Effective Intra–Inter Interaction Learning for Relational Tables

Weichen Li
weichenli@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Ken Zhong
zhongken@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Zheng Wang*
wzheng@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Li Pan
panli@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Jianhua Li
lijh888@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Abstract

Relational table learning has recently emerged as an important research direction for modeling multiple tables connected through primary key–foreign key (PK–FK) relationships. Despite recent advances, a principled modeling framework tailored to this task remains underexplored. In this paper, we propose **I**ntra–I**n**ter **R**elational **T**able **L**earning (**InRTL**), a unified framework that explicitly models dependencies both within and across relational tables. Specifically, InRTL formalizes two complementary interaction patterns: *intra-table interactions*, describing associations among rows within the same table, and *inter-table interactions*, describing dependencies between rows across PK–FK-linked tables. To model these dependencies, we develop a column-aware table encoder to generate initial row representations, followed by Transformer-based self-attention and cross-attention modules for intra-table and inter-table learning, respectively. To further improve scalability, InRTL incorporates linearized attention and heterogeneous graph neural networks to simplify the self-attention and cross-attention operations. Extensive experiments on ten datasets covering 24 real-world tasks demonstrate the effectiveness of our approach. Code is available at https://github.com/W1nterFloW/InRTL.

CCS Concepts

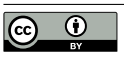
• **Information systems → Data mining**; • **Computing methodologies → Machine learning approaches**; • **Theory of computation → Design and analysis of algorithms**.

Keywords

Relational Table Learning; Data Mining; Deep Learning

ACM Reference Format:
Weichen Li, Ken Zhong, Zheng Wang, Li Pan, and Jianhua Li. 2026. Effective Intra–Inter Interaction Learning for Relational Tables. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817787

*Zheng Wang is the corresponding author.



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817787

1 Introduction

Tabular data is one of the most prevalent data modalities in real-world applications [8]. It has a simple and well-defined structure, consisting of samples (rows) described by a consistent set of features (columns). Owing to its structured format and strong interpretability, tabular data plays a central role across a wide range of domains, including medicine, finance, manufacturing, and many other application areas.

Traditional tabular learning methods primarily focus on single-table settings, aiming to capture complex interactions among table elements (like rows and columns) within the same table. Early methods mainly relied on shallow models such as decision trees, random forests, and gradient boosting machines [10], while more recent approaches employ deep neural networks to capture complex non-linear feature interactions [12]. However, extending these methods to multi-table scenarios typically requires extensive manual feature engineering to denormalize multiple tables into a single flat table [18]. This process is labor-intensive and often sacrifices the underlying relational structure, leading to performance bottlenecks.

Consequently, recent research has increasingly shifted toward relational table learning [3, 25, 28, 35]. This setting reflects modern relational databases, where data is distributed across multiple interconnected tables linked via primary-foreign key (PK-FK) relationships, forming rich semantic structures. Existing methods [6, 24] typically combine tabular neural networks (TNNs) [2] with graph neural networks (GNNs) [23]. TNNs encode each table independently to produce row-level embeddings capturing intra-table dependencies, which are then refined via GNN-based message passing to model inter-table relationships. Although these approaches have shown some effectiveness, they largely rely on generic GNN message-passing mechanisms, and a principled relational modeling framework tailored to this task remains underexplored.

In this paper, we propose **I**ntra–I**n**ter **R**elational **T**able **L**earning (**InRTL**), a unified framework that explicitly models both intra-table and inter-table dependencies. Specifically, we define two complementary interaction patterns: (1) *intra-table interactions*, which capture dependencies among rows within the same table, and (2) *inter-table interactions*, which capture dependencies between rows across PK–FK-linked tables. Based on this formulation, we develop a column-aware table encoder to generate initial row representations, followed by Transformer-based self-attention and cross-attention modules [30] for intra-table and inter-table modeling, respectively.

To improve scalability, InRTL further incorporates linearized attention and heterogeneous graph neural networks (HGNNs) to simplify the self-attention and cross-attention modules, respectively. Specifically, linearized attention reduces the computational cost of self-attention via a first-order Taylor approximation, while HGNNs reformulate cross-attention in terms of message passing to eliminate computations between rows without PK–FK relationships. As a result, InRTL scales linearly with both the number of table rows and the number of inter-table PK–FK relations, preserving expressive power while enabling efficient learning on large relational datasets.

InRTL is simple yet theoretically grounded. In contrast to traditional Transformer-based models, our method eliminates the need for positional encodings, or auxiliary training objectives, resulting in a compact and efficient learning pipeline. Furthermore, our work provides new insights into the representational power of Transformers for large-scale relational tables: by connecting linear attention with classical HGNN message passing, we offer both theoretical and practical support for leveraging GNN-inspired techniques in relational tabular learning.

In summary, we make the following contributions:

- We propose a novel relational tabular data learning method, InRTL, which explicitly defines and effectively captures both dependencies within and across tables.
- We theoretically analyze our method, including properties and error bounds of the linear attention mechanism, as well as the connection between Transformer and graph neural networks.
- We conduct extensive experiments on ten datasets from two well-established benchmarks, covering 24 real-world tasks, to validate the effectiveness of our proposed method.

2 Preliminaries

In this section, we introduce some key definitions and essential preliminary concepts used throughout this paper.

2.1 Problem Definition

In this subsection, we formally define relational table data and the intra- and inter-table interactions within it.

Definition 1 (*Relational Table Data*). *Relational table data is defined as* $(\mathcal{T}, \mathcal{K})$*, where* $\mathcal{T} = \{T_i\}_{i=1}^{|\mathcal{T}|}$ *denotes a set of tables and* $\mathcal{K} \subseteq \mathcal{T} \times \mathcal{T}$ *represents PK-FK relations among tables. Typically, each table in* $\mathcal{T}$ *contains a primary key to uniquely identify each row. If a table* T_i *contains a foreign key that references the primary key of another table* T_j*, then there exists a relation* $< T_i, T_j > \in \mathcal{K}$*.*

This definition characterizes multi-table data with primary and foreign key (PK-FK) structures, which are prevalent in practice. For instance, in the MovieLens dataset [13], the user, movie, and rating tables are linked via user and movie IDs through PK-FK relations.

Definition 2 (*Intra-table Interaction*). *Intra-table Interaction refers to explicit or implicit associations between different rows within the same table, reflecting contextual and structural relations among rows.*

Intra-table interaction typically implies that there exists some form of contextual correlation between rows within the table. For example, in the user table of MovieLens, users of the same age group tend to exhibit similar preferences in movie ratings.

Definition 3 (*Inter-table Interaction*). *Given two tables* T_i *and* T_j *connected via a PK-FK relation, inter-table interaction refers to the row-level associations arising from this linkage between two tables. This relation reveals how* T_i *and* T_j *complement each other by providing contextual information.*

Inter-table interactions capture associations between rows of different tables connected by PK–FK relations. For example, in the movies table of the MovieLens dataset, the movie ID can link to rows in the ratings table to obtain corresponding rating information. Modeling such inter-table interactions helps extract valuable contextual information from the related tables.

2.2 Heterogeneous Graph Neural Networks

Heterogeneous Graph Neural Networks (HGNNs) [36] are graph neural network frameworks specifically developed to learn representations over heterogeneous graphs. Generally, an HGNN layer aims to compute the node representation $\vec{h}_v$ for each node v, formalized as:

$$\vec{h}_v = \mathrm{UPD}_{\psi(v)}\left(\mathrm{AGG}_r\left(\left\{\mathcal{N}_v^r : r \in \mathcal{R}\right\}\right)\right) \tag{1}$$

where $\mathcal{R}$ denotes the set of edge types, and $\mathcal{N}_v^r$ represents the neighbors of node v connected via edges of type r. $\mathrm{AGG}_r(\cdot)$ is an edge-type-specific aggregation function, and $\mathrm{UPD}_{\psi(v)}(\cdot)$ is a node-type-specific update function for nodes of type $\psi(v)$.

2.3 Transformer Architecture

The Transformer [30] consists of multiple stacked layers. Each layer has an attention module and a feed-forward network (FFN). Let $X \in \mathbb{R}^{n \times d}$ denote the input to the attention module, where n is the sequence length and d is the feature dimension. The input X is linearly projected to query Q, key K, and value V matrices using projection matrices $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$. The attention is then computed as:

$$\begin{aligned} Q = XW_Q,\ K = XW_K,\ V = XW_V \\ A = \frac{QK^\top}{\sqrt{d}}, \quad \mathrm{Attn}(X) = \mathrm{Softmax}(A)V \end{aligned} \tag{2}$$

For brevity, we only describe the self-attention with single-head here, extension to multi-head and cross-attention is straightforward.

