RELICL: TRAINING-FREE RELATIONAL LEARNING WITH TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models achieve state-of-the-art performance on single-table tasks without any training. Recent work suggests that they are also well-suited for relational learning via deep feature synthesis (DFS), which flattens a relational schema into a single table by adding aggregates of the other tables' columns as features. This approach is appealing because it directly benefits from improvements to or customization of the underlying tabular foundation model. In this paper, we identify two key problems with DFS: *feature explosion* and *interaction blindness*. The first problem arises because the number of DFS features grows quickly as the schema becomes more complex, limiting scalability and performance. The second problem arises because column-wise aggregates do not account for feature interactions, limiting performance. We propose and explore an alternative method termed RelICL, which keeps the benefits of DFS but alleviates these two problems. At its heart, RelICL propagates and fuses information step by step through the schema graph, using the same tabular foundation model that is eventually used for prediction to do so. In our experimental study using RelBench tasks, RelICL was on par with the strongest approach based on deep feature synthesis.

1 INTRODUCTION

Relational data is ubiquitous in industry and science (Fey et al., 2024), spanning application areas like e-commerce, social networks, and clinical research (Robinson et al., 2024). A relational database can be understood as a collection of multiple tables, interconnected with each other via foreign key references (Codd, 1970; Garcia-Molina et al., 2009). In this paper, we explore how to leverage strong tabular foundation models for relational learning.

Tabular foundation models (TFMs) such as TabPFN (Hollmann et al., 2022; 2025), TabICL (Qu et al., 2025; 2026), and others (Ma et al., 2025; Zhang et al., 2025; Spinaci et al., 2025; Kong & Das, 2026) are pretrained on a large amount of synthetic and/or real-world tabular data. They reach state-of-the-art performance for single-table learning tasks using in-context learning, i.e., without further training. For example, on the tabular benchmark TabArena (Erickson et al., 2025), most top-ranked models[1] are TFMs, outperforming gradient-boosted trees such as XGBoost (Chen & Guestrin, 2016), deep tabular models such as RealMLP (Holzmüller et al., 2024), and AutoML systems such as AutoGluon (Erickson et al., 2020).

A natural and practically important question is whether and how TFMs can be leveraged for relational learning on a set of interconnected tables. More specifically, given a *task table* holding in-context examples and prediction targets, how can we leverage information from the remaining *context tables*? *Deep feature synthesis (DFS)* (Kanter & Veeramachaneni, 2015) provides one answer to that question: It extends the task table with additional *DFS features*, each obtained by aggregating values from a single context table column, and then applies a TFM to the resulting *extended task table*. To do so, DFS follows *meta-paths* (Sun et al., 2011), i.e., chains of foreign key references that start at the task table and end at a context table. For every such meta-path, DFS aggregates the context table's columns using multiple aggregation functions (e.g., mean, sum, median, mode, minimum, maximum). For example, in a social network task, a DFS feature may hold the average age of the friends of a given user. DFS needs no training, it can be used with any tabular model, including

[1] Latest leaderboard: https://huggingface.co/spaces/TabArena/leaderboard

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

TFMs, and it can reach state-of-the-art results for relation learning (Xu et al., 2026; Hayler et al., 2026). DFS is implemented in several frameworks such as Featuretools (Kanter & Veeramachaneni, 2015), RDBLearn (Xu et al., 2026; Zhang et al., 2026) and TabPFN-Rel (Hayler et al., 2026).

Despite its simplicity and strong empirical performance, applying DFS in practice can be challenging due to *feature explosion*: the number of DFS features grows quickly as the schema becomes more and more complex. Since DFS aggregates every column along every meta-path using multiple aggregation functions, the width of the extended task table grows with the number and width of the context tables, limiting scalability. For this reason, practical DFS approaches need to use strategies such as capping meta-paths length or pruning DFS features (Kanter & Veeramachaneni, 2015; Lam et al., 2017; Xu et al., 2026; Hayler et al., 2026), limiting performance. A second inherent limitation of DFS is *interaction blindness*: since each DFS feature is an aggregate of a single column, DFS cannot capture interactions between the columns the context tables. Any task that relies on such interactions is therefore impossible to solve with DFS.

In this paper, we explore the question of whether we can keep the benefits of DFS, i.e., training-free relational prediction using any tabular foundation model, while avoiding its drawbacks of feature explosion and interaction blindness. To do so, we propose a simple approach termed RelICL. Like DFS, RelICL builds an extended task table fed into a TFM for the final predictions. Unlike DFS, RelICL also exploits the TFM to build the extended task table. In more detail, the TFM is additionally used for *feature extraction* (row embeddings from the context tables) and *feature fusion* (propagating row embeddings step by step). RelICL does not suffer from feature explosion because the number of additional features only depends on the size of the local neighborhood instead of on the full database. RelICL does not suffer from interaction blindness because it propagates, pools, and fuses row embeddings instead of individual context-table columns. RelICL works out of the box with any TFM which provides row embeddings, e.g., most recent TFMs including TabICL (Qu et al., 2026), TabPFN (Hollmann et al., 2025), or TabFM (Kong & Das, 2026).

To summarize, the contributions of this paper are: **(C1)** We identify two major limitations of DFS, feature explosion and interaction blindness (Sec. 2) and show that there are tasks for which DFS-based predictors can do no better than random guessing. **(C2)** We propose RelICL (Sec. 3), which avoids both limitations by replacing column-wise aggregation along meta-paths with propagation and fusion of row embeddings along individual foreign key references. **(C3)** We report on an experimental study (Sec. 4) using the 21 entity tasks of RelBench (Robinson et al., 2024; Gu et al., 2026), indicating that RelICL is on par with the strongest DFS-based approach (and outperforming many alternative approaches) and that the underlying TFM can be easily swapped. We make source code and results are available through OpenReview (later through GitHub).

2 Deep Feature Synthesis and its Limitations

2.1 Preliminaries

A relational database D is a set of tables interconnected by references, see the example in Fig. 1a. We follow Fey et al. (2024); Robinson et al. (2024) and model relational learning via a *task table* T, which holds in-context examples (T_{train}, green in Fig. 1a) and *query rows* (T_{eval}, yellow), and multiple *context tables*, which hold auxiliary information. A *reference* links the rows of two tables via key columns: if two rows have the same key value, they are linked together. For example, in Fig. 1a, there is a reference between *lap times* (key *r_id*) and *results* (key *r_id*). References can be one-to-one, many-to-one (e.g., *lap times* and *results*) or many-to-many (e.g., *task* and *results*). Tables and references can be represented via a *schema graph*.

Problem statement. Given D, T_{train}, and T_{eval}, use a pretrained tabular foundation model $\mathcal{T}$ to predict the target attribute of the rows in T_{eval}.

A related task is relational deep learning (Fey et al., 2024; Robinson et al., 2024), which adds a temporal component for tasks such as forecasting. Here predictions at time t must not be influenced by data newer than t. For example, to predict whether Cathi wins the next race in the example of Fig. 1a, we should not already be aware of the results of the next race. We follow Fey et al. (2024); Robinson et al. (2024) and address such tasks by temporal filtering using a cut-off time (gray "future" rows in Fig. 1a) and temporal pooling (discussed later). In most of the following discussion, we ignore the temporal component for ease of exposition.

Our problem statement specifically requires the use of a single TFM. Approaches such as graph neural networks (Fey et al., 2024; Robinson et al., 2024; Chen et al., 2025; Dwivedi et al., 2025), which are trained per database, and relational foundation models (Ranjan et al., 2025; Kothapalli et al., 2026), which are pretrained on a relational database corpus, fall outside this setting. We focus on TFMs because they achieve state-of-the-art single-table performance, large corpora of tabular data are readily available (Gardner et al., 2024), TFMs are actively developed and continuously improved, and TFMs are easy to customize for a particular domain via continual pre-training. Any advances in TFMs thus directly translate to advances in relational learning based on TFMs.

2.2 Deep Feature Synthesis

Deep feature synthesis (Kanter & Veeramachaneni, 2015) *extends* the task table T by DFS features, i.e., columns derived from the context tables along meta-paths, and uses the resulting extended task table T_{DFS} for prediction. A *meta-path* $P = (T, R_1, \ldots, R_L)$ of length L is a sequence of $L+1$ tables starting at the task table T such that every two consecutive tables are connected by a reference. For example, (*task*, *results*, *lap times*) is a length-2 meta-path in Fig. 1a. Given a meta-path, DFS determines for every row x in T the set of rows $P(x)$ in R_L that are reachable from r by following the meta-path P.[2] The columns in $P(x)$ are then aggregated individually using multiple aggregation functions, e.g., length, mean/min/max/std (for numerical columns), mode/value frequencies (for categorical columns). The resulting aggregates then form the DFS features for meta-path P. This process is repeated for each meta-path. The number of meta paths is at least linear but can be exponential with respect to the number of tables in the database (Lam et al., 2017).

Once the extended task table has been obtained, any tabular model can be used for prediction. This includes classical models such as XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), or LightGBM (Ke et al., 2017), deep tabular models such as RealMLP (Holzmüller et al., 2024) or FT-Transformer (Gorishniy et al., 2021), and tabular foundation models as in the recent approaches RDBLearn (Xu et al., 2026; Zhang et al., 2026) and TabPFN-Rel (Hayler et al., 2026).

2.3 Feature Explosion

Limitation 1 (Feature explosion)**.** The number of DFS features grows with the complexity of the schema.

