TEMPORAL HETEROGENEOUS GRAPH PRETRAINING FOR RELATIONAL DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Relational deep learning represents database rows and foreign-key links as a heterogeneous graph, enabling prediction from record attributes and relational context. Two forms of temporal signals play distinct roles in these graphs: record age changes as the prediction cutoff advances, whereas the interval between two observed records remains fixed. Prior work has explored temporal modeling and temporal pretraining for heterogeneous graphs, but typically treats time as a single source of information or focuses on either representation or supervision in isolation. We investigate how explicitly representing both temporal signals affects the benefits of temporal pretraining on downstream tasks. Our framework pairs two complementary temporal encodings: Multi-scale Time Encoding, which captures record age through learnable time scales and type-specific projections, and Rotary Time Encoding, which encodes signed time differences between linked records through rotary transformations during graph propagation. Rather than treating these encodings as architectural additions alone, we train them through three self-supervised objectives: recovering historical relations, forecasting horizon-dependent future relation activity, and contrasting temporally valid historical subgraphs. All model inputs are restricted to information available at their observation cutoffs. We further structure pretraining into two stages: learning neighborhood representations via subgraph contrastive learning, followed by refining these representations through either relation recovery or future activity prediction. We evaluate across five RelBench datasets and 11 classification and regression tasks using representative heterogeneous graph neural network (GNN) and Graph Transformer (GT) backbones. With both temporal encodings, the best evaluated staged schedules for the two backbones outperform direct supervised training using the same encodings by 3.02% and 1.06%, respectively, and controls without pretraining or encodings by 3.24% and 2.37%, respectively.

1 INTRODUCTION

Relational databases store records in linked tables, with predictive signals distributed across both record attributes and relationships. For example, predicting whether a customer will buy again may require combining customer details, product information, and purchase times from several tables. Conventional pipelines assemble these signals through task-specific joins and aggregations (Kanter & Veeramachaneni, 2015). Relational deep learning (RDL) instead represents rows as nodes and foreign-key links as typed edges, allowing graph models to learn directly from attributes and relational neighborhoods (Fey et al., 2024).

Learning from relational graphs requires modeling both heterogeneity and time. Customer and product records describe distinct entity types, while purchase records capture interactions between them. Together, they form a heterogeneous graph with type-specific node attributes and multiple relation types. Time provides two complementary signals about these records and their relationships. *Record age* measures the time elapsed between a record's observation and the prediction cutoff. A signed *inter-record interval* measures the difference between the observation times of two linked records, capturing their temporal order and distance. For example, as the prediction cutoff advances, a purchase record becomes older, while the interval between the customer's registration and that purchase remains fixed.

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.

Temporal encodings make record age and inter-record intervals explicit in model representations (Xu et al., 2020; Hu et al., 2020c; Yu et al., 2023; Cong et al., 2023; Li et al., 2023; Robinson et al., 2024; Dwivedi et al., 2026). Pretraining guides models to integrate these temporal signals with record attributes and graph structure before adaptation to downstream tasks. Existing pretraining approaches use heterogeneous reconstruction and relation- or schema-based contrastive learning (Hu et al., 2020b; Jiang et al., 2021; Sun et al., 2025), while temporal objectives further target interaction timing or statistics of future records (Ma et al., 2026; Truong et al., 2026). Temporal encodings and pretraining objectives thus serve complementary roles: the former specify how time is represented, and the latter guide what and how the model learns from these representations. This motivates us to study how temporal heterogeneity-aware pretraining can be combined with explicit encodings of both temporal signals. Figure 2 summarizes these design choices across prior methods and our framework.

We address this challenge with a framework that integrates temporal encodings with staged pretraining. Our temporal encodings capture both record age and relative timing: Multi-scale time encoding (TIMEMIX) incorporates record-age information into node inputs through learnable time scales and type-specific projections, while Rotary time encoding (TIMEROPE) adapts prior temporal rotary attention (Peng et al., 2026) to use bounded, signed log-intervals and extends it to GNN message construction.

Building on these encodings, we introduce three complementary pretraining objectives. Temporal Subgraph Contrast (TEMP-SUB) aligns corrupted views of historical neighborhoods through type-aware pooling and relation-preserving augmentations, encouraging representations to remain stable under missing attributes or links. Historical Relation Recovery (REL-HIST) recovers hidden typed links using only the context available when those links were observed, whereas Horizon-aware Relation Activity Prediction (REL-FUTURE) predicts whether an entity will receive links of a specified relation and how many events will occur within a future window. We organize these objectives into a staged pretraining scheme: TEMP-SUB first learns neighborhood representations, which are then refined using either REL-HIST or REL-FUTURE. All model inputs respect their observation cutoffs throughout pretraining. The resulting pretrained model is subsequently fine-tuned for individual tasks within the same database.

We evaluate our framework on five RelBench databases (Robinson et al., 2024) across 11 tasks spanning binary classification, multiclass classification, and regression using representative heterogeneous GNN and GT backbones. Controlled ablations assess the complementary benefits of the temporal encodings and the effectiveness of staged pretraining (Figure 4).

