Task-Agnostic Contrastive Pretraining for Relational Deep Learning

Jakub Peleška (🖂) and Gustav Šír

Czech Technical University in Prague,
Karlovo náměstí 13, Prague, 121 35, Czechia
`jakub.peleska@cvut.cz, gustav.sir@cvut.cz`

Abstract. Relational Deep Learning (RDL) is an emerging paradigm that leverages Graph Neural Network principles to learn directly from relational databases by representing them as heterogeneous graphs. However, existing RDL models typically rely on task-specific supervised learning, requiring training separate models for each predictive task, which may hamper scalability and reuse.

In this work, we propose a novel task-agnostic contrastive pretraining approach for RDL that enables database-wide representation learning. For that aim, we introduce three levels of contrastive objectives—row-level, link-level, and context-level—designed to capture the structural and semantic heterogeneity inherent to relational data. We implement the respective pretraining approach through a modular RDL architecture and an efficient sampling strategy tailored to the heterogeneous database setting. Our preliminary results on standard RDL benchmarks demonstrate that fine-tuning the pretrained models measurably outperforms training from scratch, validating the promise of the proposed methodology in learning transferable representations for relational data.

Keywords: Relational Deep Learning · Relational Databases · Graph Neural Networks · Self-Supervised Learning · Contrastive Learning

1 Introduction

From their establishment [7], *Relational Databases* (RDBs) have played a pivotal role in ushering our society into the information age. By storing data as interconnected tables safeguarded by integrity constraints, RDBs provide a robust and highly expressive framework for managing structured information. As a result, they remain a cornerstone of critical systems across a broad spectrum of domains, ranging from healthcare [34] to government [25].

Despite their widespread adoption, RDBs are inherently misaligned with conventional Machine Learning (ML) pipelines, which assume data in the standard form of fixed-size, independent and identically distributed (i.i.d.) feature vectors—commonly referred to as the "tabular" learning format. In contrast, RDBs contain multiple interrelated tables of varying sizes, violating this assumption. To bridge the gap, traditional approaches have typically relied on *propositionalization*" [23], which is essentially a feature extraction process that aggregates relational substructures into flattened attributes (features) of the tabular format. However, this transformation inevitably results in a loss of structural and semantic information that is often crucial for relational learning.

Recent advances in *Graph Representation Learning* [16] have enabled an alternative approach. By representing an RDB as a heterogeneous (and potentially temporal) graph, where each row becomes a node and inter-table relationships are captured by edges derived from integrity constraints, it becomes possible to apply the "message-passing" principles of *Graph Neural Network* (GNN)[35] to relational data. This line of work has led to the recent emergence of the *Relational Deep Learning* (RDL) [11] field, which adapts various GNN-based message-passing schemes to the relational setting, achieving some promising initial results across a variety of supervised tasks [10,37,36,27].

However, most existing RDL methods are designed around specific downstream tasks and require training separate models for each, limiting their scalability and reusability in practical, multi-purpose database applications. To address this gap, we propose a novel contrastive pretraining methodology for RDL that enables general-purpose, task-agnostic representation learning over relational databases. At the core of the approach, we introduce a three-level contrastive objective—operating at the row, link, and context levels of the database graph—that captures both attribute semantics and structural dependencies. The resulting pretrained models can then be effectively fine-tuned for diverse downstream tasks, reducing the need for repeated task-specific training. We implement this approach within the ReDeLEx framework [28] and demonstrate its effectiveness on standard relational benchmarks. Our initial experimental results demonstrate that fine-tuning the pretrained models consistently and measurably outperforms training from scratch, validating the effectiveness of our approach in learning transferable representations for relational data, opening doors for exploring a new frontier of foundational database model training. The implementation of the approach is readily available on GitHub.[1]

2 Background

This paper builds on learning from RDBs (Sec. 2.1) with GNN-based models (Sec. 2.2), forming the backbone of the RDL paradigm (Sec. 2.3).

2.1 Relational Databases

Principles of RDBs are formally based on the *relational model* [8], which is grounded in relational logic [12]. This abstraction enables the definition of any database, regardless of specific software implementation, as a collection of n-ary relations, which are defined over the domains of their respective attributes, managed by the Relational Database Management System (RDBMS) to ensure data consistency with the integrity constraints of the database schema. The key concepts to be used in this paper are as follows.

[1] `https://github.com/jakubpeleska/ReDeLEx/tree/develop/experiments/pretraining`

Relational Database A Relational Database (RDB) $\mathcal{R}$ is defined as a finite set of relations $R_1, R_2, \ldots, R_n$. An instance of an RDB $\mathcal{R}$ is implemented through a RDBMS, enabling to perform Structured Query Language (SQL; [4]) operations, rooted in relational algebra.

Relation (Table) Formally, an n-ary relation $R_{/n}$ is a subset of the Cartesian product defined over the domains D_i of its n *attributes* A_i as $R_{/n} \subseteq D_1 \times D_2 \times \cdots \times D_n$, where $D_i = \mathsf{dom}(A_i)$. Each relation R consists of a heading (signature) $R_{/n}$, formed by the set of its attributes, and a body, formed by the values of the respective attributes, commonly represented as a *table* T_R of the relation R.

Attribute (Column) *Attributes* $\mathcal{A}_R = \{A_1, \ldots, A_n\}$ define the terms of a relation $R_{/n}$, corresponding to the *columns* of the respective table T_R. Each attribute is a pair of the attribute's name and a *type*, constraining the domain of each attribute as $\mathsf{dom}(A_i) \subseteq \mathsf{type}(D_i)$. An attribute *value* a_i is then a specific valid value from the respective domain of the attribute A_i.

Tuple (Row) An $n-$*tuple* in a relation $R_{/n}$ is a tuple of attribute values $t_i = (a_1, a_2, \ldots, a_n)$, where a_j represents the value of the attribute A_j in R. The relation can thus be defined extensionally by the *unordered* set of its tuples: $R = \{t_1, t_2, \ldots, t_m\}$, corresponding to the *rows* of the table T_R.