The extended task table holds one column per combination of (i) meta-path, (ii) column of the context table where that meta-path ends, and (iii) aggregation function. Its width is therefore $\mathcal{O}(\#\text{meta-paths} \cdot \#\text{attributes} \cdot \#\text{aggregates})$. For instance, the RelBench database rel-f1 (Robinson et al., 2024) holds nine tables with 67 columns in total (App. D.1). RDBLearn generates 7 715 columns on the `driver-top3` task with its six default aggregation functions (before pruning down to 5 460, unbounded depth), more than one hundred times the width of the schema itself.

In practice, this problem needs to be addressed. Some works do this by selecting a small set of meta-paths manually (Wang et al., 2019), whereas others aim to learn promising meta-paths (Ferrini et al., 2024). The perhaps most common approach, however, is to bounds the maximum length of the considered meta-path, the *depth* (Kanter & Veeramachaneni, 2015; Xu et al., 2026; Hayler et al., 2026). In fact, the latest versions of featuretools (1.31.0) (Kanter & Veeramachaneni, 2015), RDBLearn (1.1) (Xu et al., 2026), and TabPFN-Rel (Hayler et al., 2026) all use a maximum depth of 2 by default. Such an approach can be highly problematic, however.

Prop. 1 (Insufficiency of bounded depth)**.** *For every depth bound D, there are relational prediction tasks that no DFS-based approach can solve better than random guessing.*

This problem arises when the relevant signal for prediction is too far away from the task table, and it holds no matter how powerful the tabular model or the considered aggregation functions are. As a simple example, consider a task where the target label is directly available as a feature in a context table reachable from the task table by following a meta-path of length 3. Any DFS-based approach with depth bound 2 cannot find this signal and thus fails. Tab. 1 shows empirical results for this task (1000 training rows, 300 query rows, 3 seeds) using RDBLearn (v1.1, default configuration, i.e.,

[2] For relational deep learning, only rows that are reachable without accessing future rows are considered.

Tab. 1: Bounded depth task of Prop. 1, with RDBLearn at its default depth bound of two (AUROC (%), higher is better, mean ± SEM over three seeds).

Method	AUROC
RDBLearn (depth bound 2)	50.8 ± 0.1
RelICL (TabICL, unbounded)	100.0 ± 0.0
RelICL (TabPFN, unbounded)	100.0 ± 0.0

Tab. 2: Interaction blindness task of Prop. 2 (AUROC (%), higher is better, mean ± SEM over three seeds).

Method	AUROC
RDBLearn	46.5 ± 3.3
RelICL (TabICL)	99.3 ± 0.7
RelICL (TabPFN)	99.8 ± 0.1

depth bound 2) and RelICL (Sec. 3) with both TabICL and TabPFN as TFM backbones. RDBLearn reached a test AUROC of 50.8% on average, i.e., random guessing. RelICL consistently reached 100.0% AUROC with both backbones. See App. B.1 for details.

In addition to bounding meta-path length, prior work explored post-hoc feature selection methods, e.g., through information gain (Markovitch & Rosenstein, 2002; Piramuthu & Sikora, 2009; Fan et al., 2010), through feature importance (Shi et al., 2020), or through tests of independence with respect to the target (Lam et al., 2017). These methods milden the problem, but they also require computation of all DFS features first and they ultimately may remove relevant information from the extended task table.

2.4 Interaction Blindness

Limitation 2 (Interaction blindness)**.** DFS-based methods are oblivious to feature interactions.

Since each DFS feature is an aggregate of a single context-table column, DFS-based approaches are inherently oblivious to feature interactions. In fact, Xu et al. (2026) makes single-column aggregation their "guiding principle" according to which aggregates should compress only within but not across columns. Again, such an approach can be highly problematic.

Prop. 2 (Insufficiency of DFS features)**.** *There are relational prediction tasks that no DFS-based predictor can solve better than random guessing.*

This arises when the signal required to solve the task lies solely in feature interactions, but not feature values. Consider, for example, a relational learning task with one context table, in which every task table row references exactly three rows in the context table The three rows are either $\{(0,-1),(1,0),(2,1)\}$ (positive correlation) or $\{(0,1),(1,0),(2,-1)\}$ (negative correlation), and the task is to predict which of the two cases hold. Since each column has the same set of values, all DFS features necessarily coincide for both cases so that a DFS-based method cannot distinguish between them. Tab. 2 validates this empirically (1000 training rows, 300 query rows, 3 seeds): RDBLearn (v1.1, default configuration) is no better than random guessing, whereas RelICL (Sec. 3) has good performance. App. B.2 provides further details.

One way to combat interaction blindness is to engineer additional features, e.g., manually or via an LLM (App. A). Such an approach is challenging to get right, however, and may lead to further feature explosion.

3 RelICL

We now describe RelICL, which avoids the drawbacks of feature explosion and interaction blindness. At its heart, RelICL is a very simple approach; see Fig. 1. Its key idea is to use a TFM for both final predictions (as in DFS) but also for feature extraction and feature fusion (unlike DFS). More specifically, RelICL represents context-table rows via row embeddings extracted by the TFM. It fuses information from a referenced context table by adding pooled row embeddings as additional features. Since the number of added features depends only on the number of directly referenced tables (i.e., the local neighborhood), feature explosion is avoided. Since row embeddings capture feature interactions, interaction blindness is avoided as well.

We first discuss the assumptions underlying RelICL, and then describe the approach in more detail.

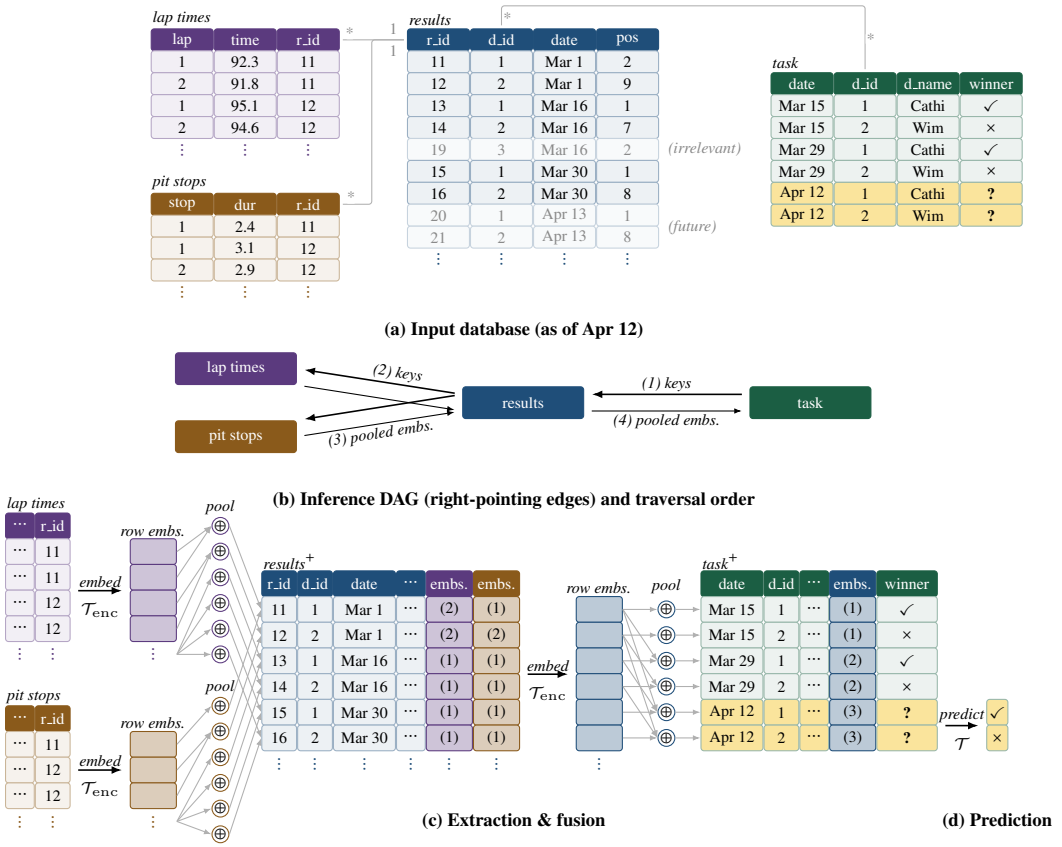


Fig. 1: A binary classification task loosely inspired by `driver-top3` (Robinson et al., 2024), predicting the drivers who win the next race. Highlighted rows are the ones being predicted. **(a)** The input database. Grayed-out rows are removed before any modeling because (i) they either fall after the evaluation cutoff or (ii) they are irrelevant for the task. **(b)** The inference DAG (Sec. 3.2), with tables ordered by their distance from the task table. The outward traversal (upper arrows) carries the keys that mark the relevant rows. The inward traversal (lower arrows) in turn carries the pooled row embeddings. **(c)** Each table pools the row embeddings of its children and gains one further column block per child. **(d)** Predict on the task table extended with the pooled row embeddings from **(c)**.

3.1 Row Embeddings are Good Features

We consider TFMs $\mathcal{T}$ based on row embeddings. To obtain predictions for a table R, $\mathcal{T}$ first uses an encoder $\mathcal{T}_{\text{enc}}$ to extract row embeddings of the training and query rows[3] (without the target attribute) and subsequently applies a prediction head $\mathcal{T}_{\text{pred}}$ (using the target attribute of the train rows). In short, $\mathcal{T}(R) = \mathcal{T}_{\text{pred}}(\mathcal{T}_{\text{enc}}(R))$. Most most recent TFMs, including TabICL (Qu et al., 2026), TabPFN (Hollmann et al., 2025), TabDPT (Ma et al., 2025), LimiX (Zhang et al., 2025), and TabFM (Kong & Das, 2026), follow this approach.