Our contributions are:

- **Complementary temporal representations.** We establish that jointly encoding record age and inter-record intervals improves aggregate performance over either alone on both backbones, with record age contributing more in component ablations.
- **Effective staged pretraining.** Our two staged schedules yield positive aggregate transfer in all four backbone–encoding settings. With both encodings, the best schedule on each backbone outperforms either constituent objective alone, demonstrating the value of staging complementary supervision.
- **Gains beyond temporal encoding alone.** Matched comparisons establish additional mean type gains of 3.02% on the GNN and 1.06% on the GT over supervised training with both encodings. Our best schedules also outperform the evaluated pretraining baselines, including their staged variants.

2 BACKGROUND AND RELATED WORK

2.1 RELATIONAL DEEP LEARNING

RDL provides an end-to-end framework for learning from linked database records (Fey et al., 2024; Dwivedi et al., 2025). It combines multimodal attributes, schema-defined relationships, and temporal observations in a graph representation. RelBench supplies predictive tasks with chronological splits (Robinson et al., 2024).

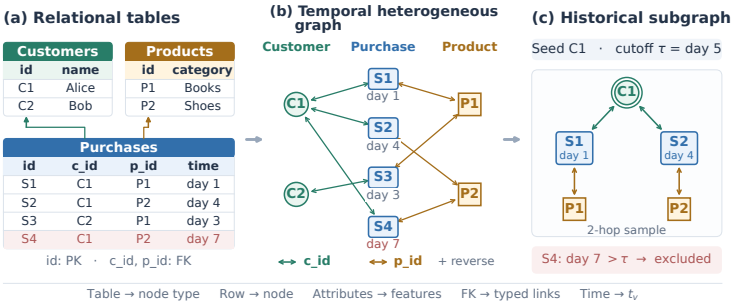

Figure 1: From relational tables to historical subgraphs for RDL. (a) Customer, purchase, and product tables: `id` denotes primary keys, `c_id` and `p_id` denote foreign keys, and `time` records purchase observation times. (b) Rows become nodes whose types follow their source tables; foreign-key references become typed edges with corresponding reverse relations for bidirectional message passing. Purchase nodes carry timestamps t_v. (c) An illustrative two-hop sample around seed C1, using only information available by cutoff $\tau =$ day 5. S4 is excluded because its timestamp exceeds τ; S3 and C2 lie outside the two-hop neighborhood.

Definitions. Let $\mathcal{D} = (\mathcal{T}, \mathcal{R})$ be a database with tables $\mathcal{T} = \{T_1, \ldots, T_n\}$ and foreign-key relations $\mathcal{R}$. Each row is identified by a primary key and contains attributes and, where applicable, foreign keys referencing other rows. A timestamp, when present, records its observation time. Following the relational entity graph formulation (Fey et al., 2024), we construct $G = (V, E, \phi, \psi, t)$, where V contains the rows, E contains their foreign-key links, $\phi : V \to \mathcal{T}$ assigns node types by source table, and $\psi : E \to \mathcal{R}$ assigns relation types. The timestamp function t is defined on timestamped nodes, with $t_v = t(v)$. We include reverse edges and corresponding relation types in E and $\mathcal{R}$ for message passing. Figure 1(a,b) illustrates this mapping: customer, purchase, and product tables become distinct node types, and the two foreign keys in each purchase define its links to a customer and a product.

2.2 Graph Models and Temporal Encoding for RDL

We consider two encoder families for RDL: GNNs based on relational message aggregation, including HeteroGNN and RelGNN (Robinson et al., 2024; Chen et al., 2025), and GTs based on heterogeneous attention, including HGT, RelGT, and Dual-Branch Temporal Heterogeneous Graph Fusion Model (THGFM) (Hu et al., 2020c; Dwivedi et al., 2026; Peng et al., 2026).

Viewed in the RDL setting, existing methods incorporate record age through temporal features added to node inputs or included in attention and sequence representations. These features use positional encodings with trainable projections in HeteroGNN, learnable trigonometric functions in TGAT, THAN, and DyGFormer, and fixed-frequency cosine functions in GraphMixer (Robinson et al., 2024; Xu et al., 2020; Li et al., 2023; Yu et al., 2023; Cong et al., 2023). RelGT incorporates seed-relative age features during record tokenization (Dwivedi et al., 2026). For inter-record intervals, HGT adds projected sinusoidal encodings of source–target time differences to source representations, while THGFM applies interval-dependent query–key rotations (Hu et al., 2020c; Peng et al., 2026). Additionally, record-age distributions and time scales vary across the evaluated RelBench databases (Appendix B.2). Motivated by these observations and the complementary roles of these two forms of temporal information, we develop two encodings and integrate both into the GNN and GT backbones.

