GAUSSIAN RELATIONAL GRAPH TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

Relational graph learning enables predictive modeling over relational databases by directly capturing dependencies across interconnected tables. While relational graph Transformers extend the receptive field beyond local message passing, a larger receptive field does not necessarily provide more useful information: sampling must preserve relational structure without introducing excessive semantically irrelevant nodes, while attention must account for the temporal relevance of the retained information. We propose GelGT, a Gaussian relational graph transformer that explicitly addresses these challenges. GelGT introduces a structure-semantic collaborative sampling strategy to preserve structural connectivity while filtering irrelevant semantic information, and incorporates a Gaussian graph attention mechanism with a learnable Gaussian bias on the sampled subgraphs to dynamically encode temporal dependencies. We provide theoretical analysis characterizing the structural preservation, semantic refinement, and temporal discrimination of these mechanisms. Experiments on **7** real-world relational datasets covering **21** prediction tasks show that GelGT consistently outperforms existing relational graph learning methods, with improvements of up to **13.8%**.

1 INTRODUCTION

Relational databases are widely used to organize interconnected entities across multiple tables in applications such as e-commerce, finance, and healthcare. Predictive modeling over such databases traditionally relies on tabular models (Chen & Guestrin, 2016; Shwartz-Ziv & Armon, 2022; Grinsztajn et al., 2022; Chen et al., 2023; Voskou et al., 2024), which typically require manually joining tables and engineering features for individual tasks (Dwivedi et al., 2025; Kanter & Veeramachaneni, 2015; Lam et al., 2021). Recent *relational graph learning* approaches instead represent rows as nodes and primary–foreign key relations as edges (Robinson et al., 2024; Chen et al., 2025; Fey et al., 2024), enabling models to learn directly from both entity attributes and cross-table dependencies (Cvitkovic, 2020; Fey et al., 2024).

Existing relational graph learning methods (e.g., RDL (Robinson et al., 2024) and RelGNN (Chen et al., 2025)) are primarily built upon message-passing. While effective for local dependencies, message passing becomes increasingly restrictive when useful information lies several hops away due to information decay during propagation (Wu et al., 2021; 2022). This limitation is particularly relevant to relational databases, where predictions can require information distributed across multiple tables. More recently, RelGT (Dwivedi et al., 2026) applies Transformer attention over sampled relational subgraphs (Figure 1). This enlarges the receptive field, making it increasingly important to determine which information should be retained and how it should contribute to prediction.

Effective sampling must preserve relational structure while avoiding irrelevant information. Existing relational graph learning methods commonly rely on random sampling (Ranjan et al., 2026; Robinson et al., 2024; Chen et al., 2025), which can break connections among sampled entities and lead to structural fragmentation (Meher & Domeniconi, 2025). Increasing the sampled neighborhood can alleviate this problem, but may also introduce nodes that contribute little useful information to the prediction (Menczer et al., 2004). Structural preservation and semantic relevance therefore need to be considered together: structure defines the candidate set, while semantics identifies which distant nodes are worth retaining.

Beyond structural and semantic relevance, the model must determine how the retained information should contribute according to temporal relevance. Useful evidence may concentrate around a

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.

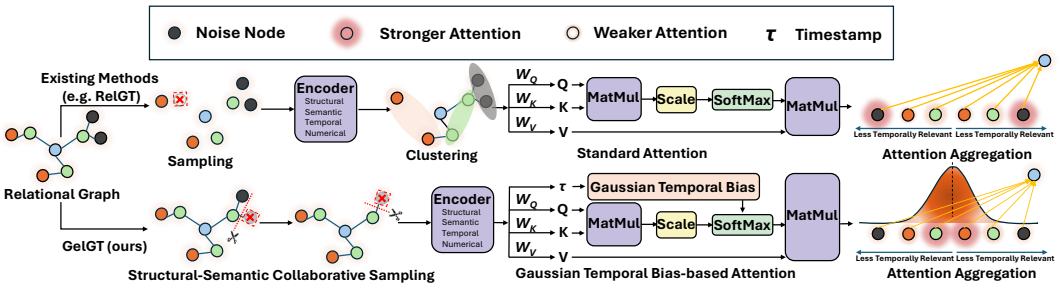

Figure 1: **Comparison of Existing Methods and Our Proposed GelGT. Structurally**, GelGT alleviates structural fragmentation by preserving node connectivity during sampling. **Semantically**, it mitigates the interference of semantically noisy nodes during sampling. **Temporally**, it employs *Gaussian Temporal Bias* to explicitly distinguish between temporally relevant and noisy nodes.

specific temporal region and extend over different temporal ranges. An inaccurate temporal focus may direct attention toward irrelevant information, while an overly narrow temporal range may miss useful dependencies and an overly broad one may introduce temporal noise (Guo et al., 2019). Effective relational modeling therefore requires both preserving structurally and semantically relevant information during sampling and explicitly distinguishing its temporal relevance during attention.

To address the aforementioned challenges, we propose the *Gaussian Relational Graph Transformer* (**GelGT** in short), as shown in Figure 1. GelGT first introduces a *structural-semantic collaborative sampling strategy* that preserves relational connectivity while refining distant nodes according to semantic relevance. For the retained information, GelGT further introduces a *Gaussian temporal bias* into graph attention, where the Gaussian center and range explicitly characterize *where* and *how broadly* the model attends over time. We provide theoretical analyses characterizing the structural preservation, semantic refinement, and temporal discrimination of these mechanisms.

Our main contributions are summarized as follows: (1) We propose **Gaussian Relational Graph Transformer (GelGT)**, a relational graph transformer that jointly addresses structural, semantic, and temporal challenges in relational graph learning through collaborative subgraph sampling and Gaussian graph attention. (2) We theoretically show that structure-semantic collaborative sampling preserves the vast majority ($\geq 99\%$) of effective structural information and can effectively mitigate semantic noise in seed node embeddings. We also prove that Gaussian graph attention assigns higher attention scores to temporally relevant nodes and lower attention scores to temporal noise nodes, compared to conventional attention without Gaussian bias (e.g., RelGT). (3) Extensive experiments on **7** datasets and **21** tasks demonstrate that GelGT achieves the state-of-the-art performance on both classification and value regression tasks, with up to a **13.8%** improvement in predictive performance.

2 PRELIMINARIES

Relational Database (RDB) (Codd, 1970; 1979). A *relational database* is formally defined as a collection of tables, denoted by $\mathcal{R} = \{\boldsymbol{T}_1, ..., \boldsymbol{T}_n\}$, where n represents the total number of tables. Each row in $\boldsymbol{T}_i$ corresponds to an entity, which is uniquely identified by a primary key and may contain references to entities in other tables through foreign keys, together with entity-specific attributes and associated timestamp information.

Relational Graph (Fey et al., 2024). The structure of a relational database inherently forms a graph representation, referred to as a *relational graph*. A relational graph is formally defined as a heterogeneous temporal graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\tau}\}$, where nodes $\mathcal{V}$ represent entities from database tables, edges $\mathcal{E}$ represent primary-foreign key relationships, $\boldsymbol{\phi}$ maps nodes to their respective types based on source tables, $\boldsymbol{\psi}$ assigns relation types to edges, and $\boldsymbol{\tau}$ captures the temporal dimension through timestamps.

RDB Task Reformulation. Based on the relational graph construction, predictive tasks on RDBs can be naturally cast as learning problems on $\mathcal{G}$. Let $\mathcal{Y}$ denote a target attribute column within a table $\boldsymbol{T}_i$. Since each row j in $\boldsymbol{T}_i$ corresponds to a unique node $v_j \in \mathcal{V}$, predicting a value in the j-th row of

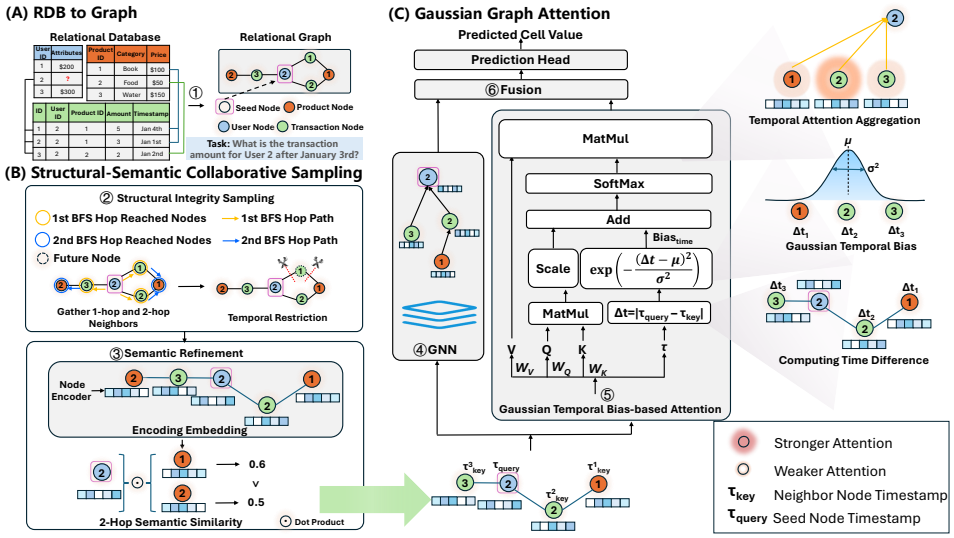

Figure 2: **Overview of the GelGT Framework**. The process consists of six main steps: ① **Graph Construction** converts the relational database into a relational graph. ② **Structural Integrity Sampling.** This stage samples subgraphs via BFS while enforcing strict timestamp constraints to mask future nodes for temporal validity. ③ **Semantic Refinement.** It filters noise by retaining only neighbors with high semantic similarity, calculated through dot products of encoded features. ④ A **GNN** module aggregates topological structural features. ⑤ **Gaussian Temporal Bias-based Attention.** It leverages a Gaussian temporal bias to assign high attention scores to temporally relevant nodes while allocating low scores to noisy nodes. ⑥ Finally, the **Fusion** module adaptively integrates the outputs of the GNN and the attention via a learnable weight (Li et al., 2025).

$\boldsymbol{T}_i$ can be formulated as learning a mapping function $f\colon \mathcal{V} \to \mathcal{Y}$ that leverages both node attributes and graph topology. Following previous work (Dwivedi et al., 2026), we focus on two paradigms: (i) *node classification*, where $\mathcal{Y}$ is a finite set of discrete categorical labels, and (ii) *node regression*, where $\mathcal{Y} \subseteq \mathbb{R}$ represents a continuous range of numerical values.

3 Gaussian Relational Graph Transformer (GelGT)

This section introduces the GelGT, as illustrated in Figure 2. GelGT consists of two components: *structure-semantic collaborative sampling* (Steps ①-③) and *Gaussian graph attention model* (Steps ④-⑥). The former constructs a query-specific computation subgraph, which enables the latter to learn temporally aware representations. Sections 3.1 and 3.2 describe the two components, respectively.

3.1 Structure-Semantic Collaborative Sampling

The goal of our sampling strategy is to construct a subgraph that preserves structural connectivity while filtering out semantically irrelevant nodes. This yields the final subgraph. The resulting subgraph explicitly defines the valid interaction scope for the subsequent attention model, and is constructed via two stages: (1) *structural integrity sampling* and (2) *semantic refinement*.

Stage 1: Structural Integrity Sampling. The goal of this stage is to preserve the local relational structure around a seed node v under a fixed sampling budget B_s. To achieve this, we employ breadth-first search (BFS) to expand from v. Under the fixed sampling budget B_s, BFS expands the neighborhood level by level, preferentially retaining information from nodes that are more closely connected to the seed node before reaching farther regions, thereby preserving the local context around v. During the expansion, we strictly traverse neighbors that satisfy temporal causality[1], and stop once the sampling budget B_s is reached. This process yields a sampled node set. Finally, we construct the subgraph induced by the sampled node set. Formally, we define the sampled node set

[1]Temporal causality (Prabhakar et al., 2010) requires that the timestamp of any sampled neighbor precedes that of the query node (i.e., $\tau_{\text{neighbor}} < \tau_{\text{seed}}$), to avoid temporal information leakage.

$\mathcal{N}_{\text{sampled}}(v)$ and the resulting sampled subgraph $\mathcal{G}_{\text{sub}}(v)$ as:

$$\mathcal{N}_{\text{sampled}}(v) = \{u_1, u_2, \ldots, u_B\}, \quad u_1 = v \tag{1}$$

$$\mathcal{G}_{\text{sub}}(v) = (\mathcal{N}_{\text{sampled}}(v), \mathcal{E}_{\text{sub}}), \tag{2}$$

where $\mathcal{E}_{\text{sub}} = \{(i,j) \in \mathcal{E} \mid i,j \in \mathcal{N}_{\text{sampled}}(v)\}$ and $\mathcal{E}$ denotes the edge set of the relational graph, and $(u_1, u_2, \ldots)$ is the BFS visitation order over nodes whose timestamps do not exceed the timestamp of node v.

Theoretical Analysis. We theoretically show that the structural integrity sampling effectively preserves the structural information of the relational graph. To quantify this, we use Katz centrality (Katz, 1953) as the topological metric and denote the Katz centrality computed on the full graph and the sampled subgraph as $C^{\mathcal{G}}(v)$ and $C^{\mathcal{G}_{\text{sub}}}(v)$, respectively. We establish the quantitative relationship between the structural loss $\Delta C(v) = |C^{\mathcal{G}}(v) - C^{\mathcal{G}_{\text{sub}}}(v)|$ arising from structural integrity sampling and the full-graph structural information in Theorem 1.

Theorem 1 (Upper Bound of Relative Structural Loss). *The ratio of the structural loss $\Delta C(v) = |C^{\mathcal{G}}(v) - C^{\mathcal{G}_{\text{sub}}}(v)|$ arising from structural integrity sampling to the full-graph structural information $C^{\mathcal{G}}(v)$ satisfies:*

$$\frac{|\Delta C(v)|}{C^{\mathcal{G}}(v)} \leq 0.01 \tag{3}$$

We provide a detailed proof of Theorem 1 in Appendix A.1. Theorem 1 demonstrates that the structural integrity sampling preserves at least 99% of the effective structural information, ensuring that the sampled subgraph provides a reliable structural basis for subsequent modeling.

