MetaRTL: Meta-path Attention Enhanced Relational Table Learning

Ken Zhong[1], Weichen Li[1], Zheng Wang[1,2*]
[1]School of Computer Science, Shanghai Jiao Tong University, Shanghai, China
[2]Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai, China
{zhongken, weichenli, wzheng}@sjtu.edu.cn

***Abstract*—Relational table learning has gained increasing attention with the widespread use of relational databases. Existing methods typically rely on deep GNN or HGNN stacks, leading to high computational costs and limited performance on large real-world databases. We propose MetaRTL, a two-stage framework for scalable and expressive relational table learning. In the first stage, MetaRTL obtains initial table embeddings via lightweight pre-training. In the second stage, it performs non-parametric message passing to derive meta-path features, which are then aggregated by an attention module, MetaAttn. By shifting computation from deep message passing to efficient meta-path aggregation, MetaRTL captures rich relational semantics while maintaining high efficiency. Experiments on 10 real-world datasets across 24 tasks demonstrate the effectiveness of the proposed method.**

***Index Terms*—Relational table learning, tabular neural networks, graph neural networks, data mining.**

I. Introduction

Relational databases serve as the backbone of data management in modern applications, supporting critical domains such as finance, healthcare, and e-commerce [1]. Consequently, machine learning on relational table data, referred to as relational table learning (RTL) [2], has attracted research attention [3]. In the era of deep learning, the goal of this task is to model relational table data in an end-to-end fashion, eliminating the need for labor-intensive feature engineering and achieving strong performance in various applications such as user profiling, behavior prediction, and click-through rate estimation [4] [5].

A common practice among existing deep RTL methods is first to construct heterogeneous graphs based on primary–foreign key (pkey–fkey) relationships [6], where each node corresponds to a single row from a table and inter-table dependencies are captured via diverse edge types. Then, to process such structured relational data, these methods [6] [2] often adopt a direct yet practical approach that builds on advances in tabular neural networks (TNNs) [7] and graph neural networks (GNNs) [8] or heterogeneous GNNs (HGNNs) [9]. Typically, TNNs straightforwardly encode per-table semantics by generating initial node embeddings for each row, which are then simply refined through stacked GNN/HGNN layers to facilitate cross-table message passing.

While this paradigm has achieved some advanced results, it still faces at least two notable challenges in practical applications. On one hand, the resulting heterogeneous graphs often exhibit high relational complexity and dense connectivity. For example, the publicly available Stack Exchange dataset [10], which originates from a real-world question answering platform, contains eight relational tables per domain. These include entities such as users, posts, and comments, with each table containing on average tens of millions of records. On the other hand, existing methods rely on deep GNN/HGNN stacks, which may incur substantial computational costs and suffer from common deep GNN issues [11] [12].

To address these challenges, we propose **MetaRTL** (Meta-path based Relational Table Learning), a novel framework designed to enhance the modeling capacity of RTL on large-scale relational tables. The core idea of MetaRTL is twofold: (1) it leverages meta-paths to capture rich semantic relationships within complex relational structures, and (2) it employs an attention mechanism to enable feature fusion, by emphasizing task-relevant meta-path features. For ease of implementation and efficiency, MetaRTL operates in two stages:

1) **Pre-training Stage**: A table encoder, consisting of a set of TNNs, is trained with substantially fewer iterations than previous RTL frameworks. Instead of relying on deep GNN/HGNN stacks, we adopt a simple two-layer HGNN and terminate training early (5 epochs in our experiments), producing stable node embeddings sufficient for subsequent meta-path aggregation.
2) **Aggregation Stage**: These embeddings are then used to compute semantic features along meta-paths on the heterogeneous graph via non-parametric propagation. To effectively integrate these features while mitigating sampling-induced semantic bias, we propose **MetaAttn**, a lightweight attention module that combines meta-path self-attention with global node cross-attention.

Intuitively, the initial embeddings from the Pre-training Stage provide a strong foundation, while the attention mechanism in the Aggregation Stage enables expressive and interpretable feature fusion. From a computational perspective, by shifting the modeling focus to the aggregation stage and leveraging precomputed meta-path features, MetaRTL avoids deep GNN stacking pipelines while maintaining efficiency. As a result, it scales linearly with the number of table rows and relations, effectively handling complex relational table data. Extensive experiments on ten real-world datasets covering 24 tasks demonstrate the effectiveness of the proposed method.

Our main contributions are summarized as follows:

- We propose MetaRTL, a novel two-stage RTL framework based on meta-paths, capturing complex relational table information.
- We design a lightweight and efficient aggregation module, MetaAttn, which integrates meta-path features with

*Corresponding author.