Integrity constraints In addition to the domain constraints $\mathsf{dom}(A_i)$, the most important integrity constraints are the primary and foreign keys. A *primary* key PK of a relation R is a minimal subset of its attributes $R[PK] \subseteq \mathcal{A}_\mathcal{R}$ that uniquely identifies each tuple: $\forall t_1, t_2 \in R : \; (t_1[PK] = t_2[PK]) \Rightarrow (t_1 = t_2)$. A *foreign* key FK_{R_2} in relation R_1 then refers to the primary key PK of another relation R_2 as $\forall t \in R_1 : \; t[FK] \in \{t'[PK] \mid t' \in R_2\}$. This constitutes the inter-relations in the database, with the RDBMS handling the *referential integrity* of $T_{R_1}[FK] \subseteq T_{R_2}[PK]$.

2.2 Graph Neural Networks

Graph Neural Networks constitute a comprehensive class of neural models designed to process graph-structured data through the concept of (differentiable) *message-passing* [35]. Given an input graph $G = (\mathcal{V}, \mathcal{E})$, with a set of nodes $\mathcal{V}$ and edges $\mathcal{E}$, let $h_v^{(l)} \in \mathbb{R}^{d^{(l)}}$ be the vector representation (embedding) of node v at layer l. The general concept of GNNs can then be defined through the following sequence of three functions:

(i) *Message* function $M^{(l)} : \mathbb{R}^{d^{(l)}} \times \mathbb{R}^{d^{(l)}} \rightarrow \mathbb{R}^{d_m^{(l)}}$ computes messages for each edge $(u, v) \in E$ as $m_{u \to v}^{(l)} = M^{(l)}(h_u^{(l)}, h_v^{(l)})$.

(ii) *Aggregation* function $A^{(l)} : \{\mathbb{R}^{d_m^{(l)}}\} \rightarrow \mathbb{R}^{d_m^{(l)}}$ aggregates the messages for each $v \in V$ as $M_v^{(l)} = A^{(l)} \left(\{m_{u \to v}^{(l)} \mid (u, v) \in E\} \right)$.

(iii) *Update* function $U^{(l)} : \mathbb{R}^{d^{(l)}} \times \mathbb{R}^{d_m^{(l)}} \rightarrow \mathbb{R}^{d^{(l+1)}}$ updates representation of each $v \in V$ as $h_v^{(l+1)} = U^{(l)}(h_v^{(l)}, M_v^{(l)})$.

The specific choice of message, aggregation, and update functions varies across specific GNN models, which are typically structured with a predefined number L of such layers, enabling the message-passing to propagate information across L-neighborhoods within the graph(s).

2.3 Relational Deep Learning

In this paper, we adopt the concept of RDL as extending mainstream deep learning models, particularly the GNNs (Sec. 2.2), for application to RDBs (Sec. 2.1). For completeness, in the relational learning community [9], a number of similar approaches combining relational (logic-based) and deep learning methods arose under a similar name of "deep relational learning" [31]. Nevertheless, for compatibility with the recently introduced frameworks [11], we hereby continue with the contemporary RDL view, where RDBs are first transformed into a graph-based representation suitable for the GNN-based learning.

Database Representation The fundamental characteristic of RDL [11] is to represent an RDB as a heterogeneous graph.[2] The graph representation can be defined as $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}^v, \mathcal{T}^e)$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{T}^v$ is a set of node types with a mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}^v$, and $\mathcal{T}^e$ is a set of edge types with a mapping $\psi : \mathcal{E} \rightarrow \mathcal{T}^e$. The node types and edge types collectively form the graph *schema* $(\mathcal{T}^v, \mathcal{T}^e)$.

Given an RDB schema $\mathcal{R}$, the node types $T \in \mathcal{T}^v$ correspond to the relations (tables) T within the database $\mathcal{T}^v \overset{1:1}{\rightarrow} \mathcal{R}$, while the edge types $\mathcal{T}^e$ represent the inter-relations between the tables, as defined by the primary-foreign key pairs: $\mathcal{T}^e = \{(R_i, R_j) \mid R_i[FK_{R_j}] \subseteq R_j[PK] \ \vee \ R_j[FK_{R_i}] \subseteq R_i[PK]\}$. For a specific *instance* of an RDB $\mathcal{R}$, the set of nodes $\mathcal{V}$ is then defined as the union of all tuples (rows) t_i from each relation $\mathcal{V} = \{v_{i,j} \mid R_i \in \mathcal{R},\ t_j \in R_i\}$, and the set of edges $\mathcal{E}$ is defined as $\mathcal{E} = \{(v_{i,k}, v_{j,l}) | \ t_k \in R_i,\ t_l \in R_j, (R_i, R_j) \in \mathcal{T}^e\}$.

The graph representation is further enriched by *node embedding matrices*, *attribute schema*, and optionally a *time mapping*. Node embedding matrix $h_v^{(l)} \in \mathbb{R}^{d \times d_{\phi(v)}}$ contains the embedding representation of a node $v \in \mathcal{V}$ in a given layer l. With an attribute schema $\mathcal{A}_T$ that provides information about the types of attributes $A_1, \dots, A_n$ associated with the nodes v of a specific node type $T \in \mathcal{T}^v$, the initial embedding tensors $h_v^{(0)} \in \mathbb{R}^{d^{(0)} \times n}$ are computed from the raw database attribute tuples $t_i = (a_1, a_2, \dots, a_n)$ through multi-modal attribute encoders [11]. Finally, the *time mapping* is a function τ that assigns a timestamp t_v to each node $\tau : v \mapsto t_v$, effectively creating a dynamically growing graph in time, enabling the use of temporal graph sampling [30].