Stage 2: Semantic Refinement. Then, we refine the sampled subgraph $\mathcal{G}_{\text{sub}}(v)$ constructed in Stage 1 to mitigate semantic noise. Specifically, we first encode the raw attributes of all nodes in the subgraph $\mathcal{G}_{\text{sub}}(v)$ to low-dimensional embeddings $\mathbf{h}$. We compute the semantic similarity between the seed node v and each candidate node u beyond its one-hop neighborhood via dot product:

$$s(v,u) = \mathbf{h}_v^\top \mathbf{h}_u, \quad \forall u \in \mathcal{N}_r = \mathcal{N}_{\text{sampled}}(v) \setminus (\{v\} \cup \mathcal{N}_1(v)), \tag{4}$$

where $\mathcal{N}_1(v)$ denotes the set of 1-hop neighbors of v satisfying the temporal causality constraint[1]. Among the remaining nodes ($\mathcal{N}_r$) in the sampled subgraph, excluding v and the nodes in $\mathcal{N}_1(v)$, we compute semantic similarity scores and use them to guide node selection. Given the final training budget $B_t < B_s$, we first include the nodes in $\mathcal{N}_1(v)$, and then select additional nodes from $\mathcal{N}_r$ in descending order of semantic similarity until the budget B_t is reached. We prioritize one-hop neighbors because the structural contribution of k-hop neighbors decays with hop distance:

$$\Delta \mathcal{C}_{k\text{-hop}}(v) = \mathcal{O}(\beta^k)^2, \tag{5}$$

where $0 < \beta < 1$ is the attenuation factor. Hence, one-hop neighbors provide the most important local structural information, motivating their preservation during semantic refinement. Finally, we construct the training subgraph $\mathcal{G}_{\text{final}}$ from these B_t selected training nodes and their edges to v that satisfy the temporal causality constraint[1].

Theoretical Analysis. Since noise nodes tend to exhibit lower semantic similarity to the seed node, semantic refinement excludes them from the final sampled subgraph $\mathcal{G}_{\text{final}}$. Let $\mathcal{I}$ denote the aggregated information from neighbors of the seed node, decomposed into semantically relevant information $\mathcal{I}_{\text{relevance}}$ and noise $\mathcal{I}_{\text{noise}}$. Semantic refinement therefore increases the relevance-to-noise ratio (by amplifying $\mathcal{I}_{\text{relevance}}$ and suppressing $\mathcal{I}_{\text{noise}}$), as formalized in Theorem 2.

Theorem 2 (Enhancement of Semantic Relevance-to-Noise Ratio). *The semantic relevance-to-noise ratio after refinement strictly exceeds that of the pre-refinement stage. Formally,*

$$\frac{\mathcal{I}_{\text{relevance}}^{\text{before}}}{\mathcal{I}_{\text{noise}}^{\text{before}}} < \frac{\mathcal{I}_{\text{relevance}}^{\text{after}}}{\mathcal{I}_{\text{noise}}^{\text{after}}} \tag{6}$$

where $\{\mathcal{I}_{\text{relevance}}^{\text{before}}, \mathcal{I}_{\text{noise}}^{\text{before}}\}$ and $\{\mathcal{I}_{\text{relevance}}^{\text{after}}, \mathcal{I}_{\text{noise}}^{\text{after}}\}$ denote the semantic relevance and semantic noise before and after the refinement process, respectively.

[2]A detailed derivation of Equation 5 is provided in Appendix A.2

Theorem 2 demonstrates that our proposed semantic refinement strategy can effectively mitigate semantic noise in seed node embeddings. A detailed proof is in Appendix A.3.

Through structural integrity sampling and semantic refinement, we obtain the final sampled subgraph $\mathcal{G}_{\text{final}}$. The resulting subgraph defines the set of structurally connected and semantically refined nodes used by the subsequent GNN and attention modules.

3.2 Gaussian Temporal Bias-based Attention

The sampling stage determines which nodes should be retained based on structural connectivity and semantic relevance. We next consider how the retained information should contribute to prediction according to its temporal relevance. As shown in Figure 2, GelGT employs a composite architecture that combines a GNN with graph attention. The GNN aggregates local topological information, while the attention module captures dependencies among the retained nodes. To explicitly model their temporal relevance, we introduce an adaptive Gaussian temporal bias into the attention mechanism.

Adaptive Gaussian Temporal Bias. A straightforward strategy for capturing temporal information, as in RelGT (Dwivedi et al., 2026), is to encode timestamps jointly with other node attributes. However, such integrated representations entangle temporal information with structural and semantic information, making it difficult for the attention mechanism to explicitly distinguish temporally relevant nodes from noise. To address this, we introduce a learnable attention bias that parameterizes a temporal preference distribution via a continuous Gaussian function, allowing the attention mechanism to selectively focus on relevant temporal information.

Formally, given the input feature matrix $\mathbf{H} \in \mathbb{R}^{N\times d}$ (derived from the encoders as shown in Appendix E), we define the standard linear projections for the Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) matrices as: $\mathbf{Q} = \mathbf{W}_Q\mathbf{H}, \quad \mathbf{K} = \mathbf{W}_K\mathbf{H}, \quad \mathbf{V} = \mathbf{W}_V\mathbf{H}$, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are learnable parameter matrices. Different from vanilla self-attention (Vaswani et al., 2017), we explicitly inject a temporal bias $\text{Bias}_{\text{time}}$ into the scaled attention score ($\frac{\mathbf{QK}^T}{\sqrt{d}}$) before normalization, as shown in Equation 7.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d}} + \text{Bias}_{\text{time}}\right)\mathbf{V}. \tag{7}$$

The adaptive temporal bias ($\text{Bias}_{\text{time}}$) is formally defined as:

$$\text{Bias}_{\text{time}} = \text{Linear}\left(\exp\left(-\frac{(\Delta t - \mu)^2}{\sigma^2}\right)\right). \tag{8}$$

Δt denotes the time interval between query and key nodes. Mean μ and standard deviation σ are learnable parameters. Temporal long-range perception is critical for relational graph learning (Lachi et al., 2025). The learned μ serves to identify the most temporally relevant information (i.e., temporal center) within the long-range temporal receptive field of a training node. Besides, the temporal relevance information decays with increasing distance from the temporal center (Guo et al., 2019). Thus, the learned σ models the information decay effect, making the temporal bias align with the underlying temporal relevance distribution.

Theoretical Analysis. In GelGT, the distribution of attention weights is jointly defined by the temporal center μ and scope σ. This parameterized structure achieves a systematic separation between temporally relevant nodes and temporal noise. First, Lemma 1 proves that the learnable parameter μ converges to the ground-truth relevant timestamp. Based on that, Theorem 3 further proves that our method assigns a higher attention score to temporally relevant nodes than to noise nodes.

Lemma 1 (**Convergence of Temporal Center**). *Let t^* be the ground-truth relevant timestamp. The gradient of the Gaussian temporal bias $\mathcal{B}(t^*)$ with respect to the learnable temporal center μ satisfies: $\frac{\partial \mathcal{B}}{\partial \mu} = \mathcal{B}(t^*) \cdot \frac{t^*-\mu}{\sigma^2}$. The term $(t^* - \mu)$ drives μ toward t^* during optimization.*

Detailed proof of Lemma 1 is in Appendix A.4. Based on the convergence property established in Lemma 1, we further quantify the disparities in attention score assignment between temporally relevant nodes and noise nodes, specifically comparing the mechanisms w/ and w/o Gaussian bias.

Table 1: **Test Set Results on the Node Classification Tasks in RelBench (Robinson et al., 2024).** Best values (based on mean performance) are in **bold**, and the second-best values are underlined. Results are reproduced based on available code. ↑: higher is better.

Dataset	Task	RDL	HGT	HGT+PE	Griffin	RelGT	LightGBM	Ours
rel-f1	driver-dnf (↑)	0.7262±0.0027	0.7077±0.0153	0.7117±0.0084	0.7091±0.0045	0.7587±0.0413	0.6886±0.0034	**0.7608±0.0175**
rel-f1	driver-top3 (↑)	0.7554±0.0154	0.7765±0.0066	0.7340±0.0018	0.7795±0.0139	0.8352±0.0342	0.7393±0.0121	**0.8408±0.0063**
rel-avito	user-clicks (↑)	0.6590±0.0195	0.6376±0.0298	0.6457±0.0099	0.6330±0.0025	0.6830±0.0602	0.5360±0.0032	**0.6844±0.0600**
rel-avito	user-visits (↑)	0.6620±0.0010	0.6432±0.0002	0.6495±0.0022	0.6468±0.0045	0.6678±0.0015	0.5305±0.0011	**0.6695±0.0060**
rel-event	user-repeat (↑)	0.7689±0.0159	0.6496±0.0220	0.6536±0.0137	0.7709±0.0062	0.7609±0.0219	0.5305±0.0211	**0.8357±0.0203**
rel-event	user-ignore (↑)	0.8162±0.0111	0.8247±0.0096	0.8161±0.0007	0.8153±0.0038	0.8157±0.0040	0.7993±0.0154	**0.8779±0.0122**
rel-trial	study-outcome (↑)	0.6860±0.0101	0.5837±0.0141	0.5921±0.0303	0.6908±0.0071	0.6861±0.0040	0.7009±0.0063	**0.7254±0.0059**
rel-amazon	user-churn (↑)	0.7042±0.0005	0.6643±0.0041	0.6619±0.0042	0.7000±0.0036	0.7039±0.0040	0.5222±0.0051	**0.7050±0.0065**
rel-amazon	item-churn (↑)	0.8281±0.0003	0.7797±0.0039	0.7803±0.0053	0.8110±0.0051	0.8255±0.0006	0.6254±0.0044	**0.8297±0.0096**
rel-stack	user-engagement (↑)	0.9021±0.0007	0.8847±0.0044	0.8817±0.0046	0.8980±0.0023	0.9053±0.0005	0.6339±0.0350	**0.9086±0.0186**
rel-stack	user-badge (↑)	0.8986±0.0008	0.8608±0.0044	0.8566±0.0068	0.8700±0.0031	0.8632±0.0018	0.6343±0.0079	**0.9044±0.0086**
rel-hm	user-churn (↑)	0.6988±0.0021	0.6695±0.0067	0.6569±0.0109	0.6804±0.0014	0.6927±0.0019	0.5521±0.0025	**0.6997±0.0017**

Theorem 3 (**Attention Score Ratio Enhancement**). *Let* $\{\alpha^A_{relevant}, \alpha^A_{noise}\}$ *denote the attention scores that the attention mechanism A assigned to temporally related nodes and noise nodes. Then,*

$$\frac{\alpha^{\text{w/ G}}_{relevant}}{\alpha^{\text{w/ G}}_{noise}} = \mathbf{e} \cdot \frac{\alpha^{\textit{w/o G}}_{relevant}}{\alpha^{\textit{w/o G}}_{noise}}, \tag{9}$$

where $\mathbf{e}$ *is Euler's number, "w/ G" and "w/o G" indicate the attention mechanism with and without the Gaussian bias, respectively.*

We provide a detailed proof of Theorem 3 in Appendix A.5. Theorem 3 demonstrates that, compared to vanilla self-attention (Vaswani et al., 2017), our mechanism increases the ratio of attention weights assigned to temporally relevant versus noisy nodes by a factor of $\mathbf{e}$. Consequently, our transformer more effectively discriminates between temporally relevant information and temporal noise.

4 Experiments

In this section, we conduct extensive experiments to answer the following research questions:
RQ1: Does GelGT outperform existing state-of-the-art methods in terms of predictive performance?
RQ2: How much does each component of GelGT contribute to its performance, and how well can the proposed components be integrated into other baseline frameworks?
RQ3: How do different configurations of GelGT affect its performance?
RQ4: How efficient is GelGT in terms of training and inference time?

4.1 Experimental Setup

Datasets GelGT is evaluated on the recently introduced RDL Benchmark (RelBench) (Robinson et al., 2024). In total, RelBench contains **30** tasks across **7** datasets, covering node classification, node regression, and recommendation. For our evaluation, we focus on **21** tasks on node classification and regression, following the task settings used in RelGT (Dwivedi et al., 2026).
Baselines We compare GelGT with six state-of-the-art baselines for node classification and node regression tasks (Tables 1 and 2). Our baselines include relational graph transformer (RelGT) (Dwivedi et al., 2026), as well as the graph-centric relational database foundation model Griffin (Wang et al., 2025). Consistent with the experimental settings in prior work (Dwivedi et al., 2026), we also evaluate two variants of the Heterogeneous Graph Transformer (HGT) (Hu et al., 2020) to further examine the effectiveness of GelGT relative to existing graph transformers. In addition, we compare against LightGBM (Ke et al., 2017), a strong baseline from the tabular learning literature.
Evaluation Metrics Node-level classification aims to predict binary labels for a given node at a specific seed time. We use the Area Under the ROC Curve (AUC) (Hanley & Mcneil, 1983) as the evaluation metric. Node-level regression involves predicting numerical labels for a node at a given seed time. We use Mean Absolute Error (MAE) as the evaluation metric.

4.2 Performance (**RQ1**)

We evaluate GelGT on **12** node classification tasks across **7** datasets, as shown in Table 1. The results demonstrate the strong performance of GelGT compared to baselines. Specifically, GelGT improves AUC by up to **6.2**% over the baselines. It ranks first in AUC on every evaluated task, indicating

Table 2: **Test Set Results on the Node Regression Tasks in RelBench (Robinson et al., 2024).** Best values (based on mean performance) are in **bold**, and the second-best values are underlined. Results are reproduced based on available code. Results marked with † are reproduced using the official code. ↓: lower is better.

Dataset	Task	RDL	HGT	HGT+PE	Griffin	RelGT	LightGBM	Ours
rel-f1	driver-position (↓)	4.022±0.119	4.2263±0.0580	4.3921±0.1382	3.9315±0.2903	3.9170±0.3448 †	4.1700±0.2200	**3.7345±0.1200**
rel-avito	ad-ctr (↓)	0.0410±0.0010	0.0462±0.0021	0.0483±0.0027	0.0433±0.0004	0.0420±0.0009 †	0.0410 ±0.0041	**0.0362±0.0009**
rel-event	user-attendance (↓)	0.258±0.006	0.2635±0.0000	0.2611±0.0043	0.3708±0.0094	0.2502±0.0033	0.2640±0.0011	**0.2423±0.0031**
rel-trial	study-adverse (↓)	44.473±0.209	45.1692±2.6927	42.6484±0.2785	59.0902±0.0022	43.9923±0.5928	44.0110±0.2020	**42.5699±0.9700**
rel-trial	site-success (↓)	0.4000±0.0200	0.4428±0.0047	0.4396±0.0083	0.3801±0.0041	0.3761±0.0178 †	0.4250±0.0008	**0.3485±0.0200**
rel-amazon	user-ltv (↓)	14.3130±0.013	15.4120±0.0447	15.8643±0.0924	19.6055±0.0239	14.2665±0.0154	16.7830±0.0188	**14.2538±0.0189**
rel-amazon	item-ltv (↓)	50.0530±0.1630	55.8683±0.6003	55.8493±0.3226	66.7938±0.0193	54.6626±0.7006 †	60.5690±0.0033	**49.2109±0.8092**
rel-stack	post-votes (↓)	0.0650±0.0000	0.0679±0.0000	0.0680±0.0000	0.1395±0.0076	0.0654±0.0002	0.0680±0.0065	**0.0648±0.0025**
rel-hm	item-sales (↓)	**0.0560±0.0000**	0.0641±0.0012	0.0639±0.0003	0.0597±0.0008	0.0563±0.0006 †	0.0760±0.0002	**0.0560±0.0009**

Table 3: **Performance on Node Classification Tasks on the SALT Dataset.** ↑: higher is better.