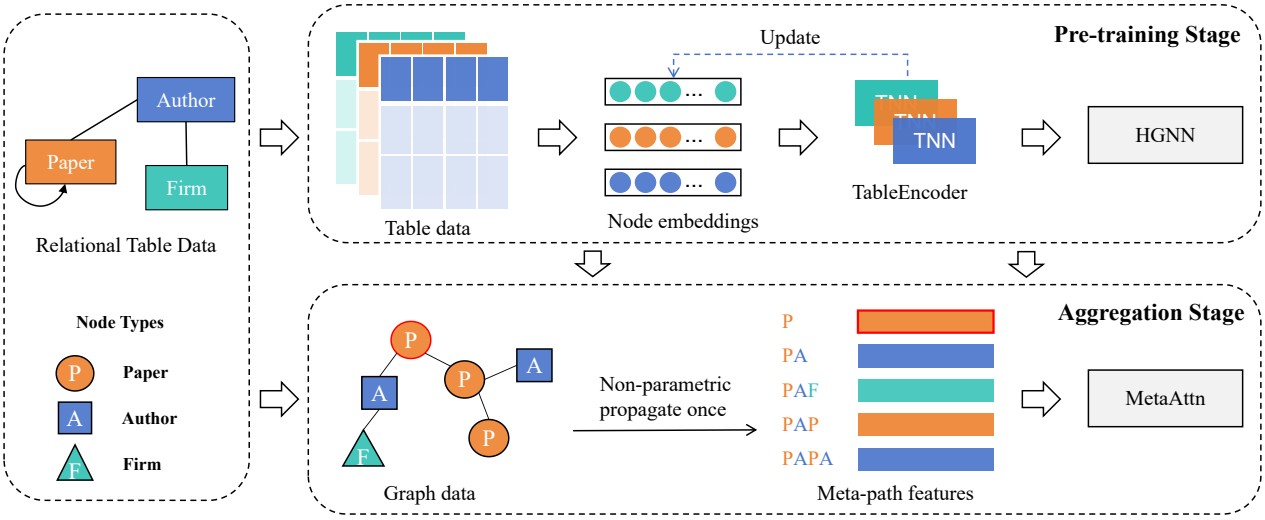

Fig. 1. Overview of the **MetaRTL** framework. Given a relational table data, we first transform them into a heterogeneous graph following Definition 1 as a preprocessing step. The framework then proceeds in two stages. In the Pre-training stage (top), a unified table encoder composed of multiple TNNs is trained jointly with a shallow HGNN for very few epochs on neighbor sampled subgraphs, producing stable and semantically meaningful node embeddings. In the Aggregation stage (bottom), these embeddings are used to compute meta-path features through non-parametric propagation on the graph. A lightweight attention module, MetaAttn, then aggregates the multiple meta-path features and global node semantics to obtain task-specific target node representations.

global node semantics.

- We conduct extensive experiments on various real-world datasets, demonstrating the superiority of MetaRTL.

II. Related Work

A. Heterogeneous Graph Neural Networks

Heterogeneous Graph Neural Networks (HGNNs), a branch of Graph Neural Networks (GNNs) research [9], are widely used for modeling multi-typed graph data. Existing methods can be broadly classified as parametric or non-parametric based on whether they rely on trainable parameters. Parametric HGNNs (like HAN [13]) employ trainable parameters to model semantic heterogeneity, by stacking architectures to aggregate multi-hop neighborhood information and compute attention weights during training. However, these models are expensive to train on large graphs and often suffer from over-smoothing [11] [12] [14]. Non-parametric HGNNs (like SeHGNN [15]) reduce the reliance on type-specific parameters by precomputing meta-path features before training. This improves efficiency, but such methods usually depend on fixed node features and are not compatible with sampling-based mini-batch training. MetaRTL builds upon this research direction and bridges the gap in applying such methods to relational table data, where node features are automatically learned from raw table entries and graph structures are induced by schema-level relations.

B. Relational Table Learning

Relational table learning (RTL) focuses on modeling structured data from real-world multi-table relational databases linked by primary–foreign key (pkey–fkey) relationships [3]. Early approaches explicitly modeled relational dependencies across tables using logic [16] or probabilistic graphical models [17]. While expressive, these methods often require manual feature engineering and do not scale well to large databases. With the rise of deep learning, recent studies leverage techniques such as tabular neural networks (TNNs) and graph neural networks (GNNs) [4], [18] For example, graph-based representations across tables have been explored [19]. RDL [6] introduces relational entity graphs and a universal TNN–GNN framework that generalizes to arbitrary multi-relational tables. BRIDGE [2] simplifies this approach by reducing the number of auxiliary tables and ignoring heterogeneity. RelGNN [20] further decomposes message passing into atomic path modeling based on relational insights. LightRDL [21] focuses on efficiency and distills relational patterns with small GNNs while relying on engineered features for temporal signals, enabling faster inference. MetaRTL differs from these approaches by employing a decoupled two-stage architecture: a shallow pre-training phase followed by meta-path feature aggregation with a lightweight attention module, yielding a more expressive solution for relational table learning.

III. Preliminary

Definition 1 ***Relational Table Data.*** Relational table data comprises multiple structured tables linked by primary–foreign key (pkey–fkey) relations. It can be modeled as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}_v, \mathcal{T}_e)$, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, and $\mathcal{T}_v$, $\mathcal{T}_e$ are their type sets [6]. Each table corresponds to a specific node type, and each row within the table represents one node of that type. Likewise, each primary–foreign key (pkey–fkey) relationship defines an edge type: for every such relationship, a typed edge is created between each pair of rows that are linked through the

pkey–fkey match. This graph-based formulation preserves the relational structure for downstream modeling.

Definition 2 ***Meta-paths.*** A meta-path [22] is a schema-level path defined as a sequence of edge types connecting a source node type to a target node type through intermediate types. Let Φ denote the set of all meta-paths under consideration. Each meta-path $\mathcal{P} \in \Phi$ is defined as:

$$\mathcal{P} : \tau_s \to \tau_1 \to \tau_2 \to \cdots \to \tau_t \tag{1}$$

where τ_s is the source node type and τ_t is the target node type. Although the nodes connected via a meta-path may not be directly adjacent in the graph, the path implies a semantically meaningful relation between them [15]. Based on Definition 1, in the context of relational table data, the target node type corresponds to the target or task table, while the source node types refer to other auxiliary tables.