3 Methodology

As shown in Figure 1, the proposed InRTL method first employs a column-aware table encoder to extract feature representations from each table independently. Then, it adopts a Transformer-like architecture to simultaneously model both intra- and inter-table interactions to get representations. Finally, these two types of representations are integrated to generate the final output.

3.1 Column-aware Table Encoder (ColATE)

This module is designed to learn high-dimensional and dense feature representations for each row in a single table. It consists of

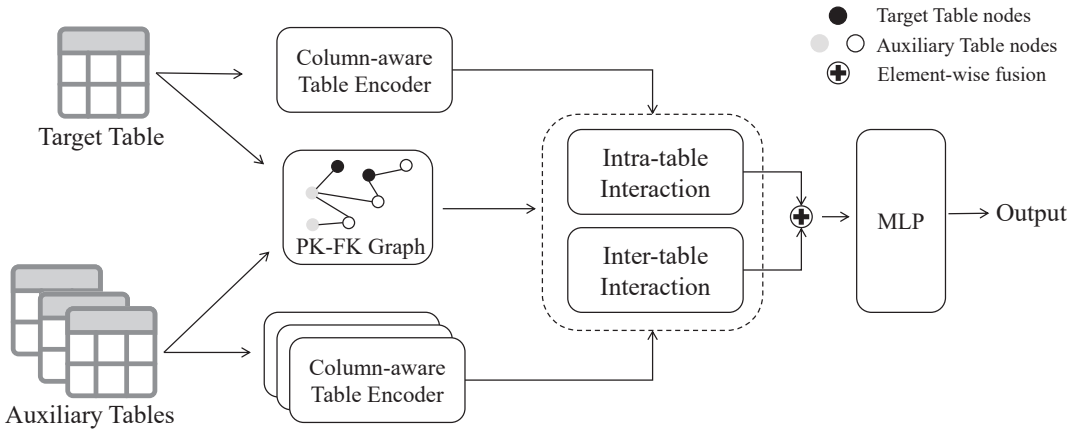


Figure 1: Overview of InRTL, where each row in the target and auxiliary tables is treated as a node in the PK-FK graph.

three components: pre-encoding module, feature-weighting module, and fusion module.

First, in the pre-encoding module, we apply a learnable parameterized mapping to embed each cell in the table into a high-dimensional space. After pre-encoding, a table with n rows and m columns is mapped to the same d-dimensional vector space, denoted as a collection of column matrices $\{Z_1, Z_2, \ldots, Z_m\}$ where $Z_i \in \mathbb{R}^{n\times d}$ represents the feature matrix of the i-th column.[1] Pre-encoding is a crucial step in tabular representation learning. We refer to [11] for more details.

Next, in the feature-weighting module, we compute a separate importance weight for each column, since different tasks may rely unevenly on different columns. Specifically, for each column matrix Z_i, we perform average pooling over all its elements to produce a scalar importance score s_i. Then, a Softmax operation is applied to normalize the scores across all columns, resulting in the importance weights α_i:

$$\alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^{m} \exp(s_j)} \tag{3}$$

Finally, in the fusion module, we first apply these computed importance weights to re-scale each column matrix, thereby dynamically focusing on more informative features. After re-weighting, a deep residual network ResNet$(\cdot)$ [14] is employed to capture higher-order interactions among columns. The final aggregated representation $E \in \mathbb{R}^{n\times d}$ is computed as:

$$E = \text{ResNet}\big(\text{Concat}(\alpha_1 Z_1,\ \ldots,\ \alpha_m Z_m)\big) \tag{4}$$

where Concat$(\cdot)$ denotes concatenation along the column dimension.

3.2 Modeling Intra- and Inter-table Interactions with Transformer

To simplify matters, we assume that there are only two tables with n rows.[2] Without loss of generality, we designate one table as the target table and the other as the auxiliary table. The encoded representations of both tables by ColATE in Section 3.1 are denoted as $E_{\text{tgt}}, E_{\text{aux}} \in \mathbb{R}^{n\times d}$. We next describe how to model intra- and inter-table features for the target table using the classical Transformer architecture.

3.2.1 Intra-table Interaction. For the target table, let $H^{(l)}_{\text{intra}} \in \mathbb{R}^{n\times d}$ denote the intra-table representation at layer l and initialize $H^{(0)}_{\text{intra}}$ with E_{tgt}. We then apply self-attention to capture contextual dependencies within the target table. Specifically, following the standard Transformer architecture with self-attention, the representation at layer l+1 is computed as:

$$\hat{H}^{(l)}_{\text{intra}} = \text{Attn}\big(\text{LN}(H^{(l)}_{\text{intra}})\big) \tag{5}$$

$$H^{(l+1)}_{\text{intra}} = \text{FFN}\big(\text{LN}(\hat{H}^{(l)}_{\text{intra}})\big) + \hat{H}^{(l)}_{\text{intra}} \tag{6}$$

where LN$(\cdot)$ denotes layer normalization.

Intuitively, self-attention allows each row to attend to all other rows within the same table, enabling the model to capture global contextual information within the table effectively.

3.2.2 Inter-table Interaction. Similarly, let $H^{(l)}_{\text{inter}} \in \mathbb{R}^{n\times d}$ denote the inter-table representation of the target table at layer l, and initialize $H^{(0)}_{\text{inter}}$ with E_{tgt}. We employ cross-attention (denoted by CrossAttn$(\cdot)$) to model the interactions between the target and auxiliary tables. Specifically, $H^{(l)}_{\text{inter}}$ serves as the query source and E_{aux} as the key and value sources. To mitigate noise from unrelated rows, we apply a binary *mask* that identifies PK–FK-linked row pairs between target table and auxiliary table, masking unrelated entries with $-\infty$. The inter-table representation at the l+1 layer is computed via a Transformer block with cross-attention as:

$$\begin{aligned}\hat{H}^{(l)}_{\text{inter}} = \text{CrossAttn}\big(&\text{LN}(H^{(l)}_{\text{inter}}),\\ &\text{LN}(E_{\text{aux}}), \mathit{mask}\big)\end{aligned} \tag{7}$$

$$H^{(l+1)}_{\text{inter}} = \text{FFN}\big(\text{LN}(\hat{H}^{(l)}_{\text{inter}})\big) + \hat{H}^{(l)}_{\text{inter}} \tag{8}$$

Intuitively, cross-attention enables each row in the target table to attend selectively to PK–FK-related rows in the auxiliary table, effectively integrating relevant information while ignoring irrelevant rows.

[1] For notational simplicity, all representation/feature dimensions are denoted as d throughout the methodology section.

[2] This assumption is made solely to simplify the presentation. As demonstrated in our experiments, our method is also applicable to scenarios involving multiple auxiliary tables with varying numbers of rows.

3.2.3 Combination of Two Interactions. Finally, we fuse the intra-table and inter-table representations and apply a multilayer perceptron (MLP) to produce the final prediction. Specifically, assuming the model has L layers, the final prediction for the target table is given by:

$$\mathrm{MLP}\big(\beta H_{\text{intra}}^{(L)} + (1-\beta) H_{\text{inter}}^{(L)}\big) \tag{9}$$

where β is a learnable weighting coefficient that balances the contributions of intra- and inter-table representations. The entire framework can then be optimized end-to-end using task-specific loss functions.

Scalability Limitation. Adopting classical Transformer blocks for both intra-table (Eq. 5) and inter-table modeling (Eq. 7) requires pairwise attention among the n rows in each case. Both involve computing an $n \times n$ attention matrix, resulting in $O(n^2)$ time complexity [30], which limits scalability to large relational table datasets.

3.3 Simplifying Transformer for Table Interactions

In this subsection, we propose simplification strategies for both intra-table and inter-table interactions to improve scalability to large tabular datasets.

3.3.1 Intra-table Interaction Simplification. To reduce the computational complexity of intra-table self-attention, we propose a linear approximation to the standard Transformer self-attention.

Theorem 1. *Let $H_{intra}^{(l)} \in \mathbb{R}^{n\times d}$ denote the intra-table representation at the l-th layer, where n is the number of rows and d is the feature dimension. Following the standard Transformer architecture (Eq. 2), define $Q = H_{intra}^{(l)} W_Q$, $K = H_{intra}^{(l)} W_K$, $V = H_{intra}^{(l)} W_V$. Assume that each row vector $\vec{q}_i$, $\vec{k}_i$, and $\vec{v}_i$ in Q, K, and V is normalized such that $\|\vec{q}_i\|_2 = \|\vec{k}_i\|_2 = \|\vec{v}_i\|_2 = 1$. By applying a first-order Taylor expansion of the exponential function at zero within the Softmax operation of Eq. 2, the intra-table attention defined in Eq. 5 becomes:*

$$\hat{H}_{intra}^{(l)} = D^{-1}\big[J_n V + Q\,(K^\top V)\big] \tag{10}$$

where $J_n = \vec{1}_n \vec{1}_n^\top \in \mathbb{R}^{n\times n}$ is an all-ones matrix, $\vec{1}_n \in \mathbb{R}^n$ is a vector of all ones, and $D = \mathrm{Diag}\left(n + Q(K^\top \vec{1}_n)\right)$ is a diagonal matrix.