Task ↑	RDL	HGT	RelGT	RGP (Lachi et al., 2025)	Ours
rel-salt-sales-group	0.20	0.20	0.35	0.34	**0.38**
rel-salt-sales-incoterms	0.70	0.75	0.73	-	**0.77**
rel-salt-sales-payterms	0.39	0.60	0.61	0.58	**0.64**
rel-salt-sales-office	0.93	0.96	0.98	-	**0.99**
rel-salt-sales-shipcond	0.59	0.76	0.75	0.81	**0.83**
rel-salt-item-incoterms	0.64	0.75	0.84	0.81	**0.87**
rel-salt-item-plant	0.90	0.91	0.97	-	**0.99**
rel-salt-item-shippoint	0.94	0.93	0.98	-	**0.99**

Table 4: **Performance Comparison with More Representative Baselines.** ↑: higher is better.

Task ↑	Rel-LLM	Nodeformer	DUALFormer	Ours
rel-f1-driver-top3	0.82	0.82	0.79	**0.84**
rel-avito-user-clicks	0.66	0.66	0.65	**0.68**
rel-event-user-repeat	0.79	0.79	0.74	**0.83**
rel-event-user-ignore	0.83	0.83	0.82	**0.87**
rel-trial-study-outcome	0.71	0.71	0.69	**0.72**
rel-stack-user-badge	0.89	0.89	0.87	**0.90**
rel-hm-user-churn	0.55	0.61	0.59	**0.69**

its strong ability to effectively distinguish positive nodes from negative ones. Besides, we evaluate GelGT on **9** node regression tasks across **7** datasets, as shown in Table 2. The results demonstrate the best performance of GelGT compared to baselines, achieving up to a **13.8**% improvement in MAE. Furthermore, on the very large-scale SALT dataset from RelBench v2 (Gu et al., 2026), GelGT achieves state-of-the-art results across all **8** classification tasks, shown in Table 3. GelGT consistently outperforms a broader set of representative baselines that incorporate recent advances across many evaluated tasks, including the LLM-based approach Rel-LLM (Wu et al., 2025), and transformer-based long-range graph models such as Nodeformer (Wu et al., 2022) and DUALFormer (Zhuo et al., 2025), as shown in Table 4. Additional details on these baselines and the SALT dataset, along with a comparison with RT (Ranjan et al., 2026), are provided in Appendix C.

4.3 Component Study (**RQ2**)

First, we perform an ablation study by removing key components of GelGT: **(1) w/o Structural Integrity Sampling**: We use random sampling instead. This does not preserve the topological structure of the relational graph. **(2) w/o Semantic Refinement**: We retain all sampled nodes, including the irrelevant ones. **(3) w/o Adaptive Gaussian Temporal Bias**: We remove the adaptive Gaussian temporal bias, reducing the attention mechanism to vanilla self-attention (Vaswani et al., 2017). Tables 5 and 6 present the ablation study results for the node classification and node regression tasks, respectively. In Tables 5 and 6, removing structural integrity sampling degrades performance, confirming the necessity of the topological structure preserved by our sampling strategy. Eliminating semantic refinement leads to a clear drop, demonstrating that GelGT effectively removes irrelevant information. Finally, excluding the adaptive Gaussian temporal bias results in an observable performance loss, indicating that simply encoding time as features is insufficient.

Besides, we evaluate the performance of RelGT enhanced with our proposed key components, including structural-semantic collaborative sampling and Gaussian temporal bias, across five different tasks to validate their effectiveness. As shown in Figure 3, this uniform performance gain across diverse tasks demonstrates that our approach effectively addresses the limitations of the original framework. In summary, the results demonstrate that our proposed modules effectively generalize to other baseline frameworks, significantly enhancing their predictive performance. In Appendix F.2, we further demonstrate through experiments that the proposed sampling strategy is model-agnostic and can improve the predictive performance of RelGNN (Chen et al., 2025).

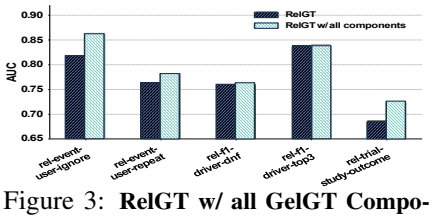

Figure 3: **RelGT w/ all GelGT Components vs. RelGT**

What is more, we conduct more fine-grained ablation studies to validate the rationale behind several of our design choices, including the GNN branch, selection rule, traversal strategy, and temporal bias as follows. **(1) GNN Branch.** As shown in Table 7, performance consistently drops across all datasets. This validates the necessity of the GNN branch in GelGT. The attention branch, even

Table 5: **Ablation Study of GelGT Components on Node Classification Tasks.** ↑: higher is better.

Dataset	Task	GelGT (Ours)	w/o Structural Integrity Sampling	w/o Semantic Refinement	w/o Adaptive Gaussian Bias
rel-avito	user-clicks (↑)	0.6844±0.0600	0.6688±0.0057	0.6677±0.0031	0.6669±0.0043
rel-avito	user-visits (↑)	0.6695±0.0060	0.6612±0.0018	0.6652±0.0040	0.6630±0.0028
rel-event	user-ignore (↑)	0.8779±0.0122	0.8691±0.0073	0.8699±0.0086	0.8638±0.0091
rel-trial	study-outcome (↑)	0.7254±0.0059	0.7161±0.0060	0.7155±0.0040	0.7192±0.0076
rel-amazon	user-churn (↑)	0.7050±0.0065	0.6812±0.0080	0.6973±0.0095	0.6945±0.0089

Table 6: **Ablation Study of GelGT Components on Node Regression Tasks.** ↓: lower is better.

Dataset	Task	GelGT (Ours)	w/o Structural Integrity Sampling	w/o Semantic Refinement	w/o Adaptive Gaussian Bias
rel-avito	ad-ctr (↓)	0.0362±0.0009	0.0372±0.0009	0.0368±0.0007	0.0370±0.0004
rel-trial	site-success (↓)	0.3485±0.0200	0.3909±0.0048	0.3491±0.0178	0.3583±0.0034
rel-hm	item-sales (↓)	0.0560±0.0009	0.0584±0.0021	0.0599±0.0014	0.0585±0.0008

with hop-distance embeddings and GNN-based positional encodings, operates on a fully connected sampled subgraph and therefore only captures relative positional information, without explicitly modeling the true PK-FK edges. In contrast, the GNN branch performs message passing directly along PK-FK edges, which preserves the original relational structure and better captures the semantic dependencies encoded in these edges. **(2) Selection Rule.** Top-k selection by the dot-product score beats both random-k and bottom-k drawn from the same candidate pool on all five tasks (Table 8 (a)), confirming that the score ranks nodes by downstream usefulness rather than acting as an arbitrary heuristic. **(3) Traversal Strategy.** BFS beats DFS on all five tasks, with the largest relative gain of 2.1% on study-outcome (Table 8 (b)). This advantage stems from their different traversal patterns under a fixed sampling budget. DFS preferentially explores a single PK–FK branch, concentrating samples on a few relational paths and leaving other direct PK–FK neighbors uncovered. In contrast, BFS expands PK–FK neighbors level by level, providing broader relational coverage within the same budget. **(4) Other Temporal Biases.** Our Gaussian bias beats both a scaled linear bias and a learnable temporal embedding on all seven datasets (Table 8 (c)). **(5) Temporal Bias under Temporal Shuffling.** Under temporal shuffling (i.e., randomly shuffling the timestamps of the dataset to make the evaluation more challenging), w/o temporal bias (i.e., RelGT's attention) experiences an average decline of 0.0441 (5.64%), while w/ temporal bias (i.e., GelGT) exhibits a smaller drop of 0.0348 (4.36%) (Table 8 (d)). This suggests that incorporating temporal bias into the attention mechanism improves its robustness to temporal perturbations.

4.4 Configuration Analysis (**RQ3**)

Sampling Size Analysis. We vary the number of sampled nodes (i.e., sampling size) within $\{100, 200, 300, 400, 500\}$, as shown in Figure 4. The performance across all tasks reaches its peak at a sampling size of 300. Specifically, when the sampling size is less than 300, the model suffers from an information bottleneck and fails to capture sufficient semantic information. Conversely, when the sampling size exceeds 300, the limited number of truly relevant neighbors forces the similarity-based mechanism to incorporate semantically irrelevant nodes, which obscures the contribution of semantically relevant nodes. Therefore, we set the sampling size to 300, as it ensures sufficient information aggregation while avoiding interference from semantically irrelevant nodes. In the Appendix F.6, we further analyze the sampling budget.

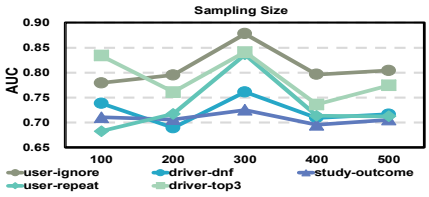

Figure 4: **Sampling Size**

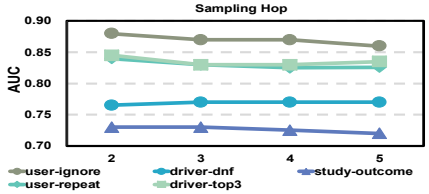

Figure 5: **Sampling Hops**

Sampling Hop Analysis. Then, we vary the sampling hops from $\{2, 3, 4, 5\}$, as summarized in Figure 5. The performance remains consistent across all tasks with minimal fluctuations, indicating that GelGT is insensitive to the sampling hops. In GelGT, long-range information is incorporated through the semantic relevance-based selection of distant nodes in Stage 2, rather than through indiscriminate topological expansion. This stability suggests that increasing the search depth beyond a moderate range provides limited benefit once semantic refinement is applied. Stage 1 exposes candidate nodes through topological expansion, while Stage 2 selectively retains distant nodes according to semantic relevance. Consequently, deeper expansion mainly increases the candidate pool and computational cost, rather than proportionally increasing useful information. Details on the rationale behind the semantic refinement design are provided in Appendix F.1.

Table 7: **Effect of the GNN Branch on Node Classification Tasks.** ↑: higher is better.

Metric / Dataset	rel-f1	rel-avito	rel-event	rel-trial	rel-amazon	rel-stack	rel-hm
Task	driver-dnf	user-clicks	user-repeat	study-outcome	item-churn	user-badge	user-churn
w/o GNN branch	0.7499	0.6751	0.8173	0.7094	0.8128	0.8917	0.6822
w/ GNN branch	**0.7608**	**0.6844**	**0.8357**	**0.7254**	**0.8297**	**0.9044**	**0.6997**

Table 8: **More Component Analysis.** (a) **Selection Rule**: Top-k selection via dot-product scores outperforms random-k and bottom-k on the same candidate nodes. (b) **Traversal Strategy**: BFS consistently surpasses DFS under matched budgets and settings. (c) **Other Temporal biases**: Comparison of AUC across different temporal bias formulations. (d) **Temporal Shuffling**: w/ temporal bias (GelGT) vs. w/o temporal bias (RelGT) under Shuffled Timestamps ↑: higher is better.

Task	**Top-k**	**Rand-k**	**Bot-k**
top3	**0.8408**	0.8311	0.8287
dnf	**0.7608**	0.7491	0.7384
repeat	**0.8357**	0.8180	0.8156
ignore	**0.8779**	0.8602	0.8564
outcome	**0.7254**	0.7111	0.7207

(a) Selection Rule

Task	**BFS**	**DFS**
top3	**0.8408**	0.8311
dnf	**0.7608**	0.7539
repeat	**0.8357**	0.8290
ignore	**0.8779**	0.8605
outcome	**0.7254**	0.7102

(b) Traversal Strategy

Task	**Linear**	**Embed.**	**Ours**
dnf	0.75	0.75	**0.76**
clicks	0.67	0.67	**0.68**
repeat	0.75	0.74	**0.76**
outcome	0.70	0.71	**0.72**
churn	0.69	0.69	**0.70**
badge	0.89	0.88	**0.90**
hm-churn	0.68	0.67	**0.69**

(c) More Temporal Biases

Task	**RelGT**	**GelGT**
top3	0.7812	**0.8025**
dnf	0.7177	**0.7332**
repeat	0.7201	**0.7998**
ignore	0.7568	**0.8513**
outcome	0.6605	**0.6798**

(d) Temporal Shuffling

4.5 Efficiency Analysis (**RQ4**)

We evaluate the efficiency of GelGT across five tasks by comparing its training time per epoch and inference time against state-of-the-art relational graph transformer models. As shown in Figure 6, GelGT consistently demonstrates lower time costs for both training and inference across all five evaluated tasks compared to baseline methods. This uniform reduction in computational overhead confirms that GelGT is more efficient for relational graph learning tasks.

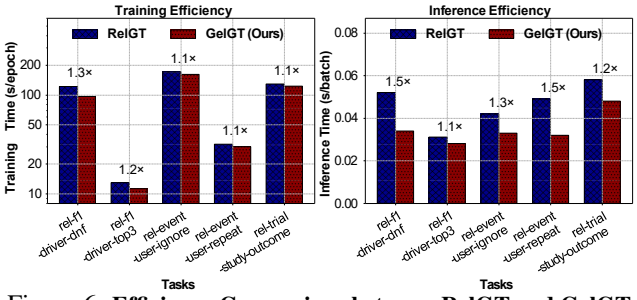

Figure 6: **Efficiency Comparison between RelGT and GelGT**

This advantage does not come at the cost of attention computation efficiency: at the 300-node context we use, our attention costs only 1.07× the epoch time of FlashAttention (Table 10). Furthermore, we evaluate our sampling efficiency in comparison with naive Random Sampling. Empirical results from Table 9 demonstrate that combining Breadth-First Search (BFS) with semantic similarity-based refinement introduces negligible latency, maintaining a computational cost highly comparable to Random Sampling. Appendix D further analyzes sampling latency, attention cost, and memory consumption under increasing context sizes.

Table 9: **Comparison of Per-Node Sampling Time between Random Sampling and GelGT**

Task	**Random Sampling**	**Ours**
rel-trial-study-outcome	14ms	14ms
rel-ratebeer-beer_rating-total_score	87ms	88ms

Table 10: **Scalability Comparison between Our Proposed Attention and FlashAttention (Dao et al., 2022) across varying Sampled Node Sizes.**

Context Size	**Ours**		**FlashAttention**	
	Epoch Time	**GPU Mem**	**Epoch Time**	**GPU Mem**
Size 300	225ms	2.2GB	210ms	1.9GB
Size 3000	1376ms	70.0GB	1004ms	53.9GB
Size 30000	— OOM —	— OOM —	— OOM —	— OOM —

5 Conclusion

We present GelGT, a Gaussian Relational Graph Transformer that improves long-range relational modeling by determining both which information to retain and how it should contribute to prediction. GelGT combines structure-semantic collaborative sampling to preserve relevant relational information with a Gaussian temporal bias to model its temporal focus and range. Experiments on 7 real-world relational datasets covering 21 prediction tasks demonstrate consistent improvements over existing relational graph learning methods.