Definition 3 ***Heterogeneous Graph Neural Networks.*** A heterogeneous graph neural network (HGNN) is a class of GNNs designed for representation learning over heterogeneous graphs. Let $\mathbf{h}_v \in \mathbb{R}^d$ denote the representation of node v at a given layer.[1] At each layer, an HGNN updates node representations following the message passing paradigm [23], where messages are aggregated from type-specific neighbor sets associated with different edge types $r \in \mathcal{T}_e$. Formally, an l-layer HGNN model can be written as:

$$\begin{aligned} \mathrm{MP}(\mathbf{h}_v, \mathcal{G}) &= \mathrm{UPD}(\{\mathrm{AGGR}_r(\mathbf{h}_v, \mathcal{N}_v^r) \,|\, r \in \mathcal{T}_e\}) \\ \mathrm{HGNN}(\mathbf{h}_v, \mathcal{G}) &= \mathrm{MP}_l \circ \cdots \circ \mathrm{MP}_1(\mathbf{h}_v, \mathcal{G}) \end{aligned} \tag{2}$$

where $\circ$ denotes function composition, $\mathrm{MP}_i(\cdot)$ is the i-th message passing layer, $\mathcal{N}_v^r$ is the neighbors of node v under edge type r, $\mathrm{AGGR}_r(\cdot)$ is an edge type specific aggregation operator (e.g. mean pooling), and $\mathrm{UPD}(\cdot)$ is an update function to integrate aggregated messages across all edge types.

Definition 4 ***Attention Mechanism.*** Attention enables dynamic weighting of input features based on pairwise relevance and serves as a core component of the Transformer architecture [24]. Given query matrix Q, key matrix K, and value matrix V, the attention output is computed as:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V. \tag{3}$$

This formulation supports both self-attention (when $Q = K = V$) and cross-attention (when Q and K, V differ), and naturally accommodates varying sequence lengths. For clarity, we omit batch dimensions and multi-head extensions, which are straightforward to incorporate.

IV. Methodology

We propose **MetaRTL**, a scalable and expressive two-stage framework for relational table learning. The proposed framework operates in two successive stages: (1) Pre-training Stage, in which a unified table encoder is trained with a two-layer HGNN using only a few training iterations to produce semantically meaningful and robust node embeddings; and (2) Aggregation Stage, in which precomputed meta-path features are efficiently fused with global node representations through a compact dual-branch attention module, **MetaAttn**, to derive the final node representations for downstream tasks.

[1]For notational convenience, we use d to denote feature dimension through this paper, although the actual dimensions may vary across modules.

A. Pre-training Stage

As illustrated on the left side of Figure 1, MetaRTL takes as input a collection of structured relational tables, which are transformed into a heterogeneous graph. Given the typically large scale of such relational table data, we adopt a uniform neighbor sampling strategy to construct a series of mini-batch subgraphs $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i, \mathcal{T}_v, \mathcal{T}_e)$ sampled from the entire graph $\mathcal{G}$ for scalable training, where corresponding table records set $\mathcal{V}_i$ and relational pkey–fkey links set $\mathcal{E}_i$ are also retrieved to provide initial input features.

To obtain node representations that are both semantically informative and stable, we introduce a unified table encoder. As illustrated in the Pre-training Stage of Figure 1, the encoder comprises a collection of table-specific TNN modules. Each $\mathrm{TNN}_\tau(\cdot)$ encodes the corresponding set of table records $\mathcal{V}_i^\tau$ in the sampled subgraph, with the table or node type $\tau \in \mathcal{T}_v$. The initial node embeddings $X_\tau \in \mathbb{R}^{|\mathcal{B}_\tau| \times d}$ are computed as:

$$X_\tau = \mathrm{TNN}_\tau(\mathcal{V}_i^\tau), \quad \forall \tau \in \mathcal{T}_v \tag{4}$$

where, $\mathcal{B}_\tau$ denotes the node index set of type τ contained in $\mathcal{V}_i$, $|\mathcal{B}_\tau|$ is the number of such nodes.

The table encoder is optimized by integrating a shallow HGNN module following the message passing paradigm defined in Definition 3. The HGNN takes as input the initial node embeddings $\{X_\tau\}_{\tau \in \mathcal{T}_v}$ with the sampled sub graph $\mathcal{G}_i$, producing task-specific node representations:

$$\mathrm{HGNN}\big(\{X_\tau\}_{\tau \in \mathcal{T}_v}, \mathcal{G}_i\big). \tag{5}$$

These representations are passed through a prediction head, and the table encoder parameters are iteratively optimized by minimizing the task-specific loss.

Crucially, in contrast to previous RTL methods that rely on deep HGNN stacks and extensive training for full convergence, our approach employs a two-layer HGNN architecture with a quite limited number of training iterations during this stage. The objective is not to achieve complete convergence, but rather to update and refine the initial node embeddings $\{X_\tau\}_{\tau \in \mathcal{T}_v}$, producing representations that are both stable and semantically rich for downstream tasks. This lightweight pre-training paradigm balances computational efficiency with adequate representational capacity.

B. Aggregation Stage

This section outlines the Aggregation Stage, as illustrated in Figure 1, which consists of two key components: the computation of meta-path features and the MetaAttn module. First, meta-path features are derived through a non-parametric propagation. Then, the one-layer lightweight MetaAttn module integrates these meta-path features with global node representations to generate the final embeddings for target nodes.

*1) **Meta-path Feature Computation**:* To capture high-order semantic dependencies across different auxiliary tables to target table within the heterogeneous graph constructed from relational data, we perform non-parametric propagation on the initial node embeddings to derive meta-path features. Formally, for an l-hop meta-path $\mathcal{P} \in \Phi$ from source node type τ_s to target node type τ_t, the corresponding meta-path adjacency matrix $A^{\mathcal{P}}$ is obtained by multiplying the associated relation matrices in reverse order:

$$A^{\mathcal{P}} = A_{\tau_t,\tau_{l-1}} \cdots A_{\tau_2,\tau_1} A_{\tau_1,\tau_s} \tag{6}$$

where $A_{\tau_l,\tau_{l-1}}$ denotes the degree-normalized adjacency matrix from node type τ_{l-1} to τ_l in the full heterogeneous graph. This computation is inherently iterative and compositional: meta-path adjacency matrices can be efficiently constructed by sequentially multiplying the intermediate products of shorter meta-path segments. By recursively applying this procedure, we obtain the complete set of meta-path adjacency matrices $\{A^{\mathcal{P}}\}_{\mathcal{P}\in\Phi}$ for all meta-paths up to a predefined maximum length. This step is independent of subgraph sampling and model training, and can be efficiently performed during data preprocessing over the entire heterogeneous graph.