[2] sometimes referred to as the "relational entity graph"

Predictive Tasks In RDL, predictive tasks are implemented through the creation of dedicated training tables T_t that extend the existing relational schema of $\mathcal{R}$. As introduced in [11], a training table T_t contains two essential components: foreign keys $T_t[FK]$ that identify the entities of interest and target labels $y \in \mathcal{A}_{T_t} \setminus T_t[FK]$. Additionally, timestamps $t_v \in \mathcal{A}_{T_t}$ that define temporal boundaries for the prediction of y can also be included.

The training table methodology supports a diverse range of predictive tasks, including node-level predictions (e.g., customer churn, product sales), link predictions between entities (e.g., user-product interactions), and, crucially, both temporal and static predictions. In the case of temporal predictions, a timestamp attribute t_v in the training table T_t specifies when the prediction is to be made, restricting the model to only consider information available up to the point t_v in time.

Neural Architecture Space Building upon the heterogeneous graph representation G, RDL models generally consist of the following four major stages.

1. **Table-level attribute encoder** creates the initial node embedding matrices $h_v^{(0)} \in \mathbb{R}^{d^{(0)} \times n}$, i.e. sequences of n embedding vectors $\mathbb{R}^{d^{(0)}_{\phi(v)}}$ for each attribute $A_1, \ldots, A_n$ of $\phi(v)$ based on its respective semantic data type.
2. **Table-level tabular model** allows to employ existing tabular learning models [5,19] to yield more sophisticated node embeddings $h_v^{(l)}$. Notably, in this stage, an RDL model *may* reduce the dimensionality of the node attribute matrix embedding $h_v^{(l)} \in \mathbb{R}^{d^{(l)} \times n}$ to a vector embedding $h_v^{(l)} \in \mathbb{R}^{d^{(l)}_{\phi(v)}}$.
3. **Graph neural model** then depends on the chosen embedding dimensionality of $h_v^{(l)}$. If there is a single embedding vector $h_v^{(l)} \in \mathbb{R}^{d^{(l)}_{\phi(v)}}$ per each node, the model can employ standard GNN (Sec. 2.2) heterogeneous message-passing [32,3], otherwise a custom message-passing scheme [27] is required.
4. **Task-specific model head** finally provides transformation of the resulting node embeddings into prediction, usually involving simple MLP layers.

3 Database-Specific Task-Agnostic Pretraining

Prior work in the emerging field of RDL has primarily focused on task-specific models trained using supervised learning [29,27,6]. However, a single relational database often supports a wide range of predictive tasks, spanning node-level classification and regression, link prediction, and temporal forecasting. Adhering to the task-specific paradigm in such settings is both computationally inefficient and operationally burdensome, as each task requires a separate model trained from scratch with its own parameters.

To overcome this limitation, we propose a database-specific, task-agnostic pretraining approach based on self-supervised contrastive learning [2]. Our method learns transferable representations by capturing structural and semantic signals across the database graph, without reliance on downstream labels. Specifically, we introduce a three-level contrastive objective that operates on different granularities of the relational data: individual rows, inter-row links, and contextual neighborhoods. This enables the model to learn rich, reusable embeddings that generalize well across multiple tasks defined on the same database.

3.1 Three-Level Contrastive Pretraining

Contrastive pretraining has become a standard technique in both tabular learning [33,38] and (heterogeneous) graph learning [21,17,24]. Since RDL inherently combines elements from both domains, it is natural to extend contrastive approaches to this setting as well. To that end, we introduce a *three-level contrastive pretraining framework*, designed to capture the heterogeneous structure of relational databases, where each level targets a different structural aspect:

1. *row-level*, capturing standard intra-tabular attribute (feature) patterns,
2. *link-level*, modeling direct inter-tabular referential relationships,
3. and *context-level*, representing broader relational context through wider neighborhoods in the derived database graph.

This multi-level formulation then enables the model to learn general-purpose representations that are both semantically meaningful and structurally aware, supporting a wide range of downstream tasks.

Row-Level Pretraining To learn robust embeddings of individual rows that reflect the heterogeneity across different database tables, we introduce a contrastive pretraining objective based on data corruption. Specifically, for each (sampled) raw database tuple $t_j = (a_1, a_2, \dots, a_n)$ from a relation R_i, we generate a corrupted version $\hat{t}_j$ by randomly selecting a subset of attribute values with uniform probability p. The selected values are then replaced with values drawn from the empirical marginal distribution of the respective attributes, defined as a uniform distribution over the values observed in the training data. This approach is inspired by SCARF [1], but extended to the relational setting.

Importantly, primary and foreign key attributes are excluded from corruption to preserve the integrity constraints of the original database schema. We then follow the standard RDL pipeline (Sec. 2.3) by passing both the original and corrupted tuples, t_j and $\hat{t}_j$, through the RDL model to obtain their respective node embeddings: h_v and $\hat{h}_v$, where $v \sim t_j \in R_i$ denotes the corresponding node.

Given a node v of type $\phi(v)$, we compute the similarity between the original embedding h_v and its corrupted counterpart $\hat{h}_v$, and contrast it against the similarity to corrupted embeddings $\hat{h}_u$ of other nodes u with the same type. This leads to the following InfoNCE-style [14] contrastive loss:

$$\mathcal{L}_v^{\mathrm{row}} = -\log \frac{\exp\left(\hat{h}_v^\top W_{\phi(v)} h_v\right)}{\sum\limits_{u \in \{v\} \cup \mathcal{V}_v^-} \exp\left(\hat{h}_u^\top W_{\phi(v)} h_v\right)}, \tag{1}$$

where $W_{\phi(v)} \in \mathbb{R}^{d \times d}$ is a *learnable* similarity matrix specific to each node type, and $\mathcal{V}_v^- \subseteq \{w \mid \phi(w) = \phi(v),\ w \neq v\}$ is a set of negative samples with cardinality $|\mathcal{V}_v^-| \leq N_{\max}^-$.