2.3 Graph Pretraining: Heterogeneity, Time, and Staging

Graph pretraining provides supervision over nodes, relations, and subgraphs, but coverage of these granularities does not by itself imply explicit use of heterogeneous types or schema (Figure 2). AttrMasking and GraphMAE learn through attribute reconstruction (Hu et al., 2020a; Hou et al., 2022; 2023); Un-SAGE and ContextPred learn from proximity and neighborhood context (Hamilton et al., 2017; Hu et al., 2020a), while GCC and GraphCL contrast subgraph or graph views (Qiu et al., 2020; You et al., 2020). Heterogeneous pretraining makes schema structure explicit in the supervision: GPT-GNN uses type-specific attribute and link generation, PT-HGNN combines relation-wise sampling

Pretraining strategy	Pretraining granularity			Temporal encoding		Schedule	Evaluated graph models
	Node; Heterogeneity	Relation; Heterogeneity	(Sub-)Graph; Heterogeneity	Recency	Interval	Staged	
TVE	●;●	–;–	◐;◐	B	–	–	GNN
TGPM	–;–	●;–	●;–	●	●	–	GT
CTRL	–;–	●;◐	●;●	●	–	–	GNN
Un-SAGE	●;–	◐;–	–;–	–	–	–	GNN
AttrMasking	●;–	●;–	–;–	–	–	●	GNN
ContextPred	●;–	–;–	●;–	–	–	●	GNN
GPT-GNN	●;●	●;●	–;–	–	B	–	GNN & GT
PHE	●;●	●;●	◐;◐	–	B	–	GNN & GT
PT-HGNN	●;●	●;●	●;●	–	B	–	GT (HGT)
Our staged schedules	●;●	●;●	●;●	●	●	●	**GNN & GT**

● Direct ◐ Indirect/limited coverage B Backbone-dependent – Not explicit

Figure 2: Comparison of representative graph pretraining strategies. In each of the first three columns, paired symbols indicate coverage of the supervision granularity and explicit use of node or relation types at that granularity, respectively. Filled and half-filled circles denote direct and indirect or limited coverage; B denotes backbone-dependent encoding, and a dash denotes no explicit coverage.

with schema-instance agreement, and PHE incorporates type-aware schema representations and semantic neighbors (Hu et al., 2020b; Jiang et al., 2021; Sun et al., 2025). HGMAE and MUG alternatively use metapaths in reconstruction or alignment (Tian et al., 2023; Shan et al., 2026). For RDL, this distinction matters because table and relation types determine which entities and links a pretraining target describe. These methods establish ways to learn from typed structure; temporal encoding and temporally defined targets introduce additional design choices.

Time can condition the representation of observed history or define what the model must predict. TGPM combines masked patch reconstruction with next-time prediction (Ma et al., 2026), while CTRL models temporal influence through heterogeneous event-subgraph and edge prediction (Li et al., 2026). In relational databases, Griffin uses completion pretraining and supervised training, while RelPrism constructs attribute-based pseudo-tasks (Wang et al., 2025; Yang et al., 2026). TVE directly targets future relational information: it predicts statistics, including counts, of records reached through schema traversals (Truong et al., 2026). These targets differ in what they predict: interaction timing, event structure, or aggregated relational statistics.

How these objectives are organized is a further question. Hu et al. (2020a) stage node-level and graph-level pretraining, whereas TGPM jointly trains temporal objectives and TVE supports weighted combinations of losses (Ma et al., 2026; Truong et al., 2026). Our schedules study a different transition: first learn stable historical representations through contrastive learning, then refine them through relation recovery or future activity prediction, evaluating these schedules with explicit temporal encodings across heterogeneous GNN and GT backbones. Together, seed-node and historical-subgraph contrast, typed relation recovery, and activity prediction cover node, relation, and subgraph supervision, with heterogeneous types explicitly used in pooling, sampling, decoding, and augmentation.

3 METHOD

For seed node s and prediction cutoff τ, let $G_{s,\leq\tau}$ denote a sampled neighborhood containing only information available by τ. We refer to this input as a *historical neighborhood* or *subgraph*. We use s to identify its seed and u, v to denote general nodes or edge endpoints. Temporal encoding and graph propagation operate on nodes and edges throughout the sampled neighborhood. In Figure 1(c), S4 is excluded by time, while S3 lies outside C1's two-hop neighborhood. Static tables contribute attributes available at the cutoff even when they have no timestamp column. The model produces the seed representation $h_s = f_\theta(G_{s,\leq\tau}) \in \mathbb{R}^d$, where d is the hidden dimension.

3.1 TEMPORAL ENCODING

Record age: TIMEMIX. TIMEMIX represents how old a record is at the prediction cutoff, allowing the model to learn how recency matters for each table type. For each timestamped node v in the sampled neighborhood, we compute its age relative to the same cutoff τ:

$$a_v = \max(0, \tau - t_v), \qquad \xi_{v,k} = \log\left(1 + \frac{a_v}{s_k}\right), \tag{1}$$

where a_v is the record age and $\xi_{v,k}$ is its feature at learned positive time scale s_k, for $k = 1, \ldots, K_{\text{time}}$. We parameterize $s_k = \text{clip}(\exp(\eta_k), s_{\min}, s_{\max})$, where $\eta_k \in \mathbb{R}$ is trainable and $0 < s_{\min} < s_{\max}$ are fixed bounds. Here K_{time} is the number of scales, and ages and scales use the same time unit. The logarithm preserves variation among recent records while compressing large ages. Future records are excluded by sampling before this encoding is applied.