AI USE STATEMENT

In this work, generative AI tools were only used to assist in language editing to improve readability. All AI-assisted text and code were thoroughly reviewed and verified by the authors, who take full responsibility for the entire content and integrity of this submission.

ETHICS STATEMENT

This paper focuses on foundational machine learning methodology and relies exclusively on publicly available benchmark datasets that do not involve personally identifiable information or human subjects. We foresee no immediate negative societal impacts or security risks resulting directly from this work, and all research was conducted in strict adherence to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we have provided the complete source code and scripts in the supplementary material. Detailed instructions for setting up the environment and running the experiments are included in the attached README file.

REFERENCES

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, 2021.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, 2016.

Michele Benzi and Christine Klymko. On the limiting behavior of parameter-dependent network centrality measures. *SIAM Journal on Matrix Analysis and Applications*, 2013.

Harald Bohr. *Almost periodic functions*. Courier Dover Publications, 2018.

Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: graph neural diffusion. In *ICML*, 2021.

Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Tien-Hao Chang. Trompt: Towards a better deep neural network for tabular data. In *ICML*, 2023.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

Tianlang Chen, Charilaos I. Kanatsoulis, and Jure Leskovec. Relgnn: Composite message passing for relational deep learning. In *ICML*, 2025.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pp. 785–794, 2016.

E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.

Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. *CoRR*, abs/2002.02046, 2020.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *NeurIPS*, 2022.

Zezhong Ding, Yongan Xiang, Shangyou Wang, Xike Xie, and S Kevin Zhou. Play like a vertex: A stackelberg game approach for streaming graph partitioning. *Sigmod*, 2024.

Vijay Prakash Dwivedi, Charilaos I. Kanatsoulis, Shenyang Huang, and Jure Leskovec. Relational deep learning: Challenges, foundations and next-generation architectures. In *KDD*, pp. 5999–6009, 2025.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico Lopez, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. In *ICLR*, 2026.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational deep learning - graph representation learning on relational databases. In *ICML*, 2024.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS*, 2022.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *SIGKDD,*, 2016.

Justin Gu, Rishabh Ranjan, Charilaos Kanatsoulis, Haiming Tang, Martin Jurkovic, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, and Jure Leskovec. Relbench v2: A large-scale benchmark and repository for relational data, 2026.

Maosheng Guo, Yu Zhang, and Ting Liu. Gaussian transformer: A lightweight approach for natural language inference. In *AAAI*, 2019.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

J A Hanley and B J Mcneil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–843, 1983.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 2020.

D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *WWW*, pp. 2704–2710, 2020.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *DSAA*, pp. 1–10, 2015.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

Liran Katzir, Gal Elidan, and Ran El-Yaniv. Net-dnf: Effective deep modeling of tabular data. In *ICLR*, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NeurIPS*, 2017.

Divyansha Lachi, Mahmoud Mohammadi, Joe Meyer, Vinam Arora, Tom Palczewski, and Eva L Dyer. Integrating temporal and structural context in graph transformers for relational deep learning. *arXiv preprint arXiv:2511.04557*, 2025.

Hoang Thanh Lam, Beat Buesser, Hong Min, Tran Ngoc Minh, Martin Wistuba, Udayan Khurana, Gregory Bramble, Theodoros Salonidis, Dakuo Wang, and Horst Samulowitz. Automated data science for relational data. In *ICDE*, pp. 2689–2692, 2021.

Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM*, 2020.

Jin Li, Zezhong Ding, and Xike Xie. Duetgraph: Coarse-to-fine knowledge graph reasoning with dual-pathway global-local fusion. In *NeurIPS*, 2025.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Dipak Meher and Carlotta Domeniconi. Inside core-kg: Evaluating structured prompting and coreference resolution for knowledge graphs, 2025.

Filippo Menczer, Gautam Pant, and Padmini Srinivasan. Topical web crawlers: Evaluating adaptive algorithms. *ACM Trans. Internet Techn.*, 2004.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Karthir Prabhakar, Sang Min Oh, Ping Wang, Gregory D. Abowd, and James M. Rehg. Temporal causality for the analysis of visual events. In *CVPR*, 2010.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *ICLR*, 2022.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *ICLR*, 2026.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. Relbench: A benchmark for deep learning on relational databases. In *NeurIPS*, 2024.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81:84–90, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Andreas Voskou, Charalambos Christoforou, and Sotirios Chatzis. Transformers with stochastic competition for tabular data modelling. In *ICML Workshop*, 2024.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. Heterogeneous graph attention network. In *WWW*, 2019.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In *ICML*, 2025.

Fang Wu, Vijay Prakash Dwivedi, and Jure Leskovec. Large language models are good relational learners. In *ACL*, pp. 7835–7854, 2025.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *NeurIPS*, 2022.

Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In *NeurIPS*, 2021.

Jiaxuan You, Jonathan Michael Gomes Selman, Rex Ying, and Jure Leskovec. Identity-aware graph neural networks. In *AAAI*, 2021.

Yiwen Yuan, Zecheng Zhang, Xinwei He, Akihiro Nitta, Weihua Hu, Manan Shah, Blaz Stojanovic, Shenyang Huang, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. Contextgnn: Beyond two-tower recommendation systems. In *ICLR*, 2025.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *ICLR*, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *NeurIPS*, 2020.

Jiaming Zhuo, Yuwei Liu, Yintong Lu, Ziyi Ma, Kun Fu, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Dualformer: Dual graph transformer. In *ICLR*, 2025.

CONTENTS

1 Introduction **1**

2 Preliminaries **2**

3 Gaussian Relational Graph Transformer (GelGT) **3**

3.1 Structure-Semantic Collaborative Sampling 3

3.2 Gaussian Temporal Bias-based Attention 5

4 Experiments **6**

4.1 Experimental Setup 6

4.2 Performance (**RQ1**) 6

4.3 Component Study (**RQ2**) 7

4.4 Configuration Analysis (**RQ3**) 8

4.5 Efficiency Analysis (**RQ4**) 9

5 Conclusion **9**

A Proofs of Theorems. **16**

A.1 Upper Bound of Relative Structural Loss 16

A.2 Exponential Decay of Structural Sensitivity 17

A.3 Enhancement of Semantic Relevance-to-Noise Ratio 17

A.4 Convergence of Temporal Center 19

A.5 Attention Score Ratio Enhancement 20

B Experimental Details. **22**

B.1 Training Setting 22

B.2 Benchmark Details. 22

B.3 Datasets 23

B.4 Tasks 23

B.5 Hyperparameter Settings 24

B.6 Hardcore Configurations 26

C Additional Baselines and Datasets **27**

C.1 Comparison with Recent Graph Transformers and Relational Foundation Models . 27

D Additional Efficiency Experiments **28**

D.1 Sampling Efficiency 28

D.2 Attention Efficiency 28

D.3 Efficiency on Extreme Graph Structures 29

E Detailed Description of Encoder Modules **30**

F Additional Analysis and Discussion **31**

F.1 Semantic Refinement Design 31

F.2 Generality of the Sampling Strategy 31

F.3 Homophily vs. Heterophily in Relational Graphs 32

F.4 Analysis of Gaussian Temporal Bias 33

F.5 Additional Discussion on Design Choices 34

F.6 Sampling Budget Analysis 36

F.7 Stability of Gaussian Bias Parameters 36

G Related Work **38**

H Limitations and Broader Impacts **38**

H.1 Limitations 38

H.2 Broader Impacts 38

A PROOFS OF THEOREMS.

A.1 UPPER BOUND OF RELATIVE STRUCTURAL LOSS

Proof. To characterize the multi-hop topological structure of nodes in the graph, we employ Katz centrality (Katz, 1953) as our analytical tool. As a classic walk-based topological metric, Katz centrality systematically captures the structural reach of a node by computing a weighted sum of all walks of varying lengths associated with it. Formally, let A be the adjacency matrix of the graph, and λ be the attenuation factor determining the weight decay of paths. Let $\mathbf{C}^{\mathcal{G}}$ denote the vector of Katz centrality for all nodes in the full graph, and let $\mathbf{1}$ be the vector of all ones. The matrix-vector form of Katz centrality is given by the Neumann series:

$$\mathbf{C}^{\mathcal{G}} = \sum_{k=1}^{\infty} (\lambda A)^k \mathbf{1}. \tag{10}$$

The structural integrity sampling strategy preserves all walks of length $k = 1$ and $k = 2$. The structural loss vector $\boldsymbol{\Delta C}$, which consists of all walks of length $k \geq 3$, can be expressed as:

$$\boldsymbol{\Delta C} = \sum_{k=3}^{\infty} (\lambda A)^k \mathbf{1}. \tag{11}$$

We observe a critical recursive relationship between the error vector and the global centrality vector. By factoring out $(\lambda A)^2$ from the error series:

$$\boldsymbol{\Delta C} = (\lambda A)^2 \left(\sum_{k=1}^{\infty} (\lambda A)^k \mathbf{1} \right) = (\lambda A)^2 \mathbf{C}^{\mathcal{G}}. \tag{12}$$

To quantify the relative magnitude of the loss, we employ the matrix 2-norm (spectral norm). Using the consistency property of matrix norms ($\|M\mathbf{x}\| \leq \|M\|\|\mathbf{x}\|$), we have:

$$\|\boldsymbol{\Delta C}\|_2 = \|(\lambda A)^2 \mathbf{C}^{\mathcal{G}}\|_2 \leq \|(\lambda A)^2\|_2 \|\mathbf{C}^{\mathcal{G}}\|_2. \tag{13}$$

Using the property that the spectral norm of a symmetric matrix is its spectral radius, i.e., $\|A\|_2 = \rho(A)$, and considering the definition $\lambda = c/\rho(A)$:

$$\|(\lambda A)^2\|_2 \leq (\lambda \|A\|_2)^2 = (\lambda \rho(A))^2 = \left(\frac{c}{\rho(A)} \cdot \rho(A) \right)^2 = c^2. \tag{14}$$

Substituting this bound back into the inequality yields:

$$\|\boldsymbol{\Delta C}\|_2 \leq c^2 \|\mathbf{C}^{\mathcal{G}}\|_2. \tag{15}$$

Rearranging the terms, we obtain the strict upper bound for the relative structural loss:

$$\frac{\|\boldsymbol{\Delta C}\|_2}{\|\mathbf{C}^{\mathcal{G}}\|_2} \leq c^2. \tag{16}$$

Following the analysis of the local regime by Benzi and Klymko (Benzi & Klymko, 2013), the scaling factor c falls within the range of ≤ 0.1.

Substituting this value into Theorem 1, the relative structural loss is upper-bounded by:

$$\frac{\|\boldsymbol{\Delta C}\|_2}{\|\mathbf{C}^{\mathcal{G}}\|_2} \leq (0.1)^2 = 1\%. \tag{17}$$

This confirms that under standard operating conditions, the structural integrity sampling preserves at least 99% of the effective structural information.

A.2 Exponential Decay of Structural Sensitivity

proof. To justify the differentiated pruning strategy in Stage 2, we analyze the sensitivity of the seed node's structural information to the removal of a specific neighbor at distance k. We define the structural sensitivity $\Delta\mathcal{C}(v, k)$ as the magnitude of change in the Katz centrality of v when a node u at distance k (i.e., $\text{dist}(v, u) = k$) is removed from the graph.

Based on the definition of Katz centrality, the contribution of a node u to v is the weighted sum of all walks between them:

$$\text{Contr}(u \to v) = \sum_{\gamma: u \rightsquigarrow v} \beta^{|\gamma|}, \tag{18}$$

where γ represents a walk from u to v, $\beta \in (0, 1)$ is the attenuation factor determining the weight decay of paths and $|\gamma|$ is its length.

When a neighbor u at distance k is pruned, all walks passing through u are eliminated. The dominant term in this summation corresponds to the shortest path (geodesic) of length k. Higher-order paths (length $> k$) also contribute, but their weights are dampened by higher powers of β. Therefore, the structural loss is dominated by the leading term:

$$\Delta\mathcal{C}_{k-hop}(v) \approx \beta^k \cdot N_{\text{paths}}(k) + \mathcal{O}(\beta^{k+1}), \tag{19}$$

where $N_{\text{paths}}(k)$ denotes the number of shortest paths of length k between v and u (typically small for local neighborhoods).

Now, we compare the sensitivity at $k = 1$ (direct neighbors) and $k = 2$ (two-hop neighbors):

$$\Delta\mathcal{C}_{1-hop}(v) = \beta \cdot 1 + \mathcal{O}(\beta^2) \sim \mathcal{O}(\beta), \tag{20}$$

$$\Delta\mathcal{C}_{2-hop}(v) = \beta^2 \cdot N_{\text{paths}}(2) + \mathcal{O}(\beta^3) \sim \mathcal{O}(\beta^2). \tag{21}$$

Note that for $k = 1$, there is exactly one direct edge, so the coefficient is 1.

The ratio of structural sensitivity between pruning a 2-hop neighbor and a 1-hop neighbor is:

$$\frac{\Delta\mathcal{C}_{2-hop}(v)}{\Delta\mathcal{C}_{1-hop}(v)} \approx \frac{\mathcal{O}(\beta^2)}{\mathcal{O}(\beta)} = \mathcal{O}(\beta). \tag{22}$$

Since the attenuation factor β is typically chosen to be small (satisfying $\beta < 1/\rho(A)$), this ratio indicates that the structural perturbation caused by removing a 2-hop neighbor is an order of magnitude smaller than that of removing a 1-hop neighbor. This theoretical result confirms that 1-hop neighbors form the non-negligible structural skeleton ($\mathcal{O}(\beta)$ sensitivity), whereas 2-hop neighbors allow for aggressive semantic pruning with bounded structural risk ($\mathcal{O}(\beta^2)$ sensitivity). □

A.3 Enhancement of Semantic Relevance-to-Noise Ratio

As we prove in Appendix A.2, pruning 2-hop neighbors has a much smaller impact on structural information compared to 1-hop neighbors. Therefore, we apply the following semantic refinement exclusively to 2-hop neighbors.

Proof. The features of the sampled neighbors are aggregated to obtain a representation for the seed node, which is then used for downstream tasks. We define a general aggregation formula as follows:

$$\mathbf{h}_{agg} = \sum_{u \in \mathcal{N}(v)} w(v, u)\mathbf{h}_u \tag{23}$$

In this formulation, v represents the seed node, u is a neighbor of v, $\mathbf{h}_u$ denotes the feature of neighbor u, $w(v, u)$ is the aggregation weight, and $\mathbf{h}_{\text{agg}}$ corresponds to the aggregated feature.

We can decompose the feature of neighbor u, $\mathbf{h}_u$, into a part that is semantically relevant to the seed node and a part that corresponds to irrelevant noise:

$$\mathbf{h}_u = \mu_{vu}\mathbf{h}_v + \mathbf{h}_u^{\text{noise}} \tag{24}$$

The first term corresponds to the part of $\mathbf{h}_u$ that is semantically relevant to the seed node v, with μ_{vu} representing the degree of relevance. The second term represents the part unrelated to v, i.e., random noise (e.g., heterophily), assumed to have variance σ^2_{noise}.

