FlexTab: A Flexible Encoder-Decoder Architecture for In-Context Learning Across Diverse Tabular Tasks

Marek Polewczyk[*1] **Maximilian Schambach**[*1] **Marco Spinaci**[*2]
Sam Thelin[1] **Johannes Höhne**[1]

[1]SAP SE [2]SAP France
{firstname.lastname}@sap.com

Abstract

We introduce FlexTab, a flexible encoder-decoder architecture for in-context learning on tabular data that pairs a single, task-agnostic encoder with a suite of task-specific decoders. Unlike existing tabular in-context learners, which entangle feature representations with a specific prediction target, our design produces *target-agnostic* row embeddings that can be leveraged across a wide range of downstream tasks within a table-native in-context learning setup. We demonstrate this flexibility on seven distinct problems: classification, regression, anomaly detection, clustering, entity matching, retrieval, and entity classification in relational databases. Both the encoder and the task-specific decoders are trained on tasks generated from a large corpus of real-world, unlabeled tables. FlexTab achieves state-of-the-art performance on classification, regression, anomaly detection, entity matching, and retrieval, while remaining competitive with specialized models on entity classification in a relational setting. These results demonstrate that a single shared encoder, paired with task-specific decoders, can serve as an effective general-purpose backbone for diverse tabular prediction problems. The inference code and checkpoints are publicly available at: `https://github.com/SAP-samples/flextab`.

1 Introduction

Recently, tabular in-context learners such as TabPFN, TabICL, Mitra, and ConTextTab [26, 27, 50, 51, 68, 61] have been shown to outperform conventional per-dataset trained and tuned models across a range of tabular prediction tasks. They outperform established baselines such as XGBoost, CatBoost, and LightGBM [8, 48, 31], as well as recent state-of-the-art deep-learning approaches such as RealMLP and TabM [28, 22], and even automated machine learning solutions such as AutoGluon [17]. As such, tabular in-context learners have broken the long-standing paradigm of training, tuning, and deploying individual models per dataset or predictive problem. Furthermore, they open up many new possibilities in both research and industry applications, for example, enabling on-the-fly predictions in agentic workflows, scientific discovery, and other fast-iteration scenarios.

However, their development and application have mostly focused on standard supervised prediction tasks, namely classification and regression. While some variations exist, in particular in the TabPFN ecosystem (e.g., embeddings, interpretability, anomaly detection, missing value imputation, data generation, and hypothesis testing) as well as problem-specific adaptations (e.g., causal inference, time-series forecasting, or graph predictions), these are mostly created in a post-hoc fashion by recasting the problem at hand as a single-table classification or regression problem, or require significant architectural adaptations and full retraining. Moreover, many of these community contributions have not yet been thoroughly studied or compared with existing methods in the literature.

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

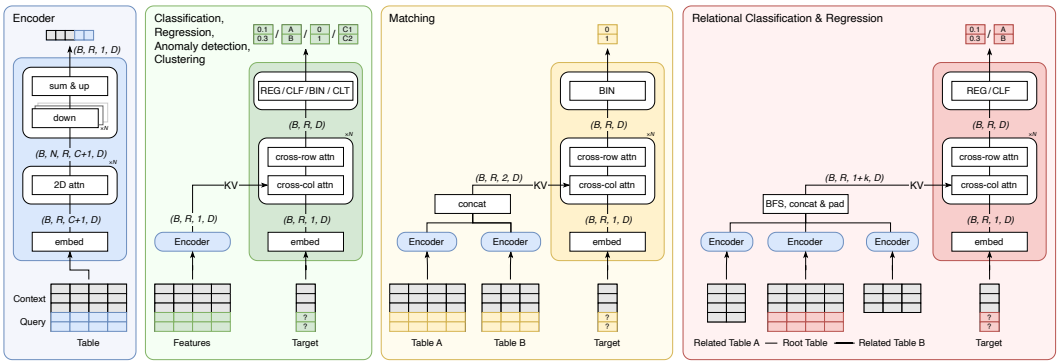


Figure 1: A simplified schematic overview of our proposed architecture, separating an in-context encoder from task-specific decoders with regression (REG), classification (CLF), binary classification (BIN) and clustering (CLT) heads. Note that details such as normalization and feed-forward layers as well as skip connections are omitted.

While effective in many cases, this general approach exhibits the following limitations: (a) the necessity to reformulate every (sub)problem as a single-table classification or regression task or perform significant modifications to an existing model architecture; (b) the tight coupling between internal latent representations and predictive modeling, which causes features to always be conditioned on targets; and (c) the inability to leverage synergies between different tasks.

We argue that these limitations cannot be jointly overcome within the design principle of current tabular in-context learners, due to the inherent entanglement of internal target-aware representations and predictive decoding. To this end, we propose a *flexible* tabular in-context learning architecture, FlexTab, based on an encoder-decoder setup that clearly separates internal latent representations from predictive paths by using a variety of problem-specific decoders. This strictly decouples target-agnostic feature representation from task-specific predictive modeling.

We train our approach on a large collection of real-world tables and evaluate on a diverse set of tasks. Beyond the well-studied classification and regression settings, these include less common tasks:

- **Anomaly detection:** identifying rows in a table that are considered outliers: unsupervised (no labels are provided), one-class (the context rows are all considered inliers), and semi-supervised (only a few rows are labeled as either inlier or anomaly).
- **Clustering:** grouping rows into coherent clusters without access to any labels, so that rows within a cluster are similar to each other.
- **Entity matching:** deciding for pairs of rows from two different tables whether they describe the same real-world entity.
- **Retrieval:** selecting context rows that are most informative for a given query row, which are then used as context for prediction.
- **Entity classification in relational databases:** predicting the target of a row by additionally leveraging related rows in other tables linked via primary-foreign key relations.

Together, these tasks highlight the flexibility of our approach in both single- and multi-table scenarios. We achieve competitive or state-of-the-art performance across all investigated scenarios, and we also highlight the need for better benchmarks and baselines in under-studied areas such as tabular anomaly detection, clustering, and matching.

2 Tabular In-Context Learning

2.1 Background

Tabular in-context learners were primarily pioneered by works on prior-fitted networks and (conditional) neural processes [45, 21], resulting in the first version of TabPFN [26]. Building on its core in-context learning principles, a multitude of works extended and refined TabPFNv1, ranging from changes in the synthetic pretraining priors [50, 68], to incorporating retrieval [39], adapting

the attention mechanism from row-wise token attention to cell-wise alternating 2D-attention [27], improving equivariance [1], accuracy, and speed [50, 51], and handling semantic features [61], to name a few. Most of the prior art focuses on tabular classification and regression tasks. While some works exist that transfer or apply tabular in-context learners to non-tabular domains such as time series forecasting [29], reinforcement learning [57], or the graph domain [24, 15, 16], applications to further tabular tasks remain scarce. Several works have investigated causal inference [54, 3], as well as unsupervised clustering [41, 69], or anomaly detection [40, 65], including very recent preprints. However, these approaches require fundamental changes to the model architecture and/or the pre-training data. We argue that tabular foundation models should natively handle and generalize to a multitude of tabular tasks beyond classification and regression, without the need to recast problems as single-table classification or regression tasks, and without substantial architectural adaptations. While some additional applications are supported in the community-maintained extensions of TabPFN (e.g., anomaly detection) and some may be approached using embeddings extracted from current models (e.g., matching via row embeddings), these approaches have not been experimentally investigated and come with their respective shortcomings. For example, embeddings from current models such as TabPFN, TabICL, or ConTextTab are *target-aware*. Obtaining embeddings from rows or tables without a target (for example, for clustering, matching, or retrieval) requires the use of dummy targets, e.g., by adding random ones, which hurts performance. To overcome these limitations, we propose a flexible architecture, built on the core principles of tabular in-context learning, that natively handles different tabular tasks and can easily be extended to additional ones in the future.

2.2 Proposed Architecture

Our proposed architecture is depicted in Figure 1 in a simplified way, with more details presented in Appendix A. At its core, our architecture separates general (target-agnostic) context and query row embeddings from task-specific decoding. The target decoding is performed by separate task-specific decoders that contextualize the targets with the feature row embeddings via cross-attention. The encoder and decoders are built around 2D-attention blocks, with alternating cross-row (vertical) and cross-column (horizontal) attention mechanisms – analogous to the design of TabPFN, Mitra, and ConTextTab – but separate the target prediction stream into different decoders to make the feature encodings target-agnostic and thus usable across a range of prediction problems.

Encoder: The encoder builds on the general setup of the ConTextTab architecture [61], adopting the same embedding strategy with separate numerical, categorical, and time embedding layers described in Appendix A. Given input tables with batch size B, R rows, and C columns, the data is embedded into a tensor of shape (B, R, C, D), where D denotes the hidden dimension. A learnable `[ROW]` token (replicated across rows) is appended to the embeddings along the column axis, resulting in a tensor of shape $(B, R, C + 1, D)$, similar to `[CLS]` tokens in BERT-like models [13, 37] and TabICL [50, 51], where these tokens are used to generate row-wise (but target-aware) embeddings. After embedding, cell tokens and `[ROW]` tokens are jointly contextualized through alternating cross-column and cross-row attention layers. When the input table consists of both context and query rows, cross-row attention is performed via cross-attention in such a way that context tokens only attend to other context tokens, and query tokens do not attend to each other. Similarly, cross-column attention is structured so that `[ROW]` tokens attend to feature tokens, while feature tokens cannot attend back to the `[ROW]` tokens.

Rather than relying solely on the final-layer representations, we collect the `[ROW]` token outputs from every encoder layer. Each layer's representation is passed through a dedicated linear projection, after which the resulting representations are summed across layers. The aggregated representation is then passed through a shared linear layer, producing the final encoder output. This design is markedly different from conventional encoder-decoder architectures, where typically only the last encoder layer's output is used [63]. We perform thorough ablations on this choice in Section 4.1. In our design, the encoder excludes any target values, resulting in general-purpose, target-agnostic row representations that are used as the cross-attention KV in the task-specific decoders.

Decoders: The different decoders consist of the same 2D-attention blocks as the encoder, with the crucial difference that cross-column attention is realized via cross-attention to the encoder outputs. As in common tabular ICL architectures, query targets are decoded from a learnable `[MASK]` token. The key distinction is that cross-row attention is applied exclusively to the target column, while the encoder's `[ROW]` tokens serve as attention KV inputs to all cross-column attention layers, with the

target column acting as the attention Q. In this way, target tokens attend to the separately encoded features via cross-attention. Again, context tokens can only attend to each other and query tokens can only attend to context. In the main classification and regression decoder, each target attends to a single `[ROW]` token, in which case the cross-attention reduces to a simple linear projection; however, in specialized decoders (e.g., for matching or relational entity classification tasks), the target may attend to multiple tokens that originate from separate tables with potentially different schemas. While not the focus of this investigation, our architecture also naturally extends to incorporate other external embeddings, similar to multi-modal LLMs. For example, the decoder could attend to embeddings produced by text embedding models or other knowledge sources, enabling the decoder to integrate heterogeneous data sources beyond tabular inputs.

Classification and Regression: The main design follows the description above, with separate prediction heads for regression and classification but otherwise shared decoder weights. The classification head is identical to that used in ConTextTab. However, unlike ConTextTab, we treat regression as a classification task and predict logits over quantile bins instead of producing a direct point estimate optimized via an L^2 loss. This simplifies the implementation and training, and enables quantile-based uncertainty estimates. In the default setting, we match the number of transformer blocks N to that of the encoder, but also perform ablations on asymmetric setups in Section 4.1. Additional details on the target embedding and prediction heads are provided in Appendix A.

Anomaly Detection: By definition, anomaly detection is a form of binary classification with imbalanced, noisy, and potentially missing labels. As such, we treat it with the same architecture as classification, but with a decoder trained on binary targets simulating an anomaly detection task. As is customary, we distinguish three different anomaly detection setups [65, 40]:

(1) *Unsupervised*, where context and query are assumed to have the same distribution of normal and anomalous points, but no explicit label is provided for any of the context rows. (2) *One-class* (a.k.a. novelty detection), where the context is known to contain only normal points; and (3) *Semi-supervised*, where context and query have the same distribution, but only a percentage of the anomalies in context are labeled, while the majority of data is unlabeled. We train a single decoder for all three tasks.

Clustering: We convert the above supervised classification framework to an unsupervised clustering one by masking the target column of all rows with `[MASK]`, including the context. The final head layer outputs an embedding vector per row, which is used for clustering.