Assumption 1 (Row embeddings are good features)**.** The performance of $\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$ matches the one of $\mathcal{T}(R)$.

This means that we can apply the encoder multiple times without loss in performance (since $\mathcal{T}(\mathcal{T}_{\text{enc}}(R)) \approx \mathcal{T}_{\text{pred}}(\mathcal{T}_{\text{enc}}(\mathcal{T}_{\text{enc}}(R)))$). This assumption is crucial to RelICL, because it allows us to replace the features of a table R by row embeddings without loss in performance. This in turn simplifies RelICL's feature fusion: use (pooled) row embeddings from context tables as additional features. Intuitively, this assumption appears reasonable: since $\mathcal{T}_{\text{pred}}$ only relies on $\mathcal{T}_{\text{enc}}(R)$ (but

[3]The row embedding of a query row r depends only on r and the entire set of train rows. It does not depend on any other query rows.

Tab. 3: Replacing the feature columns by a row embedding (test AUROC (%), higher is better, mean ± SEM over three seeds). $\mathcal{T}(R)$ runs TabICL on the table R. $\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$ replaces all feature columns by TabICL's respective row embedding. Δ is the paired difference against the original data of the same seed. The last row averages the ten datasets, so its SEM is taken across datasets rather than across seeds.

	$\mathcal{T}(R)$	$\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$	
Dataset	**AUROC**	**AUROC**	Δ
bioresponse	87.7 ± 1.0	87.5 ± 0.8	-0.3 ± 0.3
churn	92.8 ± 0.8	92.6 ± 0.6	-0.2 ± 0.2
compass	72.3 ± 0.7	72.4 ± 0.7	+0.1 ± 0.1
credit-g	82.2 ± 1.8	83.1 ± 0.4	+0.9 ± 1.4
jasmine	89.8 ± 1.1	86.8 ± 0.4	-3.0 ± 0.6
kr-vs-kp	100.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0
philippine	93.6 ± 0.1	84.7 ± 0.6	-8.9 ± 0.6
phoneme	97.6 ± 0.0	96.2 ± 0.2	-1.5 ± 0.2
qsar-biodeg	92.4 ± 2.1	91.3 ± 2.4	-1.1 ± 0.4
spambase	99.1 ± 0.0	98.8 ± 0.0	-0.4 ± 0.0
∅	90.8 ± 2.7	89.3 ± 2.6	-1.4 ± 0.9

not on R itself), the row embeddings must summarize the corresponding rows of R adequately for $\mathcal{T}$ to work well.

We tested this assumption on ten single-table binary classification datasets from OpenML (Vanschoren et al., 2014). For each task, we compared TabICL on the original table ($\mathcal{T}(R)$) against TabICL on the row embeddings ($\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$). App. C.2 provides dataset statistics and further details, Tab. 3 shows the results. We found that for most datasets (all except philippine and jasmine), there was no statistically significant loss in performance, providing some justification for our assumption.

3.2 Feature Extraction and Feature Fusion

RelICL operates in three phases:

1. Outward traversal to determine relevant context table rows
2. Inward traversal to extract and fuse features from context tables
3. Prediction on the extended task table

The first two phases rely on a partial ordering of the tables in the database, which we represent as an *inference DAG* (see Fig. 1b) with root T (the task table). An edge $R \to S$ in the inference DAG states that (i) there is a reference between table R and table S and (ii) the rows of R should be fused into the rows of S. To obtain the inference DAG, RelICL uses breadth-first search starting at the task table so that tables more distant to the task table are fused into tables closer to the task table.

Outward traversal. Not every row in every context table is *relevant* in that it is directly or indirectly referenced by a row in the task table. Irrelevant rows do not need to be propagated, and the goal of the outward traversal is to determine and prune such rows. In Fig. 1a, for instance, there is no prediction task for driver 3 so that the results of driver 3 are irrelevant. Pruning irrelevant rows is especially beneficial due to the asymptotically quadratic cost of attention (Vaswani et al., 2017) with respect to the number of rows in typical tabular TFMs (Qu et al., 2026; Hollmann et al., 2025).

During outward traversal, we traverse the inference DAG in reverse topological order, starting at the task table. When processing a table, we first determine its set of relevant rows (all for the task table) and then propagate the corresponding set of relevant keys along every reversed edge (e.g., $\{1, 2\}$ from *task→results* or $\{11\text{–}14, 15, 16, \ldots\}$ from *results→lap times*). All rows that do not contain a relevant key are marked irrelevant and dropped (e.g., row 19 in *results*).

Inward traversal. The goal of inward traversal is to propagate information from the context tables towards the task table, ultimately constructing the extended task table. This time, the inference DAG

is processed in topological order. Whenever a table R is processed, we fuse information from its children (along incoming edges) into the table to obtain R^+, extract and pool row embeddings of R^+, and finally propagate information to the parents (along outgoing edges). Each row r^+ of R^+ is meant to represent (i) the corresponding row r from R and (ii) all the rows from *all descendant tables* referenced by r.

Feature extraction. The extraction of features for some table R^+ makes use of Asm. 1: we run the encoder $\mathcal{T}_{\text{enc}}$ on R^+ to obtain a row embedding for every row of R^+ (cf. Fig. 1c). In contrast to using the rows directly, this has the advantage that (i) the size of the row embeddings is independent of the width of the table and (ii) each element of each row embedding depends on the entire row (and not just one column). The first advantage allows us to avoid feature explosion: adding features to R to obtain R^+ does not affect the size of the information propagated towards R's parents. The second advantage avoids interaction blindness.

Feature fusion. We now describe how to fuse information from R's children into R to obtain R^+. Consider a child $C \to R$ in the inference DAG. Since tables are processed in topological order, C has already been processed when processing R, so that an embedding for every row of C^+ is available. We now extend R by a block of columns holding pooled row embeddings. In more detail, for every row r in R, we pool the row embeddings of the rows in C that are referenced[4] by r (e.g., row 11 of *results* references the first two rows of *lap times*), e.g., using sum, average, or max pooling. Since the latter two methods are oblivious to the number of referenced rows, we also add an explicit *count column* with that number in the spirit of the degree scalers of Corso et al. (2020). These numbers are shown in parentheses in Fig. 1c.

Prediction. The inward traversal stops after processing the task table T to obtain the extended task table T^+. The number of features in T^+ is determined by the number of its children in the inference DAG (one in Fig. 1), not the number of tables in the entire database. RelICL does not need (nor does it use) a depth bound to avoid feature explosion. A row $t^+ \in T^+$ thus summarizes all the rows referenced from the corresponding row $t \in T$, no matter how far away. To obtain the final prediction, RelICL runs the entire tabular model on the extended task table, i.e., $\mathcal{T}(T^+)$.

4 Experiments

We performed an experimental study on RelBench in which we compared RelICL to the state-of-the-art DFS-based approaches and other methods pretrained on relational data. We found that RelICL (in its default configuration) was on par with the strongest DFS-based approaches and outperformed the pretrained relational approaches. We also provide evidence that the underlying TFM used by RelICL can be easily swapped. Our complete source code and experimental results are available via OpenReview (later via GitHub).

4.1 Experimental setup

Data. We used the 21 entity tasks from RelBench v2 (Gu et al., 2026). They span seven different relational databases in application areas such as e-commerce and retail, social and online data, racing, and clinical data. The databases vary in size from under 100 k to over 44 M rows. The benchmark consists of 12 binary classification tasks (e.g., whether a customer churns or the outcome of a clinical trial) and 9 regression tasks (e.g., a driver's finishing position or an item's future sales). App. D.1 reports database and task statistics.

Baselines. We compared RelICL against four methods, two of them based on DFS and two of them pretrained on relational data. The DFS-based methods are RDBLearn (Xu et al., 2026) and TabPFN-Rel (Hayler et al., 2026), which extends RDBLearn with feature engineering (e.g., cyclic calendar features or temporal-lag features).[5] The pretrained relational approaches are Relational Transformers (RT) (Ranjan et al., 2025) and PluRel (Kothapalli et al., 2026). Note we exclude trained models (such as RelGNN (Chen et al., 2025) or Rel-GT (Dwivedi et al., 2025)) or hosted,

[4] For relational deep learning task, pooling is temporal: we exclude any "future rows" from pooling. This is what is done in Fig. 1c to fuse *results*$^+$ into *task*.

[5] We consider the use of such features orthogonal to our work. It is conceivable, however, that they would also be beneficial for RelICL.

Tab. 4: Head-to-head wins over all regression and classification tasks combined. The baselines report different subsets of the tasks and metrics, so an average over all tasks would score each model on different tasks. We therefore compare the models pair by pair, only on the tasks both report. A cell counts the tasks the row model won over the column model divided by the total number of shared tasks. A pair shares a task only where both models report a common metric for it. RT and PluRel share no regression task with RDBLearn and TabPFN-Rel. Bold marks the model that won the pair. Tabs. 5 and 7 provide the more details.

	Pretrained (rel.)		Pretrained (tab.)		
Model	**RT**	**PluRel**	**RDBLearn**	**TabPFN-Rel**	**RelICL**
RT	–	6 / 18*	0 / 8	2 / 10	7 / 18*
PluRel	**11 / 18***	–	0 / 8	2 / 10	5 / 18*
RDBLearn	**8 / 8**	**8 / 8**	–	1 / 15	5 / 15
TabPFN-Rel	**8 / 10**	**8 / 10**	**13 / 15**	–	10 / 21
RelICL	**11 / 18***	**13 / 18***	**10 / 15**	**11 / 21**	–

* MAE results were not available for all tasks. In these cases R^2 was used.

closed-source models (such as the API version of TabPFN-Rel and KumoRFM-2 (Hudovernik et al., 2026)) from our study. The results of these models are given App. E.1 for completeness.

