0.34*	*+2.83*	5/0/2
SLASH	FT	*+0.19*	*+5.15*	*+0.53*	*+5.92*	*+3.89*	6/0/1
	ICL	*-0.62*	*-1.08*	*-0.13*	*+5.69*	*+0.09*	3/0/4

Within each row, bold and underlined scores mark the best and second-best distinct displayed values, with ties sharing the same rank.

The three-component system outperforms the strongest two-component alternative in most evaluated cases, with positive average joint gains across all four presentation-alignment methods under both FT and ICL. These gains show that downstream adaptation leaves room for further improvements through structural accumulation and integration, supporting utilization alignment as a complementary optimization target for serialization-based graph prediction.

F Multiplicity Separation and Ablation Details

We examine how SAIL separates adjacent counts in candidate-score space and how its components contribute to the prediction gains reported in Section 4.3.

F.1 Candidate-Score Visualization and Count Separation

For the visualizations in Figure 4, we use Qwen3-8B on GraphMACC and Qwen3-4B on Cora. Each setting contains 1,080 counting records at degree 4, with 270 records for each ground-truth count in $\{1, 2, 3, 4\}$. For each record, we compare the original candidate scores $s_{y'}^{\mathrm{LLM}}$ with the corrected scores $s_{y'}^{\mathrm{SAIL}}$ defined in Section 3.3, using the five valid count candidates $\{0, 1, 2, 3, 4\}$ for prediction and score correction.

For each record i and $X \in \{\mathrm{LLM}, \mathrm{SAIL}\}$, we center the score vector $\mathbf{s}_i^X \in \mathbb{R}^m$ by subtracting its mean across coordinates,

$$\widetilde{\mathbf{s}}_i^X = \mathbf{s}_i^X - \frac{1}{m}\left(\sum_{j=1}^{m} s_{i,j}^X\right)\mathbf{1}. \tag{60}$$

For each visualization, we fit a single two-dimensional PCA to all original and corrected vectors together, so that the two panels share the same projection. Class centroids and covariance ellipses are computed from all 1,080 records in each setting, with each ellipse showing the 60% probability contour of a Gaussian fitted to the projected class.

For the set $\mathcal{I}_c$ of records with ground-truth count c, we define the class centroid and the distance between adjacent centroids in the full centered score space as

$$\boldsymbol{\mu}_c^X = \frac{1}{|\mathcal{I}_c|}\sum_{i\in\mathcal{I}_c}\widetilde{\mathbf{s}}_i^X, \qquad D_c^X = \left\|\boldsymbol{\mu}_{c+1}^X - \boldsymbol{\mu}_c^X\right\|_2. \tag{61}$$

Figures 7 and 8 show the randomly sampled 30% subsets used in Figure 4, while Figures 9 and 10 show all 1,080 records in each setting. Before correction, adjacent-count distances decrease as the count increases on both datasets, matching the compression pattern discussed in Section 4.3. SAIL increases the distance between every adjacent count pair from 1 to 4 in both settings, improving separation in candidate-score space.

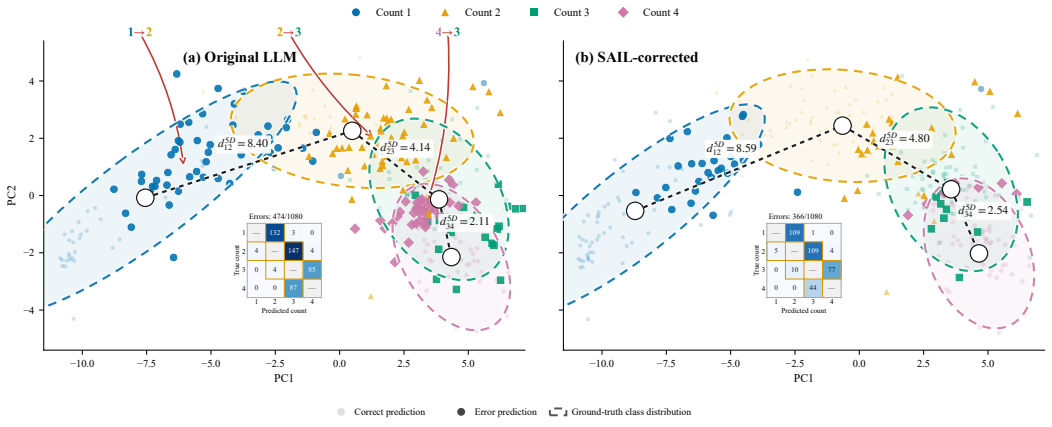


Figure 7: The panels compare candidate-score projections before and after SAIL correction on GraphMACC, reproducing the left pair of panels in Figure 4.