Matching: We consider cross-table row matching as a pairwise binary classification problem. Row pairs are treated as matching targets, and the two tables are assumed to have the same number of rows. In particular, we do not consider the steps of a full matching pipeline such as blocking [33] for creating pairwise candidate tables, but focus on the prediction step, as is common in the literature. To apply a tabular in-context learning model to pairwise cross-table matching, one could recast the task as a single-table problem by concatenating the rows into one table. However, we propose to first encode each table separately, contextualizing each table independently using only its own rows, and then decode the binary matching target using cross-attention to the two separate row embeddings in the decoder stream, as depicted in Figure 1. The binary classification head is architecturally identical to the default classification head, but with the restriction to two target classes.

Entity Classification in Relational Databases: To further demonstrate the flexibility of our architecture in multi-table scenarios, we examine entity classification tasks in relational databases. At inference time, we first select a root table that contains the target column and split it into context and query rows. For each row, we perform a breadth-first search (BFS) across the database up to a maximum of n hops to identify related rows. Typically, 2 hops are sufficient to reach all tables within the examined databases. The child rows, obtained via primary-to-foreign key relationships, are subsampled and restricted to rows whose timestamps precede the root row. Then, for each table, rows are contextualized and encoded jointly using our encoder, yielding row embeddings for all rows in all tables. Finally, for each row in the task table, we pass the concatenated and padded row embeddings of all rows discovered via BFS to the decoder. The decoder thus learns to identify patterns over sequences of embeddings corresponding to structurally related rows, enabling relational reasoning across tables. A key advantage of this approach is that it decouples representation learning at the table level from relational reasoning at the decoder level. By encoding rows within each table jointly, we obtain high-quality, context-aware embeddings that capture intra-table information. At the same time, the decoder can flexibly attend over a large and diverse set of `[ROW]` tokens, allowing it to aggregate information from multiple tables and relation paths.

3 Training Details and Task Generation

Pretraining: First, we pretrain the encoder together with the classification and regression decoder on conventional tabular prediction tasks, similar to previous tabular in-context learners. We then freeze the encoder and train the additional task-specific decoders on top using their respective task generation and loss objectives. We do this because the generation of classification and regression tasks for large-scale training of tabular in-context learners is the most well-studied and has proven effective, including using real-world data instead of synthetically generated tasks [39, 61, 38]. In principle, it is also possible to pretrain the encoder in a self-supervised fashion separately, or to train the full architecture end-to-end. Throughout, for training, we use a large corpus of ca. 300 k unlabeled tables from an internal, proprietary dataset compiled from different public sources, from which we generate the decoder-specific tasks. We report more details and perform a contamination study of this corpus in Appendix A.6. By default, we use 12 layers and a hidden dimension of 768, with 12 heads of 64 dimensions, and a feed-forward dimension of 3072, for a total of ca. 175 M encoder parameters. We train on 8 H100 GPUs, using a micro-batch size of 1 and accumulating gradients over 256 micro-batches, with the AdamW optimizer and a learning rate of 10^{-4}. We run the initial pretraining for approximately 30 M steps (roughly 7 days), and the training of the additional decoders takes between 7 M and 15 M steps each, depending on the task.

Classification and Regression: We generate classification and regression tasks from randomly sampled real-world tables in a similar fashion to prior work [61]. At each step, we randomly select 1000 rows, then sample between 50 and 900 rows as queries, and use the rest as context. Subsequently, we randomly sample feature and target columns. For both classification and regression, we use a cross-entropy loss.

Anomaly Detection: Anomaly detection tasks are sampled similarly to classification tasks, with a single column selected as target. If this column is numeric, it is transformed to categorical by selecting "optimal" cuts via K-means, with $2 \leq k \leq 10$ chosen via incremental goodness-of-variance-fit thresholding. Then, we map each category to binary (0 = normal data, 1 = anomaly) by marking rarer categories as anomalies; the fraction of anomalies is randomly chosen, not exceeding 40 % of the rows. To mimic different types of anomalies in real applications, we use multiple techniques to split the data into context and query: random splitting, selecting only normal points as context, or excluding some anomaly classes from the context. While the model output is binary, context rows have an additional label 2 for unknown labels (in unsupervised or semi-supervised settings).

Clustering: The sampling strategy follows that of the classification tasks, but target values are completely masked. The decoder outputs per-row embedding vectors tuned for unsupervised clustering. During training, we compute the cosine similarity between such vectors and apply element-wise binary cross-entropy loss to these similarities. For inference, we apply standard K-means on the embedding vectors. More details on the data sampling and training loss are provided in Appendix A.5.

Matching: To generate matching tasks from real-world tables, we use two equally weighted strategies: a same-row matching construction and a hidden target-based matching construction. For the former, we create synthetic matching pairs by splitting a randomly drawn table into two with a random overlap. We then shuffle the tables. Pairs that originate from the same original row are considered matches; others are considered non-matches. For the second approach, we do the same but consider rows as matching if they agree on a held-out target column. We describe this sampling procedure in more detail in Appendix A. The training loss is a cross-entropy loss on the predicted binary targets.

Entity Classification in Relational Databases: To simulate the incorporation of information from multiple tables, we generate examples using column-wise partitions of single tables. At each training step, we draw a table and split it into a variable number of chunks, with each chunk containing at least two columns. These chunks are batched and processed independently by the encoder, producing separate `[ROW]` embeddings for each chunk. As a result, each original table row is represented by multiple `[ROW]` tokens, and this setup enables the decoder to learn how to aggregate information from multiple tables to predict the target, which is selected in the same way as for the standard classification and regression decoder. To regularize pretraining, we apply perturbations: 20 % of `[ROW]` embeddings are randomly duplicated within each row, 20 % are randomly zeroed out, and Gaussian noise is added to 20 % of the embeddings.

Table 1: Classification and regression performance, depicting mean accuracy (Acc) for classification and (soft-clipped) R^2 score for regression tasks, in percent, as well as mean per-task rank.

Model	All	CARTE			TabArena-Lite			TALENT-Tiny			TextTab		
	Rank	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2
AutoGluon	N/A	N/A	78.8	74.0	N/A	88.6	80.7	N/A	88.9	84.7	N/A	83.5	67.0
FlexTab [bag=8] (ours)	**2.7**	**1.6**	**77.3**	**74.4**	3.6	87.1	79.5	3.2	88.0	**85.8**	**2.8**	83.8	62.7
FlexTab [bag=1] (ours)	3.2	2.3	77.0	73.6	3.8	87.0	79.1	3.4	87.9	85.5	3.2	83.7	61.4
RealMLP	3.7	4.1	74.3	69.0	3.4	88.3	79.9	3.4	**88.6**	85.0	3.8	82.3	**67.7**
ConTextTab	3.7	3.0	77.1	72.4	4.2	87.6	77.8	4.0	87.6	83.2	3.8	**84.4**	58.8
TabPFNv2.6	3.8	6.3	70.7	59.9	**1.9**	**88.7**	**80.5**	2.5	88.4	84.9	4.5	81.5	63.8
TabICLv2	3.9	6.5	70.5	55.3	2.2	**88.7**	79.9	**2.2**	88.3	85.3	4.1	82.5	60.2
CatBoost	4.0	4.3	76.3	68.3	3.8	88.2	79.0	4.2	87.1	83.5	3.3	83.7	65.4
Random forest	6.0	6.8	71.4	63.4	5.5	87.6	76.1	5.9	85.6	78.0	5.7	79.8	60.7

4 Results

Classification and Regression: We evaluate on a mix of conventional, numeric-heavy benchmarks as well as semantic ones. For the conventional benchmarks, we evaluate on the single-fold TabArena-Lite [18] and TALENT-Tiny [67]. For the semantic ones, we use CARTE [32] and TextTab [43]. Throughout, we report mean accuracy and soft-clipped R^2, as well as mean rank. We compare against a range of best-in-class baselines, including SOTA in-context learners (TabPFNv2.6, TabICLv2, and ConTextTab), as well as per-dataset trained ones (CatBoost and RealMLP), tuned and ensembled via 5-fold inner cross-validation. Further, we compare against the AutoGluon system, using its "extreme" preset, which itself stacks many of the previous models. We provide additional evaluations (win-ratios, Elo scores, and critical difference diagrams) in Appendix B and more information on the baselines in Appendix A.7. Furthermore, we refer to Appendix B.7 for an analysis of the model runtime in comparison to other baselines.

The main results, evaluating FlexTab (with and without ensembling) and baselines, are shown in Table 1. We exclude AutoGluon from the overall rank comparison due to its stacking nature, making it difficult to highlight architectural differences. Overall, our model performs competitively across non-semantic benchmarks while setting a new standard on the semantically rich CARTE and TextTab benchmarks, where it ranks best. This highlights the general efficacy of our approach. Crucially, compared to TabPFN and ConTextTab, this shows that feature context compression into single row tokens can achieve state-of-the-art results. We note that this difference is statistically significant in the case of semantic-heavy benchmarks, as shown in the critical difference diagrams in Appendix B.1.

Anomaly Detection: We evaluate on the popular ADBench benchmark [23]. We select up to 10 000 points per dataset and create context and query splits following the same procedure as in ICLAD [65]. We use AUROC as the main metric. In addition to top-performing models from ADBench, we include the foundation model FoMo-0D [59] (unsupervised and one-class) and the contemporary TACTIC approach [40]. In the semi-supervised setting, we found that the strongest baselines are in-context learning classification models (e.g., TabPFNv2.6 and TabICLv2). Results are reported in Table 2 for the unsupervised and one-class evaluations and in Table 3 for the semi-supervised case. In this last setting, we also include "semi-supervised novelty detection" [5], with the same annotated inliers as one-class, but where the remaining data is also available to the model.

Unsupervised: Our anomaly-detection-tuned variant, FlexTab-AD, shows better performance than most baselines, only slightly behind the recent SOTA TACTIC model, highlighting the effectiveness of our architecture for this task. However, based on the critical difference diagram reported in Appendix B.2, we come to the same conclusion as discussed in the original paper [23]: in the unsupervised setting, virtually all current methods are statistically indistinguishable. This calls for both continued model research and, equally important, the curation of more nuanced and diverse benchmarks.

One-class: The performance of FlexTab-AD is competitive with all baselines, surpassing the recent TACTIC. However, the classical KNN approach remains statistically not worse than any of the more modern approaches.

Table 2: Unsupervised and one-class anomaly detection: mean rank (Rk) and AUROC (in %).

Model	Unsup.		One-Class	
	Rk	Avg	Rk	Avg
KNN [52]	4.6	69.7	**3.3**	81.4
FlexTab-AD	4.5	73.8	3.7	**82.0**
TACTIC [40]	**3.9**	**74.9**	4.7	80.3
LOF [6]	5.4	64.3	4.3	79.7
OCSVM [58]	4.9	71.4	5.2	76.3
IForest [36]	4.5	72.9	5.8	75.9
PCA [60]	5.1	69.5	5.3	76.6
ECOD [35]	5.1	71.2	7.2	70.6
FoMo-0D [59]	7.0	58.0	5.4	78.3

Table 3: Semi-supervised novelty and semi-supervised anomaly detection: mean rank (Rk) and AUROC (in %) for different levels of labeled anomaly ratios.

Model	Novelty		1%		5%		25%		100%	
	Rank	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank	Avg
TabICL [51]	3.1	87.9	3.4	82.6	3.0	**88.0**	**2.5**	**91.9**	**2.2**	**94.4**
FlexTab-AD	2.7	**90.7**	**2.7**	**83.6**	**2.9**	87.3	3.5	90.7	4.9	93.2
TabPFN [27]	**2.6**	90.2	4.9	78.4	4.2	85.9	3.4	90.8	2.8	94.0
FlexTab	5.6	80.8	3.1	82.7	3.3	87.4	3.6	91.1	3.9	93.6
ConTextTab [61]	5.2	85.1	3.7	81.7	3.8	86.9	4.1	90.9	3.7	93.6
XGBOD [70]	5.8	84.3	6.1	75.7	5.9	83.2	5.9	89.2	5.4	93.2
DevNet [46]	5.6	85.7	6.9	69.3	6.9	77.0	6.7	85.4	6.7	91.0
DeepSAD [56]	6.4	83.3	6.9	72.5	7.5	77.7	7.6	83.7	7.4	89.2
FEAWAD [71]	8.0	77.1	7.4	67.9	7.7	75.2	7.7	83.9	8.0	89.3

Semi-supervised novelty detection: When the unlabeled points are presented to the model, one can apply classical supervised methods effectively, and most resulting algorithms vastly outperform one-class approaches, coherently with the results in [5]. Our dedicated FlexTab-AD achieves the overall best performance in this setting.

Semi-supervised: All in-context learning models significantly outperform the traditional baselines. FlexTab-AD is the best-performing method when very few anomalies are labeled, but models trained for supervised classification perform better once enough labeled rows are available. As shown in the critical difference diagram in the Appendix, TabICLv2 significantly outperforms other supervised methods such as TabPFN and ConTextTab. However, FlexTab natively handles all three investigated cases, whereas current in-context learners can natively be applied only to the semi-supervised case.