This theorem shows that self-attention in Transformer can be approximated with linear complexity.[3] Substituting Eq. 10 into Eq. 6, we derive the simplified formulation for modeling intra-table interactions:

$$\begin{aligned} H_{\text{intra}}^{(l+1)} = \mathrm{FFN}\big(D^{-1}\big[J_n V + Q\,(K^\top V)\big]\big) \\ + D^{-1}\big[J_n V + Q\,(K^\top V)\big] \end{aligned} \tag{11}$$

We now derive an upper bound for the approximation error of the linearized attention weights.

Corollary 1. *Continue to use the notation q, k, and n from Theorem 1. Let a_{ij} denote the original Transformer attention weight, and $\tilde{a}_{ij}$ denote the proposed linearized approximation:*

$$a_{ij} = \frac{\exp\big(\vec{q}_i^\top \vec{k}_j\big)}{\sum_{l=1}^n \exp\big(\vec{q}_i^\top \vec{k}_l\big)}, \tilde{a}_{ij} = \frac{1 + \vec{q}_i^\top \vec{k}_j}{n + \sum_{l=1}^n \vec{q}_i^\top \vec{k}_l} \tag{12}$$

[3]Due to space limitations, all proofs of the formulas presented in this paper are provided in Appendix A.

where $i, j \in \{1, \ldots, n\}$ are indices. For any i and j, we have:

$$\big|\, a_{ij} - \tilde{a}_{ij} \big| = O(\epsilon^2)$$

where ϵ is a constant such that $\max_{i,j} |\vec{q}_i^\top \vec{k}_j| \le \epsilon < 1$.

This corollary shows that the approximation error introduced in Eq. 10 is second-order with respect to the query–key dot product. Since normalized query and key vectors tend to be nearly orthogonal in high-dimensional spaces, their dot products are typically small, and the resulting approximation error is negligible in practice.

3.3.2 Inter-table Interaction Simplification. Motivated by recent studies that investigate the connection between Transformers and graph neural networks [21], we adopt the HGNN framework as an alternative to the standard Transformer for modeling inter-table interactions.

Proposition 1. *Let the bipartite graph be $\mathcal{G} = (\mathcal{V}, \mathcal{U}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{U}$ denote the sets of rows in the target and auxiliary tables (treated as nodes), and $\mathcal{E}$ denotes the set of edges induced by PK–FK relationships. The inter-table interaction calculated in Eq. 7 is equivalent to a single round of message passing over $\mathcal{G}$, with attention weights serving as edge weights.*

This proposition demonstrates that the cross-attention mechanism in Transformers can be effectively replaced by sparse HGNN layers, leading to substantial reductions in both time and space complexity. In particular, relational tables can be abstracted as a heterogeneous graph, where each PK–FK relation defines a distinct edge type. This allows the HGNN to model cross-table interactions in a computationally efficient manner.

Let the rows in the target and auxiliary tables correspond to heterogeneous graph nodes $\{v_i\}_{i=1}^n$, $\{u_j\}_{j=1}^n$ respectively, and $\vec{h}_{v_i}^{(l)}$, $\vec{h}_{u_j}^{(l)}$ denote the representations of the i-th row and j-th row treated as nodes in the target and auxiliary tables at layer l, initialized from the corresponding rows in E_{tgt} and E_{aux}. Then, by reformulating Eq.7 and Eq.8 using Eq. 1, we obtain:

$$\vec{h}_{v_i}^{(l+1)} = \mathrm{UPD}_{\psi(v_i)}\Big(\vec{h}_{v_i}^{(l)},\ \mathrm{AGG}_r\{\vec{h}_{u_j}^{(l)} \mid u_j \in \mathcal{N}_{v_i}\}\Big) \tag{13}$$

Consequently, for the target table, the final inter-table interaction representation is $H_{\text{inter}}^{(l+1)} = \big[\vec{h}_{v_1}^{(l+1)}, \ldots, \vec{h}_{v_n}^{(l+1)}\big]$.

3.3.3 Combination of Two Interactions. Similarly, after simplification, we compute both intra- and inter-table representations independently and fuse them using Eq. 9. Furthermore, it is worth noting that this simplification enables our method to be combined with mini-batch sampling, allowing interactions to be modeled only on sampled subtables and subgraphs. This leads to efficient learning on large-scale relational table data. For completeness, the overall training procedure of InRTL is summarized in Algorithm 1 in Appendix D.

4 Algorithm Analysis

In this section, we first analyze the time complexity of our method and then discuss its relation to existing approaches.

4.1 Time Complexity Analysis

The computational cost of our method arises from three main components: the column-aware table encoder, the linearized self-attention for intra-table interaction, and the HGNN message passing for inter-table interaction. First, for the column-aware table encoder, the primary computational cost arises from the ResNet, which has a complexity of $O(nmd^2)$, where n and m denote the number of rows and columns in the table, respectively. Second, for intra-table interaction, the linear attention mechanism (i.e., Eq. 10) operates with a time complexity of $O(nd^2)$. Third, for inter-table interaction, the HGNN performs message passing on the heterogeneous graph with a single-layer complexity of $O(ed)$, where e represents the number of edges induced by PK–FK relations. In summary, the overall time complexity is $O(nmd^2 + nd^2 + ed)$. In practice, the number of rows and edges are typically much larger than the number of columns and feature dimension (i.e., $n, e \gg m, d$). Therefore, the overall complexity scales linearly with the number of table rows and the number of inter-table PK–FK relations.

4.2 Relation to Prior Linear Attention Methods

In this subsection, we compare our Taylor-expanded linear attention with several prior linearization methods. To maintain notational consistency, we continue using the symbols introduced in Section 3 throughout the discussion.

Relation to Linear Transformer [19]. Linear Transformer linearizes the classical softmax attention (i.e., Eq. 2) using a kernel-based feature mapping $\phi(\cdot)$ applied independently to query and key representations:

$$\begin{aligned} \mathrm{Attn}(X) &\approx D^{-1}\phi(Q)\big(\phi(K)^\top V\big), \\ D &= \mathrm{Diag}\big(\phi(Q)(\phi(K)^\top \vec{\mathbf{1}}_n)\big) \end{aligned} \tag{14}$$

The choice of $\phi(\cdot)$ (e.g., $\mathrm{elu}(\cdot)+1$) is empirically driven without an exact theoretical derivation, making the approximation sensitive to kernel choice and input distributions. In contrast, our method (i.e., Eq. 10) is based on a strong theoretical foundation, by directly applying a first-order Taylor expansion of the exponential term in the Softmax. Our formulation is deterministic, interpretable, and requires no additional hyper-parameters. In fact, Linear Transformer only applies kernel $\phi(\cdot)$ to Q and K separately, if the kernel $\phi(\cdot)$ is applied jointly to $QK^\top$ such that $\phi(QK^\top) = J_n + QK^\top$, it will directly yield our formulation (i.e., Eq. 10).

Relation to Performer [7]. Similar to Linear Transformer mentioned above, Performer also linearizes the softmax attention by using a kernel-based feature map. Specifically, it constructs a set of random features that serve as unbiased estimators of the Softmax kernel:

$$\mathrm{Attn}(X) \approx \mathbb{E}_{\vec{\omega}}\Big[\phi_{\vec{\omega}}(Q)\big(\phi_{\vec{\omega}}(K)^\top V\big)\Big] \tag{15}$$

where $\vec{\omega} \sim \mathcal{N}(\mathbf{0}, I_d)$ is a d-dimensional standard Gaussian, $\phi_{\vec{\omega}}(\vec{\mathbf{x}}) = \exp(\vec{\omega}^\top \vec{\mathbf{x}} - \frac{1}{2}\|\vec{\mathbf{x}}\|^2)$, I_d is the $d \times d$ identity matrix, and $\mathbb{E}_{\vec{\omega}}(\cdot)$ denotes the expectation over the $\vec{\omega}$. In practice, the expectation is approximated by averaging over a few samples of $\vec{\omega}$, which introduces Monte Carlo variance. In contrast, Eq. 10 in our method attains linear complexity without relying on random features. This is accomplished through a first-order Taylor expansion, resulting in a deterministic formulation that requires no additional hyper-parameters.