Metrics. We report AUROC for classification tasks and MAE as well as R^2 for regression tasks, each on the respective RelBench test sets. For prior models, we report the results from the respective papers. Note that some papers did not publish both MAE and R^2, i.e., we cannot report all metrics.

RelICL implementation. RelICL uses a number of implementation tricks to speed up processing. First, it uses dimensionality reduction (e.g., random projections) to reduce the size of the row embeddings and hence inference costs. App. C.2 provides evidence that this reduction barely affects performance. Second, it sub-samples large tables to a fixed maximum size. Finally, it merges the tables of some many-to-one relationships, which roughly corresponds to propagating rows instead of row embeddings when no pooling is needed. App. C.1 contains more details.

Hyperparameters. We ran RelICL in its default configuration throughout. This configuration ensembles RelICL predictions using one global hyperparameter setting with one (regression) and three (classification) additional RelICL predictions, each using a hyperparameter setting differing in exactly one hyperparameter. The same ensemble members are used for all classification and regression tasks, respectively. More details are given in App. D.4.

RelICL has 11 hyperparameters (such as sample sizes or the amount of dimensionality reduction). The global hyperparameter settings have been selected via random search over 100 trials per task, optimizing for overall validation AUROC (classification) and MAE (regression) on a subset of 6 classification and 6 regression tasks from RelBench. The ensemble members differ in hyperparameters that are highly-sensitive according to this search. App. D.3 has further details.

Backbones. RelICL by default used TabICL v2 (Qu et al., 2026) as its backbone TFM. We also report results with TabPFN v3 (Hollmann et al., 2025) using the same default configuration.

4.2 Results

Main results. Tab. 4 compares the models pair by pair over the tasks each pair shares, regression and classification combined. More details are provided in Tabs. 5 and 7 which report performance numbers per task type. RelICL either outperformed or was on par with all models. The closest comparison was with TabPFN-Rel, which was published concurrently to this work. TabPFN-Rel's feature engineering clearly boosted its performance in comparison to RDBLearn (and could potentially also boost RelICL's performance). Against the models pretrained on relational data, RelICL outperformed RT and PluRel without relational pretraining, mostly on classification, despite not being tuned for the R^2 metric on which these models have been tuned.

Backbones. Tabs. 6 and 8 report RelICL's results with TabPFN v3 (in the same default configuration). TabICL v2 was ahead on most tasks of both types, although the margin was small for classification. RelICL thus did not depend on TabICL specifically.

Tab. 5: Regression, MAE (lower is better) / R^2 (%, higher is better). The table gives the per-task numbers behind the regression part of Tab. 4. It shows both metrics because RT and PluRel report only R^2, while RDBLearn and TabPFN-Rel report only MAE. RelICL reports both, so it can be compared with either group. Bold marks the best value per task and metric.

	Pretrained (rel.)		Pretrained (tab.)		
Task	**RT**	**PluRel**	**RDBLearn**	**TabPFN-Rel**	**RelICL**
item-ltv	– / 32.5	– / **40.5**	48.56 / –	47.80 / –	**43.97** / 28.6
user-ltv	– / **36.9**	– / 18.5	14.54 / –	**14.40** / –	14.78 / 21.0
ad-ctr	– / 4.5	– / 4.9	**0.03** / –	**0.03** / –	0.04 / **18.3**
user-attendance	–	–	–	**0.24** / –	**0.24** / **5.1**
driver-position	– / 52.4	– / **55.5**	–	3.76 / –	**3.68** / 25.2
item-sales	– / 14.0	– / 20.0	0.06 / –	0.06 / –	**0.04** / **69.2**
post-votes	– / **33.9**	– / 25.5	0.07 / –	0.07 / –	**0.06** / 25.6
site-success	– / 4.5	– / **38.6**	0.42 / –	**0.39** / –	0.40 / -22.0
study-adverse	– / 2.6	– / 1.6	43.91 / –	42.60 / –	**32.42** / **54.0**

– Not reported.

Tab. 6: RelICL with the TabPFN backbone.

Pretrained (tab.)
RelICL (TabPFN)
43.41 / 29.2
15.48 / 16.5
0.05 / 0.1
0.30 / 17.2
4.00 / 13.4
0.04 / 50.1
0.08 / 17.0
0.43 / -7.1
29.97 / 56.9

Tab. 7: Classification (AUROC (%), higher is better). The table gives the per-task numbers behind the classification part of Tab. 4. Bold marks the best value per task.

	Pretrained (rel.)		Pretrained (tab.)		
Task	**RT**	**PluRel**	**RDBLearn**	**TabPFN-Rel**	**RelICL**
item-churn	70.9	72.5	82.1	**82.8**	78.5
user-churn (ama)	64.0	65.0	67.6	**70.2**	65.4
user-clicks	59.5	47.9	**69.0**	61.5	64.3
user-visits	61.8	63.4	65.5	**66.9**	66.2
user-ignore	–	–	–	70.1	**88.5**
user-repeat	–	–	–	76.9	**79.3**
driver-dnf	**81.2**	81.0	–	71.5	73.6
driver-top3	**89.3**	88.4	–	79.3	81.4
user-churn (hm)	62.8	66.0	68.0	**70.6**	68.4
user-badge	80.1	82.0	85.3	**86.4**	86.3
user-engagement	75.7	86.2	89.4	**90.6**	90.0
study-outcome	51.8	51.8	71.6	73.1	**73.4**

– Not reported.

Tab. 8: RelICL with the TabPFN backbone.

Pretrained (tab.)
RelICL (TabPFN)
78.4
64.6
58.7
65.7
87.9
77.7
74.2
78.4
67.8
84.9
89.5
73.6

Limitations. Our experimental study only considered classification and regression tasks from RelBench. We did not reproduce baseline numbers, but reported numbers from the respective papers, which may differ in tuning, seeds, and evaluation protocol, and uncertainties are commonly not reported. A win in Tab. 4 may thus come from a performance gap that is not statistically significant. Asm. 1 does not hold on every table: replacing the features by the row embedding cost 8.9 AUROC points on philippine and 3.0 on jasmine (App. C.2). Finally, the synthetic tasks of Sec. 2 show that limited depth and interaction blindness are indeed weaknesses of DFS-based approaches, whereas the RelBench results indicate that DFS models may nevertheless perform well.

5 Conclusion

Feature explosion and interaction blindness are two inherent limitations of deep feature synthesis for relational learning with tabular foundation models. We explored whether these limitations can be avoided while keeping its benefits. We proposed RelICL, a simple training-free approach that uses a tabular foundation model for multiple purposes (feature extraction, feature fusion, and final prediction). RelICL does not suffer from the limitations of DFS and had strong empirical performance in our experimental study.

AI USE STATEMENT

We did not use generative AI tools for tasks with required disclosure. We used generative AI tools for (i) speeding up the creation of figures, (ii) speeding up programming, (iii) small text edits to improve readability, (iv) searching for additional sources. We have reviewed and verified all AI-assisted parts and take full responsibility for the content of this work.

ETHICS STATEMENT

Our paper, to the best of our knowledge, does not raise questions regarding the Code of Ethics.

REPRODUCIBILITY STATEMENT

The source code required to reproduce the results of this paper is provided alongside this submission on OpenReview (and later through GitHub for the final paper). The `README.md` file contains setup instructions and information on all the included directories. Besides the source code, we also publish further assets such as experimental results. Our proposed method, RelICL, is explained in Sec. 3; further details are provided in App. C. Sec. 2 describes two experiments on synthetic datasets in short; more detail is provided in Apps. B.1 and B.2. Sec. 3 describes one experiment on OpenML datasets; more details are provided in App. C.2. Sec. 4 describes our main experiments in short; more detail is provided in App. D; App. E contains additional results. Data for the RelBench experiments is publicly available and will be fetched by the provided code. Data for our synthetic datasets is generated directly within the code.

REFERENCES

Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International Conference on Machine Learning*, pp. 199–207. PMLR, 2013.

Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite Message Passing for Relational Deep Learning. In *Forty-Second International Conference on Machine Learning*, June 2025.

Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, San Francisco California USA, August 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785.

E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970. ISSN 0001-0782, 1557-7317. doi: 10.1145/362384.362685.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in neural information processing systems*, 33:13260–13271, 2020.

Milan Cvitkovic. Supervised Learning on Relational Databases with Graph Neural Networks, February 2020.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. ISSN 1098-2418. doi: 10.1002/rsa.10073.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Vijay Prakash Dwivedi and Xavier Bresson. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, January 2021.

Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico Lopez, Charilaos I. Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. Relational Graph Transformer. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data, March 2020.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A Living Benchmark for Machine Learning on Tabular Data, June 2025.

Wei Fan, Erheng Zhong, Jing Peng, Olivier Verscheure, Kun Zhang, Jiangtao Ren, Rong Yan, and Qiang Yang. Generalized and Heuristic-Free Feature Construction for Improved Accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 629–640. Society for Industrial and Applied Mathematics, April 2010. ISBN 978-0-89871-703-7 978-1-61197-280-1. doi: 10.1137/1.9781611972801.55.