Clustering: While our approach is generally effective, we defer a thorough evaluation and discussion to Appendix B.3 due to training and validation instabilities of our approach as well as when reproducing SOTA baselines such as TabClustPFN [69].

Matching: For benchmarking, we use a total of five matching datasets from the literature. Note that most public matching datasets cannot be considered tabular but are rather free-text entity matching tasks where cross-row statistics or context do not play any role. Hence, we focus on tabular matching tasks with at least two columns of relevant information that may benefit from contextualization. In particular, we evaluate on the synthetic Febrl4 dataset [10], as well as Fodors-Zagats, Bikes, eBooks, and Movies sourced from the DeepMatcher study [44] and the Magellan Data Repository [12], which we found to be the most structured. Due to the relatively small number of labeled examples, we perform 5-fold cross-validation and report average match-class F1 scores.

We compare our approach against tabular in-context learners by recasting cross-table matching as a single-table binary classification problem, as well as native matching baselines that are trained per dataset. We report the Hybrid DeepMatcher variant [44] as well as the BERT-based DITTO [34], but show additional ones in Appendix B.4. For completeness, we also compare against simple embedding-similarity-based approaches, i.e., by using the cosine similarity of the two row embeddings binarized with a threshold tuned on each dataset's (and each fold's) train split.

The results are depicted in Table 4. We observe that all tabular in-context learners, when we recast matching as a single-table problem, perform overall quite poorly. While TabPFN and TabICL seem to perform reasonably well on the synthetic exact-match Febrl4 task, their performance varies considerably on more semantically heavy tasks, such as Fodors-Zagats or Movies. ConTextTab and FlexTab, which natively handle semantic columns, show slightly more robust performance, but overall fall behind even simple baselines such as the n-gram-based embedding similarity matching. With a dedicated matching decoder, trained on matching tasks, we can see that FlexTab-Match performs very strongly. Our in-context approach even outperforms dedicated per-dataset-trained deep learning baselines such as those from the DeepMatcher study or DITTO. Overall, our approach performs best in terms of average F1 and ranks almost on par with DITTO, highlighting its effectiveness.

Moreover, we observe that the embeddings obtained from our encoder clearly outperform those from TabPFN when used for matching, highlighting the benefit of the general-purpose, target-agnostic row

Table 4: Matching benchmark F1 scores (in %, positive class) obtained as 5-fold cross-validation averages for each dataset. The average per-group rank is denoted as GRk whereas the average rank across all investigated models is denoted as Rk.

		Benchmark					Average		
	Model	Febrl4	F.-Zagat	Bikes	eBooks	Movies	F1	GRk	Rk
In-context learners	TabPFN	97.3	29.4	81.8	67.9	0.0	55.3	3.2	7.2
	TabICL	95.0	36.9	82.4	73.1	0.0	57.5	2.8	6.8
	ConTextTab	83.2	23.2	82.3	60.3	79.0	65.6	3.6	6.8
	FlexTab (ours)	79.8	12.7	84.3	72.6	61.0	62.1	3.6	7.0
	FlexTab-Match (ours)	**99.3**	**92.5**	**86.8**	**87.3**	**90.2**	**91.2**	**1.0**	**2.2**
Embedding similarity	NGramHash Embeddings	**99.7**	**88.0**	**69.8**	**88.1**	22.9	73.7	**1.4**	**4.4**
	TabPFN Embeddings	71.8	41.2	50.8	56.4	24.0	48.9	2.8	8.6
	FlexTab Embeddings (ours)	98.5	81.5	54.4	84.9	**78.9**	**79.6**	1.8	5.4
Per-dataset trained	Deepmatch Hybrid	99.8	87.1	87.7	88.5	**87.6**	**90.2**	1.6	2.2
	DITTO	**100.0**	**94.7**	**95.5**	**89.5**	44.9	84.9	**1.2**	**2.0**

embeddings produced by our architecture. Nevertheless, for many matching tasks, simple n-gram hash embeddings perform very well, while their performance strongly degrades on others (e.g., Bikes and Movies). Overall, we can already achieve reasonable results with simple cosine-similarity-based matching using FlexTab's row embeddings, which can be useful in large-scale matching scenarios. However, to achieve the best performance, the native matching decoder is preferred.

We show extended results, including all DeepMatcher baselines, as well as additional embeddings and combining TabPFN and TabICL with TF-IDF features, in Appendix B.4. There, we also ablate the pretraining strategy used, creating synthetic matching tasks from real-world tables.

Entity Classification in Relational Databases: As the de facto standard in relational prediction benchmarking, we evaluate our model on the 12 RelBench [55] entity classification tasks, following a setup similar to that of established multi-table foundation models such as Griffin [64] and Relational Transformer [53], and report mean AUROC in Table 5. We additionally include results for KumoRFM [19], a SOTA closed-source industry model, as well as the recent RDBLearn approach [66]. Aside from ours, we report results taken from the respective publications and KumoRFM-2 [30].

Our model's performance is on par with that of Relational Transformer, showcasing the competitiveness and flexibility of our approach relative to recent architectures that are purpose-built for relational prediction. Note that, unlike the other investigated relational-native approaches, ours is pretrained exclusively on chunks created from single tables, not on actual relational data. By investigating FlexTab's performance with a varying number of auxiliary rows in Appendix B.5, we verify that a growing number of row tokens from related tables and larger context size monotonically improve performance, showcasing that the decoder can successfully leverage information contained in tokens from additional auxiliary tables.

The recent RDBLearn approach [66], which uses a custom feature engineering technique, aggregating features across the relational tables and performing its prediction using a single-table in-context learner, demonstrates that relational-native models still do not show a clear gain over this simpler solution. However, this approach is primarily applicable to databases where such aggregation preserves the relevant signal, and may not generalize to settings where textual values across different tables must be jointly considered. Overall, the field of database-native predictive models as well as benchmarking remains an open and active research area, requiring continued work on table encoding [42] and relational data modeling [14].

Retrieval: To more directly demonstrate the meaningfulness of FlexTab's embeddings, we conduct retrieval experiments and report evaluation results in Table 6 (and in more detail in Table 12 in Appendix B) across varying context sizes and retrieval methods. In the default setting, we randomly sample training rows as context and show that performance improves as the context size grows. An alternative is a specialized retrieval method in which we first embed all rows with the FlexTab encoder by selecting k random training rows as context and obtaining the row embeddings for all training and test rows. For each test row, we then retrieve the top k context rows based on the L2 distance between embeddings (similar to existing approaches [39, 62]). Having retrieved the best training rows for each

Table 5: Entity classification in relational databases prediction performance: Zero-shot test AUROC (%) for 12 RelBench binary classification tasks. The target task is never seen during pretraining.

	f1		avito		event		trial	amazon		stack		hm	
Model	dnf	top3	clicks	visits	repeat	ignore	out	user	item	eng	badge	churn	Mean
Griffin	57.7	82.5	45.9	60.7	71.9	83.3	51.0	62.3	69.0	77.5	73.5	60.2	66.29
RT	81.2	89.3	59.5	61.8	73.2	77.5	51.8	64.0	70.9	75.7	80.1	62.8	70.65
FlexTab-Multi (ours)	74.6	83.3	51.9	58.5	**77.4**	79.2	65.1	62.1	76.5	81.6	80.6	60.3	70.93
RDBLearn	70.9	79.7	**69.0**	**65.5**	75.0	82.5	**71.6**	**67.6**	**82.1**	**89.5**	**85.4**	**68.1**	75.58
KumoRFM-1	**82.4**	**91.1**	64.9	64.1	76.1	**89.2**	70.8	67.3	79.9	87.1	80.0	67.7	**76.72**

Table 6: Influence of embedding methods on retrieval (top) and results of probing the embeddings via KNN (bottom), measured in terms of classification and regression scores.

Model Name	All	CARTE			TabArena			TALENT			TextTab		
	Rank	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2
Influence of embedding methods on retrieval													
FlexTab [k=1024, flextab-emb]	**1.7**	**1.9**	77.3	**75.5**	**1.4**	**87.6**	**80.4**	2.1	87.7	**86.3**	**1.5**	84.2	70.0
FlexTab [k=1024, column-hash]	1.9	2.3	77.2	75.1	1.7	86.7	80.2	**1.8**	**88.0**	86.2	1.8	84.1	**70.3**
FlexTab [k=8192, random]	2.2	2.8	77.0	73.6	1.8	87.0	79.1	1.8	87.9	85.5	2.6	83.7	61.4
FlexTab [k=1024, row-hash]	2.5	2.4	**77.7**	75.0	2.4	86.3	78.1	3.0	86.4	81.6	2.0	**84.3**	68.6
Probing the embeddings via KNN													
FlexTab [k=8192, random]	**1.1**	**1.0**	**77.0**	**73.6**	**1.2**	**87.0**	**79.1**	**1.0**	**87.9**	**85.5**	**1.2**	**83.7**	**61.4**
KNN-5 [flextab-emb]	2.2	2.1	72.2	63.2	2.4	84.9	67.2	2.2	82.8	72.1	2.2	78.6	51.8
KNN-5 [column-hashing]	2.9	3.4	67.1	51.2	2.6	82.2	66.7	2.7	78.5	68.7	3.0	76.4	45.0
KNN-5 [row-hashing]	3.5	3.4	66.7	48.9	3.4	77.4	18.7	3.9	56.0	1.2	3.5	76.7	28.3

query, we use them as context to make the prediction. We find that retrieving just 1024 dedicated rows per query, via FlexTab embedding similarity, already yields better scores than selecting 8192 random rows.

In addition, in Table 6, we show that retrieval with FlexTab embeddings outperforms the specialized retrieval based on other embedding methods. We compare against two schemes: a hashing vectorizer applied to the serialized row, and a hashing vectorizer applied to each text column combined with a standard scaler for numerical columns, where the vectorized columns are concatenated into the final embedding. This confirms that the general-purpose, target-unaware row embeddings of FlexTab effectively represent the underlying row's content. Furthermore, we directly probe the embeddings via KNN and evaluate the final predictions. We show that KNN-5 over embedded rows using the FlexTab encoder performs significantly better than KNN-5 over embedded rows using the hashing and standardization methods described above, highlighting the quality of FlexTab's general-purpose embeddings.

4.1 Architecture Ablations

We conduct two ablation studies to assess our design choices, focusing on (a) the strategy used to aggregate encoder representations and (b) the depth of the decoder. Due to the strong availability of high-quality benchmarks for classification and regression, we perform the studies on those. Results are reported in Table 7, using the same benchmarks as previously. For simplicity, we evaluate FlexTab in the non-ensembled variant here.

Encoder Aggregation: We compare five strategies: combining projected encoder representations at dimensions 768, 512, and 256; a simple mean aggregation over encoder layers; as well as using only the last encoder layer output directly. In all cases, aggregations are performed over the contextualized `[ROW]` embeddings as previously described. Surprisingly, when we used only the output of the last layer, as is common in the original Transformer encoder-decoder setup [63], we observed training collapse after roughly 2 M steps. We then investigated the different aggregation strategies. The learned aggregations, using individual projection layers, clearly outperform the non-parametric alternatives. The 768-dimensional variant achieves the best overall rank. Mean pooling across encoder layers

Table 7: Performance across two ablation groups compared to the base model: (Top) Encoder row-level embedding aggregations; (Bottom) Encoder sizes

	Model	All	CARTE			TabArena-Lite			TALENT-Tiny			TextTab		
		Rank	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2
Encoder Aggregation	FlexTab (dim 768)	**1.7**	**1.5**	**77.0**	**73.6**	**1.8**	87.0	**79.1**	**1.7**	**87.9**	**85.5**	1.8	83.7	61.4
	FlexTab (dim 512)	1.8	2.0	76.1	73.3	1.8	86.9	77.1	1.7	87.7	85.5	**1.5**	**84.0**	**61.9**
	FlexTab (dim 256)	2.3	2.8	76.2	72.8	2.0	**87.5**	78.7	2.2	87.2	85.2	1.9	83.5	61.5
	FlexTab (mean)	2.9	3.0	75.1	72.4	2.5	86.2	78.6	3.1	86.1	84.2	3.3	82.6	59.0
	FlexTab (last enc layer)	Training collapsed												
Decoder Size	FlexTab (12 layers)	**1.5**	1.8	76.1	73.3	**1.4**	86.9	**77.1**	**1.3**	**87.7**	**85.5**	**1.2**	**84.0**	**61.9**
	FlexTab (8 layers)	1.5	**1.5**	**76.7**	**73.6**	1.5	**87.3**	76.9	1.7	87.1	85.0	1.4	83.9	61.4
	FlexTab (4 layers)	2.4	2.4	76.5	72.4	2.3	87.0	74.7	2.5	85.7	82.8	2.5	83.4	58.9

noticeably lags behind, indicating that uniform weighting discards useful layer-specific information. Overall, this suggests that intermediate layers carry beneficial complementary signals.