By substituting $\mathbf{h}_u$ into the aggregation formula, we can decompose the aggregated feature into two parts:

$$\mathbf{h}_{\text{agg}} = \sum_{u \in \mathcal{N}(v)} w(v,u)\mathbf{h}_u = \underbrace{\sum_{u \in \mathcal{N}(v)} w(v,u)\mu_{vu}\mathbf{h}_v}_{\text{semantically relevant}} + \underbrace{\sum_{u \in \mathcal{N}(v)} w(v,u)\mathbf{h}_u^{\text{noise}}}_{\text{noise}} \tag{25}$$

Following the theoretical analysis in PairNorm (Zhao & Akoglu, 2020), which establishes that the information content of node features is strictly proportional to their squared norm, we measure the magnitude of the aggregated information using the L_2-norm. Therefore, the aggregated semantically relevant information is quantified as:

$$\mathcal{I}_{\text{relevance}} = \left\| \left(\sum_{u \in \mathcal{N}(v)} w(v,u)\,\mu_{vu} \right) \mathbf{h}_v \right\|^2 = \left(\sum_{u \in \mathcal{N}(v)} w(v,u)\mu_{uv} \right)^2 \cdot \|\mathbf{h}_v\|^2 \tag{26}$$

The variance directly measures the destructive information or uncertainty (Kendall & Gal, 2017) introduced by irrelevant neighbors. Therefore, we compute the semantically noisy information as:

$$\mathcal{I}_{\text{noise}} = \sum_{u \in \mathcal{N}(v)} \text{Var}(w(v,u)\mathbf{h}_u^{\text{noise}}) \tag{27}$$

Using the properties of variance, we can factor out the weight $w(v,u)$ to obtain:

$$\mathcal{I}_{\text{noise}} = \sum_{u \in \mathcal{N}(v)} w(v,u)^2\,\text{Var}(\mathbf{h}_u^{\text{noise}}) \tag{28}$$

If the variance of each individual noise is σ^2_{noise}:

$$\mathcal{I}_{\text{noise}} = \sum_{u \in \mathcal{N}(v)} w(v,u)^2\sigma^2_{\text{noise}} = \sigma^2_{\text{noise}} \sum_{u \in \mathcal{N}(v)} w(v,u)^2 \tag{29}$$

Therefore, the ratio of relevant information to noise information is:

$$\text{SNR} = \frac{\mathcal{I}_{\text{relevance}}}{\mathcal{I}_{noise}} = \frac{\left(\sum_{u \in \mathcal{N}(v)} w(v,u)\mu_{uv}\right)^2 \cdot \|\mathbf{h}_v\|^2}{\sigma^2_{\text{noise}} \sum_{u \in \mathcal{N}(v)} w(v,u)^2} = \underbrace{\frac{\|\mathbf{h}_v\|^2}{\sigma^2_{\text{noise}}}}_{\text{constant}} \cdot \frac{\left(\sum_{u \in \mathcal{N}(v)} w(v,u)\mu_{uv}\right)^2}{\sum_{u \in \mathcal{N}(v)} w(v,u)^2} \tag{30}$$

To demonstrate the effectiveness of our semantic refinement, we denote the refined 2-hop neighbors by $\mathcal{N}_{refine}(v)$ and the remaining subsets is denoted as $\mathcal{N}_{remaining}(v)$.

Let A and B denote the relevance term and noise term in SNR of our refinement strategy, respectively:

$$A = \sum_{u \in \mathcal{N}_{refine}(v)} w(v,u)\mu_{vu}, \quad B = \sum_{u \in \mathcal{N}_{refine}(v)} w(v,u)^2. \tag{31}$$

Similarly, for the remaining subset $\mathcal{N}_{remaining}(v)$, we define the relevance term a and noise term b:

$$a = \sum_{u \in \mathcal{N}_{remaining}(v)} w(v,u)\mu_{vu}, \quad b = \sum_{u \in \mathcal{N}_{remaining}(v)} w(v,u)^2 \tag{32}$$

Nodes in $\mathcal{N}_{remaining}(v)$ have negligible correlation with the seed node, implying $\mu_{vu} \approx 0$ and thus $a \approx 0$. However, since these nodes objectively exist in the graph, they contribute a strictly positive noise term, i.e., $b > 0$.

Now, we compare the SNR of our refinement strategy (SNR_{post}) against the full 2-hop neighborhood strategy (SNR_{pre}):

$$\text{SNR}_{after} = \frac{\mathcal{I}^{after}_{\text{relevance}}}{\mathcal{I}^{after}_{\text{noise}}} = \frac{\|\mathbf{h}_v\|^2}{\sigma^2_{\text{noise}}} \cdot \frac{A^2}{B} \tag{33}$$

$$\text{SNR}_{before} = \frac{\mathcal{I}^{before}_{\text{relevance}}}{\mathcal{I}^{before}_{\text{noise}}} = \frac{\|\mathbf{h}_v\|^2}{\sigma^2_{\text{noise}}} \cdot \frac{(A+a)^2}{B+b} \approx \frac{\|\mathbf{h}_v\|^2}{\sigma^2_{\text{noise}}} \cdot \frac{A^2}{B+b}. \tag{34}$$

Since b is positive and B is positive, the denominator of the full 2-hop neighborhood strategy is strictly larger while the numerator remains approximately unchanged. This leads to the inequality:

$$\frac{A^2}{B+b} < \frac{A^2}{B} \implies \text{SNR}_{before} < \text{SNR}_{after} \implies \frac{\mathcal{I}^{before}_{\text{relevance}}}{\mathcal{I}^{before}_{\text{noise}}} < \frac{\mathcal{I}^{after}_{\text{relevance}}}{\mathcal{I}^{after}_{\text{noise}}}. \tag{35}$$

This inequality mathematically guarantees that pruning low-similarity neighbors effectively removes the noise denominator b without sacrificing the relevance term numerator A, thereby strictly maximizing the ratio of relevant information to noise information of the aggregated representation. Consequently, this result rigorously confirms that our semantic refinement strategy functions as an effective denoising filter, eliminating irrelevant nodes to enhance the quality of node representation. □

A.4 Convergence of Temporal Center

In this subsection, we provide a rigorous theoretical analysis demonstrating that the proposed attention mechanism is able to focus more effectively on temporally relevant nodes compared to existing attention mechanisms.

For the model to attend to relevant historical information, the learnable temporal center μ must converge to the ground-truth relevant timestamp t^*.

proof. Consider a simplified objective where the model aims to maximize the alignment score $\mathcal{S}$ between a query and a ground-truth relevant event at time t^*. The optimization follows the gradient ascent direction $\nabla_\mu \mathcal{S}$.

The baseline attention logit is derived from the dot product of sinusoidal encodings:

$$\mathcal{S}_{\text{base}}(t^*) \approx \sum_{j=1}^{d/2} \cos(\omega_j(t^* - \mu)). \tag{36}$$

The gradient with respect to μ is:

$$\frac{\partial \mathcal{S}_{\text{base}}}{\partial \mu} = \sum_{j=1}^{d/2} \omega_j \sin(\omega_j(t^* - \mu)). \tag{37}$$

This gradient function is a superposition of high-frequency sine waves. Its sign flips repeatedly as the distance $|t^* - \mu|$ changes. If the initialization of μ is far from t^*, the gradient direction is determined by local periodic behavior rather than the global distance to t^*. Consequently, the optimization is prone to getting trapped in local optima (false peaks) nearest to the initialization.

Our method incorporates an adaptive bias term $\mathcal{B}(t^*) = \exp\left(-\frac{(t^*-\mu)^2}{2\sigma^2}\right)$. The gradient contribution from this term is:

$$\frac{\partial \mathcal{B}}{\partial \mu} = \mathcal{B}(t^*) \cdot \frac{t^* - \mu}{\sigma^2}. \tag{38}$$

Let $\Delta = t^* - \mu$ be the alignment error. The sign of the gradient is determined by $\text{sgn}(\Delta)$:

- If $\mu < t^*$ ($\Delta > 0$), the gradient is positive, pushing μ towards t^*.
- If $\mu > t^*$ ($\Delta < 0$), the gradient is negative, pushing μ towards t^*.

Unlike the baseline, the term $(t^* - \mu)$ acts as a linear **restoring force**. It provides a consistent global gradient direction that guides μ towards t^*, smoothing the optimization landscape and significantly expanding the basin of attraction for the true timestamp.

A.5 Attention Score Ratio Enhancement

Building upon the analysis in Appendix A.4, which guarantees that the learnable temporal center μ converges to the ground-truth relevant timestamp t^*, we now prove that the proposed attention mechanism effectively concentrates attention scores on these relevant nodes. Specifically, we demonstrate that our method enforces a strict probabilistic preference for events temporally close to t^*, suppressing distant noise that the baseline model fails to filter.

Proof. Let t^* be the ground-truth relevant timestamp. Based on the gradient guidance property proved in Appendix A.4, we assume the model has successfully learned the optimal temporal reference, i.e., $\mu \approx t^*$.

Consider a relevant event e_{rel} and an irrelevant noise event e_{noise}. We define their respective temporal differences relative to the ground-truth timestamp t^* as:

$$\delta_{\text{relevant}} = t_{\text{relevant}} - t^*, \tag{39}$$

and

$$\delta_{\text{noise}} = t_{\text{noise}} - t^*. \tag{40}$$

The relevant event is temporally proximal to the target t^*, while the noise event is distant. This structural difference is formalized by the inequality of their magnitudes:

$$|\delta_{\text{relevant}}| \ll |\delta_{\text{noise}}|. \tag{41}$$

The baseline attention logit is $\mathcal{S}_{\text{base}}(\delta) = \sum \cos(\omega_j \delta)$. According to Bohr's theorem on almost periodic functions (Bohr, 2018), $\mathcal{S}_{\text{base}}$ does not vanish at infinity. Specifically:

$$\limsup_{|\delta|\to\infty} \mathcal{S}_{\text{base}}(\delta) = \sup_{\delta\in\mathbb{R}} \mathcal{S}_{\text{base}}(\delta). \tag{42}$$

This implies that there exist infinitely many distant timestamps t_{noise} (where $|\delta_{noise}| \to \infty$) such that:

$$\mathcal{S}_{\text{base}}(t_{noise}) \approx \mathcal{S}_{\text{base}}(t_{relevant}). \tag{43}$$

Consequently, the baseline model cannot structurally guarantee that a relevant node receives a higher score than a distant noise node. The discriminative margin $\Delta_{\text{base}} = \mathcal{S}_{\text{base}}(t_{relevant}) - \mathcal{S}_{\text{base}}(t_{noise})$ has a lower bound of approximately 0 (or even negative values due to oscillations), indicating a failure to distinguish relevance from periodic noise.

In our method, with $\mu \approx t^*$, the Gaussian bias becomes $\mathcal{B}(\delta) = \exp(-\delta^2/2\sigma^2)$. The total logit is $\mathcal{S}_{\text{ours}}(\delta) = \mathcal{S}_{\text{base}}(\delta) + \mathcal{B}(\delta)$.

The discriminative margin between the relevant event and the noise event is:

$$\Delta_{\text{ours}} = \mathcal{S}_{\text{ours}}(t_{relevant}) - \mathcal{S}_{\text{ours}}(t_{noise}) = \Delta_{\text{base}} + \left[\mathcal{B}(\delta_{relevant}) - \mathcal{B}(\delta_{noise})\right]. \tag{44}$$

Since $|\delta_{relevant}| \ll |\delta_{noise}|$, and the Gaussian function is strictly monotonically decreasing with respect to distance from the center, we have:

$$\mathcal{B}(\delta_{relevant}) \gg \mathcal{B}(\delta_{noise}) \approx 0. \tag{45}$$

Even in the worst-case scenario for the baseline (where $\Delta_{\text{base}} \leq 0$ due to periodic misalignment), the Gaussian term introduces a strictly positive margin enhancement.

Specifically, considering the asymptotic limit where the noise becomes infinitely distant ($|\delta_{noise}| \to \infty$) and the relevant event is perfectly aligned with the learned center ($\delta_{rel} \to 0$), we have:

$$\lim_{|\delta_{noise}|\to\infty} \mathcal{B}(\delta_{noise}) = 0 \quad \text{and} \quad \lim_{\delta_{relevant}\to 0} \mathcal{B}(\delta_{relevant}) = 1. \tag{46}$$

This imposes a strict exponential penalty on the noise in the score ratio:

$$\frac{\alpha^{\text{w/ G}}_{noise}}{\alpha^{\text{w/ G}}_{relevant}} = \exp\left(\mathcal{S}_{\text{base}}(noise) - \mathcal{S}_{\text{base}}(relevant) - 1\right) = e^{-1} \cdot \frac{\alpha^{\text{w/o G}}_{noise}}{\alpha^{\text{w/o G}}_{relevant}}. \tag{47}$$

where α_{noise} and $\alpha_{relevant}$ denote the attention scores for temporal noise and related nodes, respectively; $\mathbf{e}$ is Euler's number; and w/ G" and w/o G" distinguish the attention mechanisms with and without Gaussian bias.

Combining the result that μ converges to t^* (Appendix A.4) with the Gaussian decay property, we prove that our method effectively assigns significantly higher attention scores to strictly temporally relevant nodes. Unlike the baseline, which is susceptible to distant "periodic ghosts," our mechanism imposes a structural envelope that filters out irrelevant temporal information, ensuring that the model's focus remains robustly on the learned temporal context.

B EXPERIMENTAL DETAILS.

B.1 TRAINING SETTING

GelGT is trained in an end-to-end manner without any pre-training stage to ensure a fair comparison with baseline methods such as RelGT. We do not adopt pre-training in our setting for two reasons: (i) it introduces significant computational overhead, and (ii) it often fails to transfer effectively across relational graphs (e.g., Griffin still underperforms even after fine-tuning). In contrast, direct supervised training allows the model to better adapt to the target relational structure, leading to more stable and competitive performance.

B.2 BENCHMARK DETAILS.

In this section, we present a comprehensive overview of the datasets and evaluation tasks sourced from RelBench (Robinson et al., 2024). RelBench encompasses seven datasets spanning diverse relational database domains, such as e-commerce, clinical records, social networks, and sports. These datasets are meticulously curated from their respective domains, with training set sizes ranging from 1.3K to 5.4M records, totaling 47M training instances. Each dataset supports various predictive tasks, including modeling user engagement with advertisements within a four-day window or forecasting the primary outcomes of clinical trials over a year.

Collectively, RelBench defines 30 tasks categorized into entity classification, entity regression, and recommendation. Our evaluation focuses on 21 classification and regression tasks, as GelGT is primarily designed as a node representation learning framework for Relational Deep Learning (RDL). Following pror work (Dwivedi et al., 2026), we exclude recommendation tasks in this study due to their specific requirements, such as target node identification (You et al., 2021) or the adoption of pair-wise learning architectures (Yuan et al., 2025). Detailed statistics for all evaluated datasets and tasks are summarized in Table 11.