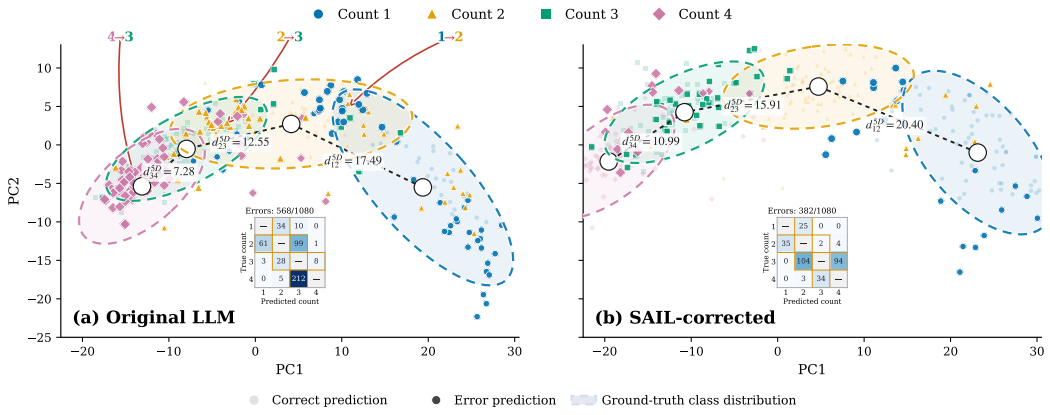


Figure 8: The panels compare candidate-score projections before and after SAIL correction for counting on Cora, reproducing the right pair of panels in Figure 4.

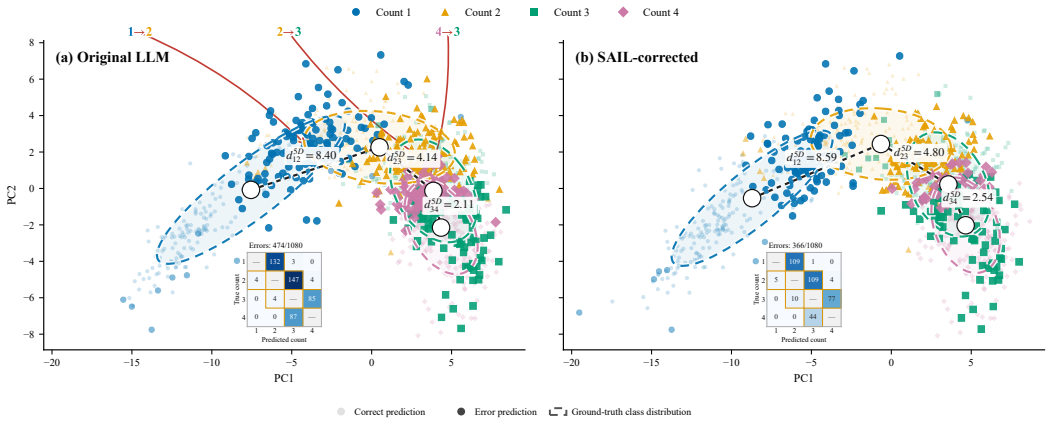


Figure 9: The panels compare candidate-score projections before and after SAIL correction on GRAPHMACC, similar to the left pair of panels in Figure 4, but using all 1,080 records.