Decoder Size: Next, we vary the decoder depth among 4, 8, and 12 layers while keeping the encoder fixed, investigating asymmetric setups. The 12- and 8-layer variants are essentially tied in overall rank. Shrinking the decoder to 4 layers causes a clear drop in performance. This gap highlights the need for adequate decoder capacity to handle both classification and regression effectively, but also demonstrates the flexibility of the approach to scale the decoder capacity independently of the encoder. We adopt the 12-layer decoder as our default, given its consistency across benchmarks, but note that the 8-layer variant offers a competitive and more efficient alternative.

5 Conclusion and Outlook

We presented FlexTab, a flexible encoder-decoder architecture for in-context learning on tabular data. By decoupling a shared, target-agnostic encoder from a family of task-specific decoders, FlexTab natively supports a broad spectrum of tabular problems – including classification, regression, anomaly detection, clustering, entity matching, and entity classification over relational databases – within a single unified framework. The entire model is trained on tasks generated from a large corpus of real-world, unlabeled tables, avoiding reliance on synthetic priors or task-specific pretraining data. Across our experiments, FlexTab achieves competitive or state-of-the-art performance on all evaluated tasks, demonstrating that general-purpose row embeddings combined with task-specific predictive modeling can serve as an effective foundation for diverse tabular prediction problems.

Limitations: Despite these encouraging results, our approach has several limitations. First, compressing each row into a single embedding may introduce an information bottleneck, particularly for wide tables or tasks requiring fine-grained cell-level reasoning. We explore this in more detail in Appendix B. Second, like other models trained on large-scale real-world data, our model evaluation may suffer from contamination. We performed a contamination study and found light contamination of evaluation datasets used during pretraining. We refer to the Appendix, where we discuss both issues in detail.

Outlook: Several promising research directions remain open. A natural next step is to replace the currently supervised pretraining of the encoder with a fully self-supervised objective, for example following the LeJEPA or data2vec approaches [4, 2]. Moreover, jointly training all decoders on top of a shared, possibly higher-capacity encoder may unlock synergies across tasks that our staged training does not currently exploit. Given the structural similarities among the different decoders, a further refinement would be to consolidate them, for instance through a mixture-of-experts design, reducing parameter redundancy while preserving task specialization. Our architecture is not limited to the tasks studied here; it extends to settings such as domain transfer, few-shot adaptation, zero-shot prediction, recommendation, and more. As shown by our matching and relational entity classification results, the specific decoders in those cases can pick up signals from multiple embeddings, opening up a range of applications that reason across multiple modalities – similar to multi-modal LLMs but with an in-context learning focus and table-native design. We view the breadth of tasks unified under a single encoder as a first step toward multi-task training at scale, which we believe is a promising and still largely unexplored route to scaling laws in the tabular domain.

References

[1] Michael Arbel, David Salinas, and Frank Hutter. 2025. EquiTabPFN: A target-permutation equivariant prior fitted network. In *Advances in Neural Information Processing Systems*.

[2] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. 2022. Data2vec: A general framework for self-supervised learning in speech, vision and language. In *International Conference on Machine Learning*.

[3] Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C Cresswell, and Rahul G Krishnan. 2025. CausalPFN: Amortized causal effect estimation via in-context learning. In *Advances in Neural Information Processing Systems*.

[4] Randall Balestriero and Yann LeCun. 2025. LeJEPA: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*.

[5] Gilles Blanchard, Gyemin Lee, and Clayton Scott. 2010. Semi-supervised novelty detection. *Journal of Machine Learning Research*, 11(99).

[6] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. In *SIGMOD*.

[7] Patricio Cerda, Gaël Varoquaux, and Balázs Kégl. 2018. Similarity encoding for learning with dirty categorical variables. *Machine Learning*, 107(8).

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16.

[9] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. 2024. Chatbot Arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*.

[10] Peter Christen. 2008. Febrl – an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proceedings of the ACM SIGMOD International Conference on Knowledge Discovery and Data Mining*.

[11] Yann Collet. 2024. xxhash: Extremely fast non-cryptographic hash algorithm. `https://github.com/Cyan4973/xxHash`. GitHub repository; accessed 2026-05-04.

[12] Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. The magellan data repository. `https://sites.google.com/site/anhaidgroup/projects/data`.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

[14] Vijay Prakash Dwivedi, Sri Jaladi, Yangyi Shen, Federico López, Charilaos I Kanatsoulis, Rishi Puri, Matthias Fey, and Jure Leskovec. 2025. Relational graph transformer. *arXiv preprint arXiv:2505.10960*.

[15] Dmitry Eremeev, Gleb Bazhenov, Oleg Platonov, Artem Babenko, and Liudmila Prokhorenkova. 2025. Turning tabular foundation models into graph foundation models. In *NeurIPS Workshop on New Perspectives in Advancing Graph Machine Learning*.

[16] Dmitry Eremeev, Oleg Platonov, Gleb Bazhenov, Artem Babenko, and Liudmila Prokhorenkova. 2026. Graphpfn: A prior-data fitted graph foundation model. *International Conference on Machine Learning*.

[17] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. 2020. AutoGluon-Tabular: Robust and accurate AutoML for structured data. *arXiv preprint arXiv:2003.06505*.

[18] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, Frank Hutter, et al. 2025. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems*.

[19] Matthias Fey, Vid Kocijan, Federico Lopez, Jan Eric Lenssen, and Jure Leskovec. 2025. KumoRFM: A foundation model for in-context learning on relational data.

[20] Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. 2024. Large scale transfer learning for tabular data via language modeling. volume 37.

[21] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Danilo Rezende Yee Whye Teh, and SM Ali Eslami. 2018. Conditional neural processes. In *International Conference on Machine Learning*.

[22] Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. 2025. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*.

[23] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. 2022. ADBench: Anomaly detection benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[24] Adrian Hayler, Xingyue Huang, Ismail Ilkan Ceylan, Michael M. Bronstein, and Ben Finkelshtein. 2025. Of graphs and tables: Zero-shot node classification with tabular foundation models. In *NeurIPS Workshop on New Perspectives in Advancing Graph Machine Learning*.

[25] Steffen Herbold. 2020. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48).

[26] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*.

[27] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. 2025. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326.

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

[29] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. 2024. The tabular foundation model TabPFN outperforms specialized time series forecasting models based on simple features. In *NeurIPS Workshop on Time Series in the Age of Large Models*.

[30] Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. 2026. KumoRFM-2: Scaling foundation models for relational learning.

[31] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*.

[32] Myung Jun Kim, Leo Grinsztajn, and Gael Varoquaux. 2024. CARTE: Pretraining and transfer for tabular learning. In *Forty-first International Conference on Machine Learning*.

[33] Pradap Konda, Sanjib Das, Paul Suganthan G C, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, et al. 2016. Magellan: Toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment*, 9(13).

[34] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60.

[35] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, and George Chen. 2022. ECOD: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*.

[36] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *IEEE International Conference on Data Mining*.

[37] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

[38] Junwei Ma, Nour Shaheen, Alex Labach, Amine Mhedhbi, Frank Hutter, Anthony L Caterini, and Valentin Thomas. 2025. Generalization can emerge in tabular foundation models from a single table. In *EurIPS Workshop on AI for Tabular Data*.

[39] Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L Caterini. 2025. TabDPT: Scaling tabular foundation models. *Advances in Neural Information Processing Systems*.

[40] Patryk Marszałek, Tomasz Kuśmierczyk, and Marek Śmieja. 2026. TACTIC for navigating the unknown: Tabular anomaly detection via in-context inference. *arXiv preprint arXiv:2603.14171*.

[41] Patryk Marszałek, Tomasz Kuśmierczyk, Witold Wydmański, Jacek Tabor, and Marek Śmieja. 2025. Zeus: Zero-shot embeddings for unsupervised separation of tabular data. In *Advances in Neural Information Processing Systems*.

[42] Joe Meyer, Divyansha Lachi, Mahmoud Mohammadi, Roshan Reddy Upendra, Eva L Dyer, Mark Li, and Tom Palczewski. 2025. RELATE: A schema-agnostic perceiver encoder for multimodal relational graphs. *arXiv preprint arXiv:2510.19954*.

[43] Martin Mráz, Breenda Das, Anshul Gupta, Lennart Purucker, and Frank Hutter. 2025. Towards benchmarking foundation models for tabular data with text. In *ICML 2025 Workshop on Foundation Models for Structured Data (FMSD)*.

[44] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 19–34.

[45] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. 2022. Transformers can do Bayesian inference. In *International Conference on Learning Representations*.

[46] Guansong Pang, Chunhua Shen, and Anton van den Hengel. 2019. Deep anomaly detection with deviation networks. In *KDD*, pages 353–362.

[47] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of Machine Learning Research*, 12:2825–2830.

[48] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.

[49] Francesco Pugnaloni, Luca Zecchini, Matteo Paganelli, Matteo Lissandrini, Felix Naumann, and Giovanni Simonini. 2025. Table overlap estimation through graph embeddings. *Proceedings of the ACM on Management of Data*, 3(3):1–25.

[50] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. 2025. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*.

[51] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. 2026. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*.

[52] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, pages 427–438.

[53] Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. 2026. Relational transformer: Toward zero-shot foundation models for relational data. In *The Fourteenth International Conference on Learning Representations*.

[54] Jake Robertson, Arik Reuter, Siyuan Guo, Noah Hollmann, Frank Hutter, and Bernhard Schölkopf. 2025. Do-PFN: In-context learning for causal effect estimation. In *Advances in Neural Information Processing Systems*.

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

[56] Lukas Ruff, Robert A. Vandermeulen, Nico Görnitz, Alexander Binder, Emmanuel Müller, Klaus-Robert Müller, and Marius Kloft. 2020. Deep semi-supervised anomaly detection. In *International Conference on Learning Representations*.

[57] David Schiff, Ofir Lindenbaum, and Yonathan Efroni. 2026. ICR-RL: Deep reinforcement learning via in-context-regression. In *International Conference on Machine Learning*.

[58] Bernhard Schölkopf, Robert C Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support vector method for novelty detection. *Advances in Neural Information Processing Systems*, 12.

[59] Yuchen Shen, Haomin Wen, and Leman Akoglu. 2025. Fomo-0d: A foundation model for zero-shot tabular outlier detection. *Transactions on Machine Learning Research*.

[60] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. 2003. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*.

[61] Marco Spinaci, Marek Polewczyk, Maximilian Schambach, and Sam Thelin. 2025. ConTextTab: A semantics-aware tabular in-context learner. In *Advances in Neural Information Processing Systems*.

[62] Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony Caterini. 2024. Retrieval & fine-tuning for in-context tabular models. In *Advances in Neural Information Processing Systems*.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

[64] Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. 2025. Griffin: Towards a graph-centric relational database foundation model. In *Forty-second International Conference on Machine Learning*.

[65] Jack Yi Wei and Narges Armanfard. 2026. ICLAD: In-context learning for unified tabular anomaly detection across supervision regimes. *arXiv preprint arXiv:2603.19497*.

[66] Linjie Xu, Yanlin Zhang, Quan Gan, Minjie Wang, and David Wipf. 2026. No need to train your RDB foundation model. In *International Conference on Machine Learning*.

[67] Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. 2024. A closer look at deep learning methods on tabular datasets. *arXiv preprint arXiv:2407.00956*.

[68] Xiyuan Zhang, Danielle C Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W Mahoney, et al. 2025. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*.

[69] Tianqi Zhao, Guanyang Wang, Yan Shuo Tan, and Qiong Zhang. 2026. TabClustPFN: A prior-fitted network for tabular data clustering. *arXiv preprint arXiv:2601.21656*.

[70] Yue Zhao and Maciej K Hryniewicki. 2018. XGBOD: Improving supervised outlier detection with unsupervised representation learning. In *International Joint Conference on Neural Networks (IJCNN)*.

[71] Yingjie Zhou, Xucheng Song, Yanru Zhang, Fanxing Liu, Ce Zhu, and Lingqiao Liu. 2021. Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6).