Francesco Ferrini, Antonio Longa, Andrea Passerini, and Manfred Jaeger. Meta-Path Learning for Multi-Relational Graph Neural Networks. In *Proceedings of the Second Learning on Graphs Conference*, pp. 2:1–2:17. PMLR, April 2024.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Position: Relational Deep Learning - Graph Representation Learning on Relational Databases. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 13592–13607. PMLR, July 2024.

Matthias Fey, Vid Kocijan, Federico Lopez, J. Lenssen, and Jure Leskovec. Kumorfm: A foundation model for in-context learning on relational data. *Technical whitepaper, Kumo AI*, 2025.

Hector Garcia-Molina, Jeffrey Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Pearson, Upper Saddle River, N.J, 2009. ISBN 978-0-13-187325-4.

Josh Gardner, Juan C. Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. *Advances in Neural Information Processing Systems*, 37:45155–45205, 2024.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting Deep Learning Models for Tabular Data. In *Advances in Neural Information Processing Systems*, November 2021.

Justin Gu, Rishabh Ranjan, Charilaos I. Kanatsoulis, Haiming Tang, Martin Jurkovič, Valter Hudovernik, Mark Znidar, Pranshu Chaturvedi, Parth Shroff, Fengyu Li, and Jure Leskovec. RelBench v2: A Large-Scale Benchmark and Relational Data Repository. *ICLR 2026 Workshop on Navigating and Addressing Data Problems for Foundation Models*, March 2026.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, Jonas Kübler, Siyuan Guo, Oscar Key, Jan Hendrik Metzen, Rylee Grace, David Salinas, Arthur Cahu, Simon Bing, Benjamin Jäger, Tuana Çelik, Mihir Manium, Vitor Monteiro, Jake Robertson, Jerry Chen, Eliott Kalfon, Tomás Pereda, Lilly Wehrhahn, Dominik Safaric, Tobias Schroeder, Georg Grab, Diana Kriuchkova, Clara Cornu, Philipp Singer, Nick Erickson, Vahid Balazadeh, Marie Salmon, Simone Alessi, Kürşat Kaya, Philipp Jund, Léo Grinsztajn, Yann LeCun, Bernhard Schölkopf, Madelon Hulsebos, Lennart Purucker, Sauraj Gambhir, Frank Hutter, and Noah Hollmann. Advancing Open and Reproducible Relational Learning: RelArena-α, TabPFN-Rel and RPI, August 2026.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*, September 2022.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, January 2025. ISSN 1476-4687. doi: 10.1038/s41586-024-08328-6.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658, December 2024.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. KumoRFM-2: Scaling Foundation Models for Relational Learning, April 2026.

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pp. 754–762. PMLR, 2014.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 1–10. IEEE, 2015.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data, September 2026.

Vignesh Kothapalli, Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin, and Jure Leskovec. PluRel: Synthetic Data unlocks Scaling Laws for Relational Foundation Models, February 2026.

Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases, June 2017.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L Caterini, and Maks Volkovs. TabDPT: Scaling Tabular Foundation Models on Real Data. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 172692–172722. Curran Associates, Inc., 2025. doi: 10.52202/085713-5748.

Shaul Markovitch and Dan Rosenstein. Feature Generation Using General Constructor Functions. *Machine Learning*, 49(1):59–98, October 2002. ISSN 0885-6125, 1573-0565. doi: 10.1023/A:1014046307775.

Jan Motl and Oliver Schulte. The CTU Prague Relational Learning Repository, March 2025.

Jakub Peleška and Gustav Šír. ReDeLEx: A Framework for Relational Deep Learning Exploration. In Rita P. Ribeiro, Bernhard Pfahringer, Nathalie Japkowicz, Pedro Larrañaga, Alípio M. Jorge, Carlos Soares, Pedro H. Abreu, and João Gama (eds.), *Machine Learning and Knowledge Discovery in Databases. Research Track*, pp. 438–456, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-05981-9. doi: 10.1007/978-3-032-05981-9_26.

Jakub Peleška and Gustav šír. Tabular Transformers Meet Relational Databases. *ACM Transactions on Intelligent Systems and Technology*, 0, 2025. doi: 10.1145/3749991.

Selwyn Piramuthu and Riyaz T. Sikora. Iterative feature construction for improving inductive learning algorithms. *Expert Systems with Applications*, 36(2):3401–3406, 2009.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. CatBoost: Unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A Tabular Foundation Model for In-Context Learning on Large Data. In *Forty-Second International Conference on Machine Learning*, June 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A Better, Faster, Scalable, and Open Tabular Foundation Model. In *Forty-Third International Conference on Machine Learning*, June 2026.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational Transformer: Toward Zero-Shot Foundation Models for Relational Data. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. RelBench: A Benchmark for Deep Learning on Relational Databases. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal Graph Networks for Deep Learning on Dynamic Graphs, October 2020.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (eds.), *The Semantic Web*, volume 10843, pp. 593–607. Springer International Publishing, Cham, 2018. ISBN 978-3-319-93416-7 978-3-319-93417-4. doi: 10.1007/978-3-319-93417-4_38.

Qitao Shi, Ya-Lin Zhang, Longfei Li, Xinxing Yang, Meng Li, and Jun Zhou. Safe: Scalable automatic feature engineering framework for industrial tasks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1645–1656. IEEE, 2020.

Gustav Šír. *Deep Learning with Relational Logic Representations*. Czech Technical University, 2021.

Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. ConTextTab: A Semantics-Aware Tabular In-Context Learner. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. PathSim: Meta path-based top-K similarity search in heterogeneous information networks. *Proc. VLDB Endow.*, 4(11):992–1003, August 2011. ISSN 2150-8097. doi: 10.14778/3402707.3402736.

Joaquin Vanschoren, Jan N. Van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, June 2014. ISSN 1931-0145, 1931-0153. doi: 10.1145/2641190.2641198.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, December 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

Minjie Wang, Quan Gan, David Wipf, Zhenkun Cai, Ning Li, Jianheng Tang, Yanlin Zhang, Zizhao Zhang, Zunyao Mao, Yakun Song, Yanbo Wang, Jiahang Li, Han Zhang, Guang Yang, Xiao Qin, Chuan Lei, Muhan Zhang, Weinan Zhang, Christos Faloutsos, and Zheng Zhang. 4DBInfer: A 4D Benchmarking Toolbox for Graph-Centric Predictive Modeling on Relational DBs, April 2024.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous Graph Attention Network. In *The World Wide Web Conference*, WWW '19, pp. 2022–2032, New York, NY, USA, May 2019. Association for Computing Machinery. ISBN 978-1-4503-6674-8. doi: 10.1145/3308558.3313562.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a Graph-Centric Relational Database Foundation Model. In *Forty-Second International Conference on Machine Learning*, June 2025.

Linjie Xu, Yanlin Zhang, Quan Gan, Minjie Wang, and David Wipf. No Need to Train Your RDB Foundation Model. In *Forty-Third International Conference on Machine Learning*, June 2026.

Jiaxuan You, Jonathan M. Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):10737–10745, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i12.17283.

Lukáš Zahradník, Jan Neumann, and Gustav Šír. A Deep Learning Blueprint for Relational Databases. In *NeurIPS 2023 Second Table Representation Learning Workshop*, October 2023.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing Structured-Data Modeling Capability for Generalist Intelligence, November 2025.

Yanlin Zhang, Linjie Xu, Quan Gan, David Wipf, and Minjie Wang. RDBLearn: Simple In-Context Prediction Over Relational Databases, February 2026.

A Extended related work

This appendix covers further related work which is not directly relevant to follow the paper.

Graph-based approaches. Rather than first flattening the schema and then running inference, graph-based approaches work on the schema directly to make predictions end-to-end. Schlichtkrull et al. (2018); Cvitkovic (2020); Šír (2021) were among the first to frame supervised learning over relational databases as a graph neural network (Scarselli et al., 2008) problem, with one node per row and one edge per foreign key. Fey et al. (2024) later formalize and coin this idea as Relational Deep Learning (RDL). RDL puts a particular focus on the time domain and hence builds upon temporal graph learning (Rossi et al., 2020). In its basic form, a heterogeneous graph neural network such as GraphSAGE (Hamilton et al., 2017) or ID-GNN (You et al., 2021) is trained from scratch for every database and task (Robinson et al., 2024). Work since then follows three lines. The first improves message passing itself. RelGNN (Chen et al., 2025) passes messages across many-to-many relationships in a single hop and thereby needs fewer message-passing layers. The second adapts Graph Transformers (Dwivedi & Bresson, 2021) to the relational domain. The Relational Graph Transformer (Dwivedi et al., 2025) tokenizes a node by its type, hop distance, time, and local structure besides its features, i.e., by properties intrinsic to relational data. The third pretrains one relational foundation model across databases and applies it to any database or task, with or without fine-tuning, with the Relational Transformer (Ranjan et al., 2025), PluRel (Kothapalli et al., 2026), Griffin (Wang et al., 2025), and KumoRFM (Fey et al., 2025; Hudovernik et al., 2026) as examples. These models attain strong performance on relational benchmarks, but they need training on relational data, whereas in-context relational prediction asks what a model reaches without any. Among graph-based approaches, Zahradník et al. (2023); Peleška & šír (2025) come closest to our work. They pass messages over the relational schema with a tabular transformer, but train it end-to-end per database, whereas RelICL reuses a frozen backbone without any training.