Table 11: Dataset and task statistics from RelBench (Robinson et al., 2024) used for our evaluation.

Dataset	Task name	Task type	#Rows of training table			#Unique	%train/test	#Dst
			Train	Validation	Test	Entities	Entity Overlap	Entities
rel-amazon	user-churn	entity-cls	4,732,555	409,792	351,885	1,585,983	88.0	—
	item-churn	entity-cls	2,559,264	177,689	166,842	416,352	93.1	—
	user-ltv	entity-reg	4,732,555	409,792	351,885	1,585,983	88.0	—
	item-ltv	entity-reg	2,707,679	166,978	178,334	427,537	93.5	—
	user-item-purchase	recommendation	5,112,803	351,876	393,985	1,632,909	87.4	12,562,384
	user-item-rate	recommendation	3,667,157	257,939	292,609	1,481,360	81.0	7,665,611
	user-item-review	recommendation	2,324,177	116,970	127,021	894,136	74.1	5,406,835
rel-avito	ad-ctr	entity-reg	5,100	1,766	1,816	4,997	59.8	—
	user-clicks	entity-cls	59,454	21,183	47,996	66,449	45.3	—
	user-visits	entity-cls	86,619	29,979	36,129	63,405	64.6	—
	user-ad-visit	recommendation	86,616	29,979	36,129	63,402	64.6	3,616,174
rel-event	user-attendance	entity-reg	19,261	2,014	2,006	9,694	14.6	—
	user-repeat	entity-cls	3,842	268	246	1,514	11.5	—
	user-ignore	entity-cls	19,239	4,185	4,010	9,799	21.1	—
rel-f1	driver-dnf	entity-cls	11,411	566	702	821	50.0	—
	driver-top3	entity-cls	1,353	588	726	134	50.0	—
	driver-position	entity-reg	7,453	499	760	826	44.6	—
rel-hm	user-churn	entity-cls	3,871,410	76,556	74,575	1,002,984	89.7	—
	item-sales	entity-reg	5,488,184	105,542	105,542	105,542	100.0	—
	user-item-purchase	recommendation	3,878,451	74,575	67,144	1,004,046	89.2	13,428,473
rel-stack	user-engagement	entity-cls	1,360,850	85,838	88,137	88,137	97.4	—
	user-badge	entity-cls	3,386,276	247,398	255,360	255,360	96.9	—
	post-votes	entity-reg	2,453,921	156,216	160,903	160,903	97.1	—
	user-post-comment	recommendation	21,239	825	758	11,453	59.9	44,940
	post-post-related	recommendation	5,855	226	258	5,924	8.5	7,456
rel-trial	study-outcome	entity-cls	11,994	960	825	13,779	0.0	—
	study-adverse	entity-reg	43,335	3,596	3,098	50,029	0.0	—
	site-success	entity-reg	151,407	19,740	22,617	129,542	42.0	—
	condition-sponsor-run	recommendation	36,934	2,081	2,057	3,956	98.4	533,624
	site-sponsor-run	recommendation	669,310	37,003	27,428	445,513	48.3	1,565,463

B.3 DATASETS

In this section, we include the details on the datasets in RelBench (Robinson et al., 2024) which we use for our evaluation.

- `rel-amazon`. The Amazon E-commerce dataset consists of product details, user information, and review interactions from Amazon's platform, including metadata like pricing and categories, along with review ratings and content.
- `rel-avito`. Avito's marketplace dataset contains search queries, advertisement characteristics, and contextual information from this major online trading platform that facilitates transactions across various categories including real estate and vehicles.
- `rel-event`. The Event Recommendation dataset from Hangtime mobile app tracks users' social planning, capturing interactions, event details, demographic data, and social connections to reveal how relationships impact user behavior.
- `rel-f1`. The F1 dataset provides comprehensive Formula 1 racing information since 1950, documenting drivers, constructors, manufacturers, and circuits with detailed records of race results, standings, and specific data on various racing sessions and pit stops.
- `rel-hm`. H&M's dataset contains customer-product interactions from their e-commerce platform, featuring customer demographics, product descriptions, and purchase histories.
- `rel-stack`. The Stack Exchange dataset documents activity from this network of Q&A websites, including user biographies, posts, comments, edits, votes, and question relationships where users earn reputation through contributions.
- `rel-trial`. The clinical trial dataset from the AACT initiative has study protocols and outcomes, containing trial designs, participant information, intervention details, and results metrics, serving as a key resource for medical research.

Temporal information in RelBench. We clarify that the temporal information follows the standard RelBench (Robinson et al., 2024) setting. We address four aspects explicitly.

What temporal information means. Each RelBench prediction sample has a seed time, which indicates when the prediction is made. Historical events or interactions may also have their own timestamps. GelGT uses the temporal relation between a historical neighbor's timestamp and the seed time.

Whether it is provided by the dataset. Yes. RelBench task tables provide Entity ID, seed time, and target label. For example, `rel-event/user-repeat` contains `user` and `timestamp`, `rel-avito/user-clicks` contains `UserID` and `timestamp`, and `rel-f1/driver-dnf` contains `driverId` and `date`.

Whether all nodes have timestamps. No. Only time-aware event or interaction tables have time columns, such as `events.start_time`, `event_attendees.start_time`, `VisitStream.ViewDate`, `results.date`, and `studies.start_date`. Static entity tables such as `UserInfo` may not have timestamps.

Single or multiple timestamps. For a single event or interaction record, the temporal information is usually one timestamp. An entity can be connected to multiple historical records, so it may be associated with multiple timestamps through its interactions.

How GelGT uses it. The seed time filters future records and prevents temporal leakage. For the retained historical neighbors, GelGT computes their time interval to the seed time. The Gaussian temporal bias then uses this interval to model whether a historical neighbor is temporally relevant to the current prediction.

B.4 TASKS

The following entity classification and regression tasks are defined in RelBench for the above datasets.

1. **rel-amazon**

(a) `user-churn`: Predict whether a user will discontinue reviewing products within the next three months.
(b) `item-churn`: Predict if a product will have no reviews in the next three months.
(c) `user-ltv`: Estimate the total monetary value of merchandise in dollars that a user will purchase and review within the next three months.
(d) `item-ltv`: Estimate the total monetary value of purchases and reviews a product will receive during the next three months.

2. **rel-avito**
   (a) `user-visits`: Predict if a user will engage with several (advertisements) ads within the upcoming four days.
   (b) `user-clicks`: Predict whether a user will interact with multiple ads through clicking within the upcoming four days.
   (c) `ad-ctr`: Estimate the interaction probability for an ad, assuming it receives an interaction within four days.
3. **rel-event**
   (a) `user-attendance`: Estimate the number of events a user will confirm attendance to (RSVP yes or maybe) within the upcoming seven days.
   (b) `user-repeat`: Predict whether a user will join an event (RSVP yes or maybe) within the upcoming seven days, provided they attended an event during the previous fourteen days.
   (c) `user-ignore`: Predict whether a user will disregard or ignore more than two events invitations within the upcoming seven days.
4. **rel-f1**
   (a) `driver-dnf`: Predict if a driver will not finish a race within the upcoming month.
   (b) `driver-top3`: Determine if a driver will achieve a top-three qualifying position in a race within the upcoming month.
   (c) `driver-position`: Estimate a driver's average finishing placement across all races in the upcoming two months.
5. **rel-hm**
   (a) `user-churn`: Predict whether a customer will not perform any transactions in the upcoming week.
   (b) `item-sales`: Estimate total revenue generated by a product in the upcoming week.
6. **rel-stack**
   (a) `user-engagement`: Predict whether a user will contribute through voting, posting, or commenting within the upcoming three months.
   (b) `user-badge`: Predict whether a user will secure a new badge within the upcoming three months.
   (c) `post-votes`: Estimate the number of votes a user's post will accumulate over the upcoming three months.
7. **rel-trial**
   (a) `study-outcome`: Predict whether a clinical trial will achieve its principal outcome within the upcoming year.
   (b) `study-adverse`: Estimate the number of patients who will experience significant adverse effects or mortality in a clinical trial over the upcoming year.
   (c) `site-success`: Estimate the success rate of a clinical trial site in the upcoming year.

B.5 Hyperparameter Settings

Sampling Module. In our experimental setup, to accommodate the varying demands for neighborhood information density across different downstream tasks, we employ a two-stage neighborhood sampling strategy.

We implement differentiated configurations for sampling sizes based on task characteristics. For the node classification task, we initially sample 300 candidate neighbors in the first stage, followed by

Table 12: Detailed hyperparameter settings and implementation specifications.

Hyperparameter	Value	Description / Note
Model Architecture		
Hidden Dimension (d)	512	Embedding size for all encoders
Global Layers (L_{global})	4	Number of model layers (from script)
Local GNN Depth	3	Layers within internal GNNStack
Attention Heads	4	Multi-head attention mechanism
Feed-forward Network (FFN) Ratio	2	Hidden dim expansion in FFN
Positional Encoding Dim	128	Learnable structural PE
Number of Centroids	4096	For vector quantization (VQ)
VQ Decay	0.99	EMA decay rate for codebook update
Optimization & Training		
Optimizer	Adam	Standard optimization algorithm
Learning Rate	1×10^{-4}	Base learning rate
Weight Decay	1×10^{-5}	L2 regularization coefficient
Batch Size	64	Adjusted for memory constraints
Dropout Rate	0.3	Applied to FFN and Attention
Max Training Steps	500	Number of steps per epoch
Total Epochs	10	Total training duration
Warmup Steps	10	Linear warmup strategy
Data Sampling & Processing		
Neighborhood Scope	2-hop	Maximum hop distance sampled
Classification Configuration	$300 \rightarrow 200$	Sample 300, select top-200
Regression Configuration	$500 \rightarrow 300$	Sample 500, select top-300

a filtering mechanism that retains 200 nodes as input. Conversely, for the node regression task—to address potentially more complex feature dependencies—we increase the initial sampling budget to 500 nodes, ultimately selecting 300 nodes for model training. This tailored configuration aims to strike an optimal balance between computational efficiency and model performance.

Model Module. Our proposed model adopts a unified hybrid architecture designed to capture both local structural topology and global temporal-semantic dependencies.

To effectively handle the heterogeneous and temporal nature of the relational graph, we employ a set of domain-specific encoders (Dwivedi et al., 2026) to project diverse input features into a unified latent space $\mathbb{R}^d$. Specifically, we utilize learnable embedding layers to encode node type heterogeneity and relative hop distances from the central node. Furthermore, continuous timestamps are processed via a `TimeEncoder` to preserve temporal dynamics, while raw tabular node features are encoded using a `TabularEncoder` based on feature statistics. To enhance subgraph structural awareness, a learnable positional encoding is also incorporated. All resulting embeddings are normalized via LayerNorm, concatenated, and projected to a dimension of $d = 512$ through a 2-layer MLP mixture network.

The core processing layer, `GelGTLayer`, processes the refined subgraph sequence X_{in} through two parallel branches to extract complementary features. The GNN Branch is designed to explicitly capture local neighborhood topology using a 3-layer GraphSAGE (Hamilton et al., 2017) backbone. Each layer applies message passing followed by LayerNorm (Ba et al., 2016), GELU (Hendrycks, 2016) activation, and Dropout ($p = 0.1$). Crucially, residual connections (He et al., 2016) are incorporated at each layer to facilitate gradient flow and prevent over-smoothing (Chen et al., 2020; Chamberlain et al., 2021; Wu et al., 2019; Li et al., 2018), yielding a structure-aware representation H_{gnn}.

Parallel to the GNN, the Attention Branch employs a Transformer-based `LocalModule` to capture long-range dependencies and temporal interactions. By utilizing Multi-Head Self-Attention (Heads=4) integrated with temporal embeddings, this branch allows the model to dynamically weigh neighbors based on both semantic similarity and temporal proximity, producing a context-aware representation H_{attn}.

To optimally combine the structural and semantic representations, we introduce a learnable gating parameter η. The final representation H_{final} is computed as a dynamic weighted sum:

$$H_{final} = \eta \cdot H_{attn} + (1 - \eta) \cdot H_{gnn} \tag{48}$$

where $\sigma(\cdot)$ denotes the Sigmoid function. This mechanism allows the model to adaptively balance the contribution of GNN and Transformer modules based on the specific requirements of the downstream task. Finally, a 2-layer MLP head generates the prediction.

The detailed hyperparameter configurations and implementation specifications are summarized in Table 12.

B.6 HARDCORE CONFIGURATIONS

We conduct all experiments with:

- Operating System: Ubuntu 20.04.6 LTS.
- CPU: Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz.
- GPU: NVIDIA Tesla A100 SMX4 with 80GB of Memory.
- Software: CUDA 12.1, Python 3.10.18, PyTorch (Paszke et al., 2019) 2.5.1.

C Additional Baselines and Datasets

To further validate the robustness, generalizability, and architectural superiority of the Gaussian Relational Graph Transformer (GelGT), we expand our empirical evaluation to include the highly challenging SALT dataset (Gu et al., 2026) and benchmark against a broader spectrum of emerging and advanced baseline models, including Rel-LLM (Wu et al., 2025), Nodeformer (Wu et al., 2022), RelGNN (Chen et al., 2025), and RGP (Lachi et al., 2025).

The SALT dataset presents a significantly more complex environment for relational deep learning. To ensure a direct and rigorous comparison with existing temporal neighbor sampling methods such as RGP, we evaluated GelGT across multiple classification tasks within the SALT dataset. We report the Mean Reciprocal Rank (MRR) to align with standard evaluation metrics used in recent literature.

As demonstrated in Table 3, GelGT consistently achieves state-of-the-art performance across all tasks. The substantial performance gap confirms that our approach—specifically the Gaussian temporal bias combined with structure-aware sampling—captures long-range relational dependencies more effectively than traditional heterogeneous methods (HGT) and recent temporal sampling strategies (RGP).

We further isolate the specific advantages of our structural-semantic-temporal collaborative modeling by comparing GelGT against two advanced architectures: Rel-LLM and Nodeformer.

We evaluated our model against Rel-LLM, a recent method leveraging large language models for relational graphs, and Nodeformer, an architecture specifically optimized for long-range dependencies. As shown in Table 4 (evaluated via AUC), GelGT consistently outperforms both baselines. This confirms that GelGT's success is not merely a byproduct of general model capacity, but fundamentally stems from our decoupled continuous Gaussian temporal bias, which precisely focuses on temporal patterns without distorting semantic embeddings.

C.1 Comparison with Recent Graph Transformers and Relational Foundation Models

In addition to the baselines reported above, we compare GelGT with Relational Transformer (RT) (Ranjan et al., 2026), a recent relational foundation model.