Let $x_v \in \mathbb{R}^d$ be the table encoder's representation of record attributes. For node type $a = \phi(v)$, we add a mixture of projected age features:

$$e_v^{\text{time}} = \sum_{k=1}^{K_{\text{time}}} \pi_k g_{a,k}(\xi_{v,k}), \qquad \widetilde{x}_v = x_v + e_v^{\text{time}}. \tag{2}$$

Here $g_{a,k} : \mathbb{R} \to \mathbb{R}^d$ is a learned linear projection for type a and scale k, and π_k is a learned nonnegative mixture weight, with $\sum_k \pi_k = 1$. The time embedding e_v^{time} augments the attribute representation, giving $\widetilde{x}_v$ as the input to graph propagation. Scales and weights are shared across node types; type-specific projections allow the same age to have different effects in different tables. For records without timestamps, we use $\widetilde{x}_v = x_v$. Appendix D gives more details.

Time differences: TIMEROPE. TIMEROPE captures the order and spacing of linked records. For a directed edge $u \to v$ in the sampled neighborhood, u is the source and v is the destination. When both endpoints have timestamps, define $\Delta t_{uv} = t_v - t_u$ as the signed time difference. We convert it to a rotation phase:

$$\rho_{uv} = \text{clip}\left(\text{sign}(\Delta t_{uv}) \log\left(1 + \frac{|\Delta t_{uv}|}{s_{\text{R}}}\right), -\rho_{\max}, \rho_{\max}\right), \tag{3}$$

where s_{R} is a fixed positive time scale and $\rho_{\max} > 0$ is a fixed hyperparameter that bounds the magnitude of the rotation phase. The sign distinguishes temporal order; the logarithm and clipping limit the effect of long gaps. Edges with an untimestamped endpoint use the backbone's message or attention operation without rotation.

For each node v, let $y_v \in \mathbb{R}^d$ denote its input to the current graph layer; in the first layer, $y_v = \widetilde{x}_v$. Let $R(\rho)$ rotate the j-th consecutive pair of feature dimensions by angle $\omega_j \rho$, where ω_j is the corresponding RoPE frequency (Su et al., 2023). Our GNN backbone constructs messages using opposite half-phase rotations and aggregates them by relation:

$$\begin{aligned} m_{u \to v} &= \tfrac{1}{2}\left[R(-\rho_{uv}/2)y_u + R(+\rho_{uv}/2)y_v\right], \\ h_v^{(r)} &= W_{\text{neigh}}^{(r)} \left(\frac{1}{|\mathcal{N}_r(v)|} \sum_{u \in \mathcal{N}_r(v)} m_{u \to v}\right) + W_{\text{self}}^{(r)} y_v. \end{aligned} \tag{4}$$

Here $\mathcal{N}_r(v)$ contains incoming neighbors under relation r, and $h_v^{(r)}$ is the relation's output. The learned matrices $W_{\text{neigh}}^{(r)}$ and $W_{\text{self}}^{(r)}$ transform averaged messages and node inputs, respectively.

For our GT backbone, we adapt the query–key rotation mechanism of THGFM's Rotary Temporal Attention (Peng et al., 2026), using the bounded, signed log-interval ρ_{uv} defined above as the temporal phase. Thus, TIMEROPE incorporates inter-record timing into GNN message construction and GT attention weighting. Appendix D provides the rotation convention and backbone implementation details.

3.2 TEMPORAL PRETRAINING OBJECTIVES

Historical link recovery: REL-HIST. REL-HIST asks whether two records were linked, given the context available when the link was observed. We sample an observed link $u \to v$ of relation r and use its assigned observation time as cutoff τ. We then sample K_{neg} negative destinations of the same type as v that are available at τ and are not known destinations of u under r in the pretraining data.

We encode historical neighborhoods centered on the source and on each candidate destination at the same cutoff τ, obtaining their node representations. We remove the target edge and its reverse from every input so that the encoder cannot directly observe the answer. For endpoint representations

h_u and h_v, a relation-specific bilinear decoder assigns the score $s_{\text{hist}}(u, v, r) = h_u^\top W_r h_v$, where $W_r \in \mathbb{R}^{d\times d}$ is a learned matrix for relation r. The mean softplus ranking loss $\mathcal{L}_{\text{Rel-Hist}}$ encourages the observed destination to score above the sampled negatives. Appendix E gives the timestamp assignment and full loss.

Future relation activity: REL-FUTURE. REL-FUTURE predicts whether a seed node s will receive links of relation r in a future window and how many. Each forward foreign-key link $e = (u \to s)$ represents an event, with time $t_e = t_u$ given by the source record's timestamp. Let $\mathcal{E}_r(s)$ be the set of these events. For cutoff τ and horizon δ, we define the event count and prediction targets as

$$\begin{aligned} c_r(s;\tau,\delta) &= \sum_{e\in\mathcal{E}_r(s)} \mathbf{1}\{\tau < t_e \le \tau+\delta\}, \\ y_{\text{act}} = \mathbf{1}\{c_r(s;\tau,\delta) > 0\}, &\qquad y_{\text{count}} = \log\big(1 + c_r(s;\tau,\delta)\big). \end{aligned} \tag{5}$$

To construct a positive example, we uniformly sample an eligible relation and then an eligible event of that relation. Let s^+ be its destination and t_{evt} its time. We choose a candidate horizon δ and set $\tau = t_{\text{evt}} - \delta$, requiring $t_{\text{first}}(s^+) \le \tau$, where t_{first} denotes the entity's first observation time. This ensures that the seed is available at the cutoff and the window contains at least the sampled event. For the same (r, τ, δ), we sample negative seeds s^- of the same type as s^+ that are available at τ and have no events in the window.