Link-Level Pretraining To capture the specific semantics of relationships between rows connected via primary and foreign key constraints, we introduce a contrastive learning objective at the level of individual edges. Each such relationship corresponds to a particular edge type in the graph representation of the RDB (Sec. 2.3).

Given a directed edge $e_{u,v}$ from source node u to target node v with edge type $\psi(e)$, we aim to distinguish the actual linked pair (u, v) from randomly sampled negative pairs. Specifically, we compare the similarity between the true source and target embeddings (h_u, h_v) to the similarity between h_v and other nodes h_w of the same type as u that are not connected to v via an edge of type $\psi(e)$. This leads to the following contrastive loss:

$$\mathcal{L}_{e_{u,v}}^{\text{link}} = -\log \frac{\exp\left(h_u^\top W_{\psi(e)} h_v\right)}{\sum\limits_{w \in \{u\} \cup \mathcal{V}_{e_{u,v}}^-} \exp\left(h_w^\top W_{\psi(e)} h_v\right)}, \tag{2}$$

where $W_{\psi(e)} \in \mathbb{R}^{d \times d}$ is a *learnable* similarity matrix specific to each edge type $\psi(e)$, and $\mathcal{V}_{e_{u,v}}^- \subseteq \{w \mid \phi(w) = \phi(u),\ w \neq u,\ (w, v) \notin \mathcal{E}_{\psi(e)}\}$ is a set of negative samples of cardinality $|\mathcal{V}_{e_{u,v}}^-| \leq N_{\max}^-$.

Context-Level Pretraining While the link-level loss captures local relational dependencies between pairs of nodes, it remains limited to a single type of connection. To model more complex, higher-order structural interactions, we introduce a *context-level* contrastive objective, informed by previous successful works [18,21].

For each node v of type $\phi(v)$, we first compute its *context embedding* c_v as the average of transformed embeddings of its neighboring source nodes:

$$c_v = \frac{1}{|\mathcal{N}_v|} \sum_{u \in \mathcal{N}_v} h_u^\top W_{\phi(u)}, \tag{3}$$

where $\mathcal{N}_v$ is the set of all source neighbors of node v, and $W_{\phi(u)} \in \mathbb{R}^{d \times d}$ is a *learnable* transformation matrix specific to the node type $\phi(u)$. Notably, the transformation matrices used here are distinct from those used in the row-level loss (Eq. 1), allowing the model to specialize its representations for context aggregation.

Next, we compute the similarity between the input node embedding h_v and its aggregated context embedding c_v, and contrast it against similarities with context embeddings of other nodes of the same type. This yields the following context-level contrastive loss:

$$\mathcal{L}_v^{\text{context}} = -\log \frac{\exp\left(c_v^\top h_v\right)}{\sum\limits_{u \in \{v\} \cup \mathcal{V}_v^-} \exp\left(c_u^\top h_v\right)}, \tag{4}$$

where $\mathcal{V}_v^- \subseteq \{w \mid \phi(w) = \phi(v),\ w \neq v\}$ is a set of negative samples with cardinality $|\mathcal{V}_v^-| \leq N_{\max}^-$.

3.2 Pretraining Pipeline

Sampling The sheer scale of many relational databases, which often contain millions of interconnected rows [26,29], makes training on their full graph representations computationally infeasible. This necessitates sampling smaller, tractable subgraphs for training. While neighborhood sampling [15] based on breadth-first search is a common technique for tasks centered on a single node type predictions (e.g., customer churn), it is ill-suited for our pretraining objectives. On heterogeneous graphs, such as those generated from RDBs, neighborhood sampling creates subgraphs with a heavily skewed distribution of node and edge types. This imbalance is problematic for our global, task-agnostic objectives, which rely on a balanced view of the graph's structure (Sec. 3.1). To address this, we employ a variant of HGSampling [20], specifically designed to create subgraphs with an equal representation of node and edge types.

We utilize a single mini-batch for both positive and negative samples; consequently, there is minimal computational overhead associated with the generation of negative samples. More precisely, for the *row-level* loss, we augment each row (Sec. 3.1) in the mini-batch and subsequently take the negative samples for each original row as the augmented rows from the same table, excluding the row corresponding to the original entry. Negative samples for the *context-level* loss are generated in a similar manner, with the exception of replacing the augmentation process with the computation of context embeddings. Lastly, the *link-level* loss negative samples for a given edge type are drawn from the complement of the edges present in the mini-batch.

Combined Loss The proposed contrastive losses are designed to encapsulate the heterogeneous nature of relational databases; however, this very heterogeneity makes it difficult to combine them into a single loss function for training the RDL model. The challenge primarily arises from the varying number of entities (rows) involved in the computation of each loss component. While HGSampling can partially address these discrepancies, a fundamental difference in scale remains between the loss components: the link-level loss is driven by edges (links), whereas the row-level and context-level losses are driven by nodes (rows).

To mitigate the variance between these components, we apply a dynamic normalization factor, μ, based on the number of negative samples utilized for a given input. This factor is defined as:

$$\mu(N^-) = -\log \frac{1}{N^- + 1}, \tag{5}$$

where N^- is the number of negative samples used in the loss computation.