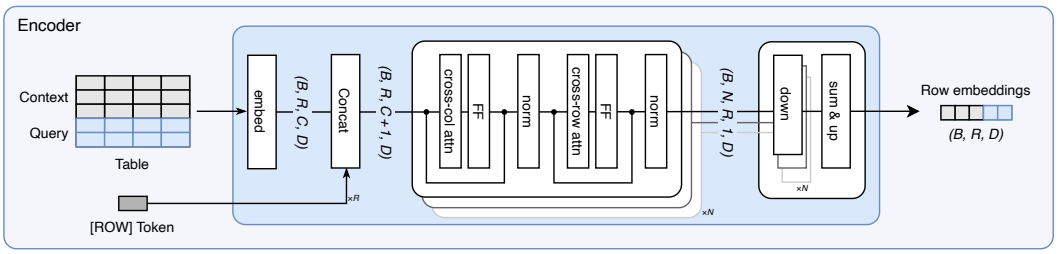


Figure 2: Detailed schematic depiction of the encoder architecture. Note that cross-row attention is performed in a way such that query rows cannot attend to each other, analogous to existing architectures. The learnable `[ROW]` tokens are repeated along the row axis and then appended to the table embeddings along the column axis. Although depicted in parallel, the transformer blocks are executed sequentially using the output of the previous layer serving as the input to the next, as is common. However, the N individual `[ROW]` token outputs of each transformer block are separately down-projected, then summed and up-projected to obtain the final row embeddings.

A Architectures and Pretraining Details

A.1 Encoder

We follow the embedding strategy of ConTextTab [61], encoding each cell according to its modality: text, date, or numeric. Text cells are embedded with a pretrained `all-MiniLM-L6-v2` model, and a learnable linear layer then projects the resulting vector to the FlexTab transformer's input dimension of $d = 768$. Dates are decomposed into day, month, and year; each component is embedded separately, and the three vectors are summed. Numeric columns are discretized into 50 quantile bins, following the procedure described in the decoder's regression section below, and are encoded by piecewise linear interpolation over a set of learned reference vectors. In addition, column headers serve as a form of positional encoding: each header is embedded with a text embedding model, passed through its own learnable linear layer, and added to every cell in its column. Finally, the summed embeddings are normalized with layer normalization.

A more detailed depiction of our proposed encoder backbone is shown in Figure 2. While depicted in parallel, the 2D Transformer blocks are executed sequentially. The visualization further illustrates that each block's output is separately (down-)projected, then summed and (up-)projected to obtain the final row embeddings of the input rows.

In all cases, cross-row attention is performed in such a way that context rows can only attend to each other and query rows can only attend to context rows, preventing any query row leakage. This is achieved efficiently using a cross-attention formulation. Given an input sequence of shape (B, S, D) with a batch size B, sequence length S, and hidden dimension D, where the first C elements in the sequence are context examples and the last Q elements are query examples, we achieve the desired attention pattern by using the context-truncated sequence of shape (B, C, D) as attention KV and the full sequence of shape $(B, S, D) = (B, C + Q, D)$ as attention Q. If no context/query split is present, all sequence elements are effectively treated as context, and the formulation falls back to full self-attention between all rows.

A.2 Decoder

All decoders are built around the same 2D attention block, with intermediate feed-forward and normalization layers as depicted in Figure 2. Details on the individual heads are depicted in Figure 3. In the main classification and regression decoder, each target attends to a single `[ROW]` token. Due to the softmax operation, a standard Multi-Head Attention applied over a KV sequence of length 1 produces an attention weight equal to 1.0. This ultimately reduces to a linear projection of V. Combined with the skip connection, the cross-column attention block output is therefore equivalent to the query input plus a linear projection of V (`[ROW]` token). However, this formulation allows the model to attend to multiple `[ROW]` tokens without any change to the architecture.

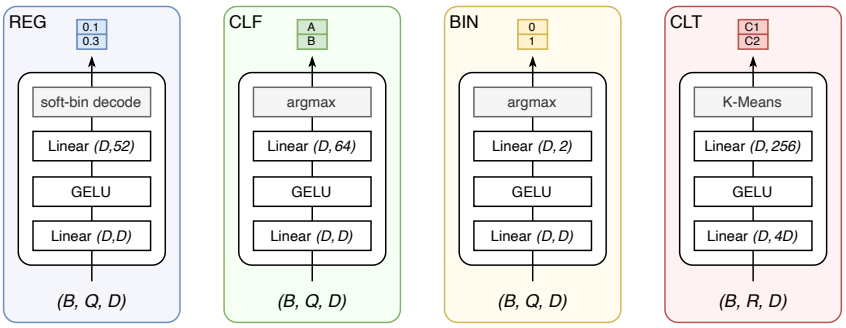


Figure 3: Details of the used heads for all different investigated decoders, including the regression (REG) head, classification (CLF) head, binary classification (BIN) head, and clustering (CLT) head. Grey blocks denote inference-only blocks, whereas for training, the raw logit outputs are used and passed to the loss function.

A.3 Classification and Regression

In this section, we describe the decoder architecture used for both classification and regression tasks. The decoder consists of the main backbone and two separate MLP-based heads: one dedicated to classification and the other to regression.

Classification: For classification, we employ a standard cross-entropy loss applied to the output of an MLP with a fixed output dimension of 64, corresponding to the maximum number of supported classes. As a result, at inference time, the model cannot predict more classes than this predefined number without resorting to less efficient approaches such as hierarchical classification (e.g., as used in TabPFN's many-class extension or TabICL). To ensure the model can associate semantic meaning with class indices (e.g., class "0"), it must learn an internal representation of these identifiers. To facilitate this, we introduce a dedicated input encoding specifically for the target column, in addition to the encodings used for feature columns. Although this design choice breaks class order permutation equivariance, we find it to be effective in typical few-class classification settings.

Regression: We encode numerical values via piecewise linear interpolation over a set of learned reference vectors. Specifically, given the training data (context rows), we compute $n{=}50$ quantiles at orders $\frac{1}{2n}, \frac{3}{2n}, \ldots, \frac{2n-1}{2n}$ and extend them with a (clipped) minimum and maximum, yielding $n{+}2$ reference points $q_0 < q_1 < \cdots < q_{n+1}$. Each reference point q_i is associated with a learnable vector v_i. A value $x \in [q_i, q_{i+1}]$ is then encoded as $(1{-}\lambda)\, v_i + \lambda\, v_{i+1}$, where $\lambda = \frac{x-q_i}{q_{i+1}-q_i}$; values outside $[q_0, q_{n+1}]$ are clipped to the boundary.

For the regression target, this encoding is used on the input side during training. On the output side, regression is cast as classification over the $n{+}2$ reference points: the model produces a probability distribution $(p_0, \ldots, p_{n+1})$ via a softmax, and the final prediction is the expectation $\sum_i p_i\, q_i$.

A.4 Matching

For the matching decoder training, we employ two separate pretraining and task generation strategies:

First, from a random table, we generate two column-wise splits with a random overlap of *at most* 50%. These act as the matching Table A and Table B depicted in Figure 1. With a random probability between 5% and 33% (drawn and fixed for the current table), we then mark each row pair as a match by drawing from the Bernoulli distribution. This marking serves as the binary target label of the current sample. We then keep the order of rows marked as matched in place, while randomly shuffling the remaining ones. This strategy resembles cases of pairwise row matches with partially identical column entries. That is, in most cases, the prediction can be made by exact matching, $\hat{y} = (A_i == B_j)$ for some column A_i in Table A and B_j in Table B respectively. However, note that this cannot trivially be exploited in our architecture as both rows are first encoded into latent `[ROW]` tokens which are contextualized over separate statistics and table schemas. Hence, we found this strategy to provide reasonably challenging matching tasks that closely resemble many real-world matching tasks with exact or fuzzy column matches.

As for the second strategy, we use a more indirect semantic matching target: Again, using a randomly drawn table, we use the same strategy as in the classification training to generate a feature-target split $X, \tilde{y}$ of a corresponding classification target. We then again create two column-wise splits from the feature matrix X with *at least* 50% overlap. The resulting tables are then shuffled. The ground truth matching target is then defined as the agreement of the original classification target between the rows in Tables A and B, $y = (\tilde{y}_A == \tilde{y}_B)$. That is, two rows are considered matching, if they share their underlying classification target. This results in a much more indirect, semantic matching task as opposed to the direct column matching above.

A.5 Clustering

For unsupervised clustering, most of the decoder architecture is inherited from the classification decoder, in order to minimize the number of parameters to be trained from scratch. Target embeddings are also inherited by using the `[MASK]` token in all of the rows, including the context. The prediction head is replaced by an MLP trained from scratch, outputting embedding vectors of dimension 256 for each row, which are further normalized.

Training data is sampled similarly to classification or regression data, by choosing a random table and selecting one random column as the target, with the following additional criteria (applied independently to each column, of any data type): missing values are dropped; only values that appear at least 5 times are retained; among those, only the 10 most numerous classes are retained. If, after these steps are applied, at least 50% of the original rows survive, this column is used as the target. A random subset of the remaining columns is used as features, and the target is only used to construct labels (never shown to the model itself).

During training, the row-embedding vectors obtained above are transformed into a square matrix containing the cosine similarities s_{ij} between any two rows (rescaled by a constant factor of 10). These values are then compared against the ground truth (1 meaning "same hidden class", 0 meaning "different class"), that is:

$$\text{Loss} = -\sum_{\substack{i=1,\dots,R \\ j\in\text{context}}} \Big[\log(s_{ij})\delta_{c_i=c_j} + \log(1-s_{ij})\delta_{c_i\neq c_j}\Big], \quad s_{ij} = \sigma\Big(10\cdot\frac{x_i\cdot x_j}{\|x_i\|\|x_j\|}\Big) \tag{1}$$

Note that the loss is computed over all rows i (both context and query). In this way, the same trained model can support both the inductive (using context+query, with query rows not influencing each other) and transductive settings (using context only; the latter is the only setting supported by, e.g., TabClustPFN).

At inference time, the normalized embeddings are fed to K-means. This is a natural choice: for normalized vectors, the cosine similarity is directly linked to the Euclidean distance. For example, if a set of points achieves zero loss as computed in (1) (which is only possible if there are exactly two clusters), then embeddings of points in the same cluster will have distance 0, and those of points in different clusters will have distance 2 (being normalized vectors pointing in opposite directions), so they will be clustered correctly by K-means (or other Euclidean-distance-based algorithms). Also, K-means requires an explicit k to be provided; we only implement and test the "oracle" setting where this is provided by the user.

A.6 Pretraining Data

Data Statistics: Our pretraining corpus consists of roughly 300 k raw, unlabeled tables. These tables were sourced and curated at scale from various public sources with a permissive license. An overview of the size distribution is presented in Figure 4, comparing the row, column, and data-type distributions of our corpus with those of The Tremendous TabLib Trawl (T4) collection [20]. We observe that our pretraining corpus is more broadly spread, particularly in terms of table width and height, and overall has a higher average rate of string columns. Note that these data type rates are averages of per-table rates. That is, we first measure the data type rate across columns of each table and then collect averages across tables.

Contamination: We perform a contamination study to estimate overlap between the pretraining corpus and the evaluated benchmark data for classification and regression. To that aim, we employ two different methodologies.

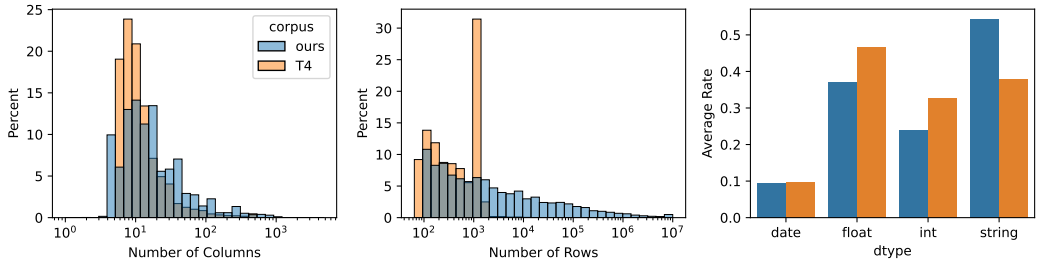
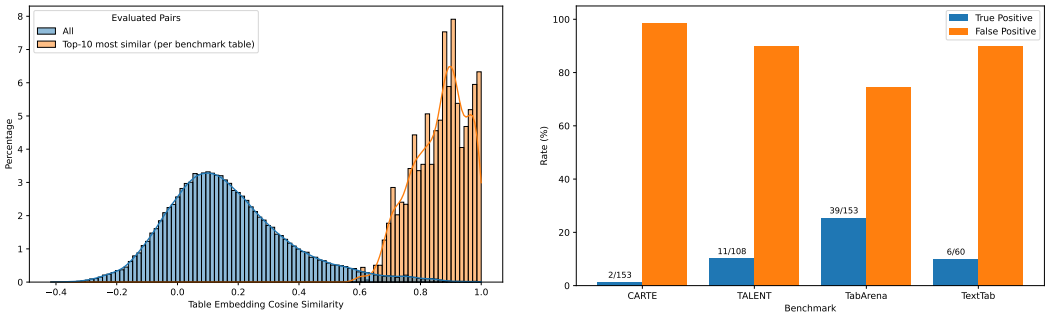


Figure 4: Column and row count distribution, as well as per-column data type rate of our pretraining data corpus and the T4 corpus. Our corpus consists of roughly 300 k tables in total, sourced from various public sources. The T4 corpus was downsampled to 500 k tables for its statistics calculation.