GelGT outperforms RT on all five tasks, with the largest relative gain of 32.9% on study-outcome. The relative gains are 13.2% on item-churn, 6.7% on user-churn, 12.4% on user-clicks, 6.9% on user-visits, and 32.9% on study-outcome. This directly shows that GelGT is not a RelBench-specific tuned tool: it remains stronger than a recent relational foundation model under the same downstream evaluation.

Table 13: Comparison with Relational Transformer (RT), a recent relational foundation model. ↑: higher is better.

Task	GelGT	RT
item-churn	**0.8297**	0.7330
user-churn	**0.7050**	0.6610
user-clicks	**0.6844**	0.6090
user-visits	**0.6695**	0.6260
study-outcome	**0.7254**	0.5460

D Additional Efficiency Experiments

In this section, we provide a detailed analysis of the computational efficiency of the proposed GelGT architecture. We specifically evaluate the overhead introduced by our structure-semantic collaborative sampling strategy and the Gaussian temporal bias-based attention mechanism, comparing them against standard baselines and optimized kernels.

D.1 Sampling Efficiency

A potential concern regarding our proposed sampling mechanism is whether the Breadth-First Search (BFS) combined with semantic similarity-based refinement introduces significant computational latency compared to naive Random Sampling. However, empirical results demonstrate that our approach maintains a cost highly comparable to Random Sampling.

The efficiency of our sampling strategy stems from its dual-stage design:

- **Stage 1 (Structural Integrity Sampling):** Under our standard configuration (e.g., a 300-node limit), the overhead of managing the BFS queue is marginal. It remains on the same scale as the random number generation and deduplication operations required by Random Sampling.
- **Stage 2 (Semantic Refinement):** The similarity calculation and selection process are executed in parallel for all subgraphs within a batch, ensuring minimal additional latency.

To validate this, we measured the single-node sampling times across two RelBench (Gu et al., 2026) datasets, including rel-ratebeer, a massive-scale graph with tens of millions of nodes. As shown in Table 14, our method remains just as efficient as Random Sampling, proving its feasibility for large-scale relational databases.

Table 14: Comparison of per-node sampling time between random sampling and the proposed method.

Task	Random Sampling	Ours
rel-trial-study-outcome	14ms	14ms
rel-ratebeer-beer_rating-total_score	87ms	88ms

D.2 Attention Efficiency

We also analyze the computational cost of our Gaussian Temporal Bias-based Attention. While modern optimized attention kernels like FlashAttention significantly accelerate vanilla attention, they are often less compatible with arbitrary pairwise biases. We demonstrate that our method remains highly competitive through two key perspectives: practical scalability and context size requirements.

We compared the training runtime (Epoch Time) and GPU memory consumption of our attention mechanism against FlashAttention across different context sizes (Table 15). For our standard context size of 300 nodes, FlashAttention yields only a marginal $1.07\times$ speedup, as the small context does not trigger typical memory bottlenecks. When aggressively scaling the sample size up to 3,000, our method's runtime remains in the same order of magnitude as FlashAttention. At an extreme scale of 30,000 nodes, both methods inevitably face Out-Of-Memory (OOM) errors due to hardware limitations. This confirms that the practical training overhead of our attention mechanism scales similarly to optimized vanilla attention.

A common assumption is that larger graphs inherently require substantially more context (e.g., thousands of nodes). However, our structure-semantic collaborative sampling effectively filters out irrelevant noise, eliminating the need for massive context windows. As demonstrated on the rel-ratebeer regression task (a dataset with over 13.7 million nodes), GelGT successfully captures the necessary structural information and achieves SOTA performance (Table 16) while strictly maintaining an extremely small context size of 300 nodes.

By improving sampling quality and strictly controlling the sample size, GelGT fundamentally avoids the quadratic computational bottlenecks of attention mechanisms, rendering the lack of FlashAttention compatibility a non-issue in practical deployments.

Table 15: Scalability comparison between our proposed attention and FlashAttention across varying sampled node sizes.

Context Size	Ours		FlashAttention	
	Epoch Time	GPU Mem	Epoch Time	GPU Mem
Size 300	225ms	2.2GB	210ms	1.9GB
Size 3000	1376ms	70.0GB	1004ms	53.9GB
Size 30000	— OOM —	— OOM —	— OOM —	— OOM —

Table 16: Performance comparison on the rel-ratebeer dataset (regression task). Best values (in terms of mean performance) are in **bold**. ↓ indicates lower is better.

Task ↓	RDL	RelGT	LightGBM	Ours
rel-beer_ratings-total_score	0.36	0.33	0.47	**0.32**

D.3 Efficiency on Extreme Graph Structures

Our sampling strategy is constrained by dual factors: a maximum BFS depth and a fixed candidate budget. This design ensures consistent efficiency across graph structures with varying densities.

In dense graphs, BFS quickly reaches the fixed budget (300 nodes), which bounds the cost of subsequent semantic filtering. In sparse graphs, the BFS traversal terminates earlier due to depth limitation, naturally reducing the candidate pool and further accelerating computation.

To validate this behavior, we construct extreme graph settings following the skewness-based generation protocol in (Ding et al., 2024). We report the average sampling time in Table 17.

Table 17: Sampling efficiency under extreme graph structures.

Sparse Graph	trial-study-outcome	Dense Graph
11 ms	14 ms	15 ms

The results indicate that our sampling strategy maintains stable efficiency under dense conditions and even improves slightly in sparse settings, demonstrating robustness across graph distributions.

E Detailed Description of Encoder Modules

In this section, we describe the specific encoder modules designed to process the heterogeneous attributes associated with nodes in our relational graph framework. Following previous work (Dwivedi et al., 2026), these encoders transform raw input features—including node types, hop distances, temporal information, tabular attributes, and structural positions—into dense vector representations suitable for downstream learning tasks.

Neighbor Node Type Encode. The Neighbor Node Type Encoder maps discrete heterogeneity information into a continuous latent space using a learnable lookup table (Embedding Layer). It instantiates a parameter matrix $\mathbf{E}_{\text{type}} \in \mathbb{R}^{|\mathcal{T}| \times d}$, where $|\mathcal{T}|$ is the number of node types. The encoder takes integer indices as input and performs a direct embedding lookup to project categorical entity constraints into dense vector representations.

Neighbor Hop Encoder. The Neighbor Hop Encoder introduces structural inductive bias by encoding the shortest-path distance using a dedicated Embedding Layer. To accommodate 0-based or 1-based indexing, the scalar hop counts are shifted and mapped via an embedding matrix $\mathbf{E}_{\text{hop}} \in \mathbb{R}^{K_{\max} \times d}$. This non-linear mapping allows the model to learn distance-dependent feature transformations rather than treating distance merely as a scalar weight.

Neighbor Time Encoder. The Neighbor Time Encoder transforms continuous scalar time intervals using a Sinusoidal Positional Encoding (PE) mechanism followed by a Linear Projection. Specifically, the module first projects scalar time differences Δt into $\mathbb{R}^d$ using fixed Fourier feature frequencies (Sinusoidal PE). The output is then passed through a learnable Linear Layer (nn.Linear). To handle invalid temporal data (e.g., negative time deltas), the module utilizes a learnable Mask Parameter, combining the linear output and the mask vector via a gating mechanism.

Neighbor Tabular Feature Encoder. The Neighbor Tabular Feature Encoder processes heterogeneous tabular attributes (numerical, categorical, and timestamps) using a ResNet backbone adapted for tabular data via the torch frame library. For each node type, the encoder first applies specific column encoders—Linear Encoders for numerical features and Embedding Encoders for categorical features. These processed features are then fed into a deep ResNet (consisting of multiple residual blocks with Linear, BatchNorm, and ReLU layers) to extract high-order interactions between columns.

GNN Positional Encoder. The GNN-based Positional Encoder captures local topological structures using a Graph Isomorphism Network (GIN) backbone. Node features are initialized either with random noise or Laplacian Eigenvector Positional Encodings (LapPE) projected via a Linear layer. The structure is processed by stacked GIN Convolutional Layers, where each layer utilizes a Multi-Layer Perceptron (MLP) as the aggregation function. This MLP is composed of Linear $\rightarrow$ BatchNorm $\rightarrow$ ReLU $\rightarrow$ Linear blocks. Residual connections are applied across layers, and the final embedding is obtained by aggregating layer-wise outputs via a generic pooling function (e.g., concatenation or max-pooling) followed by a final linear transformation.

GNN Branch: Architecture, Output, and Fusion. The GNN branch uses a 3-layer GraphSAGE (Hamilton et al., 2017) backbone. Each layer performs message passing followed by LayerNorm, GELU, and Dropout, and residual connections are incorporated at each layer, yielding a structure-aware representation $\mathbf{H}_{\text{gnn}}$.
Output. The output of the GNN branch is $\mathbf{H}_{\text{gnn}}$. It is not a single graph-level vector; it contains node-level structure-aware representations for all nodes in the refined sampled subgraph.
Fusion inside the branch. Since the branch uses GraphSAGE, the internal fusion is the standard GraphSAGE neighbor aggregation and message passing along PK–FK edges. In other words, the branch fuses local neighbor information through edge-based aggregation, and then uses multi-layer message passing with residual connections to produce $\mathbf{H}_{\text{gnn}}$.
Fusion across branches. The outputs of the GNN branch and the Gaussian temporal bias-based attention branch are combined by a learnable gate, so that the model can interpolate between structural and attentional evidence rather than committing to either branch a priori. Table 7 and Table 26 show that neither branch alone matches the fused model.

F Additional Analysis and Discussion

F.1 Semantic Refinement Design

We describe the design of semantic refinement from two aspects: (1) its exclusion at 1-hop neighbors, and (2) its behavior under non-homophilic settings.

Exclusion at 1-hop neighbors. Semantic refinement is not applied to 1-hop neighbors in order to preserve high-quality first-order signals. In relational databases, 1-hop neighbors correspond to direct PK-FK relations and provide the most reliable structural and semantic information for the target node. Filtering at this stage may remove essential information and degrade the quality of initial node representations. Preserving complete first-order neighborhoods is also consistent with prior findings (e.g., LightGCN (He et al., 2020)), which emphasize the importance of retaining full local connectivity for effective representation learning.

Behavior under non-homophilic settings. Semantic refinement is designed to operate beyond strict homophily assumptions. Instead of relying on node-type similarity, the method evaluates neighbors in a shared latent space (Wang et al., 2019) constructed from tabular attributes and positional encodings. In this space, similarity reflects task-relevant semantic correlation. Consequently, heterogeneous neighbors are retained as long as they contribute useful information to the target task. The refinement mechanism mainly acts on higher-hop neighbors to suppress less relevant or noisy signals, while preserving informative dependencies even in heterophilic scenarios.

Validation of the selection score. The dot product is applied to encoded node embeddings, not raw attributes, so it measures relevance in the learned task space, as in dot-product attention and dense retrieval (Karpukhin et al., 2020). From the same nodes, we compare top-(k), random-(k), and bottom-(k) selection ranked by our dot-product score. Top-(k) performs best on all five tasks, showing that the score ranks nodes by downstream usefulness rather than acting as an arbitrary heuristic.

Table 18: Comparison of selection rules applied to the same candidate pool. Top-k ranked by the dot-product score outperforms both random-k and bottom-k on all five tasks. ↑: higher is better.

Task	**Top-k**	**Random-k**	**Bottom-k**
driver-top3	**0.8408**	0.8311	0.8287
driver-dnf	**0.7608**	0.7491	0.7384
user-repeat	**0.8357**	0.8180	0.8156
user-ignore	**0.8779**	0.8602	0.8564
study-outcome	**0.7254**	0.7111	0.7207

Overall, semantic refinement balances information preservation and noise reduction by maintaining complete first-order neighborhoods and selectively filtering higher-order neighbors based on task-relevant semantic signals.

F.2 Generality of the Sampling Strategy

To demonstrate the generality of the proposed sampling strategy, we further evaluate it on another relational graph learning architecture, RelGNN. We integrate our sampling strategy into RelGNN without modifying its original model design, and compare the performance with its vanilla version.

The results are summarized in Table 19. We observe that the proposed sampling strategy consistently improves the performance of RelGNN across multiple tasks. This suggests that the effectiveness of our method is not tied to a specific model architecture.

We attribute these gains to the ability of our semantic similarity-based filtering to reduce noisy or less informative neighbors, thereby providing cleaner and more relevant neighborhood signals for downstream relational learning.

Traversal strategy. Under a fixed sampling budget, the traversal strategy determines which nodes enter the subgraph, and BFS better matches our goal. Our sampling aims to preserve local PK–FK

Table 19: Performance comparison (AUC) between RelGNN and RelGNN with the proposed sampling strategy.

Dataset	RelGNN	RelGNN (w/ Ours)
rel-avito-user-visits	0.6517	**0.6673**
rel-event-user-repeat	0.7661	**0.7898**
rel-trial-study-outcome	0.7003	**0.7290**
rel-amazon-item-churn	0.8043	**0.8118**
rel-stack-user-engagement	0.8975	**0.9088**

relations. BFS expands layer by layer, first covering 1-hop neighbors and then higher-hop nodes. DFS may go deep along few paths and miss other direct 1-hop relations, weakening local coverage. This is consistent with node2vec (Grover & Leskovec, 2016), where BFS-style sampling preserves nearby neighborhoods and supports structural-equivalence-oriented representations. Under the same hop number, budget, semantic refinement, and training setting, BFS outperforms DFS on all five tasks, with the largest relative gain of 2.1% on study-outcome.

Table 20: Effect of the graph traversal strategy. BFS consistently outperforms DFS under an identical sampling budget, hop number, and training setting. ↑: higher is better.

Task	BFS	DFS
driver-top3	**0.8408**	0.8311
driver-dnf	**0.7608**	0.7539
user-repeat	**0.8357**	0.8290
user-ignore	**0.8779**	0.8605
study-outcome	**0.7254**	0.7102

Generality without the GNN branch. Long-range dependency is mainly modeled by the Gaussian attention branch. We therefore repeat the evaluation inside an attention-only model, in which the GNN branch is removed and all remaining settings are unchanged. As shown in Table 21, removing structural integrity sampling, semantic refinement, or adaptive Gaussian bias consistently reduces the attention-only model on all five tasks. These results show that the gains of GelGT are not tied to the GNN branch and are not obtained by simply increasing the receptive field; they come from enabling useful long-range dependency through structural preservation, semantic denoising, and temporal weighting.

Table 21: Ablations inside the attention-only branch. The three modules remain beneficial even when the GNN branch is removed, showing that the long-range gain does not come from simply enlarging the receptive field. ↑: higher is better.

Task	Attn-only	w/o Struct.	w/o Sem.	w/o Gauss.
driver-top3	**0.8327**	0.8237	0.8193	0.8122
driver-dnf	**0.7499**	0.7370	0.7201	0.7186
user-repeat	**0.8173**	0.7998	0.7947	0.7951
user-ignore	**0.8501**	0.8465	0.8384	0.8322
study-outcome	**0.7094**	0.7024	0.6969	0.6978

F.3 Homophily vs. Heterophily in Relational Graphs

In this section, we investigate whether relational graphs derived from databases exhibit homophilic or heterophilic properties, and clarify the role of low-similarity neighbors in our learning framework.