For each mini-batch subgraph generated by neighbor sampling, the precomputed meta-path adjacency matrices $A^{\mathcal{P}}$ are sliced according to the indices of the source and target node types present in the current subgraph, denoted by $\mathcal{B}_{\tau_s}$ and $\mathcal{B}_{\tau_t}$, respectively. The sliced adjacency matrices are then multiplied by the initial embeddings X_{τ_s} of the meta-path source node type $\tau_s \in \mathcal{T}_v$ to yield meta-path feature $H^{\mathcal{P}} \in \mathbb{R}^{|\mathcal{B}_{\tau_t}|\times d}$:

$$H^{\mathcal{P}} = A^{\mathcal{P}}[\mathcal{B}_{\tau_t}, \mathcal{B}_{\tau_s}] X_{\tau_s} \tag{7}$$

where $[,]$ denotes the slicing operation. The resulting meta-path feature set $\{H^{\mathcal{P}}\}_{\mathcal{P}\in\Phi}$ effectively encodes the semantic information carried from auxiliary tables τ_s to the target table τ_t along all meta-paths within the sampled subgraph $\mathcal{G}_i$.

*2) **MetaAttn Module**:* To effectively aggregate the meta-path features of target table corresponding nodes while incorporating global semantic information, we design a lightweight, single-layer attention-based module named MetaAttn. As illustrated in Figure 2, it consists of a meta-path self-attention module that captures the relative semantic importance among multiple meta-path features, a global node cross-attention module that incorporates full-graph semantic context, and a feature aggregation step that combines the outputs to form the final target node representations.

Meta-path Self-attention (MPSA).

Motivated by the observation that auxiliary tables contribute unequally to relational table learning, we introduce a meta-path self-attention mechanism along the meta-path axis to selectively aggregate semantic features from different auxiliary tables into the target table. This mechanism highlights task relevant information while suppressing noise from less informative tables, as illustrated in the left part of Figure 2.

Given the precomputed meta-path features $\{H^{\mathcal{P}}\}_{\mathcal{P}\in\Phi}$ from Equation 7, we utilize the mapping that associates each meta-path $\mathcal{P}$ with its corresponding source node type τ_s. Based on this association, we group the meta-path features by their source types and apply type-specific linear transformations to each group. The transformed representations are then concatenated along a new axis to construct a aggregated meta-path feature tensor $H_{\text{meta}} \in \mathbb{R}^{|\mathcal{B}_{\tau_t}|\times|\Phi|\times d}$:

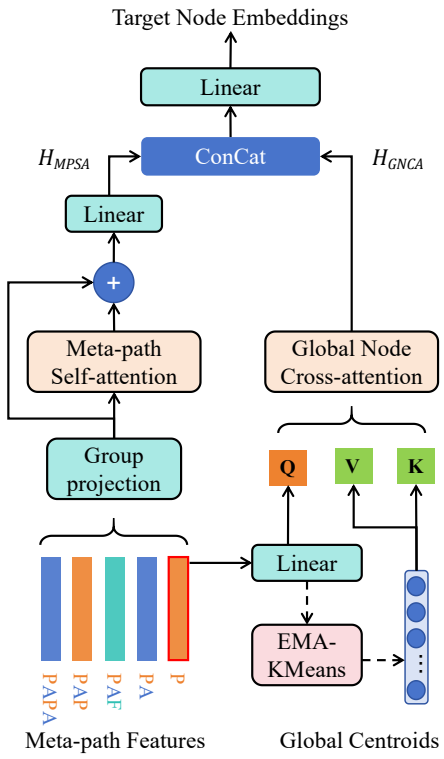

Fig. 2. Architecture of the MetaAttn Module.

$$H_{\text{meta}} = \Big\|_{\tau_s\in\mathcal{T}_v} \text{Linear}_{\tau_s}\left(\{H^{\mathcal{P}} \mid \mathcal{P} \mapsto \tau_s\}\right) \tag{8}$$

where $\mathcal{P} \mapsto \tau_s$ denotes the mapping that indicates each meta-path $\mathcal{P}$ to its source node type τ_s, $\text{Linear}_{\tau_s}(\cdot)$ denotes the linear layer associated with type τ_s, and $\|$ represents the concatenation operation. Then to capture the relative importance of different auxiliary tables, a self-attention mechanism formulated in Definition 4 is applied over the meta-path dimension on the meta-path feature tensor computed above, followed by a residual connection and a linear projection. The resulting meta-path semantic aggregation representation $H_{\text{MPSA}} \in \mathbb{R}^{|\mathcal{B}_{\tau_t}|\times d}$ is computed as:

$$H_{\text{MPSA}} = \text{Linear}\Big(\text{SelfAttn}(H_{\text{meta}}) + H_{\text{meta}}\Big) \tag{9}$$

where, the attention output H_{MPSA} selectively encodes the aggregated semantic information from all auxiliary tables for target table, i.e. each target node.