(a) Cosine similarity distribution between the used pretraining tables and all evaluated benchmark tables

(b) True and false positives among the top three most similar table pairs as estimated by Claude Opus 4.6

Figure 5: Table overlap and false positive rates based on Armadillo embeddings and false-positive estimated using Claude Opus 4.6. Out of those marked true positive, manual confirmation showed 48 confirmed true positive matches for 25 of the used evaluation tables.

Row-wise Hashing: This scalable approach looks at the full pretraining data, by flagging tables where rows match exactly (up to column permutation; table headers are also ignored). To that aim, we first convert all cells to strings, then encode them to 64 bits via the xxHash algorithm [11]. These hashes are then interpreted as `uint64` numbers, i.e., integers modulo 2^{64}, and the final (permutation-invariant) fingerprint of each row is obtained by summing them together. We then check for duplicates by first filtering for tables having the same number of columns, sorting the fingerprints for all the pretraining data matching this condition, and finally looking for the (much fewer) fingerprints appearing in evaluation tables via binary search. Evaluation tables that match at least 10% of their rows to a given table are flagged as leaking. This is the strongest lower bound, flagging tables that are reported verbatim (including both features and targets) in the training data. It is, however, not robust to some minor distortions, like type casting (e.g., `int` to `float`) or dropping one or more columns.

This exact equality highlighted that 8 of the 158 evaluation tables do appear in the pretraining data; 4 of these are part of the TabArena benchmark, and 4 more of the TALENT-Tiny one.

Table-level Embeddings: As a more robust, but less scalable approach, we perform a pair-wise comparison between all used evaluation datasets and a maximum 10 k-row sample of each pretraining dataset using table-level embeddings obtained via Armadillo [49]. Armadillo is a Graph Neural Network pretrained such that the cosine similarity between two table embeddings corresponds to their estimated overlap. Note that the obtained embeddings are invariant against permutations of rows and columns, but trained for exact, non-fuzzy matches.

That is, we calculate table-level embeddings for all 158 tables used during evaluation (we focus on the classification and regression datasets here) and all roughly 300 k tables used during pretraining. We then calculate the pair-wise cosine similarity between all embeddings and collect the top 10 most-similar matches for each evaluation table. The resulting similarity distribution is shown in Figure 5a.

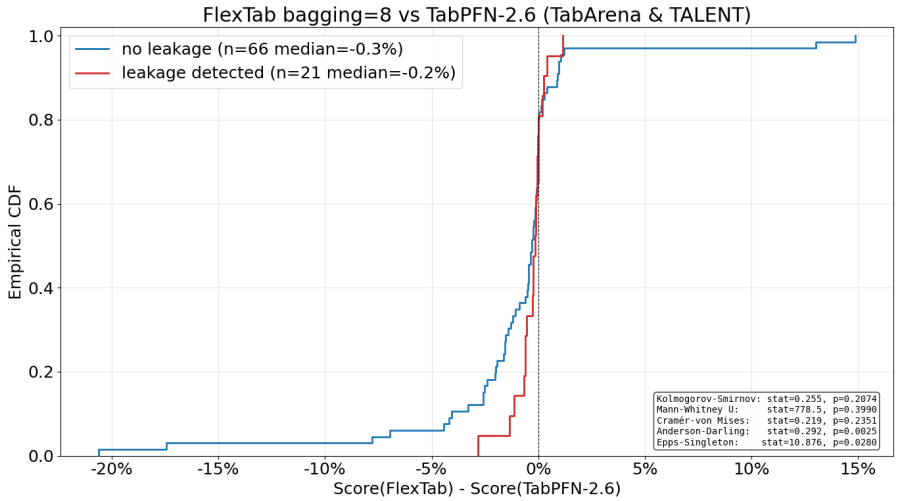


Figure 6: Cumulative distribution functions of the differences between FlexTab scores vs. TabPFNv2.6 scores on the contaminated (red) and non-contaminated (blue) subsets of numeric-heavy datasets.

Overall, the embeddings obtained via Armadillo show a relatively large overlap, with a median of about 14 %. Among the most-similar matches, we observe a much higher similarity, indicating that almost all tables have at least one matching pretraining table with more than 50 % estimated overlap, according to the Armadillo embeddings. However, importantly, a manual inspection showed many non-matching pairs despite the high degree of estimated overlap. Hence, as a further step, for each evaluation table, we passed the top three most similar found matches to an AI agent (Claude Code using Claude Opus 4.6) to estimate whether the pair is an actual match or a false positive. The resulting distribution is depicted in Figure 5b. As a final verification step, we then manually went through all pairs marked as true-positives and found 48 pairs of those to be actually true positives. However, even out of those, nine matches do not contain the original task's target column but are matches in the raw data instead.

Concluding, in total, we found matching pretraining tables for 25 of the used evaluation tables (including the 8 found with the previous methodology), in detail:

- 2 from the CARTE benchmark (with one having only matches that do not contain the target)
- 15 from TabArena
- 2 from TextTab
- 6 from TALENT-Tiny (with one having only matches that do not contain the target)

In particular, the high number of numerical tasks in TabArena accounted for many false positives due to the high probability of any large mostly-numeric table to have a large cell-wise overlap.

While this overlap is considerable, we believe the impact on our reported results to be small: Each table is sampled only a small number of times during training and subsequently augmented, subsampled, and used with a random prediction target. Given the comparatively small model size and in-context learning training objective, we believe that current tabular in-context learners, including our proposed architecture, do not suffer or gain from in-weight data memorization in any meaningful way. We can observe this with our model performing noticeably better than the state of the art on the CARTE benchmark, where we only found contamination for 2 of the used evaluation datasets, whereas the performance on TabArena is relatively worse despite the much larger amount of contamination found.

To confirm this hypothesis, we have performed a comparative analysis on the downstream performance on classification and regression tasks. We compared our FlexTab model against TabPFNv2.6 that, being trained on synthetic data, serves as a contamination-free baseline. We analyze separately the performance on contaminated and non-contaminated subsets of the evaluation datasets, compute the score difference between the two models, and investigate whether the two resulting distributions are statistically distinguishable. Given that the contamination is mostly present on the numeric-heavy datasets (TabArena and TALENT-Tiny) where our model is comparatively weaker and less

on text-heavy ones (CARTE, TextTab) where it is stronger, this analysis only focuses on the former datasets, to avoid biasing the results. The cumulative distribution function of the differences are plotted in Figure 6. While the two distributions differ on the tails (as highlighted by the low p-value of the Anderson-Darling test), there is no consistent bias of FlexTab performing better on average on contaminated datasets. The median score delta only drops from a –0.2 % (non-contaminated) to a –0.3 % (contaminated) gap, well within statistical noise. This is confirmed by statistical tests that are more sensitive to the center of the distribution, like Kolmogorov-Smirnov ($p = 0.21$) and Cramér-von Mises ($p = 0.24$), neither of which suggests a systematic difference between the two distributions.

Overall, while we have performed due diligence to estimate the amount of contamination in our pretraining dataset, estimating the impact on training and downstream performance is a challenging problem. Generally, as the community moves forward, investigating tabular models trained on real-world data, the topic of benchmark contamination with pretraining data used to train large tabular models is an open and important research topic.

A.7 Baselines

Overall, we follow the evaluation protocol and baseline implementations of ConTextTab [61], but with updated baseline versions:

TabPFN: We evaluate TabPFNv2.6 using the official PyPI package version 7.1.1. For classification tasks with more than 10 classes, we use the many-class extension from `tabpfn-extensions` in version 0.2.2+ at the latest git commit `452708bdf2e644dd1b2ebe938862a9ea52ac7e21` which adds compatibility with TabPFNv2.6. We use default initialization parameters, in particular, using 8 ensembled estimators for inference.

TabICL: We evaluate the recent TabICLv2 model, using the PyPI package version 2.0.1. As opposed to the original TabICL variant, TabICLv2 natively supports both classification and regression. We use default initialization parameters, in particular, using 8 ensembled estimators for inference.

ConTextTab: We evaluate the recent ConTextTab model using the official repository and checkpoints at git version tag v1.1.2[1].

AutoGluon: We evaluate AutoGluon version 1.5 in the `extreme` preset with a time limit of 1h, fitted on a 40-core node with 320 GB of RAM and a single H100 GPU. We also evaluated the `best` preset with a time limit of 4h but found the former to be slightly better performing, likely due to the better use of recent tabular in-context learners in the model pool.

PytabKit: Again, we follow the setup in ConTextTab, but use the recent 1.7.3 version of the PyPI `pytabkit` package [28] with updated search spaces following the TabArena setup [18]. For RealMLP, we report the best-in-class tuned Caruana-ensembled results [18]. For the extended results, we also show the tuned-defaults (TD) variants.

Scikit-learn: For the naive, KNN, and Random Forest baselines, we use the corresponding scikit-learn [47] estimators with their default settings. We use scikit-learn version 1.5.2.

LimiX: In the extended results, we also report the evaluation of LimiX (2M and 16M) using the corresponding retrieval configs. We used the code from the official repository[2] at the latest commit hash `375b49eb1d377620943aa93e4654a63e542f0d2b`.

A.8 Evaluation Metrics

For all metric calculations, we use the corresponding scikit-learn implementations [47].

For classification and regression results, we report mean accuracy and soft-clipped R^2: due to the dominance of outliers, we soft-clip the negative R^2 values to the range $[-1, 0)$ via $\tanh$, which is smoothly differentiable and retains the relative ordering of negative scores [61].

For the calculation of the Elo scores, we utilize the implementation of TabArena [18], which itself is based on ChatBot Arena's implementation [9].

[1] https://github.com/SAP-samples/contexttab/

[2] https://github.com/limix-ldm-ai/LimiX

Table 8: Extended classification and regression performance, depicting mean accuracy (Acc) for classification and (soft-clipped) R^2 score for regression tasks, in percent, as well as mean per-task rank.

Model Name	All	CARTE			TabArena-Lite			TALENT-Tiny			TextTab		
	Rank	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2
AutoGluon	N/A	N/A	78.8	74.0	N/A	88.6	80.7	N/A	88.9	84.7	N/A	83.5	67.0
FlexTab [bag=8]	**4.5**	**2.0**	**77.3**	**74.4**	6.1	87.1	79.5	5.5	88.0	**85.8**	5.3	83.8	62.7
FlexTab [bag=1]	5.2	3.0	77.0	73.6	6.8	87.0	79.1	5.9	87.9	85.5	5.9	83.7	61.4
RealMLP [HPO, CV=5]	5.3	5.7	74.3	69.0	5.0	88.3	79.9	5.2	88.6	85.0	5.0	82.3	**67.7**
LimiX-16M [etrieval]	5.8	6.2	75.5	69.5	5.9	86.8	78.1	5.9	**88.9**	84.4	**4.4**	83.8	65.0
ConTextTab	6.1	4.0	77.1	72.4	7.3	87.6	77.8	7.1	87.6	83.2	6.4	**84.4**	58.8
CatBoost [HPO, CV=5]	6.1	6.7	76.3	68.3	5.7	88.2	79.0	6.4	87.1	83.5	5.0	83.7	65.4
TabPFNv2.6	6.6	12.6	70.7	59.9	**2.5**	88.7	**80.5**	3.5	88.4	84.9	7.2	81.5	63.8
TabICLv2	6.9	13.3	70.5	55.3	3.0	**88.7**	79.9	**3.2**	88.3	85.3	7.2	82.5	60.2
LightGBM [HPO, CV=5]	7.1	7.7	73.6	67.4	6.6	88.0	78.7	7.5	86.9	81.5	6.3	81.5	65.5
XGBoost [HPO, CV=5]	7.7	8.6	73.5	66.9	7.4	87.9	78.8	7.4	87.0	81.1	7.1	81.4	65.4
TabDPT	8.2	9.8	73.0	65.7	5.4	88.1	80.1	8.6	87.0	82.7	10.6	80.8	53.7
LimiX-2M [retrieval]	8.2	8.5	74.9	68.2	7.5	84.4	77.5	8.0	88.1	81.5	9.8	77.0	59.4
CatBoost [TD]	10.3	10.5	74.7	65.9	10.3	87.6	76.4	10.3	86.2	82.3	9.4	83.3	53.2
LightGBM [TD]	10.4	11.5	72.7	64.9	10.0	87.4	76.8	9.4	86.4	78.3	10.5	80.9	60.8
XGBoost [TD]	11.6	12.8	72.3	64.6	11.0	87.4	76.6	11.4	86.3	78.0	10.3	81.1	60.4
TabM	11.7	15.4	70.7	60.7	9.1	87.2	76.8	10.1	85.7	82.0	12.1	79.6	55.9
Random forest	11.8	13.8	71.4	63.4	10.7	87.6	76.1	11.4	85.6	78.0	10.4	79.8	60.7
RealMLP [TD]	12.1	15.6	70.6	59.7	10.7	87.3	75.4	9.8	86.0	79.6	11.2	80.0	57.8
KNN [k=5]	18.1	18.3	65.6	34.5	17.3	81.8	57.6	18.4	80.3	66.9	19.4	72.5	4.8
Naive	18.7	19.9	53.0	−1.7	17.3	75.0	−7.3	19.5	53.4	−21.0	17.6	70.4	−5.5

For all ranking results (aside from those used in the critical difference diagrams), we use a robust ranking algorithm. That is, models that lie within 0.1 percentage points (in terms of either Accuracy, AUROC, or R^2) are considered ties. The same robustification also applies to our Elo score and win-ratio calculations.