Heterophily of relational graphs. As discussed in the main paper (Line 90) and established in prior studies (Zhu et al., 2020; Robinson et al., 2024; Dwivedi et al., 2026), relational graphs constructed

Table 22: Edge homophily ratio across datasets.

Task	Edge Homophily Ratio
rel-f1-driver-dnf	0.23
rel-avito-user-clicks	0.17
rel-event-user-repeat	0.13
rel-trial-study-outcome	0.22
rel-amazon-item-churn	0.21
rel-stack-user-badge	0.21
rel-hm-user-churn	0.25

from real-world databases are typically heterogeneous and temporal in nature. To further quantify this property, we compute the Edge Homophily Ratio (Zhu et al., 2020) on all evaluated datasets.

As shown in Table 22, all datasets consistently exhibit low homophily ratios (all < 0.5), indicating strong heterophily. This empirical evidence supports the conclusion that relational graphs in our setting are inherently heterophilous rather than homophilous.

Role of low-similarity neighbors. A key design choice in our method is to filter out low-similarity neighbors, which are treated as noisy or less informative connections. This design is supported by both empirical observations and the underlying modeling mechanism.

From an empirical perspective, as shown in Figure 4 of the main paper, increasing the number of sampled neighbors—thereby introducing more low-similarity neighbors—leads to a consistent degradation in performance. This indicates that such neighbors tend to introduce noise rather than useful signal for prediction.

From a mechanistic perspective, following the formulation in (Wang et al., 2019), our model projects tabular attributes and positional encodings into a shared semantic latent space, where similarity reflects task-relevant semantic correlation rather than structural homophily. Under this view, neighbors with low similarity in the learned space are more likely to be task-irrelevant and thus act as noise. Filtering them therefore improves representation quality and predictive performance.

Overall, these results suggest that low-similarity neighbors are generally not beneficial for the considered tasks under our semantic similarity space, and instead may negatively affect model performance when included indiscriminately.

F.4 Analysis of Gaussian Temporal Bias

In this section, we analyze the design choice of the proposed Gaussian temporal bias and compare it with alternative temporal modeling strategies to better understand its advantages.

Comparison with alternative temporal bias mechanisms. We first compare our method with two representative alternatives: (1) scaled linear temporal biases (Press et al., 2022), and (2) learnable temporal embeddings (Li et al., 2020).

Scaled linear biases impose a monotonically increasing penalty with respect to temporal distance, effectively favoring recent interactions. However, such a design is limited in modeling non-monotonic or periodic temporal dependencies commonly observed in real-world relational data. In contrast, our Gaussian formulation introduces a learnable mean parameter, which allows the attention peak to shift adaptively toward task-relevant historical periods rather than strictly prioritizing recent events.

Learnable temporal embeddings, on the other hand, discretize time into bins and assign independent parameters to each interval. While expressive, this approach increases parameter complexity linearly with the temporal range and may introduce redundancy. In comparison, our Gaussian bias only introduces two learnable scalars per attention head (mean and variance), making it significantly more parameter-efficient and scalable across varying time spans.

Table 23: Performance comparison (AUC) of different temporal bias mechanisms.

Task	Scaled Linear Bias	Temporal Embedding	Ours
f1-driver-dnf	0.75	0.75	0.76
avito-user-clicks	0.67	0.67	0.68
event-user-repeat	0.75	0.74	0.76
trial-study-outcome	0.70	0.71	0.72
amazon-item-churn	0.69	0.69	0.70
stack-user-badge	0.89	0.88	0.90
hm-user-churn	0.68	0.67	0.69

Empirical comparison. We further evaluate the effectiveness of different temporal bias designs across multiple tasks. As shown in Table 23, our Gaussian temporal bias consistently outperforms both baselines.

Range and robustness of the temporal bias. A natural concern is that the Gaussian kernel is bounded in $[0, 1]$, so the injected bias might be too small to influence attention. This concern comes from a misunderstanding of how the bias is used in GelGT. The raw Gaussian kernel is indeed in $[0, 1]$, but GelGT does not add this raw value directly. We use a learnable linear mapping,

$$\mathrm{Bias}_{\mathrm{time}} = \mathrm{Linear}\left(\exp\left(-\frac{(\Delta t - \mu)^2}{\sigma^2}\right)\right), \tag{49}$$

so the final temporal bias added to attention is not restricted to $[0, 1]$; its scale is learned during training. Moreover, the bias is added before the softmax,

$$\mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathrm{Bias}_{\mathrm{time}}\right), \tag{50}$$

so it is part of the attention logit rather than a post-processing term. Even a logit change of 1 can change the relative attention weight by a factor of e, so its effect is not inherently small. Consistently, removing the adaptive Gaussian bias degrades performance in Tables 5 and 6, and Table 23 shows that our Gaussian bias outperforms both a scaled linear bias and a temporal embedding.

To further verify that this gain comes from meaningful temporal relevance rather than from the mere presence of a bias term, we randomly shuffle timestamps in the test graph. Performance drops on all five tasks, with the largest relative drop of 6.3% on study-outcome. This confirms that the benefit depends on the actual temporal ordering of neighbors, rather than on the bias term acting as an unconditional additive constant.

Table 24: Effect of randomly shuffling timestamps in the test graph. Destroying temporal ordering degrades performance on all five tasks. ↑: higher is better.

Task	w/o Shuffle	w/ Shuffle
driver-top3	**0.8408**	0.8025
driver-dnf	**0.7608**	0.7332
user-repeat	**0.8357**	0.7998
user-ignore	**0.8779**	0.8513
study-outcome	**0.7254**	0.6798

Discussion. The results demonstrate that the Gaussian temporal bias effectively captures more flexible temporal patterns compared to monotonic or discretized formulations. This flexibility, together with its parameter efficiency, enables consistent improvements across diverse relational prediction tasks.

F.5 Additional Discussion on Design Choices

In this section, we further clarify the design motivations behind the proposed components and highlight their relevance to the unique challenges of relational deep learning (RDL).

Gaussian temporal modeling. Timestamped interactions in relational databases often exhibit continuous, irregular, and non-monotonic temporal dynamics. To effectively capture such patterns, we propose a Gaussian temporal bias mechanism, which introduces only two learnable scalar parameters per attention head.

Compared to scaled linear biases (Press et al., 2022), which enforce a strictly monotonic temporal decay, and learnable temporal embeddings (Li et al., 2020), which discretize time and scale parameters with temporal granularity, our formulation provides a more flexible and compact representation of temporal influence. In particular, the learnable mean enables adaptive shifting of the attention peak toward task-relevant historical periods, while the variance controls the temporal sensitivity range. This design offers a principled and parameter-efficient alternative for continuous temporal modeling in relational graphs.

Temporal-aware BFS sampling. Our sampling strategy is also carefully designed for relational databases and is not a direct application of standard BFS. Instead, we incorporate explicit temporal constraints into the BFS traversal process. This design ensures that sampled subgraphs preserve both (i) complete primary–foreign key relational structures and (ii) strict temporal causality.

Unlike prior RDL sampling methods (Zhu et al., 2020; Robinson et al., 2024; Dwivedi et al., 2026), which either ignore temporal ordering or fail to jointly preserve structural and temporal consistency, our approach enables the extraction of causally valid relational subgraphs. This makes it particularly suitable for learning over dynamic relational systems where both structure and time are critical.

Interaction between the three components. The ablation numbers in Tables 5 and 6 are close because each variant removes only one component while keeping the other two, so all variants remain strong. The key point is the consistent direction: removing structural sampling, semantic refinement, or Gaussian temporal bias always reduces performance. To further isolate the relative contribution of each component, we additionally conduct a full combination ablation over all $2^3 = 8$ settings on driver-dnf.

Table 25: Full 2^3 combination ablation on driver-dnf. Structural sampling, semantic refinement, and adaptive Gaussian temporal bias each contribute positively, and the full model performs best. ↑: higher is better.

Sampling	Semantic	Temporal	driver-dnf
w/	w/	w/	**0.7608**
w/	w/	w/o	0.7495
w/	w/o	w/	0.7422
w/	w/o	w/o	0.7413
w/o	w/	w/	0.7486
w/o	w/	w/o	0.7430
w/o	w/o	w/	0.7398
w/o	w/o	w/o	0.7333

Compared with the corresponding one-module-removed variant, structural sampling brings 1.6% relative gain, semantic refinement 2.5%, and Gaussian temporal bias 1.5%. The full model performs best, improving by 3.8% over removing all three.

Contribution of the two branches. We further add the GNN-only result under the same encoder, sampling strategy, hidden size, training budget, and random seed. The comparison includes GNN-only, attention-only, and full GelGT. Compared with the full model, GNN-only drops by up to 3.9%, and attention-only drops by up to 3.2%, confirming that the two branches are complementary. The GNN branch captures local PK–FK structure, while attention models global and temporal interactions. The full model outperforming both single-branch variants shows that fusion is necessary.

Summary. Overall, the proposed designs are specifically motivated by the characteristics of relational databases, where both temporal irregularity and schema-driven structure play central roles. These components jointly provide a unified and efficient framework for relational representation learning.

Table 26: Branch-wise comparison. GNN-only and attention-only are both inferior to the fused model, showing that the two branches are complementary. ↑: higher is better.

Task	Full	GNN-only	Attn-only
driver-top3	**0.8408**	0.8318	0.8327
driver-dnf	**0.7608**	0.7311	0.7499
user-repeat	**0.8357**	0.8044	0.8173
user-ignore	**0.8779**	0.8588	0.8501
study-outcome	**0.7254**	0.7113	0.7094

F.6 Sampling Budget Analysis

To justify the choice of a fixed candidate size in our original design, we analyze the effect of sampling budget on both performance and efficiency. To better understand the effective neighborhood size, we additionally introduce an auxiliary diagnostic mechanism with adaptive soft-gating, which is used solely for analysis and is not involved in training or inference of the proposed model. This analysis reveals that the effective number of retained neighbors consistently stabilizes around a fixed range, providing empirical support for our initial design choice.

As shown in Table 27, the average number of retained effective neighbors across datasets ranges between 249 and 347, with a mean value close to 300. This indicates that a sampling budget around 300 is sufficient to capture informative relational semantics while avoiding excessive redundancy.

Table 27: Average effective number of retained neighbors under adaptive soft-gating.

Dataset	Avg. Effective Neighbors
f1-driver-dnf	298
avito-user-clicks	275
event-user-repeat	324
trial-study-outcome	307
amazon-item-churn	249
stack-user-badge	333
hm-user-churn	347

Table 28: Evolution of Gaussian bias parameters during training.

Epoch	μ	σ
1	33.31	9.98
3	33.27	9.94
5	33.24	9.90
7	33.20	9.86
9	33.16	9.83
10	33.16	9.83

From an efficiency perspective, increasing the sampling budget beyond 300 yields marginal performance gains but significantly increases computational and memory overhead due to quadratic attention complexity. In contrast, smaller budgets (e.g., below 200) tend to truncate informative relational signals, particularly in dense or heterogeneous graphs.

Overall, a budget of 300 achieves a favorable trade-off between semantic coverage, efficiency, and cross-dataset stability. This also aligns with our adaptive mechanism, where the learned effective neighborhood size naturally stabilizes around this range.

F.7 Stability of Gaussian Bias Parameters

We empirically study the optimization behavior of the Gaussian temporal bias parameters. The results show that these parameters exhibit fast and stable convergence during training, rather than instability.

From a theoretical perspective, as shown in Lemma 3.3, the gradient contains a linear restoring term that provides a global directional force towards a stable equilibrium. This design ensures well-conditioned optimization dynamics and prevents oscillatory updates.

We further track the parameter evolution on the trial-study-outcome dataset. As shown in Table 28, the parameters rapidly converge within the first few epochs and remain stable thereafter.

These results demonstrate that the Gaussian bias parameters are not sensitive to optimization instability and instead converge reliably to a stable region.

G Related Work

Traditional Relational Learning. Relational learning on relational databases was heavily reliant on manual feature engineering, requiring domain experts to transform multi-table records into flat feature vectors to accommodate tabular learning methods (Dwivedi et al., 2026), e.g., LightGBM (Ke et al., 2017), TabNet (Arik & Pfister, 2021) and DNF-Net (Katzir et al., 2020). Specifically, TabNet treats attributes from distinct tables merely as a homogeneous feature set, while DNF-Net models interactions solely as property combinations within a single row. However, these methods will break the relationships between tables in the RDB. This reduction of hierarchical tables to flat vectors transforms *Primary Key-Foreign Key* connections from explicit referential links to disconnected feature columns, effectively obscuring the hierarchical dependencies between tables. Relational graph learning aims to address these challenges (Cvitkovic, 2020; Fey et al., 2024).

Relational Graph Learning (RGL). In RGL, the relational database is modeled as a relational graph, which effectively captures the relational dependencies. Most RGL methods are built upon the message-passing-based mechanisms, utilizing message-passing mechanisms to propagate information across tables. For example, RelGNN (Chen et al., 2025) and RDL (Robinson et al., 2024) adopt an iterative message-passing mechanism where row-level representations are updated by recursively aggregating transformed feature messages from connected neighbors across tables via foreign keys. However, these methods struggle to capture such long-range dependencies effectively (Wu et al., 2021; 2022), limiting their ability to model complex and multi-hop relationships for downstream predictive tasks. Therefore, existing methods shift to graph Transformer-based architectures, including RelGT (Dwivedi et al., 2026), Griffin (Wang et al., 2025) and HGT (Hu et al., 2020). Nevertheless, current approaches fall short of addressing the complex structural, semantic, and temporal demands of RGL. For example, RelGT relies on random sampling, which disregards explicit connections, leading to structural fragmentation and degrading predictive performance. Although Griffin (Wang et al., 2025) preserves all connected nodes, such full retention is often impractical in real-world databases, where a single row can be associated with thousands of records (Chen et al., 2025). By indiscriminately incorporating such massive neighborhoods without semantic filtering, it inevitably introduces semantic noise (Wang et al., 2019). In addition, they commonly flatten dynamic temporal patterns into static latent representations, ignoring the intrinsic temporal pattern and lacking the capacity to capture precise temporal dependencies. Our GelGT explicitly addresses the above challenges, achieving state-of-the-art performance.

H Limitations and Broader Impacts

H.1 Limitations

While GelGT achieves state-of-the-art predictive performance and strong computational efficiency on standard relational databases, its deployment in extreme ultra-high-frequency scenarios—such as millisecond-level financial trading—remains an open challenge. Such specialized environments impose extreme physical latency constraints that fall outside the typical scope of current relational graph learning frameworks, presenting an interesting direction for future engineering efforts.

H.2 Broader Impacts

Our work proposes a novel graph transformer architecture for relational graph learning, offering robust support for predictive tasks in relational databases and advancing the field of Machine Learning. This work is foundational research in Machine Learning. We do not foresee any immediate negative societal impacts specific to this work.