For each seed s, the encoder uses only $G_{s,\le\tau}$ to compute h_s. Two decoders, conditioned on r and δ, map h_s to an occurrence logit and a nonnegative log-count prediction. We optimize

$$\mathcal{L}_{\text{Rel-Future}} = \mathcal{L}_{\text{act}} + \lambda_{\text{count}}\mathcal{L}_{\text{count}}, \tag{6}$$

where $\mathcal{L}_{\text{act}}$ is the binary occurrence loss, $\mathcal{L}_{\text{count}}$ is the Smooth L1 loss on log-count targets, and λ_{count} weights the count loss. Each loss gives equal weight to the positive example and the mean over negatives. Appendix E provides the timestamp assignment, sampling rules, and full losses.

Temporal subgraph contrast: TEMP-SUB. TEMP-SUB learns representations that remain consistent when some features or links are missing from a historical neighborhood. Let τ_{pre} be the global pretraining cutoff, set to the dataset's validation timestamp. For timestamped seeds, we require $t_s \le \tau_{\text{pre}}$ and set $\tau = t_s$; for seeds without timestamps, we set $\tau = \tau_{\text{pre}}$. For each input $G_{s,\le\tau}$, we create two independently corrupted views sharing the same seed node s and cutoff τ by randomly masking input embedding entries and dropping edges. We either keep or drop each edge together with its corresponding reverse edge, and each view retains at least one edge per represented forward relation.

To summarize each view, we first pool nodes within each type, then combine the type summaries. Let H be an augmented subgraph, $\mathcal{T}(H)$ its set of node types, and $V_a(H)$ its nodes of type a. We compute

$$\mu_{H,a} = \frac{1}{|V_a(H)|}\sum_{u\in V_a(H)} h_u, \qquad c_{H,a} = \tanh(P_a\mu_{H,a} + e_a)\,, \tag{7}$$

where h_u is computed within H, $\mu_{H,a}$ is the mean representation of type a, and $c_{H,a}$ is its type summary. The learned projection $P_a \in \mathbb{R}^{d\times d}$ and type embedding $e_a \in \mathbb{R}^d$ adapt the summary to its table type. A shared learned linear scoring function $g_{\text{pool}} : \mathbb{R}^d \to \mathbb{R}$ then assigns attention weights $\beta_{H,a}$ to combine the summaries into z_H:

$$\beta_{H,a} = \frac{\exp(g_{\text{pool}}(c_{H,a}))}{\sum_{b\in\mathcal{T}(H)}\exp(g_{\text{pool}}(c_{H,b}))}, \qquad z_H = \sum_{a\in\mathcal{T}(H)} \beta_{H,a}c_{H,a}. \tag{8}$$

Averaging within types prevents larger tables from dominating solely through their node counts, while attention learns how much each type contributes to the neighborhood representation.

We apply contrastive learning at two levels: the whole neighborhood and its seed node s. For neighborhood-level contrast, we map the pooled representations of both views through the same projection head. Each neighborhood in the first view is trained to match its counterpart in the second view, with other neighborhoods in the batch serving as negatives. We compute an InfoNCE

loss (van den Oord et al., 2019) using cosine similarity divided by temperature T. We then repeat the matching from the second view to the first and average the two losses to obtain $\mathcal{L}_{\text{graph}}$. For seed-level contrast, we take the representation of s from each view and apply the same matching procedure through a separate projection head, obtaining $\mathcal{L}_{\text{seed}}$. To complement contrastive learning in the projection space, we additionally regularize the seed representations with a VICReg penalty (Bardes et al., 2022) that discourages collapse by maintaining variance across seeds and reduces redundancy across feature dimensions. This penalty acts directly on the encoder outputs before the seed projection head and is averaged over the two views, giving $\mathcal{R}_{\text{reg}}$. The full objective is

$$\mathcal{L}_{\text{Temp-Sub}} = \mathcal{L}_{\text{graph}} + \mathcal{L}_{\text{seed}} + \mathcal{R}_{\text{reg}}. \tag{9}$$

Appendix C and Appendix E provide more details.

4 Experiments and Results

4.1 Experimental Setup

Datasets and evaluation. We evaluate on 11 tasks from five RelBench databases, Arxiv, Avito, F1, H&M, and Event (Robinson et al., 2024), using the original chronological training, validation, and test splits. We report accuracy for multiclass classification, ROC–AUC for binary classification, and MAE for regression. All experiments are repeated with four random seeds, $\{42, 43, 44, 45\}$; reported task metrics are the means and standard deviations across these runs, with each run evaluated at its best validation checkpoint. Appendix A provides dataset and task details.