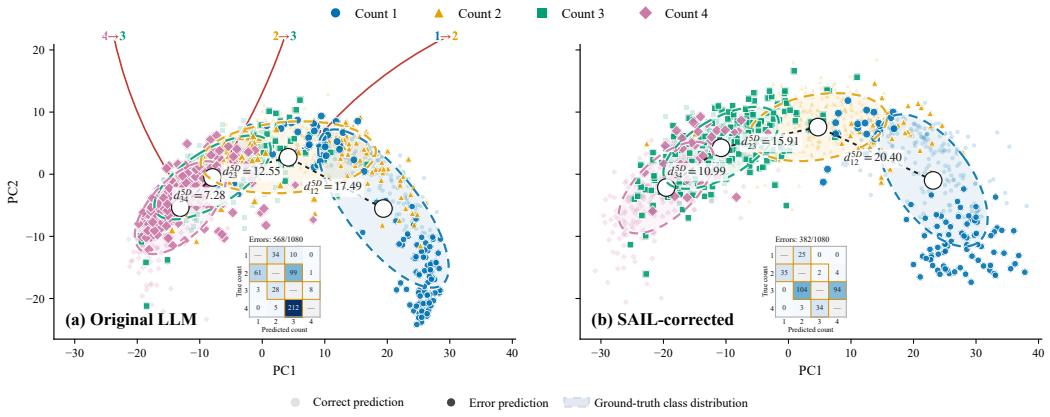


Figure 10: The panels compare candidate-score projections before and after SAIL correction for counting on Cora, similar to the right pair of panels in Figure 4, but using all 1,080 records.

F.2 Prediction Changes between Multiplicity Classes

We compare the original and corrected predictions for all 1,080 records in each setting from Appendix F.1, with 270 records per ground-truth count. An error $a \to b$ denotes a true count a predicted as b, where $a \neq b$, and is adjacent when $|a - b| = 1$. The inset matrices show errors among counts 1 to 4, while the total error counts also include predictions of 0. Table 37 reports the complete confusion counts and per-class accuracies.

Table 37: Count predictions and accuracies are shown as original LLM / SAIL.

True count	Predicted count					Accuracy (%)
	0	1	2	3	4	
GRAPHMACC (Qwen3-8B)						
1	8 / 7	127 / 153	132 / 109	3 / 1	0 / 0	47.04 / 56.67
2	0 / 0	4 / 5	115 / 152	147 / 109	4 / 4	42.59 / 56.30
3	0 / 0	0 / 0	4 / 10	181 / 183	85 / 77	67.04 / 67.78
4	0 / 0	0 / 0	0 / 0	87 / 44	183 / 226	67.78 / 83.70
Controlled Cora counting (Qwen3-4B)						
1	96 / 79	130 / 166	34 / 25	10 / 0	0 / 0	48.15 / 61.48
2	9 / 2	61 / 35	100 / 227	99 / 2	1 / 4	37.04 / 84.07
3	1 / 0	3 / 0	28 / 104	230 / 72	8 / 94	85.19 / 26.67
4	1 / 0	0 / 0	5 / 3	212 / 34	52 / 233	19.26 / 86.30

We identify repaired errors as originally incorrect predictions that become correct after SAIL correction, and newly introduced errors as originally correct predictions that become incorrect. Table 38 summarizes these changes, together with the total number of errors and the percentage attributable to adjacent counts.

Table 38: Errors repaired and introduced by SAIL. Paired entries show original LLM / SAIL results.

Dataset	Total errors	Adjacent errors (%)	Repaired	Introduced
GRAPHMACC	474 / 366	98.52 / 98.63	125	17
Cora	568 / 382	94.72 / 97.64	358	172

Adjacent-count confusions dominate the original errors in both settings, and SAIL corrects substantially more predictions than it makes incorrect. The increased separation in candidate-score space is therefore accompanied by lower overall counting error on both tasks, supporting the contribution of structural accumulation to more accurate count prediction.

F.3 Ablation Variants and Controlled Comparisons

We define the ablation variants relative to SAIL in Section 3, including the fixed structural scaling in Appendix C.4. We evaluate the ablations on all three GRAPHMACC tasks using Qwen3-4B, with detailed results reported in Table 5. For module ablations, we generally remove both the module and any connections that could convey similar information. Then, we organize the ablations into two groups, corresponding to the TrajAgg and QueryInt modules of SAIL.

TrajAgg.

Mean aggregation averages the occurrence-level trajectory messages to replace the original sum operation and replaces role counts with role proportions to prevent potential leakage of sum-based information through the role encodings. Sum and Mean use the same fixed structural scale estimated from the Sum fitting states, along with the same QueryInt architecture and training settings.