Table 1: Statistical analysis of the used benchmark datasets.

Benchmark	Dataset	#Tables	#Rows
SJTUTables	TACM12K	4	97,774
	TLF2K	3	101,773
	TML1M	3	1,010,132
RelBench	rel-f1	9	97,606
	rel-trial	15	5,852,157
	rel-avito	8	20,679,117
	rel-amazon	3	24,291,489
	rel-hm	3	33,265,846
	rel-stack	7	38,109,828
	rel-event	5	41,328,337

Table 2: Classification results on SJTUTables (Accuracy %, ↑).[4]

Model	TML1M	TLF2K	TACM12K	Avg. Rank
LightGBM	26.66	35.52	35.70	5.7
TabPFN	25.20	43.20	34.10	5.3
FT-Transformer	28.60	15.80	14.80	8.3
TabNet	24.40	15.50	21.20	9.3
SAINT	27.90	13.70	16.50	9.0
Trompt	27.80	13.40	12.40	10.3
ExcelFormer	31.90	20.60	17.10	6.7
BRIDGE	36.20	42.20	25.60	3.3
RDL	35.60	28.70	19.20	5.3
RelGNN	27.12	20.31	18.19	8.0
LightRDL	24.94	39.17	35.77	5.7
InRTL	**40.60**	**45.80**	**48.40**	**1.0**

Relation to Linformer [31]. Linformer reduces the quadratic complexity of self-attention by introducing a low-rank projection on the key and value matrices:

$$\mathrm{Attn}(X) \approx \mathrm{softmax}\left(\frac{Q(\hat{K}K)^\top}{\sqrt{d}}\right)(\hat{V}V) \tag{16}$$

where $\hat{K}, \hat{V} \in \mathbb{R}^{d' \times n}$ are learnable projection matrices with $d' \ll n$. This formulation assumes that the attention matrix is inherently low-rank, thus approximating the Softmax weights through dimension reduction. However, a small projection dimension inevitably restricts the expressive power of the resulting attention scores, leading to potential information loss. In contrast, as shown in Eq. 12, our Taylor-based approach directly linearizes the exponential kernel: $e^{\vec{q}_u{}^\top \vec{k}_v} \approx 1 + \vec{q}_u{}^\top \vec{k}_v$, yielding a closed-form expression, which achieves linear complexity $O(nd^2)$ without imposing any low-rank assumption. Therefore, our method preserves all pairwise interactions in full-rank space while maintaining efficiency comparable to Linformer.

[4]The updated TLF2K dataset excludes artist embeddings, reducing accuracy from 50.10% to 45.80%.

Table 3: Binary classification results on RelBench (ROC-AUC %, ↑).

Dataset	Task	LightGBM	TabPFN	FT-Trans	Trompt	ExcelFormer	RDL	RelGNN	BRIDGE	LightRDL	InRTL
rel-amazon	user-churn	52.22	60.51	52.30	53.10	52.09	70.42	65.44	70.51	69.17	**70.53**
	item-churn	62.54	63.48	64.15	59.82	62.50	82.81	82.64	79.93	81.20	**82.83**
rel-avito	user-visits	53.05	65.34	52.10	56.11	55.01	65.99	65.47	66.16	62.39	**66.25**
	user-clicks	53.60	64.58	51.32	54.23	55.32	65.76	66.13	65.01	64.11	**67.03**
rel-event	user-repeat	68.04	65.85	69.10	67.17	69.73	75.60	73.07	75.30	71.32	**78.73**
	user-ignore	79.93	83.24	75.69	77.31	77.92	75.14	79.11	76.16	76.30	**85.80**
rel-f1	driver-dnf	68.56	68.38	68.31	67.52	67.20	71.25	67.06	70.16	68.63	**74.60**
	driver-top3	73.92	72.18	73.62	73.17	75.22	77.23	80.86	77.13	79.16	**83.21**
rel-hm	user-churn	55.21	69.75	50.13	54.19	53.95	70.11	68.10	69.33	67.13	**70.32**
rel-stack	user-engagement	63.39	88.34	60.02	60.73	64.20	90.47	89.95	89.74	89.02	**90.56**
	user-badge	63.43	86.73	60.36	62.82	67.13	88.86	88.98	87.90	86.71	**89.02**
rel-trial	study-outcome	70.09	67.88	62.39	65.73	67.31	69.02	69.16	68.98	68.15	**70.20**
Avg. Rank		6.92	6.00	8.75	8.25	7.58	3.33	3.92	4.08	5.17	**1.00**

In summary, prior linear attention methods achieve efficiency through various approximate strategies that may introduce uncontrolled uncertainty, such as heuristic kernel design, stochastic random features, or structural low-rank constraints. Our method, in contrast, leverages deterministic Taylor linearization in inner-product space, improving stability and theoretical completeness.

5 Experiments

In this section, we conduct comprehensive experiments to evaluate our method against state-of-the-art baselines.

5.1 Experimental Setup

Datasets. As summarized in Table 1, we conduct experiments on two public relational table learning benchmarks. The first is SJTUTables [25], which includes three datasets covering domains such as online movies, online music, and academic paper citations. Following the original benchmark configuration, we report accuracy as the evaluation metric for its three multi-class classification tasks. The second benchmark, RelBench [28], includes seven datasets covering areas such as e-commerce, healthcare, and online communities. This benchmark involves 21 tasks, including 12 binary classification tasks and 9 regression tasks. In accordance with the benchmark guidelines, we adopt ROC-AUC for classification tasks and Mean Absolute Error (MAE) for regression tasks. Additionally, for all experiments, we follow the standard data preprocessing and train/val/test split provided by each benchmark.

Baselines. We compare our method with various baselines, categorized into three main groups. The first group comprises traditional decision tree models, represented by LightGBM [20], a gradient boosting framework known for its efficiency and strong performance on tabular data. The second group includes deep learning-based single-table models: TabPFN [15], FT-Transformer [12], TabNet [1], SAINT [29], Trompt [5], and ExcelFormer [4]. These methods leverage attention mechanisms and advanced representation learning techniques to capture intra-table dependencies. The third group consists of relational table learning methods like BRIDGE [25], RDL [28], LightRDL [24] and RelGNN [6]. The details about these baselines can be found in Appendix B.

Implementation Details. We re-ran and carefully tuned all models using their official implementations. For RelGNN and LightRDL, which lacked complete training scripts, we re-implemented the missing components to ensure consistent evaluation. All reported results are averaged over 20 independent runs. Our experimental pipeline is built on the open-source relational table learning library relationLLM (rLLM)[5], and additional implementation details are provided in Appendix C.

5.2 Performance Evaluation on Benchmarks

Table 2 presents the multi-class classification results on SJTUTables, while Table 3 and Table 4 report the binary classification and regression results on RelBench, respectively. Best values are in bold. We summarize the following key observations from the results.

First, we find that relational table learning methods generally outperform single-table approaches, including both shallow and deep learning models. This highlights that leveraging multiple tables and their PK–FK relationships can significantly improve task performance, underscoring the strong potential of relational table learning. Second, our method achieves consistently strong performance across all datasets and tasks. This suggests that the proposed intra- and inter-table interaction mechanisms effectively capture multidimensional information to enhance performance. This also demonstrates the generalizability of our approach and its potential for widespread application. Third, we observe significant variation in the performance gains of our method across different datasets and tasks. For example, substantial improvements are observed on TACM12K, rel-f1, and rel-event, whereas the gains are smaller on rel-amazon and rel-stack. This variability likely stems from dataset-specific factors, such as differences in table size, sparsity, and PK–FK graph topology. These insights motivate further investigation into how such data factors influence model performance, which presents a promising avenue for future research. Last, on the SJTUTables

[5] https://github.com/rllm-project/rllm

Table 4: Regression results on RelBench (MAE, ↓).