Backbones and baselines. We benchmark five backbones: two GNNs, HeteroGNN and RelGNN, and three GTs, HGT, RelGT, and THGFM. The controlled encoding and pretraining study focuses on HeteroGNN and THGFM, selected as the best-performing representatives of their respective families by mean validation rank across the 11 tasks (Appendix B.1). Selecting one backbone per family is a trade-off between architectural coverage and computational cost. All backbones use three layers with hidden dimension 128, and GTs use eight attention heads. We compare against nine pretraining baselines: AttrMasking, ContextPred, Un-SAGE, GPT-GNN, PT-HGNN, PHE, CTRL, TGPM, and TVE. Single-stage pretraining and downstream fine-tuning each use 10 epochs with 10,000 steps per epoch. Two-stage pretraining uses 10 epochs with 5,000 steps per epoch in each stage, matching the total single-stage pretraining budget of 100,000 steps. Models trained without pretraining use the same downstream budget. Appendix C provides the full architecture and training configuration.

4.2 Main Results

Aggregate performance across backbones. We denote the fixed day-based single-scale record-age encoding used by HeteroGNN (Robinson et al., 2024) as DayPE and use it as the baseline for multi-scale TimeMix on both backbones (Appendix B.2). Figure 3 uses the same-backbone supervised control with only DayPE and without TimeRoPE or pretraining as its reference (A0) throughout, measuring the effects of pretraining and, where enabled, our temporal encodings. In each of the four backbone–encoding settings, one of our two proposed staged schedules achieves the highest mean type gain among the evaluated methods. With both encodings, TS→Rel-Hist leads on the GNN at $+3.24\%$, and TS→Rel-Future leads on the GT at $+2.37\%$. The strongest baselines, including TS-initialized variants, reach $+2.96\%$ on the GNN (AttrMasking) and $+1.15\%$ on the GT (TS→PT-HGNN). Without our encodings, TS→Rel-Future leads on the GNN ($+2.51\%$), while TS→Rel-Hist leads on the GT ($+1.58\%$).

Performance under task-balanced aggregation. The benchmark contains one multiclass, six binary, and four regression tasks, so equal weighting of task types gives individual tasks different weights. We therefore also report average ranks with all 11 tasks weighted equally. Our staged schedules rank best in all four panels, with average ranks of 4.73, 7.14, 4.95, and 6.45, respectively. TS→Rel-Hist leads on the GNN with both encodings, and TS→Rel-Future leads in the other three panels; with both encodings, our two staged schedules occupy the top two positions on each backbone. Thus, the aggregate advantage is supported by both gain and rank summaries.

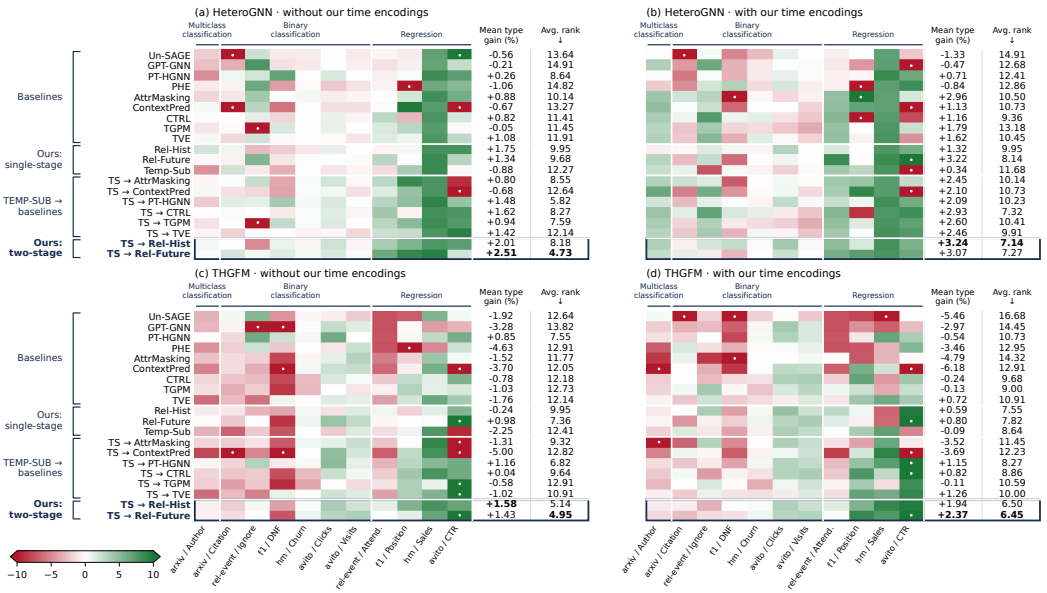

Figure 3: Relative test-performance gains (%) across 11 tasks, computed from task metrics averaged over four random seeds. All panels use the same-backbone supervised control without our time encodings (only with DayPE) or pretraining as reference (A0). Mean type gain averages gains within each task type and then equally across types; average rank weights all tasks equally (lower is better). Colors saturate at $\pm 10\%$; white dots mark values beyond this range. TS denotes TEMP-SUB; outlined rows identify our two-stage schedules, and bold summary values mark the best result in each panel.