Global Node Cross-attention (GNCA). To alleviate potential information loss introduced by neighbor sampling, we adopt a global node cross-attention mechanism inspired by the attention module in previous work [25], [26], as shown on the right side of Figure 2. This mechanism enables each target node in a sampled subgraph to attend to a set of shared global centroids $\mathcal{C}$, which compactly capture the overarching semantic

TABLE I
STATISTICAL ANALYSIS OF TWO BENCHMARK DATASETS

Dataset	#Tables	#Rows	#Columns	Max hop
TACM12k	4	97,774	12	2
TLF2K	3	101,773	15	1
TML1M	3	1,010,132	20	2
rel-f1	9	97,606	77	3
rel-trial	15	5,852,157	140	3
rel-avito	8	20,679,117	43	2
rel-amazon	3	24,291,489	15	2
rel-hm	3	33,265,846	37	2
rel-stack	7	38,109,828	51	3
rel-event	5	41,328,337	128	2

TABLE II
PERFORMANCE ON SJTUTABLES (ACCURACY% ↑)

Group	Method	TACM12k	TLF2K	TML1M
Single-table	LightGBM	35.70	35.52	26.66
	FTTransformer	14.88	15.80	28.62
	Trompt	12.42	13.41	27.85
	ExcelFormer	17.13	20.61	31.90
Multi-table	RDL	19.23	19.58	26.93
	RelGNN	18.19	20.31	27.12
	BRIDGE	25.60	42.20	36.20
	LightRDL	24.94	39.17	35.77
	MetaRTL	**46.54**	**43.17**	**37.09**

structure of the entire heterogeneous graph. The centroids are updated by clustering the current target-node features during training, thereby providing a holistic semantic context that complements the local neighborhood information.

For the 0-hop meta-path, the feature $H^{\mathcal{P}}_{\tau_t} = X_{\tau_t}$ reduces to the original node features of the target table and is directly used as the input. The resulting global semantic representation, denoted as $H_{\text{GNCA}} \in \mathbb{R}^{|\mathcal{B}_{\tau_t}| \times d}$, is computed as:

$$\begin{aligned} H_{\text{GNCA}} &= \text{CrossAttn}\left(\text{Linear}(H^{\mathcal{P}}_{\tau_t}), \mathcal{C}\right) \\ \mathcal{C} &\leftarrow \text{KMeans}\left(H^{\mathcal{P}}_{\tau_t}, \mathcal{C}\right) \end{aligned} \tag{10}$$

where $\text{KMeans}(\cdot)$ denotes the centroid-update rule following the exponential moving average strategy, and $\text{CrossAttn}(\cdot)$ denotes the cross-attention operation in which the linearly transformed node features serve as the query matrix, while the global centroids $\mathcal{C}$ (stacked along the first dimension) are used as both key and value matrices.

Feature Aggregation. Finally, the meta-path self-attention semantic feature H_{MPSA} obtained from Equation 9 and the global node cross-attention semantic feature H_{GNCA} obtained from Equation 10 are concatenated and aggregated through a linear layer to produce the target node embeddings output, formulated as $\text{Linear}([H_{\text{MPSA}} \parallel H_{\text{GNCA}}])$.

Depending on the downstream task, such as regression or classification, different output layers are applied to generate the final predictions. The model is trained end-to-end by minimizing the corresponding loss via backpropagation.

V. EXPERIMENTS

A. Experimental Setup

*1) **Datasets**:* We evaluate MetaRTL on two public relational table learning benchmarks: SJTUTables [2] and RelBench [6]. SJTUTables provides three small to medium scale multi-class classification datasets: `TACM12k`, `TLF2K`, and `TML1M`. RelBench consists of 7 large-scale datasets covering 12 binary classification and 9 regression tasks across real-world domains such as e-commerce, social networks, and healthcare. During training, we apply temporal uniform neighbor sampling, and all data preprocessing, sampling strategies, and evaluation metrics strictly adhere to the official benchmark settings. The detailed statistics of the datasets in both benchmarks are summarized in Table I.

*2) **Baselines**:* We compare MetaRTL with a diverse set of representative baselines, which can be broadly categorized into single-table and multi-table methods. **Single-table methods** operate only on the target table, including LightGBM [27], a gradient boosting baseline, FTTransformer [28], which applies row-wise Transformers over tokenized features, Trompt [29], a prompt-based tabular model, and ExcelFormer [30], which enhances column selection and feature interactions via attention. **Multi-table methods** exploit auxiliary tables via pkey-fkey relations, including RDL [6], combining ResNet-based TNN with heterogeneous GraphSAGE, BRIDGE [2], using a single table with standard GCN, RelGNN [20], with composite relational message passing, and LightRDL [21], which incorporates temporal relational graphs and heterogeneous GraphSAGE with LightGBM derived embeddings.

For single-table methods, following the original RelBench setup, since the task tables contain only indices and labels without feature columns, we perform a left join operation to merge the feature tables with the target tables. Since the official implementations of RelGNN and LightRDL are incomplete, we reproduce them based on the released code and publications. BRIDGE cannot generalize to relational data involving more than two entity tables, and thus, in our experiments on RelBench, we use only the auxiliary table that is most relevant to the target table. In addition, LightRDL requires timestamped datasets for proper data splitting. As some datasets in SJTUTables lack temporal features, we substitute them with randomly generated timestamps. The main experimental pipeline is implemented based on the open-source relational table learning library relationLLM (rLLM)[2].