Relational benchmarks. Benchmarks for relational data have moved from plain collections of databases to standardized prediction tasks. The CTU Prague Relational Learning Repository (Motl & Schulte, 2025) is the largest such collection, with almost 150 SQL databases. It proposes a target column for many of them, but no splits or evaluation protocol. RelBench (Robinson et al., 2024; Gu et al., 2026) adds both and pairs seven databases with 30 tasks. Each task is an SQL query evaluated at a cutoff time, so a model only sees data that existed at prediction time. Wang et al. (2024) release 4DBInfer at the same time. It varies four dimensions: dataset, task, graph extraction, and base predictor. We evaluate on RelBench since all baselines we compare against report on it. Further benchmarks aim to consolidate several approaches into one. ReDeLEx (Peleška & Šír, 2026) serves over 70 CTU databases through the RelBench interface, so their targets become RelBench-style tasks. Those tasks may be static, whereas RelBench tasks are always temporal. RelBench v2 (Gu et al., 2026) grows to 11 databases and adds the 4DBInfer tasks. It also introduces autocomplete tasks, which predict missing values inside a table. RelArena (Hayler et al., 2026) defines an evaluation protocol on RelBench. It follows TabArena (Erickson et al., 2025) from the tabular domain.

B Deep Feature Synthesis and its Limitations

B.1 Prop. 1: Insufficiency of Bounded depth

Proof of Prop. 1. Let the task table reference a table R_1, and let each R_i be connected to a table R_{i+1} for $i \leq d$, every connection a one-to-one reference. Only R_{d+1} holds a column besides its keys, namely a uniformly random bit, and the target of a task row is that bit. Every feature that follows at most d references is then a function of key values alone, which are independent of the target. □

Data. The database holds the task table and a chain of three further tables, R_1, R_2, and R_3 (Fig. 2). The task table references R_1, R_1 references R_2, and R_2 references R_3. Every reference is one-to-one. The only non-key column in the database is a bit in R_3, which equals the label of the task row. The label therefore sits three references away from the task table, which is the construction of Prop. 1 for $d = 2$. Training and test rows are disjoint, both splits are balanced, and we used 1000 training and 300 test rows. We ran three seeds for each method.

task

label	r1_id
0	1
1	2
?	3

R_1

r1_id	r2_id
1	2
2	1
3	3

R_2

r2_id	r3_id
1	2
2	1
3	3

R_3

r3_id	bit
1	0
2	1
3	1

Fig. 2: The bounded depth task for three task rows. Each line links a key column to the matching column of the next table, and every reference is one-to-one. The label of a task row equals the bit in R_3, three references away. The highlighted query row is the one the model has to predict.

task

entity_id	label
1	1
2	0
3	?

signals

signal_id	entity_id	a	b
1	1	0	−1
2	1	1	0
3	1	2	1
4	2	0	1
5	2	1	0
6	2	2	−1
7	3	0	−1
8	3	1	0
9	3	2	1

Fig. 3: Showcases a pathological task which cannot be solved through DFS features. Entity 1 is positive, so b increases with a; entity 2 is negative, so b decreases. Column a holds $0, 1, 2$ and column b holds $-1, 0, 1$ in both classes, so the classes are separated only by which value of b is next to which value of a. Entity 3 is the highlighted query row, whose label the model has to predict from its three signal rows.

B.2 Prop. 2: Insufficiency of DFS features

Data. The database holds the task table and a *signals* table (Fig. 3). Training and test entities are disjoint, both splits are balanced, and we used 1 000 training and 300 test entities. We ran three seeds for each method, each of which set the random draws of the data generator and of the model.

Setup and results. We ran RDBLearn at its default configuration. Deep feature synthesis generated nine aggregates over the *signals* table: the row count, together with the minimum, maximum, mean, and standard deviation of each of the two columns. Each of them takes a single distinct value across all training entities, so the feature generator inside RDBLearn's preprocessor discarded all nine and the base estimator received only the join key. RDBLearn therefore can only randomly guess labels.

C Method details

C.1 Schema rewrites

A *schema rewrite* is a view on the database that removes tables before fusion (Sec. 3.2). Every remaining table costs one forward pass of the tabular model. The views read the temporally filtered tables, so RelICL applies the rewrites once, and each timestamp only changes the filter.

Inlining. RelICL left-joins leaf tables into every table that references it and then drops it, which adds columns but no rows. In an entity task in RelBench, every task row references one *entity table* that describes the entity the prediction is about. RelICL inlines that table into the task table and rewires its remaining references to the task table.

Junction tables. A *junction table* R references two tables C_1 and C_2 and is referenced by none. RelICL replaces R by two copies (Fig. 4). The copy R_1 carries the columns of C_2 inline and references C_1. The copy R_2 carries the columns of C_1 inline and references C_2. Pooling R_1 into C_1 then passes the attributes of C_2 to C_1 in one step instead of two, and likewise for R_2. This follows RelGNN (Chen et al., 2025), which also passes messages between the two neighbors of a junction table in a single step.

C.2 Asm. 1: Row embeddings are good features

This appendix supports Asm. 1.

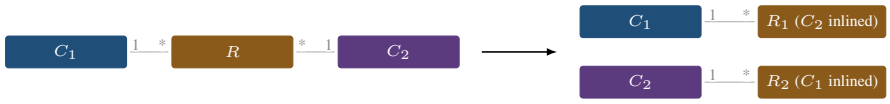


Fig. 4: Junction table elimination. The junction table R is on the many side of its references to C_1 and C_2. RelICL replaces it by two copies, each of which keeps one reference and carries the columns of the other neighbor inline.

Tab. 9: The ten OpenML datasets of App. C.2.

Dataset	OpenML ID	Rows	Features	Positive class (%)
bioresponse	4134	3751	1776	54.2
churn	40701	5000	20	14.1
compass	42193	5278	13	47.0
credit-g	31	1000	20	70.0
jasmine	41143	2984	144	50.0
kr-vs-kp	3	3196	36	52.2
philippine	41145	5832	308	50.0
phoneme	1489	5404	5	29.3
qsar-biodeg	1494	1055	41	33.7
spambase	44	4601	57	39.4

Setup. We took ten binary classification datasets from OpenML (Vanschoren et al., 2014), listed in Tab. 9. Each one is a single table. The setting $\mathcal{T}(R)$ ran TabICL on exactly that table R. The alternative replaced all feature columns by a row embedding. Both settings started from the same data, so $\mathcal{T}(R)$ is the upper bound this construction can reach. We ran the replacement twice. The setting $\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$ appended the encoder output as it is, at $d = 512$. The setting $\mathcal{T}(\text{reduce}(\mathcal{T}_{\text{enc}}(R)))$ then applied the Gaussian random projection that RelICL used in the main paper, at $d' = 32$ (see App. D.3). Every configuration used a single estimator and ran with three seeds. Each seed determines both the stratified 80/20 split and the models' randomness.

Results. Tab. 10 reports test AUROC and the regret of the replacement. On four of the ten datasets the replacement cost nothing measurable. Eight of the ten lost at most 1.5 points. The outliers are philippine at 8.9 points and jasmine at 3.0. Width alone does not explain them, since bioresponse is the widest table of the ten at 1776 columns and lost nothing measurable.

D Experimental setup

D.1 Dataset and task statistics

Tab. 11 summarizes the seven RelBench databases used in our experiments (Robinson et al., 2024; Gu et al., 2026). Tabs. 12 and 13 report, per task, the number of rows in each split. The last column reports, on the test split, the positive rate (classification) or the target's mean and standard deviation (regression). Both task tables are restricted to the 21 entity tasks. RelBench ships further task types (e.g., autocompletion or recommendation) that we did not evaluate.

D.2 Baselines

The published numbers in Sec. 4 and App. E.1 stem from GraphSAGE (Robinson et al., 2024), RelGNN (Chen et al., 2025), RelGT (Dwivedi et al., 2025), RT (Ranjan et al., 2025), PluRel (Kothapalli et al., 2026), KumoRFM-2 and DS+LightGBM (Hudovernik et al., 2026), RDBLearn (Xu et al., 2026), and TabPFN-Rel (Hayler et al., 2026). RT (target DB) is RT after continued pretraining on the target database. The papers differ in tuning, seeds, and evaluation protocol, so the tables give only a rough comparison.

Tab. 10: Replacing the feature columns by a row embedding (test AUROC (%), higher is better, mean ± SEM over three seeds). $\mathcal{T}(R)$ runs TabICL on the table R. $\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$ replaces all feature columns by the row embedding ($d = 512$). $\mathcal{T}(\text{reduce}(\mathcal{T}_{\text{enc}}(R)))$ then applies a Gaussian random projection ($d' = 32$, see App. D.3). Both Δ are paired differences against the original data of the same seed. The last row averages the ten datasets, so its SEM is taken across datasets rather than across seeds.