No relation encoding removes both $\mathbf{R}_{\text{count}}$ and $\mathbf{R}_{\text{map}}$ from the structural input while retaining the summed trajectory messages.

Structure only replaces QueryInt with a linear model that predicts candidate scores only from the accumulated state $\mathbf{T}_{q,G}^{(\ell)}$, keeping TrajAgg fixed and training only the linear readout. Full SAIL achieves the highest overall, counting, and comparison accuracy in Table 39.

Table 39: TrajAgg ablations and a structure-only comparison on GRAPHMACC with Qwen3-4B. Scores are mean ± SD, with differences from Full SAIL (percentage points) in parentheses.

Variant	Overall ↑	Acc_E ↑	Count ↑	Compare ↑
Full SAIL	**70.41** ± *0.34* *(0.00)*	92.61 ± *0.20* *(0.00)*	**47.10** ± *0.41* *(0.00)*	**71.53** ± *0.85* *(0.00)*
Mean aggregation	64.40 ± *0.22* *(−6.01)*	92.42 ± *0.12* *(−0.19)*	44.51 ± *0.44* *(−2.59)*	56.26 ± *0.62* *(−15.27)*
No relation encoding	<u>66.05</u> ± *0.29* *(−4.36)*	**92.70** ± *0.19* *(+0.09)*	<u>47.02</u> ± *0.28* *(−0.08)*	58.44 ± *0.63* *(−13.09)*
Structure only	62.49 ± *0.45* *(−7.92)*	88.68 ± *0.39* *(−3.93)*	38.90 ± *0.43* *(−8.20)*	<u>59.90</u> ± *1.01* *(−11.63)*

Mean aggregation reduces counting and comparison accuracy, while removing relation encoding mainly hurts comparison, with both variants retaining high existence accuracy. These patterns support the joint contribution of structural accumulation and role encoding to the use of category-count information beyond presence recognition. The Structure only results show a marked drop in accuracy across all three tasks, supporting the integration of accumulated structural information into the full LLM prediction process.

QueryInt.

Response only removes the accumulated structural input $\mathbf{T}_{q,G}^{(\ell)}$ while retaining the query representation h_q, candidate embeddings $\mathbf{e}_{y'}$, and original candidate scores $s_{y'}^{\mathrm{LLM}}$, allowing QueryInt to learn residual corrections without the structural trajectories.

No candidate embedding removes $\mathbf{e}_{y'}$ while retaining the accumulated structural state, query representation, and original candidate scores, with the residual correction architecture unchanged. Full SAIL achieves the highest overall, counting, and comparison accuracy in Table 40.

Table 40: QueryInt ablations on GRAPHMACC with Qwen3-4B. Results are mean ± SD, with changes from Full SAIL in percentage points shown in parentheses. Best and second-best means are bold and underlined, respectively.

Variant	Overall↑	Acc_E ↑	Count↑	Compare↑
Full SAIL	**70.41** ± *0.34* *(0.00)*	92.61 ± *0.20* *(0.00)*	**47.10** ± *0.41* *(0.00)*	**71.53** ± *0.85* *(0.00)*
Response only	<u>64.70</u> ± *0.33* *(-5.71)*	**92.85** ± *0.26* *(+0.24)*	<u>43.30</u> ± *0.43* *(-3.80)*	<u>57.97</u> ± *0.59* *(-13.56)*
No candidate embedding	57.18 ± *0.88* *(-13.23)*	91.96 ± *0.26* *(-0.65)*	33.31 ± *0.38* *(-13.79)*	46.25 ± *0.21* *(-25.28)*

Response only retains high existence accuracy but reduces counting and comparison accuracy, supporting the contribution of accumulated structural information to learned score correction. Removing candidate embeddings produces an even larger drop in counting and comparison accuracy, showing the value of integrating this information with candidate representations.

G GENERALIZATION AND LIMITATIONS

G.1 CROSS-MODEL APPLICABILITY

We extend the evaluation to 11 additional backbones from the Llama, Qwen, and Gemma families, ranging from 0.5B to 14B parameters. Each backbone remains frozen and has its own trained SAIL module, with the extraction layer and attention heads selected for that backbone. We follow the data splits and graph input construction in Appendix D.2 and the training procedure in Appendix D.4.