Dataset	Task	LightGBM	TabPFN	FT-Trans	Trompt	ExcelFormer	RDL	RelGNN	BRIDGE	LightRDL	InRTL
rel-amazon	user-ltv	16.783	19.413	16.930	16.606	17.001	14.313	16.783	15.374	14.310	**14.250**
	item-ltv	60.569	70.796	60.319	61.424	59.983	50.053	48.826	**47.944**	48.112	48.805
rel-avito	ad-ctr	0.041	0.044	0.049	0.045	0.050	0.041	0.038	0.039	0.040	**0.037**
rel-event	user-attendance	0.264	0.410	0.270	0.281	0.264	0.258	0.248	0.258	0.260	**0.241**
rel-f1	driver-position	4.170	5.068	4.210	4.191	4.112	4.172	4.250	4.179	4.079	**3.904**
rel-hm	item-sales	0.076	0.094	0.092	0.073	0.073	0.056	0.054	0.060	**0.044**	0.054
rel-stack	post-votes	0.068	0.079	0.074	0.080	0.070	0.065	0.065	0.066	0.064	**0.063**
rel-trial	study-adverse	44.011	45.609	44.490	45.397	45.552	44.473	44.461	44.773	43.910	**43.522**
	site-success	0.425	0.460	0.431	0.440	0.429	0.400	0.355	0.390	0.411	**0.331**
Avg. Rank		5.94	9.56	7.89	7.94	7.11	4.28	3.94	4.17	2.78	**1.39**

dataset, we observe that two relational table learning methods (RDL and RelGNN) perform relatively poorly. This may be because their implementations force each text column to use simple GloVe [27] embeddings independently. In contrast, following BRIDGE, our method utilizes preprocessed embeddings from BERT [9] for entire rows of auxiliary text tables, enhancing textual understanding. Text encoder choice can materially affect relational table learning. Therefore, evaluating under different settings helps separate architectural gains from encoder gains. Further experiments on different text embedding choices, including GloVe and a lightweight BERT-family encoder (all-MiniLM-L6-v2 [32]), are provided in Appendix C.4.

5.3 Ablation Study

This part is designed to assess the contribution of three major components in our method: the column-aware table encoder, the linear-attention-based intra-table interaction, and the HGNN-based inter-table interaction (denoted as ColATE, LIN-ATTN, and HGNN). We remove one component at a time from the full model and measure the performance to assess its individual contribution. Due to space limitations, we perform ablation studies on three representative datasets: rel-f1, TML1M, and rel-amazon, covering small-, medium-, and large-scale classification tasks. Similar results are found on other datasets as well.

The results are presented in Table 5. We observe that each of the three components contributes to the overall performance of InRTL. Removing any module results in a noticeable performance degradation. This demonstrates the effectiveness of the three core components in our method, underscoring the necessity of jointly modeling both intra- and inter-table interactions to fully exploit relational structures in tabular data.

5.4 Hyper-parameter Analysis

This subsection investigates the impact of two key hyper-parameters in our method. These experiments are all performed on the rel-f1 dataset under its default classification settings (driver-dnf and driver-top3). Consistent trends are observed across other datasets.

Effect of Model Depth. We examine two depth-related factors: (i) the number of linear attention layers for intra-table interaction and (ii) the number of HGNN layers for inter-table interaction. For each

Table 5: Ablation results, where TML1M is evaluated using accuracy %, and the others are evaluated using ROC-AUC %.

Dataset	Task	InRTL	Without		
			ColATE	LIN-ATTN	HGNN
TML1M	age-cls	**40.60**	37.70	38.60	39.80
rel-f1	driver-dnf	**74.60**	73.10	72.40	71.40
	driver-top3	**83.90**	80.90	80.70	80.40
rel-amazon	user-churn	**70.53**	70.49	70.45	56.07
	item-churn	**82.80**	82.40	82.60	64.05

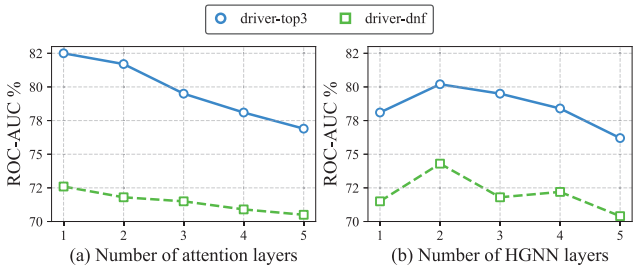


Figure 2: Performance variation with respect to model depth. (a) Linear attention layers for intra-table modeling. (b) HGNN layers for inter-table modeling.

setting, we independently vary the number of layers, while keeping all other components fixed. As shown in Figure 2, we observe that increasing the number of linear attention layers consistently degrades performance. We attribute this to the expressive power of multi-head attention, which captures sufficient intra-table interactions in the early stages, while deeper layers may introduce noise. On the other hand, HGNN performance is non-monotonic: with increasing depth, the performance first rises then falls, peaking at two layers. This aligns with well-known findings in GNNs. Specifically, a single layer generally lacks sufficient receptive field to capture informative multi-hop dependencies across relational tables, while deeper networks risk exponential neighbor growth or

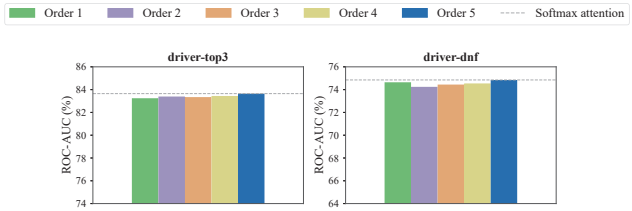


Figure 3: ROC-AUC comparison of attention methods with different taylor series orders.

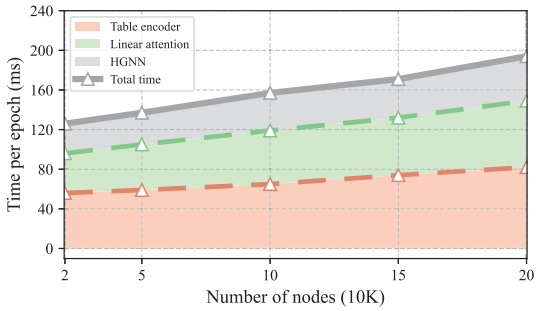


Figure 4: Scalability test of training time per epoch w.r.t. graph sizes (a.k.a. node numbers).

over-smoothing [33]. Therefore, using two HGNN layers achieves a good trade-off between expressiveness and efficiency.

Effect of Order Number in Taylor Expansion. We investigate the impact of the Taylor expansion order on attention optimization for modeling intra-table interactions. In this experiment, all other components of InRTL are fixed, and we compare different attention mechanisms by varying the order of the Taylor expansion of the exponential function in Softmax. As an important baseline, we also evaluate the original Transformer architecture (i.e., without any Taylor expansion). As shown in Figure 3, the performance gradually approaches that of the original Softmax attention as the Taylor expansion order increases. We also find that the overall difference remains small, even with the first-order expansion. Since only the first-order expansion has linear complexity (higher orders are all quadratic), we adopt it to achieve an effective balance between computational efficiency and modeling precision.

5.5 Scalability Analysis

To evaluate scalability, we synthetically generated several large graph datasets. In these datasets, each node is treated as a row in a table with 5 columns on average, and the total number of nodes ranges from 10K to 200K, with an average node degree of 5. We ran InRTL on these datasets and recorded the runtime of each stage for one epoch to assess its scalability.

As shown in Figure 4, we can clearly see that the overall training time of InRTL grows approximately linearly with the number of nodes. The table encoder and linear attention module components dominate the runtime and exhibit similar growth patterns as the dataset size increases. This is because the table encoder part processes each feature column of every node, while the linear attention part operates on the node features, leading both components to scale roughly proportionally with the number of nodes and feature dimensions. In contrast, the HGNN component contributes a relatively small fraction of the total runtime, and its growth is moderate due to the fixed average node degree in the synthetic datasets. These experimental results confirm our theoretical expectations and demonstrate that InRTL scales efficiently to large datasets.

6 Related Work

In this section, we review prior work on deep learning for single-table and multi-table tabular data. This section is organized into single-table modeling and multi-table modeling to clarify their assumptions and limitations.

6.1 Deep Learning on Single Table

Deep learning for single-table tabular data primarily focuses on learning expressive feature representations and capturing complex feature interactions without manual feature engineering [2]. A prominent line of research is based on attention mechanisms [30], where features are treated as tokens and modeled through self-attention. Representative methods such as TabTransformer [16] and FT-Transformer [12] learn contextualized feature representations by modeling pairwise and higher-order feature dependencies within a table. Another line of work emphasizes feature selection and interpretability. Models such as TabNet [1] and ExcelFormer [4] dynamically select and reweight informative features while suppressing irrelevant ones, thereby improving both predictive performance and interpretability. More recently, prompt-inspired approaches such as Trompt [5] incorporate feature semantics and instance-level importance into unified representations, bridging ideas from language modeling and tabular learning. Despite their effectiveness, these methods are fundamentally designed for isolated single tables and do not explicitly model relational dependencies across multiple tables.