	$\mathcal{T}(R)$	$\mathcal{T}(\mathcal{T}_{\text{enc}}(R))$		$\mathcal{T}(\text{reduce}(\mathcal{T}_{\text{enc}}(R)))$	
Dataset	**AUROC**	**AUROC**	Δ	**AUROC**	Δ
bioresponse	87.7 ± 1.0	87.5 ± 0.8	-0.3 ± 0.3	86.1 ± 1.0	-1.6 ± 0.4
churn	92.8 ± 0.8	92.6 ± 0.6	-0.2 ± 0.2	91.4 ± 0.6	-1.4 ± 0.2
compass	72.3 ± 0.7	72.4 ± 0.7	+0.1 ± 0.1	72.3 ± 0.7	0.0 ± 0.1
credit-g	82.2 ± 1.8	83.1 ± 0.4	+0.9 ± 1.4	79.4 ± 0.4	-2.8 ± 1.5
jasmine	89.8 ± 1.1	86.8 ± 0.4	-3.0 ± 0.6	86.5 ± 0.8	-3.3 ± 0.4
kr-vs-kp	100.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0	100.0 ± 0.0	0.0 ± 0.0
philippine	93.6 ± 0.1	84.7 ± 0.6	-8.9 ± 0.6	83.5 ± 0.8	-10.1 ± 0.8
phoneme	97.6 ± 0.0	96.2 ± 0.2	-1.5 ± 0.2	96.2 ± 0.3	-1.4 ± 0.3
qsar-biodeg	92.4 ± 2.1	91.3 ± 2.4	-1.1 ± 0.4	91.1 ± 1.9	-1.3 ± 0.3
spambase	99.1 ± 0.0	98.8 ± 0.0	-0.4 ± 0.0	98.8 ± 0.0	-0.3 ± 0.0
∅	90.8 ± 2.7	89.3 ± 2.6	-1.4 ± 0.9	88.5 ± 2.8	-2.2 ± 0.9

Tab. 11: RelBench databases

Dataset	**Tables**	**Rows**	**Columns**
rel-amazon	3	23,218,245	15
rel-avito	8	24,653,915	42
rel-event	5	44,822,992	131
rel-f1	9	97,606	67
rel-hm	3	16,931,173	37
rel-stack	7	5,399,818	51
rel-trial	15	5,852,157	140

Tab. 12: RelBench's (binary) classification tasks

Dataset	**Task**	**Train**	**Val**	**Test**	**Pos. rate (%)**
rel-amazon	item-churn	2,536,014	177,689	166,842	36.9
rel-amazon	user-churn	4,708,383	409,792	351,885	60.6
rel-avito	user-clicks	59,454	21,183	47,996	1.5
rel-avito	user-visits	86,619	29,979	36,129	85.1
rel-event	user-ignore	19,239	2,013	1,958	13.0
rel-event	user-repeat	3,842	268	246	44.7
rel-f1	driver-dnf	11,411	566	702	70.5
rel-f1	driver-top3	1,353	588	726	17.6
rel-hm	user-churn	3,832,692	76,556	74,575	82.6
rel-stack	user-badge	3,386,276	247,398	255,360	2.6
rel-stack	user-engagement	1,360,850	85,838	88,137	2.7
rel-trial	study-outcome	11,994	960	825	58.5

Tab. 13: RelBench's regression tasks

Dataset	Task	Train	Val	Test	Target (mean ± SD)
rel-amazon	item-ltv	2,707,679	166,978	178,334	77.13 ± 664.61
rel-amazon	user-ltv	4,708,383	409,792	351,885	16.78 ± 55.44
rel-avito	ad-ctr	5,100	1,766	1,816	0.05 ± 0.11
rel-event	user-attendance	19,239	2,013	1,958	0.26 ± 0.64
rel-f1	driver-position	7,453	499	760	11.93 ± 5.21
rel-hm	item-sales	5,488,184	105,542	105,542	0.08 ± 0.58
rel-stack	post-votes	2,453,921	156,216	160,903	0.07 ± 0.37
rel-trial	site-success	151,407	19,740	22,617	0.46 ± 0.48
rel-trial	study-adverse	43,335	3,596	3,098	57.93 ± 249.73

Tab. 14: Tasks the hyperparameter search ran on.

Regression	Classification
`rel-f1/driver-position`	`rel-f1/driver-top3`
`rel-event/user-attendance`	`rel-event/user-repeat`
`rel-trial/study-adverse`	`rel-event/user-ignore`
`rel-trial/site-success`	`rel-trial/study-outcome`
`rel-avito/ad-ctr`	`rel-avito/user-clicks`
`rel-hm/item-sales`	`rel-hm/user-churn`

D.3 Hyperparameter optimization

We ran a simple hyperparameter optimization (HPO) scheme to determine a good hyperparameter configuration for the tasks in RelBench.

Tasks. The search ran independently on six classification and six regression tasks of the 21 RelBench entity tasks (see Tab. 14). The other 9 of the 21 tasks were not used to find a suitable default setting.

Search space. Tab. 15 lists all 11 hyperparameters, their search space, and the value selected for the configuration RelICL reports throughout the paper, unless declared otherwise.

Selection. We chose each hyperparameter on its own rather than taking the best single trial. Within each task, we ranked the trials by validation score, since raw scores are not comparable across tasks (Demšar, 2006; Bardenet et al., 2013). For each value of a hyperparameter, we averaged these ranks over all trials and tasks that used this value. This averages over the other hyperparameters, similar to the marginals of Hutter et al. (2014). We then kept the value with the best mean rank (last column in Tab. 15).

Regression specifics. The target transform was the one exception kept per task type: it is undefined for classification, so it was fixed to `none` there.

Run time. Across the 12 studies, the search consumed 121 GPU-hours in total (1200 completed trials), with a mean trial duration of 361 s (median 303 s). Per-task mean trial duration ranged from 55 s (`rel-avito/ad-ctr`) to 887 s (`rel-hm/item-sales`).

D.4 Configuration ensembling

The reported results in Tabs. 5 and 7 ensembled over configurations. Concretely, this combined per task type a small, fixed set of configuration variants with the *default values*, i.e., the configuration selected by the hyperparameter search (App. D.3).

Configurations. Regression tasks combined the default values with one variant (2 members); classification tasks combined the default values with three (4 members). The same combination was used for every task of a given type: selection below is joint across all tasks of that type, not selected individually per task. Each variant changed exactly one setting from the default values. For regression, the variant adds windowed count features over the last 1, 7, and 28 days. For classification,

Tab. 15: Hyperparameters, their search space, and the value selected for RelICL's reported configuration.

Hyperparameter	**Search space**	**Selected (default values)**
Anchor budget fraction[a]	$\{0.25, 0.75\}$	0.75
Train sample size	$\{16384, 32768, 65536\}$	32768
Context table sample size	$\{32768, 65536\}$	65536
Count features after optimization[b]	{true, false}	true
Count feature windows (days)[b]	$\{\text{none}, [1, 7, 28], [1, 3, 7, 28]\}$	none
Relative time encoding	{true, false}	true
Early-fusion reduction dim.	$\{32, 64\}$	32
Early-fusion reduction method	{GRP[c], PCA, first-k[d]}	GRP[c]
Key pooling[e]	{mean, max, sum}	mean
Outlier clipping threshold[f]	$\{4.0, 8.0\}$	4.0
Target transform[*]	{none, log1p}	log1p / none

[a] Share of the training-row budget reserved for rows sharing a key with the test row.
[b] Adds per-key counts of referencing rows as extra features, optionally split into rolling windows over the last N days.
[c] GRP: Gaussian random projection (Johnson & Lindenstrauss, 1984; Dasgupta & Gupta, 2003).
[d] first-k: keep the first k embedding dimensions, dropping the rest.
[e] How row embeddings sharing a key are aggregated into one.
[f] TabICL specific.
[*] Only defined for regression; fixed to `none` for classification.

Tab. 16: All published models, regression, MAE (lower is better) / R^2 (%, higher is better)

Kind	**Model**	**item-ltv**	**user-ltv**	**ad-ctr**	**user-attendance**	**driver-position**	**item-sales**	**post-votes**	**site-success**	**study-adverse**
Trained	GraphSAGE	50.05 / –	14.31 / –	0.04 / –	0.26 / –	4.02 / –	0.06 / –	0.07 / –	0.40 / –	44.47 / –
Trained	RelGNN	48.77 / –	14.23 / –	0.04 / –	0.24 / –	3.80 / –	0.05 / –	0.07 / –	**0.30** / –	44.46 / –
Trained	RelGT	48.92 / –	14.27 / –	0.04 / –	0.25 / –	3.92 / –	0.05 / –	0.07 / –	0.33 / –	43.99 / –
Trained	DS+LightGBM	**41.12** / –	13.93 / –	0.04 / –	0.28 / –	3.96 / –	0.04 / –	0.07 / –	0.41 / –	40.58 / –
Pretrained (rel.)	RT	– / 32.5	– / 36.9	– / 4.5	–	– / 52.4	– / 14.0	– / 33.9	– / 4.5	– / 2.6
Pretrained (rel.)	RT (target DB)	– / 32.2	– / **38.3**	– / 8.0	–	– / **58.7**	– / 30.9	– / **35.0**	– / 5.1	– / 3.1
Pretrained (rel.)	PluRel	– / **40.5**	– / 18.5	– / 4.9	–	– / 55.5	– / 20.0	– / 25.5	– / **38.6**	– / 1.6
Pretrained (rel.)	KumoRFM-2	46.99 / –	**13.92** / –	**0.03** / –	**0.23** / –	**2.85** / –	**0.03** / –	**0.06** / –	0.43 / –	43.29 / –
Pretrained (tab.)	RDBLearn	48.56 / –	14.54 / –	**0.03** / –	–	–	0.06 / –	0.07 / –	0.42 / –	43.91 / –
Pretrained (tab.)	TabPFN-Rel (API)	46.80 / –	14.40 / –	**0.03** / –	0.24 / –	3.77 / –	0.06 / –	0.07 / –	0.41 / –	39.80 / –
Pretrained (tab.)	TabPFN-Rel	47.80 / –	14.40 / –	**0.03** / –	0.24 / –	3.76 / –	0.06 / –	0.07 / –	0.39 / –	42.60 / –
Pretrained (tab.)	RelICL (TabICL)	43.97 / 28.6	14.78 / 21.0	0.04 / **18.3**	0.24 / 5.1	3.68 / 25.2	0.04 / **69.2**	**0.06** / 25.6	0.40 / -22.0	32.42 / 54.0
Pretrained (tab.)	RelICL (TabPFN)	43.41 / 29.2	15.48 / 16.5	0.05 / 0.1	0.30 / **17.2**	4.00 / 13.4	0.04 / 50.1	0.08 / 17.0	0.43 / -7.1	**29.97** / **56.9**

– Not reported.

the first variant drops the reference count features entirely. The second disables the schema rewrites (App. C.1). The third reduces the embeddings of each backbone estimator separately instead of merging all estimators before reducing.