B. Performance Evaluation on Benchmarks

*1) **Results on SJTUTables**:* Table II presents the results on the three datasets from the SJTUTables benchmark. We can clearly see that MetaRTL consistently achieves the best performance across all datasets, significantly outperforming the baselines. On `TACM12k`, the absolute accuracy of MetaRTL improves over LightGBM by nearly 11% and surpasses the strongest deep learning baseline BRIDGE by approximately 21%. It also demonstrates superior generalization on `TLF2K` and `TML1M`. This improvement is primarily due to MetaRTL's design: by leveraging the rich features during pretraining, it

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

TABLE III
RESULTS ON RELBENCH CLASSIFICATION TASKS (ROC-AUC ↑)

Dataset	Task	Single-table				Multi-table				
		LightGBM	FTTransformer	Trompt	ExcelFormer	RDL	RelGNN	BRIDGE	LightRDL	MetaRTL
rel-f1	driver-top3	0.7392	0.7362	0.7317	0.7522	0.7723	0.8086	0.7713	0.7916	**0.8415**
	driver-dnf	0.6856	0.6831	0.6752	0.6720	0.7125	0.6706	0.7016	0.6863	**0.7418**
rel-trial	study-outcome	0.7009	0.6239	0.6573	0.6731	0.6902	0.6916	0.6898	0.6815	**0.7013**
rel-avito	user-clicks	0.5360	0.5132	0.5423	0.5532	0.6576	0.6613	0.5501	0.6411	**0.6614**
	user-visits	0.5305	0.5210	0.5611	0.5501	**0.6599**	0.6547	0.5616	0.6239	0.6587
rel-amazon	user-churn	0.5222	0.5230	0.5310	0.5209	0.7042	0.6544	0.5351	0.6917	**0.7101**
	item-churn	0.6254	0.6415	0.5982	0.6250	0.8281	0.8264	0.6993	0.8120	**0.8290**
rel-hm	user-churn	0.5521	0.5013	0.5419	0.5395	**0.7011**	0.6810	0.5933	0.6713	0.6795
rel-stack	user-engagement	0.6339	0.6002	0.6573	0.6920	0.9047	0.8995	0.7974	0.8902	**0.9120**
	user-badge	0.6343	0.6036	0.6282	0.6713	0.8886	0.8898	0.8790	0.8671	**0.8898**
rel-event	user-repeat	0.6804	0.6910	0.6717	0.6973	0.7560	0.7307	0.7530	0.7132	**0.7604**
	user-ignore	0.7993	0.7569	0.7731	0.7792	0.7514	0.7911	0.7616	0.7630	**0.8703**
Average ranking		6.25	8.00	7.33	6.83	2.92	3.33	4.83	4.17	**1.33**

TABLE IV
RESULTS ON RELBENCH REGRESSION TASKS (MAE ↓)

Dataset	Task	Single-table				Multi-table				
		LightGBM	FTTransformer	Trompt	ExcelFormer	RDL	RelGNN	BRIDGE	LightRDL	MetaRTL
rel-f1	driver-position	4.170	4.210	4.191	4.112	4.172	4.250	4.179	4.170	**3.987**
rel-trial	study-adv1erse	44.011	44.490	45.397	45.552	44.473	44.461	44.773	43.910	**43.102**
	site-success	0.425	0.431	0.440	0.429	0.400	0.355	0.431	0.411	**0.350**
rel-avito	ad-ctr	0.041	0.049	0.045	0.050	0.041	**0.039**	0.044	0.040	0.040
rel-amazon	user-ltv	16.783	16.930	16.606	17.001	14.313	16.783	15.374	14.310	**14.102**
	item-ltv	60.569	60.319	61.424	59.983	50.053	48.826	60.441	48.112	**46.176**
rel-hm	item-sales	0.076	0.092	0.073	0.073	0.056	0.054	0.071	0.044	**0.039**
rel-stack	post-votes	0.068	0.074	0.080	0.070	0.065	0.065	0.066	0.064	**0.060**
rel-event	user-attendance	0.264	0.270	0.281	0.264	0.258	0.248	0.259	0.260	**0.241**
Average ranking		5.72	7.72	7.72	6.78	3.89	3.78	5.50	2.61	**1.67**

obtains stable initial node semantic embeddings, and its meta-path based lightweight aggregation prevents over-smoothing of neighbor information, thereby preserving feature signals and enhancing predictive performance.

We further observe that RDL and RelGNN underperform on SJTUTables. This can be primarily attributed to a mismatch between their complex model architectures and the simpler patterns in the benchmark. SJTUTables contains a limited number of tables with relatively shallow and sparse relations, under which the deep HGNN backbones in RDL and RelGNN may introduce excessive modeling capacity, resulting in overfitting. In contrast, MetaRTL better leverages these compact but informative representations through non-parametric propagation and meta-path-aware aggregation.

*2) **Results on RelBench**:* Tables III and IV present the detailed results of classification and regression tasks, respectively, across 21 tasks from the RelBench benchmark. The tables report ROC-AUC for classification tasks, MAE for regression tasks, and the average ranking of each method, providing a comprehensive comparison of baseline approaches and our proposed MetaRTL. MetaRTL consistently achieves the lowest average ranks on both classification and regression tasks, and attains the best performance on 18 out of 21 tasks. For the remaining tasks, MetaRTL still produces results that are close to the top-performing method. Overall, these results highlight not only the effectiveness of MetaRTL in individual tasks but also its strong generalization capability across diverse relational table datasets, demonstrating its robustness in modeling complex real-world relational structures.