6.2 Deep Learning on Multiple Tables

Recent efforts have increasingly explored deep learning on multi-table data, which can be broadly categorized into two lines of research. The first line focuses on transfer learning across independent tables without explicit relational structure. These methods aim to capture shared representations or inductive biases across datasets to improve generalization to new tasks. Classical transfer learning approaches [22, 34] learn cross-dataset commonalities to enhance performance on downstream targets. More recently, tabular foundation model approaches such as TabPFN [15] pretrain on large collections of tabular datasets, enabling in-context learning and rapid adaptation to unseen tasks. The second line focuses on relational tabular data [35], where multiple tables are connected via foreign-key relationships. A common paradigm is to represent relational schemas as graphs, where tabular neural networks encode individual tables and graph neural networks capture inter-table dependencies [6, 24]. Recent benchmarks and systems such as RelBench [28] and rLLM [25] further demonstrate the effectiveness of this paradigm in real-world relational database scenarios. Our proposed method falls into this second category. Unlike prior work, our approach explicitly defines and jointly models both intra-table

and inter-table dependencies within a unified framework, enabling a clear and task-specific paradigm for relational table learning.

7 Conclusion

In this paper, we propose a novel relational table learning method InRTL. Compared to traditional approaches, our method introduces a clear, task-specific objective designed to model relational tables effectively. Specifically, we formalize intra-table and inter-table interactions as explicit modeling objectives, implemented via Transformer-based self-attention and cross-attention modules. To improve scalability, we propose two theoretically grounded simplification mechanisms that substantially reduce computational overhead, making InRTL practical for large-scale datasets. Extensive experiments demonstrate that InRTL consistently outperforms state-of-the-art baselines on diverse relational table tasks.

Limitations and Future Work. Our evaluation mainly focuses on well-curated relational databases with explicit table relationships. Extending relational table learning to open and heterogeneous data lakes remains challenging due to weak and noisy inter-table relationships [26]. Developing robust and scalable methods for such scenarios is an important direction for future research.

References

[1] Sercan Ö Arik and Tomas Pfister. 2021. TabNet: Attentive interpretable tabular learning. In *AAAI conference on Artificial Intelligence*, Vol. 35. AAAI Press, Washington, DC, 6679–6687.

[2] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2022. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 35, 6 (2022), 7499–7519.

[3] Feiyang Chen, Ken Zhong, Aoqian Zhang, Zheng Wang, Li Pan, and Jianhua Li. 2026. TLSQL: Table Learning Structured Query Language. arXiv:2601.14109 [cs.DB] https://arxiv.org/abs/2601.14109

[4] Jintai Chen, Jiahuan Yan, Qiyuan Chen, Danny Z. Chen, Jian Wu, and Jimeng Sun. 2024. Can a Deep Learning Model be a Sure Bet for Tabular Prediction?. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 288–296.

[5] Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Darby Tien-Hao Chang. 2023. Trompt: towards a better deep neural network for tabular data. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 173, 43 pages.

[6] Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. 2025. RELGNN: composite message passing for relational deep learning. In *Proceedings of the 42nd International Conference on Machine Learning* (Vancouver, Canada) *(ICML'25)*. JMLR.org, Article 314, 17 pages.

[7] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking Attention with Performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[8] Marco Cremaschi, Blerina Spahiu, Matteo Palmonari, and Ernesto Jimenez-Ruiz. 2024. Survey on Semantic Interpretation of Tabular Data: Challenges and Directions. arXiv:2411.11891 [cs.AI] https://arxiv.org/abs/2411.11891

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186.

[10] Jerome H Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232.

[11] Yury Gorishniy, Ivan Rubachev, and Artem Babenko. 2022. On embeddings for numerical features in tabular deep learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 24991–25004.

[12] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, Vol. 34. 18932–18943.

[13] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5, 4 (2015), 1–19.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European Conference on Computer Vision*. Springer, 630–645.

[15] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*.

[16] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. arXiv:2012.06678

[17] Hugging Face and sentence-transformers. 2021. sentence-transformers/all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. Model card, accessed 2026-05-19.

[18] Kaggle. 2022. Kaggle Data Science & Machine Learning Survey, 2022. https://www.kaggle.com/code/paultimothymooney/kaggle-survey-2022-all-results/notebook.

[19] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *International Conference on Machine Learning*, Vol. 119. PMLR, 5156–5165.

[20] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, Vol. 30. 3146–3154.

[21] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, Vol. 35. 14582–14595.

[22] Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. 2024. CARTE: Pretraining and Transfer for Tabular Learning. In *International Conference on Machine Learning*, Vol. 235. PMLR, 23843–23866.

[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

[24] Veronica Lachi, Antonio Longa, Beatrice Bevilacqua, Bruno Lepri, Andrea Passerini, and Bruno Ribeiro. 2025. Boosting relational deep learning with pretrained tabular models. *arXiv preprint arXiv:2504.04934* (2025).

[25] Weichen Li, Xiaotong Huang, Jianwu Zheng, Zheng Wang, Chaokun Wang, Li Pan, and Jianhua Li. 2024. rLLM: Relational Table Learning with LLMs. arXiv:2407.20157 [cs.AI] https://arxiv.org/abs/2407.20157

[26] Feiyu Pan, Tianbin Zhang, Aoqian Zhang, Yu Sun, Zheng Wang, Lixing Chen, Li Pan, and Jianhua Li. 2026. LakeMLB: Data Lake Machine Learning Benchmark. *arXiv preprint arXiv:2602.10441* (2026).

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*. 1532–1543.

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

[29] Gowthami Somepalli, Avi Schwarzschild, Micah Goldblum, C. Bayan Bruss, and Tom Goldstein. 2022. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. In *NeurIPS 2022 First Table Representation Workshop*.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. 5998–6008.

[31] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[32] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. arXiv:2012.15828 [cs.CL] https://arxiv.org/abs/2012.15828

[33] Zheng Wang, Hongming Ding, Li Pan, Jianhua Li, Zhiguo Gong, and Philip S. Yu. 2025. From Cluster Assumption to Graph Convolution: Graph-based Semi-Supervised Learning Revisited. *IEEE Transactions on Neural Networks and Learning Systems* 36, 7 (2025), 12952–12963.

[34] Zifeng Wang and Jimeng Sun. 2022. TransTab: Learning Transferable Tabular Transformers Across Tables. In *Advances in Neural Information Processing Systems*, Vol. 35. 2902–2915.

[35] Lukáš Zahradník, Jan Neumann, and Gustav Šír. 2023. A deep learning blueprint for relational databases. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

[36] Xin Zheng, Yi Wang, Yixin Liu, Ming Li, Miao Zhang, Di Jin, Philip S. Yu, and Shirui Pan. 2026. Graph Neural Networks for Graphs With Heterophily: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2026), 1–20. doi:10.1109/tkde.2026.3680353

A Proofs

A.1 Proof of Theorem 1

Proof. We begin by applying the first-order Taylor expansion of the exponential function at $x = 0$. For any real number x, we have:

$$\exp(x) = 1 + x + \Omega(x) \tag{17}$$

$$\Omega(x) = \frac{\exp(\xi)}{2}x^2, \quad \xi \in [0, x] \tag{18}$$

where $\Omega(x)$ denotes the second-order remainder term of the Taylor expansion, and ξ is a point in the interval $[0, x]$, related to the actual approximation error.

As the queries and keys in Transformer have been assumed to be normalized (i.e., $\|\vec{q}_i\|_2 = \|\vec{k}_i\|_2 = 1$), their inner product satisfies $|\vec{q_u}^\top \vec{k_v}| \leq 1$ and is typically small in practice. Applying the first-order Taylor approximation to the exponential of this inner product, we can rewrite Eq. 17 as:

$$\exp(\vec{q_u}^\top \vec{k_v}) \approx 1 + \vec{q_u}^\top \vec{k_v} \tag{19}$$

Using this approximation, the standard Transformer's attention weight (in Eq. 2) becomes:

$$a_{uv} = \frac{\exp(\vec{q_u}^\top \vec{k_v})}{\sum_{i=1}^{n} \exp(\vec{q_u}^\top \vec{k_i})} \approx \frac{1 + \vec{q_u}^\top \vec{k_v}}{\sum_{i=1}^{n} \left(1 + \vec{q_u}^\top \vec{k_i}\right)} \tag{20}$$

Since $|\vec{q_u}^\top \vec{k_v}| \leq 1$, we have $1 + \vec{q_u}^\top \vec{k_v} \geq 0$, ensuring that the approximated attention weights remain non-negative.

Then, we can express the numerator of the approximate attention weight in Eq. 20 in matrix form:

$$A = [a_{uv}]_{u,v=1}^{n} = J_n + QK^\top \tag{21}$$

Meanwhile, the denominator of the approximate attention weight in Eq. 20 corresponds to a diagonal matrix formed by row sums of A:

$$D = \mathrm{Diag}(A\vec{\mathbf{1}}_n) = \mathrm{Diag}(n + Q(K^\top \vec{\mathbf{1}}_n)) \tag{22}$$

Combining Eq. 21 and Eq. 22, the linearized attention matrix is given by $D^{-1}A$. Substituting this into the attention output (in Eq. 5) and omitting layer normalization for simplicity, we obtain:

$$D^{-1}AV = D^{-1}\left[J_n V + Q(K^\top V)\right] \tag{23}$$

This completes the proof of the validity of the linear attention approximation.