Task coverage and sources of improvement. With both encodings, the best staged configuration for each backbone improves over A0 on 8 of the 11 tasks for the GNN and 7 for the GT, including all four regression tasks on both backbones. Regression provides the largest task-type gains: $+5.12\%$ on the GNN and $+8.55\%$ on the GT. On the latter, gains span driver position ($+8.74\%$), item sales ($+7.54\%$), and ad CTR ($+17.76\%$). Classification gains are less uniform: the GNN schedule improves author-category accuracy by 3.40% and binary classification by 1.21% on average; the GT schedule gains 0.26% on binary classification but loses 1.70% on author-category accuracy. The positive aggregates therefore reflect benefits across several regression targets alongside more variable classification transfer. Appendix B.3 reports detailed test results and standard deviations.

4.3 Ablation Studies

Contributions of the temporal encodings. Removing TIMEMIX restores DAYPE, the single-scale baseline introduced in Section 4.2, regardless of whether TIMEROPE is enabled. In Figure 4a, removing TIMEMIX reduces mean type gain by 0.59 percentage points on the GNN and 1.18 on the GT, compared with 0.28 and 0.46 points from disabling TIMEROPE. Removing both lowers gains by 0.46 and 1.29 points, respectively. The combined configuration therefore achieves the highest mean type gain on both backbones, with removing TIMEMIX causing the larger decrease. Effects vary across task types: disabling TIMEROPE slightly improves multiclass performance but reduces binary and regression gains on both backbones.

Effect of staged pretraining. Figure 4b measures pretraining gains over supervised controls with the same backbone and temporal encodings. Both proposed schedules yield positive mean type gains in all four settings. With both encodings, TS→REL-HIST achieves $+3.02\%$ on the GNN, while TS→REL-FUTURE achieves $+1.06\%$ on the GT, demonstrating benefits beyond temporal encoding alone. Under the same total pretraining budget, adding an initial TEMP-SUB stage improves over the corresponding single objective in seven of the eight comparisons for REL-HIST and REL-FUTURE. Notably, on the GT with both encodings, staging turns negative transfer into positive gains for both objectives, whereas all six TS-initialized baselines shown remain below the supervised control.

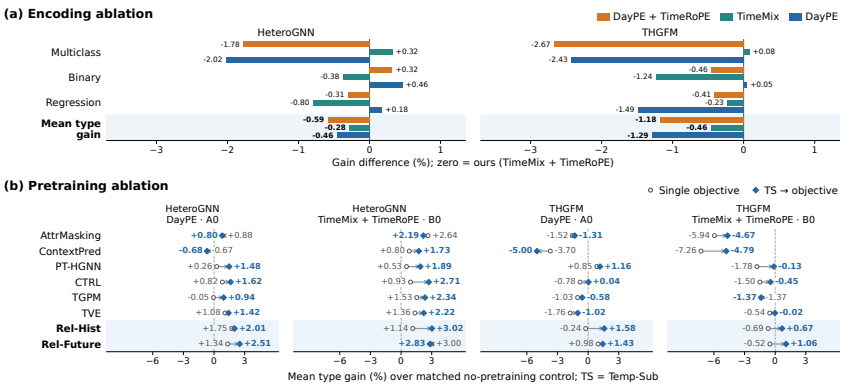

Figure 4: Contributions of temporal encoding and staged pretraining. (a) Gain differences from ours (TIMEMIX+TIMEROPE), without pretraining. DAYPE is the single-scale baseline for TIMEMIX. (b) Pretraining gains over matched supervised controls (A0: DAYPE; B0: TIMEMIX+TIMEROPE). Arrows connect single- and two-stage results with equal total pretraining steps.

The benefit nevertheless depends on the configuration: REL-FUTURE alone slightly outperforms its staged counterpart on the GNN with both encodings ($+3.00\%$ versus $+2.83\%$). Among the baselines, staging improves PT-HGNN, CTRL, TGPM and TVE in all four settings, while its effects on AttrMasking and ContextPred are mixed. One possible explanation is that the type-aware representations learned by TEMP-SUB better support relation- and schema-based supervision (Figure 2).

4.4 DISCUSSION AND LIMITATIONS

Temporal scales and horizon design. The statistics in Appendix B.2 reveal substantial variation in record-age distributions across databases. Avito includes records observed within one day before the prediction cutoff, with age variation at second, minute, and hour scales. Record ages in Event and H&M are concentrated at scales of days to months, whereas Arxiv and F1 include records dating back years. These differences motivate the multi-scale record-age representation in TIMEMIX.

The supported prediction horizons also vary across databases. Avito supports only 1- and 7-day horizons, while Event and H&M also support windows spanning weeks to months. In Arxiv and F1, 67.7% and 87.1% of eligible events, respectively, support a 365-day horizon. These differences motivate using multiple horizons for REL-FUTURE: short windows provide training examples where history is limited, while longer windows provide supervision on relation activity over months or a year where sufficient history is available. We therefore use candidate horizons from one day to one year.

Limitations. Our framework focuses on within-database training; transfer to unseen databases remains to be explored. Single- and two-stage pretraining use the same total number of training steps, although computational cost per step can vary across objectives. Comparisons under matched computational costs and with alternative stage orders would further clarify the contributions of objective composition and stage order.