For the critical difference (CD) diagrams, we use the `autorank` library [25]. So as not to influence the statistical guarantees of the significance test used, we do not alter or robustify its internal rank calculation. Hence, the ranks shown in the CD diagrams can deviate from those shown in the result tables or figures.

A.9 Datasets

Classification and Regression: We evaluate all models on CARTE, TextTab, TabArena-Lite, and TALENT-Tiny. We use the same split creation protocol as ConTextTab, except for TabArena-Lite, for which we use the predefined splits.

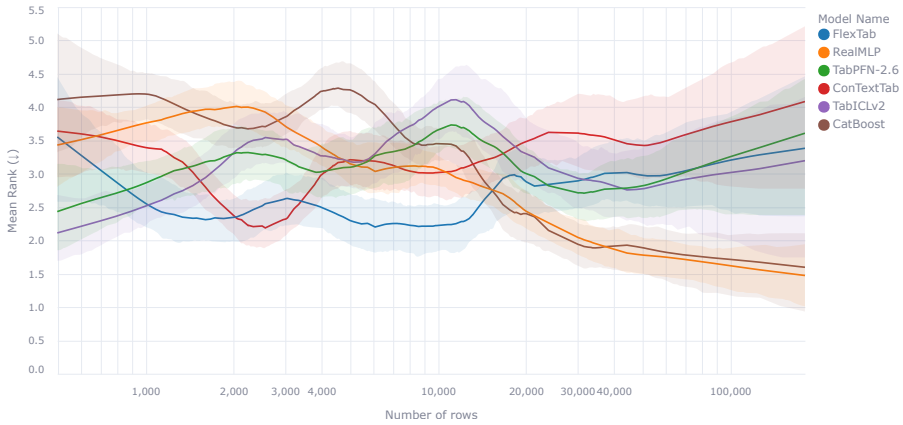
Matching: We evaluate on the synthetic Febrl4 dataset from the Freely Extensible Biomedical Record Linkage project [10], as well as Fodors-Zagats, Bikes, eBooks, and Movies sourced from the DeepMatcher study [44] and the Magellan Data Repository [12]. Due to the relatively small size of these datasets with available ground truth match labels, we perform 5-fold cross-validation over all available labeled examples and report average match-class F1 scores.

Relational Prediction: We use the RelBench benchmark from the official `relbench` PyPI package in the latest version 2.1.1 using the predefined splits.

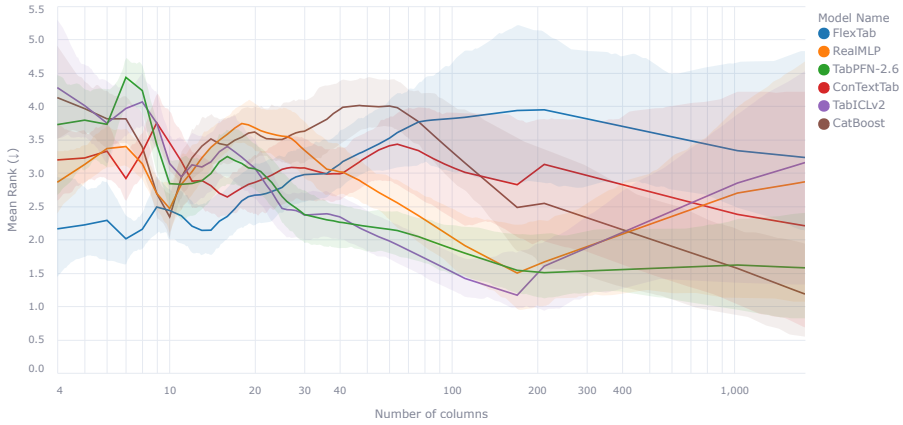
B Additional Results

B.1 Classification and Regression

Additional Baseline Evaluations: Additional results including LimiX, and other gradient-boosted tree variants and scikit-learn estimators, are shown in Table 8. The overall results are in line with the selected presentation in the main paper and serve only to illustrate a complete picture. Across all evaluated models and benchmarks, FlexTab ranks best overall and performs particularly well on



(a) Performance dependence on row count.



(b) Performance dependence on column count.

Figure 7: Relation between number of training dataset rows (top) and columns (bottom) and model performance, obtained as a LOWESS regression in the plane $\log(n_{\text{rows/cols}}, \text{rank})$. The confidence bands are the 80% confidence intervals obtained via bootstrapping.

semantic-heavy benchmarks. This is in line with the results of ConTextTab, which our architecture builds upon, while narrowing the gap on numeric-heavy regression, such as on TabArena-Lite, but remaining behind in classification.

Relation between Dataset Size and Model Performance: We plot the average rank of each model as a function of the dataset size expressed in number of rows and the number of columns in Figure 7 across all evaluated datasets.

We observe that FlexTab performs particularly well in the 1 k to 10 k row count regime. For smaller tables, TabICL and TabPFN tend to perform better, but we also note the large confidence intervals due to the comparatively small support set in those areas. For larger tables, the in-context learners tend to converge to a similar performance and start being outperformed by CatBoost and RealMLP which are extensively trained and tuned for each dataset.

In terms of the dependence on the number of columns or features of each task, we observe that FlexTab excels at very narrow tables but shows decreasing performance for wider ones (relative to the other models). Note again the potentially small support set and wide confidence bands; ranking scores can exaggerate such differences, and the absolute differences may not be as large. Nevertheless, this points to a potential bottleneck in our architecture: As we compress each row into a single embedding that is fed to the prediction decoder, wider tables naturally undergo a relatively larger compression than narrow ones. This should be investigated in future work, for example, by increasing the hidden dimension or investigating hybrid approaches that utilize multiple tokens per row for wider tables.

Win ratios, CD diagrams, and Elo scores: Additionally, we show the win ratio confusion matrix, CD diagrams, and Elo scores across all investigated benchmarks (CARTE, TabArena-Lite, TALENT-Tiny, and TextTab) in Figure 8. Wins are calculated based on accuracy for classification and R^2 for regression datasets. Ties are not counted as wins.

For fine-grained insights, we also plot per-benchmark-type diagrams:

- for semantic-heavy benchmarks (CARTE and TextTab) in Figure 9,
- for numeric-heavy benchmarks (TabArena-Lite and TALENT-Tiny) in Figure 10.

Across all benchmarks investigated, FlexTab shows very strong performance, outperforming other baselines, even hyperparameter-optimized ones with inner CV ensembles. The ensembled FlexTab [bag=8] variant achieves the highest Elo score across the full set of 158 datasets (Figure 8), followed closely by FlexTab [bag=1], with the critical difference diagram confirming that both variants rank significantly above Random Forest, CatBoost, ConTextTab, and TabICLv2.

The per-benchmark-type analysis reveals a more nuanced picture. On the 71 semantic-heavy datasets from CARTE and TextTab (Figure 9), FlexTab's advantage is particularly pronounced: FlexTab [bag=8] reaches an Elo of 1506, more than 100 points ahead of the next-best in-context learner (ConTextTab) and more than 400 points ahead of TabPFNv2.6 and TabICLv2, whose handling of semantics is more limited. The corresponding critical difference diagram shows that both FlexTab variants significantly outperform all other methods. On the 87 numeric-heavy datasets from TabArena-Lite and TALENT-Tiny (Figure 10), the picture is tighter: TabPFNv2.6 and TabICLv2 take the top two Elo positions, with FlexTab ranking fourth but remaining competitive with all tuned and ensembled per-dataset baselines. Notably, the critical difference diagram indicates that the top three models in this regime are statistically indistinguishable.

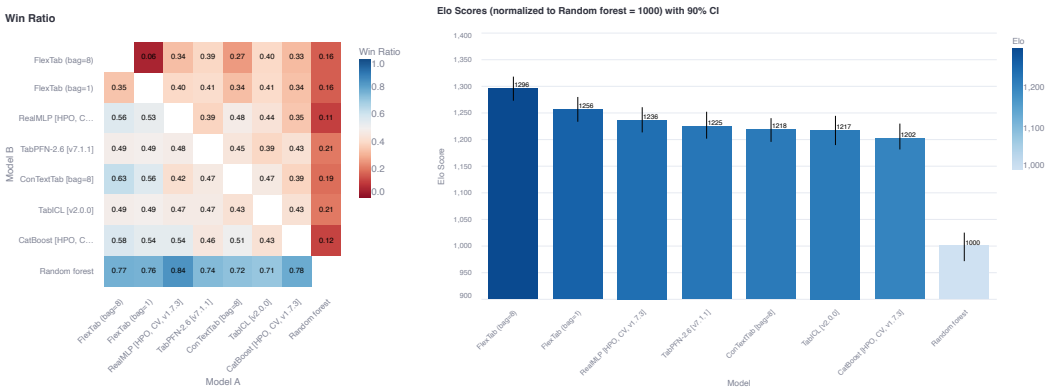
These results show that FlexTab is the strongest method overall across the investigated benchmarks, with a clear and statistically significant edge on semantic-heavy tasks and competitive performance on numeric-heavy ones. This supports our central claim that target-agnostic row embeddings, combined with task-specific decoders, provide an effective and broadly applicable foundation without sacrificing performance on the well-studied classification and regression regimes.

B.2 Anomaly Detection

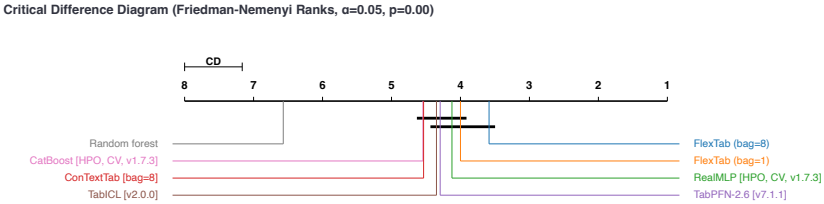
The checkpoint has been selected via early stopping based on a manually curated validation set of 89 datasets from standard classification (52) and regression (37) tasks (including CARTE, TabArena, cc18, ctr23, and TextTab). For classification data, rare classes were chosen as anomalies and common classes as normal points; for regression data, after plotting a distribution with density smoothing, an antimode between two local clusters (if present) is chosen as the split value. In both cases, datasets are retained only if one of the thus-constructed classes is substantially smaller than the other. The three scenarios (unsupervised, one-class, and semi-supervised) are all evaluated during validation, and the best checkpoint by average score is selected; this corresponds to 7.2M training steps. For the semi-supervised setup, we also test the original FlexTab version trained for classification and regression. To distinguish between the two, we denote the version trained for anomaly detection as FlexTab-AD.

Unsupervised: The unsupervised setup (where no labels are provided to the model) is a notoriously challenging one. The results are reported in Figure 11 (top). Overall, TACTIC slightly outperforms competing models across all considered AUROC-based metrics (average, median, and rank), with FlexTab scoring a close second. However, by and large, we share the same conclusion as in the original ADBench paper: in the unsupervised setting, essentially all methods end up being statistically equivalent, as the critical difference diagram in the top-right part of the figure highlights.