Time complexity. Computing $Q(K^\top V)$ requires $O(nd^2)$ operations. Computing $J_n V$ by summing the columns of V and broadcasting the result requires only $O(nd)$. Thus the overall complexity remains $O(nd^2)$ and is substantially lower than the $O(n^2 d)$ cost of standard Transformer attention. In the regime $n \gg d$ the dominant cost scales as $O(n)$. □

A.2 Proof of Corollary 1

Proof. To derive the error bound for the approximation between standard softmax-based attention and our linearized form, we fix a row index u and have

$$x_i = \vec{q_u}^\top \vec{k_i}, \quad |x_i| \leq \epsilon < 1, \quad i = 1, \ldots, n. \tag{24}$$

Let a_i and $\tilde{a}_i$ denote the standard softmax-based attention weight and its linearized approximation. By Eq. 20 in A.1, they can be expressed as:

$$a_i = \frac{\exp(x_i)}{\sum_{i=1}^{n} \exp(x_i)}, \quad \tilde{a}_i = \frac{1 + x_i}{\sum_{i=1}^{n}(1 + x_i)} \tag{25}$$

Let $D_{\exp}$ and D_{lin} denote the denominators for the standard and linearized attention weights:

$$D_{\exp} = \sum_{i=1}^{n} \exp(x_i) \tag{26}$$

$$D_{\mathrm{lin}} = \sum_{i=1}^{n}(1 + x_i) = n + \sum_{i=1}^{n} x_i \tag{27}$$

Since $|x_i| \leq \epsilon$, applying first-order Taylor approximation of the exponential function (i.e., Eq. 17 and Eq. 18 in A.1) to Eq. 26, we obtain:

$$D_{\exp} = \sum_{i=1}^{n}(1 + x_i + \Omega(x_i)) = D_{\mathrm{lin}} + \sum_{i=1}^{n} \Omega(x_i) \tag{28}$$

$$|\Omega(x_i)| \leq \tfrac{1}{2}\exp(\epsilon)\epsilon^2, \ \left|\sum_{i=1}^{n} \Omega(x_i)\right| \leq \tfrac{n}{2}\exp(\epsilon)\epsilon^2 \tag{29}$$

Then, a_i and $\tilde{a}_i$ in Eq. 25 can be rewritten as:

$$a_i = \frac{\exp(x_i)}{D_{\exp}} = \frac{1 + x_i + \Omega(x_i)}{D_{\exp}}, \quad \tilde{a}_i = \frac{1 + x_i}{D_{\mathrm{lin}}} \tag{30}$$

Their difference can be decomposed as:

$$a_i - \tilde{a}_i = \underbrace{\frac{\Omega(x_i)}{D_{\exp}}}_{(I)} + \underbrace{(1 + x_i)\left(\frac{1}{D_{\exp}} - \frac{1}{D_{\mathrm{lin}}}\right)}_{(II)} \tag{31}$$

For term (I), from Eq. 28, it follows that $|\Omega(x_i)| \leq \frac{1}{2}\exp(\epsilon)\epsilon^2$. Substituting this into Eq. 29 yields: $D_{\exp} \geq D_{\mathrm{lin}} - \left|\sum_i \Omega(x_i)\right| \geq n - \frac{n}{2}\exp(\epsilon)\epsilon^2$. Accordingly, the following bound of term (I) holds:

$$|(I)| \leq \frac{\frac{1}{2}\exp(\epsilon)\epsilon^2}{n - \frac{n}{2}\exp(\epsilon)\epsilon^2} = O(\epsilon^2) \tag{32}$$

For term (II), observe that

$$\frac{1}{D_{\exp}} - \frac{1}{D_{\mathrm{lin}}} = \frac{D_{\mathrm{lin}} - D_{\exp}}{D_{\exp} D_{\mathrm{lin}}} = -\frac{\sum_i \Omega(x_i)}{D_{\exp} D_{\mathrm{lin}}} \tag{33}$$

From Eq. 29, we have established that $\left|\sum_i \Omega(x_i)\right| \leq \frac{n}{2}\exp(\epsilon)\epsilon^2$. Furthermore, applying the bound $|x_i| \leq \epsilon$ from Eq. 24 to Eq. 27, we obtain $D_{\mathrm{lin}} = n + \sum_i x_i \geq n(1 - \epsilon)$. In addition, as shown in the analysis of (I), we have $D_{\exp} \geq n - \frac{n}{2}\exp(\epsilon)\epsilon^2$. Substituting the bounds on $\sum_i \Omega(x_i)$, D_{lin} and $D_{\exp}$ into Eq. 33 gives:

$$|(II)| \leq \frac{\frac{n}{2}\exp(\epsilon)\epsilon^2}{n(1 - \epsilon) \cdot (n - \frac{n}{2}\exp(\epsilon)\epsilon^2)} = O(\epsilon^2) \tag{34}$$

Therefore, combining both Eq. 32 and Eq. 34, we conclude:

$$\left|a_i - \tilde{a}_i\right| = O(\epsilon^2) \tag{35}$$

This completes the proof. □

A.3 Proof of Proposition 1

Proof. Consider the bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{U}, \mathcal{E})$, which contains exactly two node types, $\psi(v_i)$ and $\psi(u_i)$, and a single edge type r. We only consider target nodes here.

Based on the edge set $\mathcal{E}$, we define a masking matrix mask $\in \mathbb{R}^{n\times n}$ as:

$$\text{mask}_{ij} = \begin{cases} 0, & \text{if } (u_i, v_j) \in \mathcal{E}, \\ -\infty, & \text{otherwise.} \end{cases} \tag{36}$$

For each target node v_i, the aggregation operator $\text{AGG}_r(\cdot)$ defined in Eq. 1 collects features from all neighboring auxiliary nodes. The aggregated message at layer l, denoted by $msg_{v_i}^{(l)}$, is computed as:

$$\begin{aligned} msg_{v_i}^{(l)} &= \text{AGG}_r\left(\left\{\vec{h}_{u_j}^{(l)} \mid u_j \in \mathcal{N}_{v_i}^r\right\}\right) \\ &= \sum_{u_j \in \mathcal{N}_{v_i}^r} a_{v_i u_j} \vec{h}_{u_j}^{(l)} \end{aligned} \tag{37}$$

where the weight is defined as

$$a_{v_i u_j} = \frac{\exp\left(\vec{h}_{v_i}^\top \vec{h}_{u_j}\right)}{\sum_{u_k \in \mathcal{N}_{v_i}^r} \exp\left(\vec{h}_{v_i}^\top \vec{h}_{u_k}\right)}$$

We further define the update operator $\text{UPD}_{\psi(v_i)}$ in Eq. 1 as the identity mapping. The updated representation of the target node v_i at layer l+1, denoted $h_{v_i}^{(l+1)}$, is given by:

$$\vec{h}_{v_i}^{(l+1)} = \text{UPD}_{\psi(v_i)}(\vec{h}_{v_i}^{(l)}, msg_{v_i}^{(l)}) = msg_{v_i}^{(l)} \tag{38}$$

Therefore, after one round of message passing using Eq. 37 and Eq. 38, the representation of node v_i is updated as:

$$\vec{h}_{v_i}^{(l+1)} = \sum_{u_j \in \mathcal{N}_{v_i}^r} a_{v_i u_j} \vec{h}_{u_j}^{(l)} \tag{39}$$

Let $Q = \left[\vec{h}_{v_1}^{(l)}, \ldots, \vec{h}_{v_n}^{(l)}\right]$, $K = V = \left[\vec{h}_{u_1}^{(l)}, \ldots, \vec{h}_{u_n}^{(l)}\right]$. Then, using the mask defined in Eq. 36, Eq. 39 can be reformulated in matrix form as:

$$\text{Softmax}\left(QK^\top + \text{mask}\right) V \tag{40}$$

which is exactly equivalent to the Transformer cross-attention mechanism used in Eq. 7, omitting layer normalization and temperature scaling (i.e., division by $\sqrt{d}$). □

B Baselines

To conduct a rigorous and comprehensive empirical evaluation, we compare our method against a diverse set of competitive and widely recognized baselines. These baselines are organized into three categories: *traditional decision tree models*, *deep learning-based single table learning methods*, and *relational table learning methods*.