For GRAPHMACC, we follow the evaluation protocol in Appendix A.6.2, keeping each backbone's training set, selected configuration, and trained checkpoint fixed across the three test partitions. For the eleven backbones evaluated on native benchmarks, we follow the data protocol in Appendix D.2 and repeat training with five fixed random seeds on the same data partitions, following Appendix D.4. We report means and sample standard deviations across the three test partitions for GRAPHMACC and the five runs for native benchmarks. We compute gains from paired scores before averaging as SAIL minus the original LLM, except for $\Delta G = G_{\mathrm{LLM}} - G_{\mathrm{SAIL}}$, which measures gap reduction. For native benchmarks, each SAIL training run is compared with the same fixed original-LLM score on the same test set. Table 6 summarizes the cross-model gains, while Tables 41 and 42 provide the detailed results for GRAPHMACC and the native benchmarks, respectively.

Table 41: SAIL gains on GRAPHMACC are reported as mean ± SD in percentage points.

Model	$\Delta\mathrm{Acc}_E \uparrow$	$\Delta\mathrm{Acc}_{\mathrm{Count}} \uparrow$	$\Delta\mathrm{Acc}_{\mathrm{compare}} \uparrow$	$\Delta U \uparrow$	$\Delta G \uparrow$
Llama					
Llama-3.2-1B	*+45.37 ± 1.30*	*+52.74 ± 4.64*	*+27.04 ± 1.75*	*+39.89 ± 1.55*	*−5.48 ± 2.31*
Llama-3.2-3B	*+42.73 ± 1.65*	*+39.86 ± 3.23*	*+25.09 ± 1.72*	*+32.48 ± 2.01*	*−10.25 ± 3.46*
Llama-2-7B	*+33.66 ± 3.59*	*+35.90 ± 2.37*	*+23.17 ± 2.16*	*+29.53 ± 1.19*	*−4.13 ± 3.64*
Llama-2-13B	*+27.96 ± 2.90*	*+58.08 ± 1.79*	*+37.86 ± 0.97*	*+47.97 ± 1.10*	*+20.01 ± 2.29*
Qwen					
Qwen2.5-0.5B	*+43.39 ± 1.64*	*+57.41 ± 1.15*	*+20.72 ± 1.02*	*+39.06 ± 0.97*	*−4.32 ± 2.58*
Qwen2.5-3B	*+3.10 ± 1.88*	*+26.27 ± 3.60*	*+17.77 ± 2.17*	*+22.02 ± 2.67*	*+18.92 ± 3.53*
Qwen2.5-7B	*+6.25 ± 2.46*	*+48.65 ± 3.97*	*+26.03 ± 2.94*	*+37.34 ± 2.39*	*+31.08 ± 3.41*
Qwen2.5-14B	*+0.39 ± 1.91*	*+22.37 ± 2.61*	*+13.89 ± 2.23*	*+18.13 ± 1.85*	*+17.74 ± 3.02*
Gemma					
Gemma-3-1B	*+41.24 ± 1.31*	*+54.94 ± 1.32*	*+24.49 ± 2.87*	*+39.71 ± 1.75*	*−1.52 ± 2.98*
Gemma-3-4B	*+29.58 ± 2.50*	*+41.00 ± 2.37*	*+26.75 ± 1.88*	*+33.88 ± 1.33*	*+4.30 ± 3.09*
Gemma-3-12B	*+33.33 ± 1.87*	*+26.41 ± 1.43*	*+18.33 ± 1.07*	*+22.37 ± 1.13*	*−10.96 ± 2.07*

Table 42: SAIL gains on native benchmarks are reported as mean ± SD in metric points.