E Additional results

E.1 All published models

Tabs. 16 and 17 compare RelICL with every model for which we collected published RelBench results.

E.2 Head-to-head wins

Tabs. 18 and 19 count wins pair by pair over the models of Tabs. 5 and 7, and Tabs. 20 and 21 do the same over every published model. Regression compares on MAE where both models report it and on R^2 otherwise, because RT and PluRel report only R^2 while the trained baselines report only MAE. An asterisk marks a cell in which R^2 decided at least one task. A tie counts as a win for

Tab. 17: All published models, classification (AUROC (%), higher is better)

Kind	Model	item-churn	user-churn (ama)	user-clicks	user-visits	user-ignore	user-repeat	driver-dnf	driver-top3	user-churn (hm)	user-badge	user-engagement	study-outcome
Trained	GraphSAGE	**82.8**	70.4	65.9	66.2	81.6	76.9	72.6	75.5	69.9	88.9	90.6	68.6
Trained	RelGNN	82.6	**71.0**	68.2	66.2	86.2	79.6	75.3	85.7	**70.9**	**89.0**	**90.8**	71.2
Trained	RelGT	82.5	70.4	68.3	66.8	81.6	76.1	75.9	83.5	69.3	86.3	90.5	68.6
Trained	DS+LightGBM	81.8	67.6	64.3	64.5	84.2	75.4	69.8	82.4	69.0	86.2	90.3	72.0
Pretrained (rel.)	RT	70.9	64.0	59.5	61.8	–	–	81.2	89.3	62.8	80.1	75.7	51.8
Pretrained (rel.)	PluRel	72.5	65.0	47.9	63.4	–	–	81.0	88.4	66.0	82.0	86.2	51.8
Pretrained (rel.)	KumoRFM-2	82.2	69.1	67.4	**69.4**	**90.8**	**81.7**	**84.6**	**92.2**	69.3	87.2	89.4	72.0
Pretrained (tab.)	RDBLearn	82.1	67.6	**69.0**	65.5	–	–	–	–	68.0	85.3	89.4	71.6
Pretrained (tab.)	TabPFN-Rel (API)	**82.8**	70.9	67.5	66.8	87.9	75.9	73.2	77.1	70.5	88.0	90.6	**76.5**
Pretrained (tab.)	TabPFN-Rel	**82.8**	70.2	61.5	66.9	70.1	76.9	71.5	79.3	70.6	86.4	90.6	73.1
Pretrained (tab.)	RelICL (TabICL)	78.5	65.4	64.3	66.2	88.5	79.3	73.6	81.4	68.4	86.3	90.0	73.4
Pretrained (tab.)	RelICL (TabPFN)	78.4	64.6	58.7	65.7	87.9	77.7	74.2	78.4	67.8	84.9	89.5	73.6

– Not reported.

Tab. 18: Head-to-head wins, regression

		Pretrained (rel.)		Pretrained (tab.)		
Kind	Model	RT	PluRel	RDBLearn	TabPFN-Rel	RelICL
Pretrained (rel.)	RT	–	3 / 8*	–	–	**5 / 8***
Pretrained (rel.)	PluRel	**5 / 8***	–	–	–	3 / 8*
Pretrained (tab.)	RDBLearn	–	–	–	0 / 7	2 / 7
Pretrained (tab.)	TabPFN-Rel	–	–	**6 / 7**	–	4 / 9
Pretrained (tab.)	RelICL	3 / 8*	**5 / 8***	**5 / 7**	**5 / 9**	–

* R^2 decided at least one task of this cell.
⁻ Same model, or no shared task.

neither model, so the two cells of a pair can sum to less than the number of shared tasks. Bold marks a pair the row model wins; ties are underlined.

Tab. 19: Head-to-head wins, classification

		Pretrained (rel.)		Pretrained (tab.)		
Kind	Model	RT	PluRel	RDBLearn	TabPFN-Rel	RelICL
Pretrained (rel.)	RT	–	3 / 10	0 / 8	2 / 10	2 / 10
Pretrained (rel.)	PluRel	**6 / 10**	–	0 / 8	2 / 10	2 / 10
Pretrained (tab.)	RDBLearn	**8 / 8**	**8 / 8**	–	1 / 8	3 / 8
Pretrained (tab.)	TabPFN-Rel	**8 / 10**	**8 / 10**	**7 / 8**	–	6 / 12
Pretrained (tab.)	RelICL	**8 / 10**	**8 / 10**	**5 / 8**	6 / 12	–

– Same model.

Tab. 20: Head-to-head wins against all published models, regression

		Trained				Pretrained (rel.)				Pretrained (tab.)			
Kind	**Model**	**GraphSAGE**	**RelGNN**	**RelGT**	**DS+LightGBM**	**RT**	**RT (target DB)**	**PluRel**	**KumoRFM-2**	**RDBLearn**	**TabPFN-Rel (API)**	**TabPFN-Rel**	**RelICL**
Trained	GraphSAGE	–	0/9	0/9	3/9	–	–	–	1/9	**4/7**	4/9	3/9	1/9
Trained	RelGNN	**8/9**	–	**5/9**	4/9	–	–	–	1/9	**4/7**	**5/9**	**5/9**	4/9
Trained	RelGT	**8/9**	2/9	–	4/9	–	–	–	1/9	**4/7**	4/9	4/9	3/9
Trained	DS+LightGBM	**5/9**	4/9	4/9	–	–	–	–	3/9	**6/7**	**5/9**	**5/9**	3/9
Pretrained (rel.)	RT	–	–	–	–	–	1/8 *	3/8 *	–	–	–	–	**5/8** *
Pretrained (rel.)	RT (target DB)	–	–	–	–	**7/8** *	–	**6/8** *	–	–	–	–	**5/8** *
Pretrained (rel.)	PluRel	–	–	–	–	**5/8** *	2/8 *	–	–	–	–	–	3/8 *
Pretrained (rel.)	KumoRFM-2	**8/9**	**8/9**	**8/9**	**6/9**	–	–	–	–	**5/7**	**5/9**	**6/9**	**6/9**
Pretrained (tab.)	RDBLearn	3/7	3/7	3/7	1/7	–	–	–	1/7	–	0/7	0/7	2/7
Pretrained (tab.)	TabPFN-Rel (API)	**5/9**	4/9	**5/9**	4/9	–	–	–	4/9	**7/7**	–	**5/9**	3/9
Pretrained (tab.)	TabPFN-Rel	**6/9**	4/9	**5/9**	4/9	–	–	–	3/9	**6/7**	3/9	–	4/9
Pretrained (tab.)	RelICL	**8/9**	**5/9**	**6/9**	**6/9**	3/8 *	3/8 *	**5/8** *	3/9	**5/7**	**6/9**	**5/9**	–

* R^2 decided at least one task of this cell.
– Same model, or no shared task.

Tab. 21: Head-to-head wins against all published models, classification

		Trained				Pretrained (rel.)			Pretrained (tab.)			
Kind	**Model**	**GraphSAGE**	**RelGNN**	**RelGT**	**DS+LightGBM**	**RT**	**PluRel**	**KumoRFM-2**	**RDBLearn**	**TabPFN-Rel (API)**	**TabPFN-Rel**	**RelICL**
Trained	GraphSAGE	–	2/12	**7/12**	**9/12**	**8/10**	**8/10**	5/12	**6/8**	3/12	**7/12**	6/12
Trained	RelGNN	**10/12**	–	**9/12**	**11/12**	**8/10**	**8/10**	6/12	**6/8**	**8/12**	**9/12**	**9/12**
Trained	RelGT	5/12	3/12	–	**10/12**	**8/10**	**8/10**	4/12	**6/8**	4/12	5/12	**8/12**
Trained	DS+LightGBM	3/12	1/12	2/12	–	**8/10**	**8/10**	1/12	**5/8**	1/12	3/12	5/12
Pretrained (rel.)	RT	2/10	2/10	2/10	2/10	–	3/10	0/10	0/8	2/10	2/10	2/10
Pretrained (rel.)	PluRel	2/10	2/10	2/10	2/10	**6/10**	–	0/10	0/8	2/10	2/10	2/10
Pretrained (rel.)	KumoRFM-2	**7/12**	6/12	**7/12**	**11/12**	**10/10**	**10/10**	–	**7/8**	5/12	**7/12**	**10/12**
Pretrained (tab.)	RDBLearn	2/8	2/8	2/8	3/8	**8/8**	**8/8**	1/8	–	1/8	1/8	3/8
Pretrained (tab.)	TabPFN-Rel (API)	**9/12**	4/12	**8/12**	**11/12**	**8/10**	**8/10**	**7/12**	**7/8**	–	**8/12**	**8/12**
Pretrained (tab.)	TabPFN-Rel	5/12	3/12	**7/12**	**9/12**	**8/10**	**8/10**	5/12	**7/8**	4/12	–	6/12
Pretrained (tab.)	RelICL	6/12	3/12	4/12	**7/12**	**8/10**	**8/10**	2/12	**5/8**	4/12	6/12	–

– Same model.