- **LightGBM** [20] is a scalable tree-based gradient boosting framework optimized for efficient tabular data learning.
- **TabPFN** [15] is a tabular foundation model based on transformers that performs Bayesian-style in-context prediction using the Prior-Data Fitted Network paradigm.
- **FT-Transformer** [12] adapts the standard Transformer architecture for tabular data by projecting heterogeneous features into a shared embedding space for self-attention.
- **SAINT** [29] enhances tabular learning by applying joint attention across both rows and columns alongside contrastive self-supervised pre-training.
- **Trompt** [5] disentangles intrinsic column semantics from instance-conditioned feature importance to flexibly model tabular data.
- **ExcelFormer** [4] is a robust single-table model combining semi-permeable attention, data augmentation, and a gated attentive feed-forward module.
- **RDL** [28] enables end-to-end relational learning by encoding a database as a heterogeneous graph and integrating neural table encoders with GNNs.
- **BRIDGE** [25] simplifies relational table learning as a GCN-based baseline that focuses on a single relational table while disregarding inter-table heterogeneity.
- **RelGNN** [6] constructs atomic relational message routes based on primary-foreign key links using a hybrid message-passing and attention mechanism.
- **LightRDL** [24] extends RDL to capture temporal dynamics by constructing snapshotted relational graphs and applying heterogeneous GraphSAGE propagation.

C Implementation details

In this section, we present experimental details, including data preparation, baselines details, model details, and hyper-parameter settings, to facilitate the reproducibility of our results.

C.1 Data Preparation

For the SJTUTables benchmark, we follow the original graph construction procedure proposed in its benchmark. Given the dataset's modest size, we employ full-batch training by feeding the entire graph into the model during each iteration. For RelBench, we adopt the data processing pipeline outlined in the original paper, where relational tables are transformed into heterogeneous temporal graphs. Owing to the large scale of the datasets, we apply a mini-batch training strategy based on temporal neighbor sampling. Specifically, we apply temporal neighbor sampling proposed in the original RelBench paper to generate subtables and subgraphs, compute loss, and update parameters on each sampled subgraph.

Each dataset is split into training, validation, and test sets, strictly following the splits defined in their original papers. We train models on the training set, evaluate performance on the validation set, and report final metrics on the test set. We employ cross-entropy loss for classification tasks and L1 loss for regression tasks. All models are trained on a single Nvidia RTX A6000 GPU with 48GB of memory. All reported results are averaged over 20 independent runs.

C.2 Baseline Details

Our baselines include traditional decision tree models, deep learning-based single-table models, and relational table learning methods. For decision tree and single-table models, which cannot natively handle multi-relational inputs, we adopt a Flattening strategy. This applies a sequence of LEFT JOIN operations on the target table to incorporate information from all auxiliary tables, resulting in a single, feature-rich table used for training and evaluation.

For relational table learning methods, we follow their original design constraints. In particular, BRIDGE supports only unary relationships between two tables; thus, its graph construction includes

Table 6: Unified text embeddings comparison on SJTUTables (Accuracy %, ↑).

Embedding	Method	TML1M	TLF2K	TACM12K
GloVe	InRTL	**39.83**	**49.67**	**46.80**
GloVe	RDL	35.60	28.65	19.50
GloVe	RelGNN	28.43	21.34	20.20
all-MiniLM-L6-v2	InRTL	**40.30**	**50.33**	**48.17**
all-MiniLM-L6-v2	RDL	36.40	31.67	19.80
all-MiniLM-L6-v2	RelGNN	28.60	24.70	20.43

only one auxiliary table connected to the target table. Other relational baselines are allowed to use all available auxiliary tables.

To ensure fairness, we re-implemented all baselines based on the official code or their original papers and performed a comprehensive grid search over key hyper-parameters (see Appendix C.5 for details). For methods with incomplete or non-runnable public implementations (e.g., RelGNN and LightRDL, whose training scripts were not fully provided), we re-implemented the missing training components to enable consistent evaluation. We report the best performance for each baseline under our experimental setup.

C.3 Our Model Details

Our model comprises three key components: a column-aware table encoder, a linear attention module for intra-table interactions, and a HGNN module for inter-table interactions. We describe each component below.

The column-aware table encoder follows the suggestion in RelBench, using specialized pre-encoding strategies for different column types. Categorical features are embedded using a lookup table that maps discrete values into a shared embedding matrix. Column offsets are added to distinguish between different columns, enabling the model to generate dense, column-specific representations. For textual features, we apply pretrained embeddings such as GloVe [27] or transformer-based language models like MiniLM [17, 32] to obtain vector representations. The number of ResNet layers used in the fusion module is configurable.

The linear attention module extends Eq. 11 into a multi-head variant during training. Each attention head projects inputs into a separate subspace, allowing the model to capture diverse interaction patterns in parallel. This design improves both the representational capacity and the robustness of intra-table modeling.

The HGNN module models inter-table relationships based on PK–FK constraints. These are represented as undirected edges in a heterogeneous graph. The aggregation function in Eq. 13 is implemented as an element-wise summation over neighboring node features. The update function applies a linear transformation to the aggregated result, combined with the node's own representation.

C.4 Comparison with Unified Text Embeddings

To remove the potential confounding effect of text initialization, we reran SJTUTables comparisons using unified text embeddings for the main methods InRTL, RDL, and RelGNN.

Specifically, as shown in Table 6, we report results under two shared embedding settings: (1) GloVe, and (2) lightweight BERT-family encoder all-MiniLM-L6-v2. As shown in Table 6, under both unified settings, InRTL consistently remains the best on all three SJTUTables datasets, indicating that the performance gain is not solely due to text embedding choice.

C.5 Hyper-parameters Settings

We tune all model hyper-parameters based on validation performance. Unless otherwise stated, the hyper-parameters are selected from the following candidate values using exhaustive grid search.

- batch size: {128, 256, 512}
- learning rate: {1e-3, 5e-3, 1e-2}
- hidden dimension: {32, 64, 128, 256, 512}
- weight decay: {1e-5, 1e-4, 1e-3, 1e-2}

The search spaces for other model-specific hyper-parameters are listed below:

- LightGBM: max depth ∈ {3, 4, 5, 6, 7, 8, 9, 10}, number of leaves ∈ {2, 32, 64, 128, 256, 512}.
- FT-Transformer: number of heads ∈ {4, 8, 16}, number of layers ∈ {2, 3, 4, 6}.
- TabNet: number of steps ∈ {3, 4, 5, 6}, $\gamma \in \{1.0, 1.5, 2.0\}$.
- Trompt: number of Trompt Cells ∈ {1, 3, 6, 12}.
- ExcelFormer: number of heads ∈ {4, 8, 16, 32}.
- BRIDGE: layers of TabTransformer ∈ {2, 3, 4}, layers of GCN ∈: {1, 2, 3, 4}.
- RDL: layers of heterogeneous GraphSAGE ∈: {1, 2, 3, 4}.
- InRTL: number of layers in linear attention ∈ {1, 2, 3, 4, 5}, number of heads in linear attention ∈ {2, 4, 8, 16}, number of layers in HGNN ∈ {1, 2, 3, 4, 5}, initial value of learnable coefficient $\beta \in \{0.5, 0.8\}$.

D Pseudocode of InRTL

For clarity, we provide the pseudocode of the overall InRTL training procedure in Algorithm 1.

Algorithm 1 InRTL

Input: Relational tables $\mathcal{T}$, PK-FK graph $\mathcal{G}$, number of layers L, ground-truth labels y, sampling function Sampler()
Output: Trained InRTL model

Initialize the parameters of InRTL
for each training epoch **do**
  for all mini-batches $(\mathcal{T}_\mathcal{B}, \mathcal{G}_\mathcal{B}) \sim \text{Sampler}(\mathcal{T}, \mathcal{G})$ **do**
    Obtain E_{tgt} and E_{aux} for $\mathcal{T}_\mathcal{B}$ as described in Sec. 3
    $H^{(0)}_{\text{intra}} \leftarrow E_{\text{tgt}}, \quad H^{(0)}_{\text{inter}} \leftarrow E_{\text{tgt}}$
    for $l = 0$ to $L - 1$ **do**
      $Q \leftarrow H^{(l)}_{\text{intra}} W_Q,\ K \leftarrow H^{(l)}_{\text{intra}} W_K,\ V \leftarrow H^{(l)}_{\text{intra}} W_V$
      Update $H^{(l+1)}_{\text{intra}}$ according to Eq. 11
      Update $H^{(l+1)}_{\text{inter}}$ with $\mathcal{G}_\mathcal{B}$ according to Eq. 13
    end for
    Compute prediction $\hat{y}_\mathcal{B}$ using Eq. 9
    Obtain mini-batch labels $y_\mathcal{B}$ from y
    Compute loss on $\hat{y}_\mathcal{B}$ and $y_\mathcal{B}$, and update parameters
  end for
end for
return Trained InRTL model