Model	Cora ΔAcc. ↑	PubMed ΔAcc. ↑	WikiCS ΔAcc. ↑	Arxiv ΔAcc. ↑	Citation2 ΔMRR ↑	MolHIV ΔAUC ↑
Llama						
Llama-3.2-1B	*+0.46 ± 1.29*	*+1.66 ± 4.78*	*−0.83 ± 0.92*	*+0.48 ± 0.59*	*+1.150 ± 0.400*	*+6.20 ± 14.38*
Llama-3.2-3B	*+2.66 ± 5.18*	*+8.56 ± 8.02*	*+1.53 ± 2.95*	*+2.94 ± 3.34*	*−0.482 ± 0.053*	*+27.44 ± 5.77*
Llama-2-7B	*+0.34 ± 0.84*	*+12.70 ± 11.93*	*−3.33 ± 3.84*	*+0.29 ± 0.39*	*+0.079 ± 0.147*	*+32.80 ± 1.88*
Llama-2-13B	*−0.04 ± 0.09*	*+4.94 ± 5.50*	*−0.75 ± 0.80*	*+0.21 ± 0.57*	*+0.277 ± 0.185*	*+5.83 ± 15.14*
Qwen						
Qwen2.5-0.5B	*+1.94 ± 7.05*	*−4.26 ± 5.27*	*+5.54 ± 5.29*	*+0.13 ± 0.12*	*+0.214 ± 0.129*	*+7.31 ± 1.12*
Qwen2.5-3B	*+0.12 ± 1.09*	*+0.22 ± 0.37*	*+1.04 ± 1.27*	*+3.57 ± 4.19*	*+0.051 ± 0.101*	*+3.10 ± 1.89*
Qwen2.5-7B	*+0.74 ± 0.93*	*+0.06 ± 0.05*	*−0.40 ± 0.39*	*+2.96 ± 8.31*	*+0.091 ± 0.066*	*+3.89 ± 3.95*
Qwen2.5-14B	*−0.06 ± 0.24*	*0.00 ± 0.00*	*+0.18 ± 0.14*	*−0.09 ± 0.06*	*+0.002 ± 0.005*	*+9.43 ± 0.60*
Gemma						
Gemma-3-1B	*+2.26 ± 2.96*	*+0.18 ± 4.63*	*+0.04 ± 2.23*	*+3.44 ± 2.60*	*+0.066 ± 0.064*	*+9.55 ± 2.52*
Gemma-3-4B	*−3.38 ± 7.17*	*0.00 ± 0.00*	*−0.21 ± 1.95*	*+0.02 ± 0.15*	*+0.252 ± 0.135*	*+8.76 ± 1.95*
Gemma-3-12B	*−0.10 ± 0.22*	*0.00 ± 0.00*	*+0.66 ± 0.62*	*−0.03 ± 0.13*	*−0.093 ± 0.097*	*+6.37 ± 13.19*

SAIL improves recognition, counting, and comparison across all evaluated GRAPHMACC backbones, indicating that the benefits of structural accumulation extend across model families and sizes. The native results show gains in 49 of 66 settings across node, edge, and graph prediction, supporting the applicability of utilization alignment beyond the two Qwen3 backbones in the main experiments.

G.2 MULTIPLICITY RANGE AND EXTRAPOLATION

We evaluate counting with Qwen3-4B on graphs of fixed degree 22, varying the training cutoff m from 3 to 10. Training and validation use counts and candidate answers $1, \ldots, m$, while testing extends both to $1, \ldots, m+2$. For each cutoff, we use five fixed splits with disjoint training, validation, and test graph replicates within each split, and train QueryInt with five fixed random seeds per split. We select checkpoints by validation cross-entropy, average the five runs within each split, and report means and 95% Student-t confidence intervals across the five split means.

Figure 5 reports relative error reduction within the training range,

$$\mathrm{RER} = \frac{\mathrm{Acc}_{\mathrm{SAIL}} - \mathrm{Acc}_{\mathrm{LLM}}}{1 - \mathrm{Acc}_{\mathrm{LLM}}} \times 100\%, \tag{62}$$

where accuracies are expressed as fractions and the reduction accounts for both repaired and newly introduced errors. For each adjacent pair of in-range counts, we measure separation as the distance between class centroids of centered candidate-score vectors divided by their pooled within-class spread. The pooled spread is $\sqrt{\max\{(\operatorname{tr}\widehat{\Sigma}_c + \operatorname{tr}\widehat{\Sigma}_{c+1})/2, 10^{-12}\}}$, where $\widehat{\Sigma}_c$ is the sample covariance matrix of centered candidate-score vectors for count c. Separation recovery is the relative increase in this normalized distance over the original LLM, averaged equally over adjacent pairs, and can therefore exceed 100%. We compute both metrics separately for each run before averaging over seeds and splits. Their positive values across the evaluated cutoffs show that SAIL reduces prediction errors while improving adjacent-count separation within each training range.