For a graph representation $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}^v, \mathcal{T}^e)$ of a relational database and a sampled subgraph $G' = (\mathcal{V}', \mathcal{E}', \mathcal{T}^v, \mathcal{T}^e)$, we define the combined loss for the subgraph as:

Dataset	Tab	FK	Cols	Rows	Links	Loop	Dom.	Cat	Num	Text	Time
rel-f1	9	13	45	97.6k	227.7k	✓	Sport	16	13	9	7
rel-stack	7	12	33	5.4M	7.5M	✓	Edu.	10	1	13	7

Table 1: Characteristic features of the databases used for experiments.

$$\mathcal{L} = \frac{1}{|\mathcal{V}'|}\sum_{v\in\mathcal{V}'}\frac{\mathcal{L}_v^{\text{row}}}{\mu_v} + \frac{1}{|\mathcal{E}'|}\sum_{e_{u,v}\in\mathcal{E}'}\frac{\mathcal{L}_{e_{u,v}}^{\text{link}}}{\mu_{e_{u,v}}} + \frac{1}{|\mathcal{V}'|}\sum_{v\in\mathcal{V}'}\frac{\mathcal{L}_v^{\text{context}}}{\mu_v}, \tag{6}$$

where $\mu_v = \mu(|\mathcal{V}_v^-|)$ and $\mu_{e_{u,v}} = \mu(|\mathcal{V}_{e_{u,v}}^-|)$.

4 Experiments

In this section we present our initial experimental assessment of the proposed pretraining methodology and discuss its strengths and weaknesses.

4.1 Experimental Setup

We evaluate our method across a range of hyperparameters selected according to standard practices in graph representation learning and previous research in relational deep learning. These hyperparameters influence both the model architecture and the learning process itself. As the overall approach is significantly resource-intensive, we focus our evaluation on two databases from the established RelBench benchmark [29], whose characteristics are detailed in Table 1.[3]

Backbone RDL Models The backbone models used for the experiments follow the RDL blueprint and the outlined neural architecture space, as described in Sec. 2.3. Nevertheless, the backbone models incorporate only the first three stages, thereby omitting the task-specific model head. To ensure comparability across the experiments, we present two models featuring distinct table-level architectures while utilizing the same attribute encoders for numerical, categorical, multi-categorical, textual, and temporal values, as well as the same graph neural model.

GraphSAGE with Linear Transformation This model applies a linear transformation on top of a concatenation of the attribute $a_1, \ldots, a_n$ embeddings $h_v^{(0)} \in \mathbb{R}^{n \cdot d^{(0)}}$ to yield a single embedding vector $h_v^{(1)} = W h_v^{(0)}$ for each node v. The projected node embeddings $h^{(1)} \in \mathbb{R}^{d_{\phi(v)}}$ then form input into the GraphSAGE [15] model, forming the GNN level.

[3] A comprehensive description of the features is available in the Appendix A.3.

GraphSAGE with Tabular ResNet This model is similar to the previous, but it employs a more sophisticated tabular ResNet model [13] for the tabular-level stage to reduce the dimensionality of the node embeddings before passing them to the GraphSAGE layer.

Pretraining In our pretraining experiments, we trained the backbone models by optimizing the proposed three-level contrastive loss (Eq. 6) using the Adam optimizer [22] with a learning rate of 0.001. To prevent data leakage from the validation and test sets, we pruned the database to contain only data available up to the predefined validation timestamp from RelBench.

We sampled the database's graph representation using HGSampling, seeded with a single input node type corresponding to the table with the highest number of foreign keys (i.e., 'results' for 'rel-f1' and 'posts' for 'rel-stack'). HGSampling was configured to sample 64 nodes for each node type over 3 iterations. Internally, we set the maximum number of negative samples per input N^{-}_{max} to a constant value of 256 for all loss levels. All models had 128 internal hidden channels and were evaluated over a hyperparameter grid, that included the number of message-passing layers (2 or 3), the aggregation function (summation or mean), and the cell value corruption probability (0.2, 0.4, or 0.6), as derived from prior work [1].

The models were trained with a batch size of 64 for a maximum of 2000 steps, with a time limit of 4 hours. We employed an early stopping procedure with validation performed every 50 steps and a patience of 10 rounds. Each validation was performed on 50 new samples, randomly sampled from the database using the described HGSampling setup.

Task-Specific Fine-Tuning The pretrained models were further trained on the standardized supervised downstream tasks from RelBench [29], which include entity binary classification and regression. For each task, we equipped the backbone model with a distinct, task-specific head formed by an MLP with a single hidden layer of 128 channels, batch normalization and a ReLU activation function. The model was then trained by optimizing either the Cross-Entropy loss for binary classification or the Mean Squared Error loss for regression, respectively, using the Adam optimizer with a learning rate of 0.0001.

Similar to the pretraining phase, we limited the training to a maximum of 2000 steps and a time limit of 2 hours. We also employed an early stopping procedure, with validation performed every 100 steps and a patience of 5 epochs. For graph sampling, we used Neighborhood Sampling with a batch size of 512 and 128 neighbors per node, with a sampling depth equal to the number of the model's message-passing layers.

Binary Classification								
Model AUC-ROC								
Dataset	**Task**	**Split**	**Linear SAGE**			**ResNet SAGE**		
			B	P	P&F	B	P	P&F
rel-f1	driver-dnf	Val	72.91	74.39	73.25	74.69	74.59	78.35
		Test	75.58	72.93	73.76	78.07	76.57	78.69
	driver-top3	Val	78.02	79.67	78.91	77.40	82.51	82.16
		Test	84.60	79.16	82.24	84.57	82.11	71.37
rel-stack	user-badge	Val	89.35	87.72	89.63	89.29	87.25	89.46
		Test	88.12	85.96	88.51	87.89	85.56	88.21
	user-engagement	Val	90.02	86.93	90.08	89.94	87.06	90.18
		Test	90.34	86.99	90.47	90.28	86.55	90.32
Regression								
Model MAE								
rel-f1	driver-position	Val	3.202	3.227	3.118	3.113	3.208	3.076
		Test	3.686	3.424	3.591	3.329	3.581	3.394
rel-stack	post-votes	Val	0.099	0.104	0.079	0.089	0.102	0.085
		Test	0.105	0.110	0.086	0.095	0.108	0.090

Table 2: Best results on the entity binary classification (AUC-ROC, higher is better) and regression tasks (MAE, lower is better).