5 CONCLUSION

We present an RDL framework that encodes record age and inter-record intervals and stages subgraph contrast before historical relation recovery or future activity prediction. Across five RelBench databases and 11 tasks, the two encodings together outperform either alone in aggregate on both backbones, and staged pretraining yields additional gains over supervised controls with the same encodings on both GNN and GT backbones. These results support combining explicit temporal representations with staged self-supervision for within-database prediction. Generalization to unseen databases and comparisons of stage orders under matched computational budgets remain directions for future work.

AI USE STATEMENT

Generative AI tools assisted with language editing and polishing. The authors are responsible for the final content, claims, and artifacts in this work.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, Section 2.1 specifies the graph construction and observation rule, and Section 3 describes the encodings, objectives, and training schedule. Appendix A documents database statistics, task definitions, and temporal splits; Appendix B reports task-level metrics and their variation; Appendix C specifies the architecture, optimization, and four-seed evaluation settings; Appendix D provides temporal implementation details; Appendix E specifies the decoders, losses, and sampling details for the pretraining objectives; and Appendix F presents the optimization curves. The appendix will be provided as a separate PDF in the Supplementary Materials. All code needed to reproduce the reported results will be provided as supplementary code in the Supplementary Materials, including instructions for downloading and processing the publicly available RelBench datasets (Robinson et al., 2024) used in our experiments.

REFERENCES

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022.

Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite message passing for relational deep learning, 2025. URL https://arxiv.org/abs/2502.06784.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks?, 2023. URL https://arxiv.org/abs/2302.11636.

Vijay Prakash Dwivedi, Charilaos Kanatsoulis, Shenyang Huang, and Jure Leskovec. Relational deep learning: Challenges, foundations and next-generation architectures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5999–6009, 2025.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational graph transformer. In *International Conference on Learning Representations*, volume 2026, pp. 154348–154369, 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 *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13592–13607, 2024.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 594–604, 2022.

Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. GraphMAE2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM web conference 2023*, pp. 737–746, 2023.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1857–1867, 2020b.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pp. 2704–2710, 2020c.

Xunqiang Jiang, Tianrui Jia, Yuan Fang, Chuan Shi, Zhe Lin, and Hui Wang. Pre-training on large-scale heterogeneous graph. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 756–766, 2021.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Chenglin Li, Yuanzhen Xie, Chenyun Yu, Junfeng Zhao, Yu Xia, Beibei Kong, Zang Li, and Di Niu. CTRL: Continuous-time representation learning on temporal heterogeneous information network. *Know.-Based Syst.*, 339(C), June 2026. ISSN 0950-7051. doi: 10.1016/j.knosys.2026.115514. URL https://doi.org/10.1016/j.knosys.2026.115514.

Longhai Li, Lei Duan, Junchen Wang, Chengxin He, Zihao Chen, Guicai Xie, Song Deng, and Zhaohang Luo. Memory-enhanced transformer for representation learning on temporal heterogeneous graphs. *Data Science and Engineering*, 8:98–111, 2023.

Yijun Ma, Zehong Wang, Weixiang Sun, and Yanfang Ye. Temporal graph pattern machine, 2026. URL https://arxiv.org/abs/2601.22454.

Yixin Peng, Diego Collarana, Er Jin, and Stefan Decker. THGFM: Dual-branch temporal heterogeneous graph fusion model, 2026. URL https://arxiv.org/abs/2607.27303.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '20, pp. 1150–1160. ACM, August 2020. doi: 10.1145/3394486.3403168. URL http://dx.doi.org/10.1145/3394486.3403168.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan E Lenssen, Yiwen Yuan, Zecheng Zhang, et al. RelBench: A benchmark for deep learning on relational databases. *Advances in Neural Information Processing Systems*, 37:21330–21341, 2024.

Lianze Shan, Jitao Zhao, Dongxiao He, Yongqi Huang, Zhiyong Feng, and Weixiong Zhang. MUG: Meta-path-aware universal heterogeneous graph pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 25260–25268, 2026.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Shengyin Sun, Chen Ma, and Jiehao Chen. PHE: Structure and semantic enhanced pre-training of graph neural networks for large-scale heterogeneous graphs. *ACM Transactions on Knowledge Discovery from Data*, 20(1):1–26, 2025.

Yijun Tian, Kaiwen Dong, Chunhui Zhang, Chuxu Zhang, and Nitesh V Chawla. Heterogeneous graph masked autoencoders. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 9997–10005, 2023.

Quang Truong, Zhikai Chen, Mingxuan Ju, Tong Zhao, Neil Shah, and Jiliang Tang. A pre-training framework for relational data with information-theoretic principles. *Advances in Neural Information Processing Systems*, 38:8122–8153, 2026.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model, 2025. URL https://arxiv.org/abs/2505.05568.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020.

Jinyu Yang, Cheng Yang, Junze Chen, Zedi Liu, Muhan Zhang, Hanyang Peng, and Chuan Shi. RelPrism: A multi-faceted pre-training framework with self-generated tasks for relational databases, 2026. URL https://arxiv.org/abs/2605.23241.

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

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 67686–67700. Curran Associates, Inc., 2023. doi: 10.52202/075280-2960. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d611019afba70d547bd595e8a4158f55-Paper-Conference.pdf.