At unseen counts, Figure 11 shows negative mean accuracy changes for six of the eight cutoffs at $c = m + 1$ and for all eight at $c = m + 2$, indicating that the in-range gains do not ensure reliable extrapolation to larger counts. This may be related to the lack of training examples at larger counts or to saturation in QueryInt, and we will investigate both possibilities in future work.

G.3 MOLECULAR CASE STUDY

We examine MolHIV-2191 (PubChem CID 247952), an active molecule in the ogbg-molhiv test set containing two topologically equivalent aryl-imidazoline motifs (Figure 12). The original Qwen3-4B predicts "inactive", with an active-minus-inactive candidate-score margin of -0.25, whereas

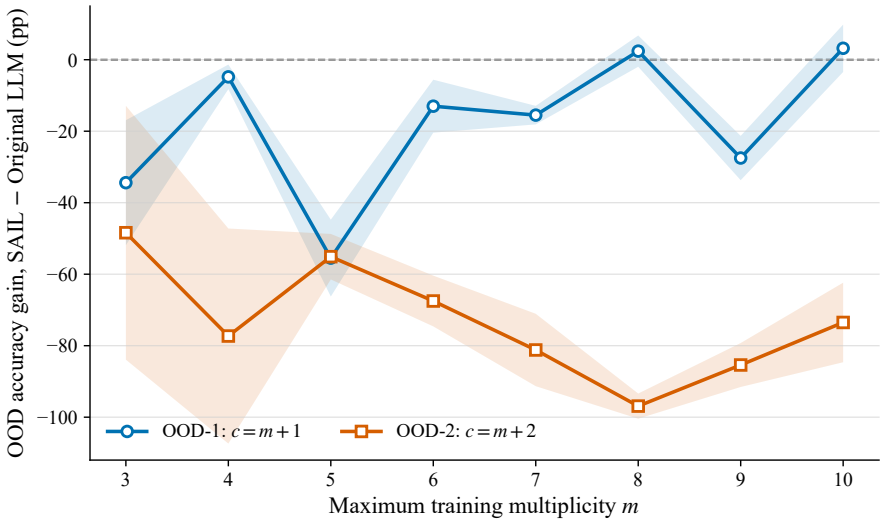


Figure 11: The curves show SAIL's accuracy changes relative to the original LLM at unseen counts $c = m + 1$ and $c = m + 2$, measured in percentage points. Shading shows 95% confidence intervals across five splits after averaging five training runs within each split.

SAIL makes this margin positive and recovers the active label in all five runs. The repeated motifs in this example also have biological relevance, as related bis-imidazoline HIV inhibitors illustrate how two imidazoline groups can jointly occupy the CXCR4 binding cavity and contribute to receptor binding and antiviral activity (Zhou et al., 2023). SAIL's consistent correction illustrates its benefit on a molecular graph with biologically relevant repeated structures, and we will investigate the role of these motifs in the candidate-score change in future work.

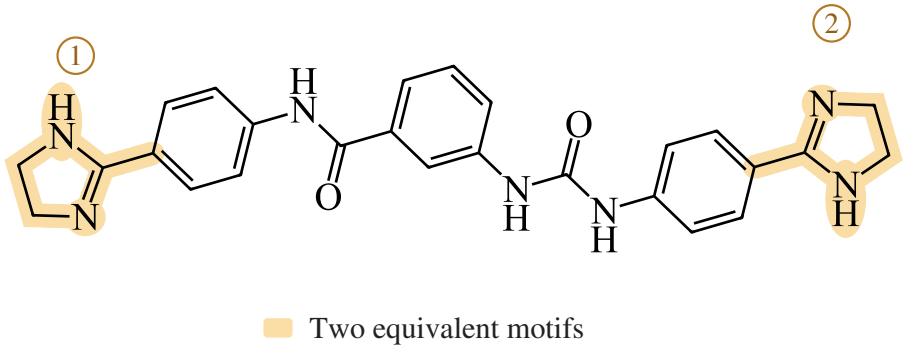


Figure 12: Complete molecular structure of MolHIV-2191, with its two equivalent aryl-imidazoline motifs highlighted.