4.2 Results

Table 3 presents the main results[4] of our experiments, using the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) metric for the binary classification tasks and the Mean Absolute Error (MAE) metric for the regression tasks, respectively. We evaluated the models in three distinct scenarios to assess the impact of our pretraining method:

1. A *baseline* model (B) trained from scratch on the downstream task.
2. A *pretrained* model (P) where the backbone parameters were *frozen*, and only the task-specific head was trained.
3. A *pretrained* model that was fully *fine-tuned* (P&F) on the downstream task, updating the parameters of both the backbone and the head.

Our results show that on the smaller ‘rel-f1‘ dataset, the frozen pretrained model (P) provides promising performance, performing comparably to the baseline model (B) and even outperforming it on the ‘driver-top3‘ task within a validation split. Nevertheless, on the larger ‘rel-stack‘ dataset, the same frozen models fell behind the performance of the baseline. This might be due to a number of reasons, such as the limited capacity of the model, too short a training period, or a missed aspect of the data, which we discuss further in Sec. 5.

[4] Additional, more granular, results can be found in Appendix A.1.

The pretrained and fine-tuned models (P&F), however, consistently outperformed the baseline models (B) on the validation splits across all tasks. Importantly, these models provided a significant performance boost on the larger 'rel-stack' database, highlighting the ability of our pretraining procedure to learn meaningful and transferable representations, agnostic of the downstream task.

5 Conclusion

In this work, we introduced a self-supervised pretraining method for representation learning on relational databases, building on the emerging paradigm of Relational Deep Learning (RDL). The approach learns effective relational representations by implementing a three-level contrastive objective—at the row, link, and context levels—without the need for downstream supervision. The evaluated RDL models, coupled with type-balanced heterogeneous graph sampling, demonstrated promising results across both classification and regression tasks. Particularly, our findings indicate that pretraining offers measurable benefits, especially in low-data regimes, and generally enhances model performance when fine-tuned, paving the way towards the development of foundational models for relational data.

Future Work Despite these promising initial findings, the scope of our experiments was significantly restricted, and a more thorough study is needed to fully assess the capabilities of the pretraining regime. Notably, pretrained models without the finetuning on the downstream task fell behind in performance on the larger dataset. We hypothesize that this limitation is due to capacity constraints of the model and, in the future, we would like to enhance the experiments with larger Transformer-based models with higher representation dimensionality [27]. Additionally, our pretraining method currently neglects the temporal dimension of relational data, and incorporating time-aware mechanisms is a promising direction for encoding more complex relationships. Finally, extending the pretraining paradigm to support cross-database generalization or continual learning could dramatically improve its usability in real-world applications.

Acknowledgments. This work has received funding from the Czech Science Foundation under the grant No. 24-11664S.

References

1. Bahri, D., Jiang, H., Tay, Y., Metzler, D.: Scarf: Self-supervised contrastive learning using random feature corruption. In: The Tenth International Conference on Learning Representations, ICLR 2022 (2022)
2. Baldi, P., Pineda, F.: Contrastive Learning and Neural Oscillations. Neural Computation **3**(4), 526–545 (Dec 1991)
3. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: International Conference on Learning Representations (2022)

4. Chamberlin, D.D., Boyce, R.F.: SEQUEL: A structured English query language. In: Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control. pp. 249–264. SIGFIDET '74, Association for Computing Machinery, New York, NY, USA (1974)
5. Chen, K.Y., Chiang, P.H., Chou, H.R., Chen, T.W., Chang, T.H.: Trompt: Towards a better deep neural network for tabular data (2023)
6. Chen, T., Kanatsoulis, C., Leskovec, J.: RelGNN: Composite Message Passing for Relational Deep Learning (Feb 2025), arXiv:2502.06784 [cs]
7. Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**(6), 377–387 (6 1970)
8. Codd, E.F.: The relational model for database management: version 2. Addison-Wesley Longman Publishing Co., Inc. (1990)
9. Cropper, A., Dumančić, S., Muggleton, S.H.: Turning 30: New ideas in inductive logic programming. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. pp. 4833–4839 (2020)
10. Cvitkovic, M.: Supervised learning on relational databases with graph neural networks (2020)
11. Fey, M., Hu, W., Huang, K., Lenssen, J.E., Ranjan, R., Robinson, J., Ying, R., You, J., Leskovec, J.: Position: Relational deep learning - graph representation learning on relational databases. In: Forty-first International Conference on Machine Learning (2024)
12. Gallier, J.H.: Logic for computer science: foundations of automatic theorem proving. Courier Dover Publications (2015)
13. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. pp. 18932–18943. NIPS '21, Curran Associates Inc., Red Hook, NY, USA (2021)
14. Gutmann, M., Hyvärinen, A.: Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 297–304. JMLR Workshop and Conference Proceedings (Mar 2010)
15. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp. 1024–1034 (2017)
16. Hamilton, W.L.: Graph Representation Learning. Morgan & Claypool Publishers (Sep 2020)
17. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: Proceedings of the 37th International Conference on Machine Learning. ICML'20, vol. 119, pp. 4116–4126. JMLR.org (2020)
18. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HJlWWJSFDH
19. Hu, X., Tang, W., Hsieh, C.K., Shi, S.: Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)
20. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer (2020)
21. Jiang, X., Jia, T., Fang, Y., Shi, C., Lin, Z., Wang, H.: Pre-training on Large-Scale Heterogeneous Graph. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. pp. 756–766. KDD '21, Association for Computing Machinery, New York, NY, USA (2021)