On the RelBench benchmark, which better reflects real-world scenarios, single-table methods show clear differentiation. When individual tables contain sufficient information (e.g., `rel-event`), they can achieve performance comparable to some multi-table methods. However, on datasets with less informative tables and richer textual features (e.g., `rel-stack` and `rel-amazon`), their performance drops significantly. This underscores the importance of leveraging pkey–fkey based multi-table relationships. In contrast, MetaRTL demonstrates strong performance by effectively extracting informative signals from auxiliary tables while suppressing noise. It achieves notable gains on datasets with complex multi-table structures (e.g., `rel-f1`), while improve-

TABLE V
RESULTS OF ABLATION STUDY

Variant	driver-top3↑	driver-dnf↑	driver-position↓
w/o TNN Pretrain	0.7311	0.6761	4.319
w/o Meta-path Features	0.6294	0.6013	4.870
w/o Global Attention	0.8389	0.7216	4.110
MetaRTL	**0.8415**	**0.7418**	**3.987**

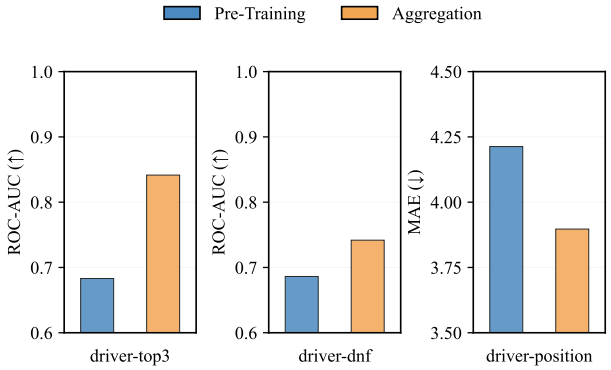

Fig. 3. Prediction Performance: Pre-training Stage vs. Aggregation Stage.

ments are smaller on simpler datasets (e.g., `rel-hm`), where it may even slightly under-perform some baselines. Overall, these results highlight the effectiveness of meta-path-based aggregation for modeling complex inter-table relationships.

C. Ablation of Model Components

1) ***Impact of Model Components****:* To quantify the contribution of each module in MetaRTL, we perform ablation studies on the `rel-f1` dataset. Similar trends are observed on other datasets. Three variants are considered to evaluate the core components: (i) without TNN pretraining (w/o TNN Pretrain), (ii) without meta-path-based features (w/o Meta-path Features), and (iii) without the global node cross-attention mechanism (w/o Global Attention). As shown in Table V, removing either TNN pretraining or meta-path features leads to a substantial drop in performance, highlighting the importance of stable node initialization and structural semantics. Omitting the global attention module also slightly degrades performance, reflecting its complementary role in aggregating global context beyond local neighborhoods. Overall, these results demonstrate that each component of MetaRTL contributes meaningfully to the model's effectiveness.

2) ***Effect of Pretraining and Aggregation Stages****:* Further, to verify that the performance gains of MetaRTL primarily stem from the Aggregation Stage rather than the initial Pre-training Stage, we compare their prediction results in Figure 3. Across all three tasks, the Pre-training Stage yields lower performance than the second stage. This shows that the Pre-training Stage, with limited training iterations and a shallow HGNN, mainly produces stable initial node features containing basic semantic information, rather than a fully converged, final semantic representation. Building on this, the Aggregation Stage leverages precomputed meta-path features with meta-path self-attention to integrate local heterogeneous

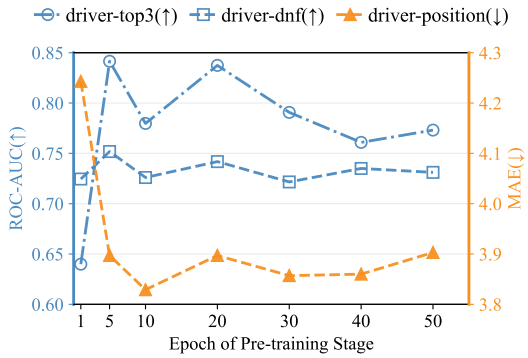

Fig. 4. MetaRTL Performance vs. Pre-training Stage Epochs.

semantics effectively. The global node cross-attention further mitigates biases introduced by neighbor sampling. Together, these mechanisms significantly boost model performance. In summary, MetaRTL's two-stage design distinctly decouples semantic stabilization from deep semantic fusion, resulting in superior predictive accuracy.

D. Training Efficiency and Scalability Analysis

1) ***Effect of Pretraining Duration****:* To systematically evaluate the efficiency of the Pre-training Stage, we assess variants on three downstream tasks from the `rel-f1` dataset. As shown in Figure 4, when pre-training is too short particularly at 1 epoch, the table encoder is underfitted and fails to capture sufficient structural and semantic information. This results in low quality initial node embeddings for the Aggregation Stage and significantly degraded downstream performance. Extending pre-training to 5–10 epochs substantially improves and stabilizes performance, indicating that a moderate number of iterations allows the encoder to effectively extract essential patterns from tabular inputs. However, further increasing pre-training leads to a gradually slight performance decline, which can be attributed to overfitting: excessive training causes the encoder to memorize spurious patterns and introduce strong inductive biases that cannot be easily corrected during downstream learning. In summary, the number of epochs has a critical impact on MetaRTL, with too few causing underfitting and too many leading to harmful bias.

2) ***Scalability Test****:* To experimentally validate the scalability of our method, we generated synthetic graphs with a fixed average out-degree, varying the total number of nodes, with target nodes accounting for 1% of the total. As shown in Figure 5, the overall training time of MetaRTL increases approximately linearly with the number of nodes. Specifically, the Pre-training Stage initially consumes a small fraction of the total time but grows faster as the graph size increases, while the Aggregation Stage remains relatively lightweight. Non-parametric meta-path computation contributes negligibly to the overall time. These results indicate that MetaRTL scales efficiently across graphs of varying sizes, even under variations in table features and their distributions.

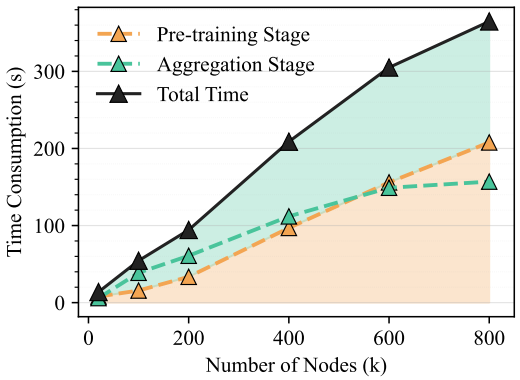

Fig. 5. Training Time vs. Number of Nodes (Fixed Node Out-Degree).