22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
23. Krogel, M.A., Rawles, S., Železný, F., Flach, P.A., Lavrač, N., Wrobel, S.: Comparative evaluation of approaches to propositionalization. Springer (2003)
24. Li, H., Cao, J., Zhu, J., Luo, Q., He, S., Wang, X.: Augmentation-Free Graph Contrastive Learning of Invariant-Discriminative Representations. IEEE Transactions on Neural Networks and Learning Systems **35**(8), 11157–11167 (Aug 2024)
25. Maali, F., Cyganiak, R., Peristeras, V.: Enabling Interoperability of Government Data Catalogues. In: Wimmer, M.A., Chappelet, J.L., Janssen, M., Scholl, H.J. (eds.) Electronic Government. pp. 339–350. Springer, Berlin, Heidelberg (2010)
26. Motl, J., Schulte, O.: The ctu prague relational learning repository. arXiv preprint arXiv:1511.03086 (2015)
27. Peleška, J., šír, G.: Tabular transformers meet relational databases. ACM Trans. Intell. Syst. Technol. (2025). `https://doi.org/10.1145/3749991`
28. Peleška, J., Šír, G.: REDELEX: A Framework for Relational Deep Learning Exploration (2025). `https://doi.org/10.48550/arXiv.2506.22199`
29. Robinson, J., Ranjan, R., Hu, W., Huang, K., Han, J., Dobles, A., Fey, M., Lenssen, J.E., Yuan, Y., Zhang, Z., He, X., Leskovec, J.: Relbench: A benchmark for deep learning on relational databases. In: The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2024)
30. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.: Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637 (2020)
31. Šír, G.: Deep Learning with Relational Logic Representations. Czech Technical University (2021)
32. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
33. Wang, Z., Sun, J.: TransTab: Learning Transferable Tabular Transformers Across Tables. Advances in Neural Information Processing Systems **35**, 2902–2915 (Dec 2022)
34. White, J.: PubMed 2.0. Medical Reference Services Quarterly **39**(4), 382–387 (Oct 2020)
35. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems (2020)
36. Zahradník, L., Neumann, J., Šír, G.: A deep learning blueprint for relational databases. In: NeurIPS 2023 Second Table Representation Learning Workshop (2023)
37. Zhang, H., Gan, Q., Wipf, D., Zhang, W.: Gfs: Graph-based feature synthesis for prediction over relational databases. arXiv preprint arXiv:2312.02037 (2023)
38. Zhu, B., Shi, X., Erickson, N., Li, M., Karypis, G., Shoaran, M.: XTab: Cross-table Pretraining for Tabular Transformers. In: Proceedings of the 40th International Conference on Machine Learning. pp. 43181–43204. PMLR (Jul 2023), iSSN: 2640-3498

A Additional Details and Results

A.1 Additional Results

Dataset	Task	Split	Linear SAGE B	Linear SAGE P	Linear SAGE P&F	ResNet SAGE B	ResNet SAGE P	ResNet SAGE P&F
Binary Classification								
Model AUC-ROC								
rel-f1	driver-dnf	val	72.420 ± 0.565	73.847 ± 0.488	73.052 ± 0.190	71.635 ± 2.267	74.133 ± 0.382	76.505 ± 1.231
		test	76.533 ± 0.699	75.577 ± 2.040	74.392 ± 2.370	77.782 ± 0.345	75.551 ± 1.094	76.811 ± 2.293
	driver-top3	val	76.511 ± 1.145	79.010 ± 0.532	77.515 ± 0.954	75.568 ± 2.382	79.245 ± 1.899	80.951 ± 0.799
		test	84.115 ± 0.717	82.443 ± 2.441	80.921 ± 4.466	85.213 ± 1.010	82.092 ± 0.809	79.930 ± 4.919
rel-stack	user-badge	val	88.850 ± 0.555	87.278 ± 0.348	89.087 ± 0.399	88.821 ± 0.459	86.970 ± 0.220	89.177 ± 0.317
		test	87.616 ± 0.612	85.696 ± 0.159	87.925 ± 0.448	87.596 ± 0.411	85.263 ± 0.348	88.000 ± 0.236
	user-engagement	val	89.409 ± 0.497	86.639 ± 0.415	89.947 ± 0.168	89.329 ± 0.615	86.295 ± 0.636	90.034 ± 0.262
		test	89.709 ± 0.555	86.347 ± 0.665	90.284 ± 0.237	89.589 ± 0.608	85.916 ± 0.522	90.275 ± 0.316
Regression								
Model MAE								
rel-f1	driver-position	val	3.317 ± 0.181	3.274 ± 0.037	3.175 ± 0.034	3.549 ± 0.536	3.229 ± 0.017	3.101 ± 0.024
		test	3.587 ± 0.196	3.596 ± 0.162	3.554 ± 0.138	3.871 ± 0.567	3.638 ± 0.170	3.469 ± 0.110
rel-stack	post-votes	val	0.107 ± 0.006	0.109 ± 0.006	0.084 ± 0.003	0.108 ± 0.015	0.109 ± 0.004	0.091 ± 0.004
		test	0.113 ± 0.005	0.115 ± 0.005	0.090 ± 0.003	0.113 ± 0.014	0.114 ± 0.004	0.096 ± 0.004

Table 3: Average results with standard deviation of top 5 runs on the entity binary classification (AUC-ROC, higher is better) and regression tasks (MAE, lower is better).

A.2 Computational Requirements

All experiments in Sec. 4 utilized one core of the `AMD EPYC 7742 64-Core` Processor. Figures 1, 2, 3, and 4 display the mean training time for dataset tasks—self-supervised pretraining (blue) and specific downstream tasks (other colors). The bar heights indicate total training time per dataset.

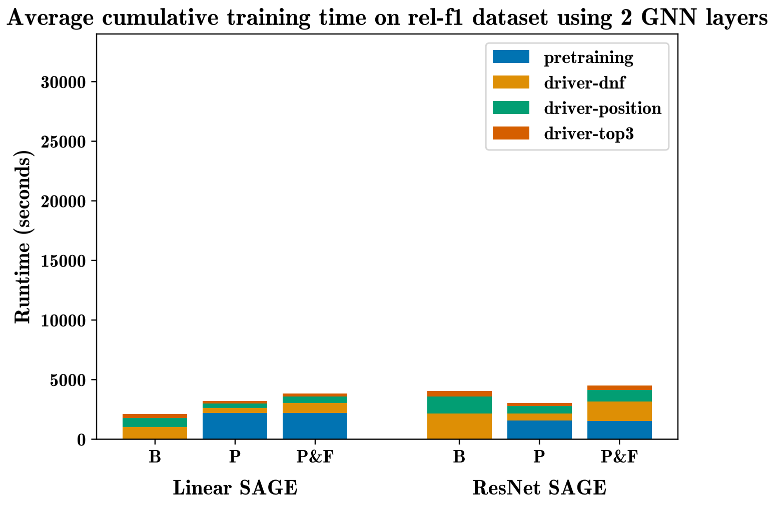

Fig. 1: Average cumulative training time (seconds) on 'rel-f1' dataset using 2 message-passing layers.

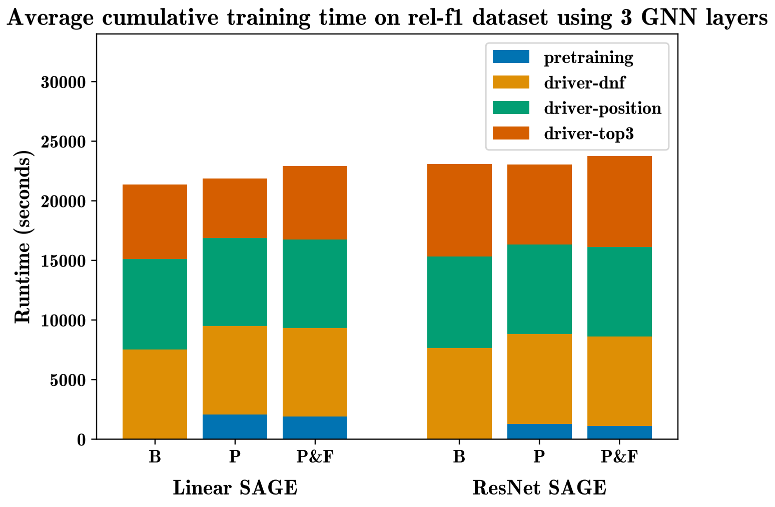

Fig. 2: Average cumulative training time (seconds) on 'rel-f1' dataset using 3 message-passing layers.

The relative time spent on the pretraining appears to be negligible when compared to the time required for training on downstream tasks. The HGSampling [20] employed during pretraining generates significantly less dense subgraphs than neighborhood sampling [15]. We hypothesize that this discrepancy is the primary factor contributing to the fast convergence during pretraining; however, this fast convergence may also arise from the underutilization of the self-supervised pretraining task.

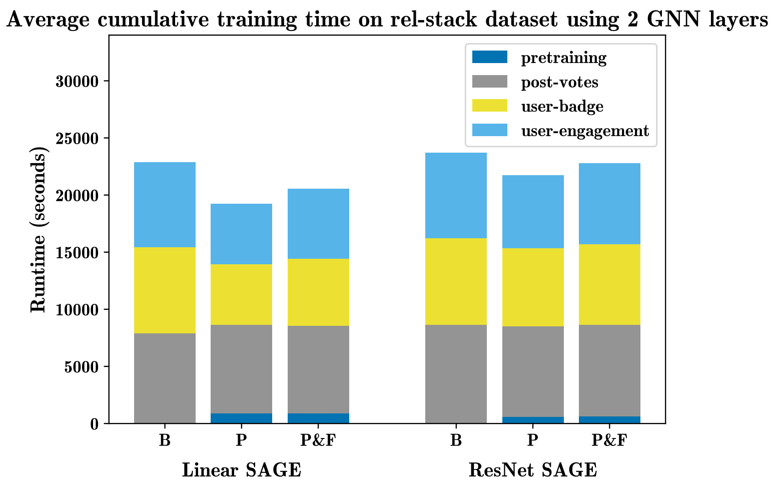

Fig. 3: Average cumulative training time (seconds) on 'rel-stack' dataset using 2 message-passing layers.

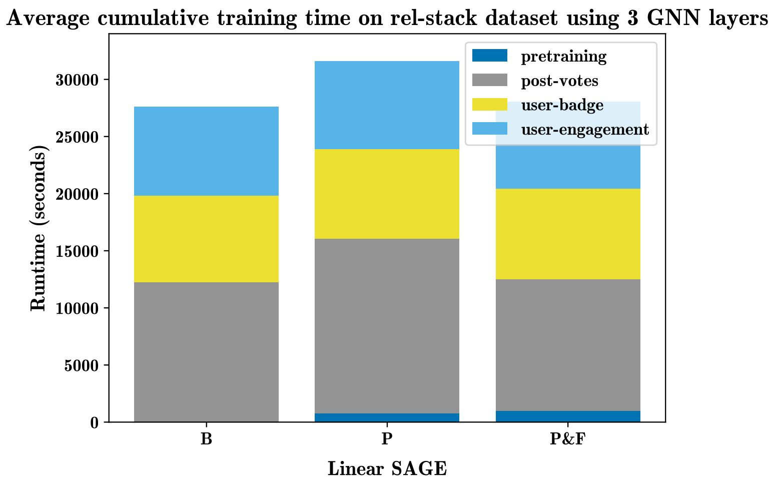

Fig. 4: Average cumulative training time (seconds) on 'rel-stack' dataset using 3 message-passing layers.

A.3 Database Characteristics

A comprehensive description of the features presented in Table 1 is provided below:

- Tab—number of tables;
- FK—number of primary-foreign key column pairs;
- Cols—total number of columns excluding primary and foreign key columns;
- Rows—total number of rows;
- Links—total number of primary-foreign key pairs;
- Loop—indicates whether the database schema contains a cycle;
- Dom—domain of the contained data;
- Cat—total number of categorical columns;
- Num—total number of numerical columns;
- Text—total number of textual columns;
- Time—total number of time columns.