VI. Conclusion

We present MetaRTL, a novel framework for relational table learning that effectively models complex relational structures induced from relational table data. Instead of relying on deep, parameter-heavy message passing, MetaRTL adopts a two-stage design: it first learns stable node representations through shallow pretraining, and then aggregates precomputed meta-path features using a lightweight attention module, MetaAttn. This decoupled architecture enables expressive relational modeling while preserving scalability across large and diverse relational databases. Extensive experiments demonstrate that MetaRTL consistently outperforms existing methods, establishing new state-of-the-art results in relational table learning. As future work, we plan to extend this framework to data lake scenarios [31].

Acknowledgments

This work was supported by the Natural Science Foundation of Shanghai under Grant 23ZR1434000.

References

[1] T. Halpin and T. Morgan, *Information modeling and relational databases*. Morgan Kaufmann, 2010.

[2] W. Li, X. Huang, J. Zheng, Z. Wang, C. Wang, L. Pan, and J. Li, "rLLM: Relational table learning with llms," *arXiv preprint arXiv:2407.20157*, 2024.

[3] S. Džeroski, *Relational Data Mining*. Boston, MA: Springer US, 2010, pp. 887–911.

[4] L. Zahradník, J. Neumann, and G. Šír, "A deep learning blueprint for relational databases," in *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[5] F. Chen, K. Zhong, A. Zhang, Z. Wang, L. Pan, and J. Li, "TLSQL: Table learning structured query language," *arXiv preprint arXiv:2601.14109*, 2026.

[6] M. Fey, W. Hu, K. Huang, J. E. Lenssen, R. Ranjan, J. Robinson, R. Ying, J. You, and J. Leskovec, "Position: Relational deep learning – graph representation learning on relational databases," in *International Conference on Machine Learning*, vol. 235, 2024, pp. 13 592–13 607.

[7] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 6, pp. 7499–7519, 2022.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[9] A. Zou, J. Ji, M. Lei, J. Liu, and Y. Song, "Exploring brain effective connectivity networks through spatiotemporal graph convolutional models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 6, pp. 7871–7883, 2022.

[10] I. Stack Exchange, "Stack exchange data dump," Internet Archive, https://archive.org/details/stackexchange, 2024, accessed: Nov. 20, 2025.

[11] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 3538–3545.

[12] Z. Wang, H. Ding, L. Pan, J. Li, Z. Gong, and P. S. Yu, "From cluster assumption to graph convolution: Graph-based semi-supervised learning revisited," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 7, pp. 12 952–12 963, 2025.

[13] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The Web Conference*, 2019, p. 2022–2032.

[14] Z. Wang, J. Wang, Y. Guo, and Z. Gong, "Zero-shot node classification with decomposed graph prototype network," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1769–1779.

[15] X. Yang, M. Yan, S. Pan, X. Ye, and D. Fan, "Simple and efficient heterogeneous graph neural network," in *AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, 2023, pp. 10 816–10 824.

[16] K. Kersting, "An inductive logic programming approach to statistical relational learning," *AI Communications*, vol. 19, no. 4, pp. 389–390, 2006.

[17] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning*. Cambridge, MA, USA: MIT Press, 2007.

[18] W. Wang, M. Zhang, G. Chen, H. V. Jagadish, B. C. Ooi, and K.-L. Tan, "Database meets deep learning: Challenges and opportunities," *SIGMOD Record*, vol. 45, no. 2, p. 17–22, Sep. 2016.

[19] M. Cvitkovic, "Supervised learning on relational databases with graph neural networks," *arXiv preprint arXiv:2002.02046*, 2020.

[20] T. Chen, C. Kanatsoulis, and J. Leskovec, "RelGNN: Composite message passing for relational deep learning," in *International Conference on Machine Learning*, vol. 267, 2025, pp. 8296–8312.

[21] V. Lachi, A. Longa, B. Bevilacqua, B. Lepri, A. Passerini, and B. Ribeiro, "Boosting relational deep learning with pretrained tabular models," *arXiv preprint arXiv:2504.04934*, 2025.

[22] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, p. 135–144.

[23] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, vol. 70, 2017, pp. 1263–1272.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[25] K. Kong, J. Chen, J. Kirchenbauer, R. Ni, C. B. Bruss, and T. Goldstein, "GOAT: A global transformer on large-scale graphs," in *International Conference on Machine Learning*, vol. 202, 2023, pp. 17 375–17 390.

[26] V. P. Dwivedi, Y. Liu, A. T. Luu, X. Bresson, N. Shah, and T. Zhao, "Graph transformers for large graphs," *arXiv preprint arXiv:2312.11109*, 2023.

[27] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[28] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," in *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 18 932–18 943.

[29] K. Chen, P. Chiang, H. Chou, T. Chen, and T. Chang, "Trompt: Towards a better deep neural network for tabular data," in *International Conference on Machine Learning*, vol. 202, 2023, pp. 4392–4434.

[30] J. Chen, J. Yan, Q. Chen, D. Z. Chen, J. Wu, and J. Sun, "Can a deep learning model be a sure bet for tabular prediction?" in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, p. 288–296.

[31] F. Pan, T. Zhang, A. Zhang, Y. Sun, Z. Wang, L. Chen, L. Pan, and J. Li, "LakeMLB: Data lake machine learning benchmark," *arXiv preprint arXiv:2602.10441*, 2026.