One Class and semi-supervised novelty detection: We group the results for these two settings, because the labeled data is the same in both cases; the only difference is that models in the latter category are allowed to operate in a transductive manner (i.e. to see all the test points at once and make use of that information). This changes the class of models that can be used entirely: in the inductive setting, only dedicated models can be used. Among these, the traditional KNN method appears as valid a choice as any more sophisticated approach, among which FlexTab-AD performs the best (with best overall median score in this class). Once test points can be exploited, though, one can obtain significantly higher accuracy by means of any binary classifier. In this setting we can see

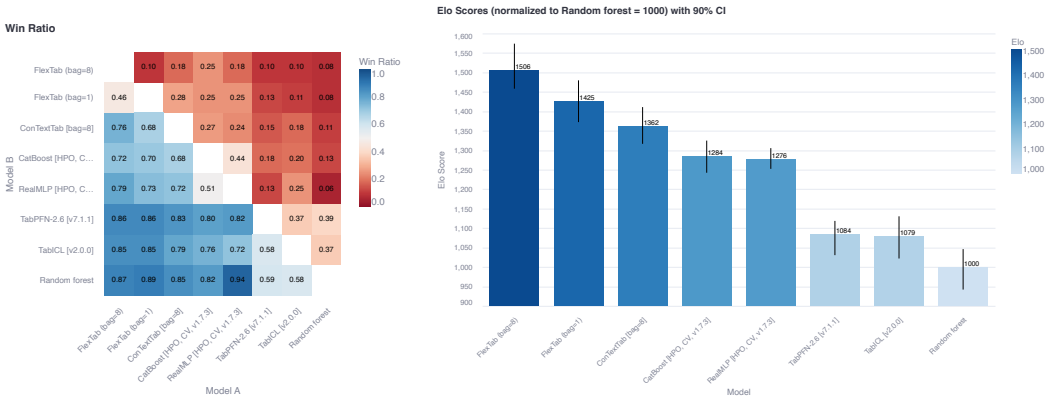


(a) Win ratio matrix with Model A wins over Model B (left) and Elo scores (right).

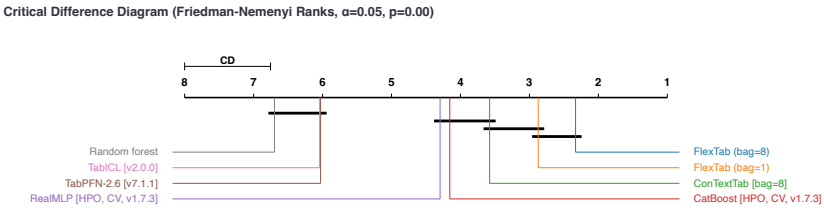


(b) Critical difference (CD) diagram.

Figure 8: Win ratio confusion matrix, Elo scores, and CD diagram of the main investigated models across all 158 datasets from CARTE, TabArena-Lite, TALENT-Tiny, and TextTab benchmarks.

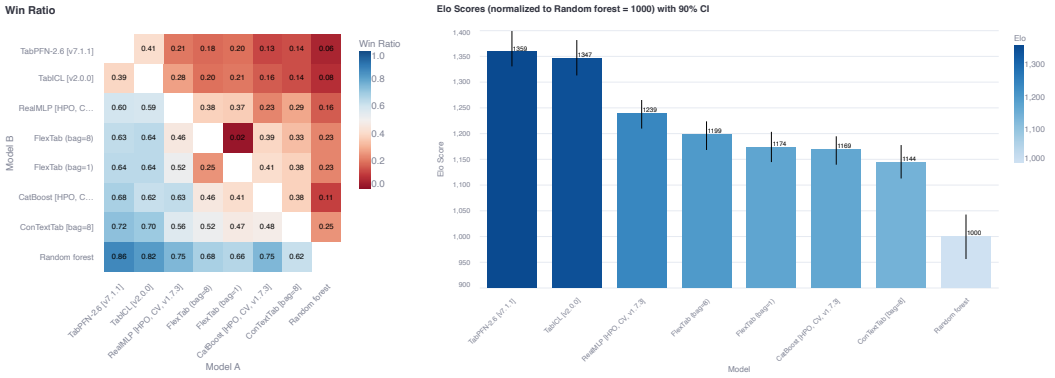


(a) Win ratio matrix with Model A wins over Model B (left) and Elo scores (right).

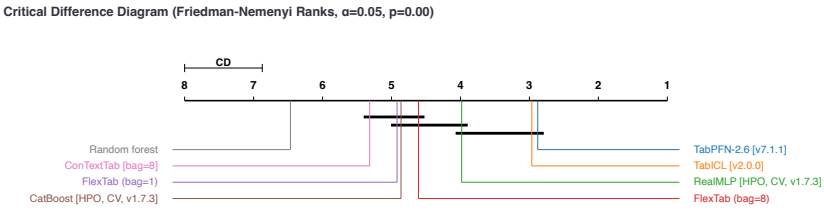


(b) Critical difference (CD) diagram.

Figure 9: Win ratio confusion matrix, Elo scores, and CD diagram of the main investigated models across all 71 semantic-heavy datasets from CARTE and TextTab benchmarks.



(a) Win ratio matrix with Model A wins over Model B (left) and Elo scores (right).



(b) Critical difference (CD) diagram.

Figure 10: Win ratio confusion matrix, Elo score, and CD diagram of the main investigated models across all 87 numeric-heavy datasets from TabArena-Lite and TALENT-Tiny benchmarks.

that the two best models are TabPFNv2.5 and FlexTab-AD. They have very close average rank, only rivaled by TabICLv2. FlexTab-AD scores first in terms of average and median AUC.

Semi-supervised: In the semi-supervised setting, some of the anomalies are labeled as such, while all the other anomalies are labeled in the same way as normal points. Therefore, this setting can also be thought of as noisy binary classification. Looking at the original ADBench evaluation paper [23], the best semi-supervised method turned out to be XGBOD. In fact, since our metric of choice (AUROC) only requires the model to rank the points according to their likelihood of being anomalies, any classification algorithm can be used for this task; XGBOD itself is only a small addition on top of a traditional XGBoost classifier. For this reason, to provide a challenging baseline, we evaluate against state-of-the-art (classification) tabular foundation models that are able to pick up the signal starting from very few labeled anomalies. As already mentioned in Table 3 and further detailed in Figure 12, TabICLv2 in particular performs well with just 5% or 10% of the points being labeled, as does our FlexTab model trained for classification and regression (with TabPFNv2 and ConTextTab following closely). The FlexTab-AD model trained for anomaly detection has a lead in the nearly-unsupervised setup of just 1% of labeled anomalies, but falls slightly behind as the setting approaches full supervision (which is just a standard, if imbalanced, binary classification task).

B.3 Clustering

To evaluate FlexTab for clustering, we use the same codebase and data as TabClustPFN [69]. This consists of a total of 44 datasets of small size (at most 1000 rows, 64 features, and 10 clusters), manually collected from UCI and OpenML sources (22 each). We evaluate against the same baselines presented there (most of which are based on the scikit-learn implementation [47], with the exception of ZEUS [41] for which we use the official implementation), as well as TabClustPFN itself. The metric of choice is ARI.

We remark that clustering evaluation in general, and this evaluation benchmark specifically, is very noisy. Being totally unsupervised, with (classification) labels withheld from the model, it is entirely possible that multiple valid classification tasks could be defined over the same features, and the evaluated metric would change dramatically depending on which one is picked by the model. For

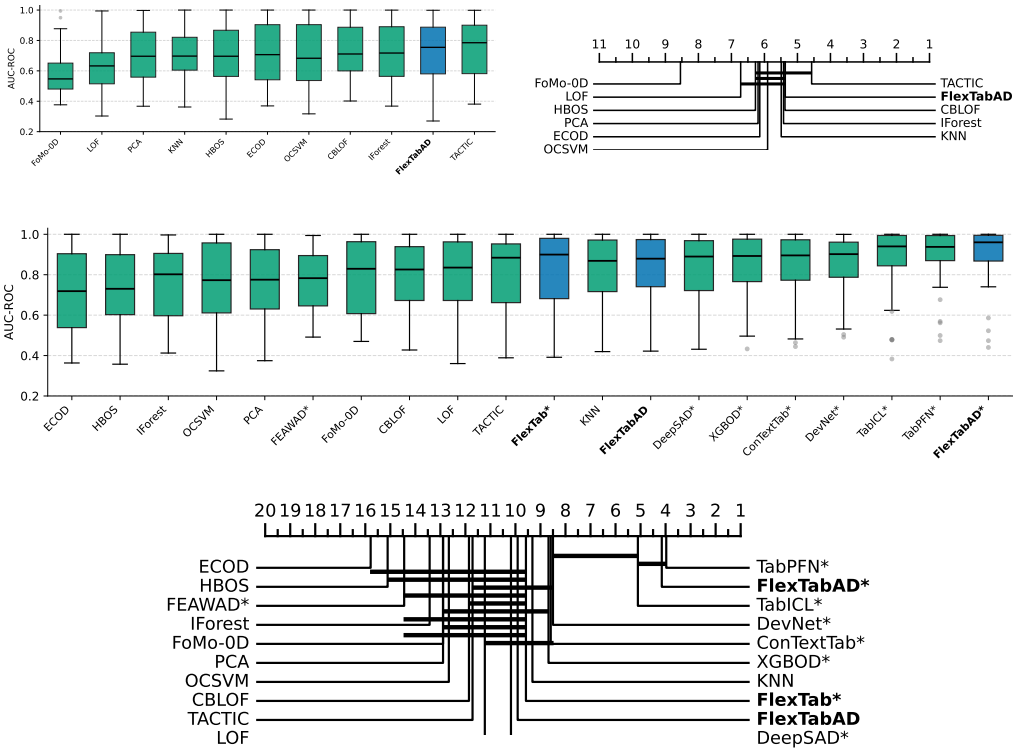


Figure 11: Results for unsupervised anomaly detection (top) and one-class + semi-supervised novelty detection (bottom). The models in the second group (i.e. those allowed to see the test set in a transductive manner) are marked with a (*). Left: box and whisker plots, sorted by average score. Right: critical difference diagrams based on [25].

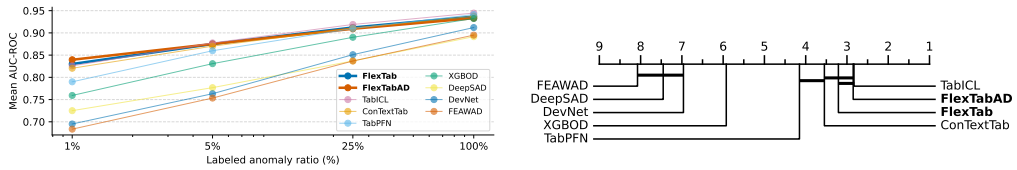


Figure 12: Results for semi-supervised anomaly detection. Left: Average AUROC scores per level of supervision. Right: critical difference diagram (all levels combined) based on [25].

this reason, we chose not to present these results in the main body of the paper, because multiple trainings differing in apparently minor details could lead to significantly different results. Similarly, when retraining TabClustPFN with the officially provided training code, results oscillate significantly (though to a lesser extent than when training FlexTab; probably due to synthetic data being more regular than real training data) between consecutive checkpoints.

For the same reason, the choice of validation set and checkpoint can affect results significantly. We chose to select classification tasks from various benchmarks (TabArena, CC18, TALENT-Tiny) with the only filter that, after applying skrub's TableVectorizer [7] to handle non-numeric fields, the k-means algorithm resulted in an ARI score of at least 0.1. After removing duplicates and any dataset also appearing in the TabClustPFN evaluation set, this resulted in a total of 38 tasks. The best checkpoint based on median ARI was selected after 15.6M steps.

The results are reported in Table 9 (separately for UCI and OpenML) and Figure 13 (jointly). Overall, TabClustPFN shows a small edge in both rank and average, although differences are minimal, not uniform across datasets, and not statistically significant. In fact, several methods, including at least FlexTab and k-means, end up being equivalent, and these results suggest that the clustering task, at

Table 9: Results for clustering: Mean rank and average ARI per dataset.

Model	UCI		OpenML	
	Rank	Avg ARI	Rank	Avg ARI
TabClustPFN	**2.91**	0.44	**3.61**	**0.47**
FlexTab	3.27	**0.47**	4.61	0.44
KMeans	4.86	0.40	3.98	0.45
BIRCH	4.30	0.40	5.59	0.34
Agglomerative	5.30	0.38	4.64	0.38
BisectingKMeans	5.77	0.36	5.02	0.39
ZEUS + KMeans	5.91	0.36	6.45	0.26
AffinityProp	7.95	0.15	8.00	0.13
HDBSCAN	8.41	0.15	8.14	0.15
MeanShift	8.84	0.15	8.20	0.17
DBSCAN	10.02	0.08	9.11	0.10
OPTICS	10.45	0.04	10.64	0.02

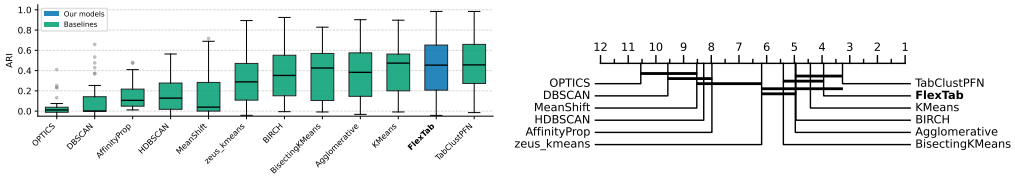


Figure 13: Results for clustering. Left: box and whisker plots, sorted by average score. Right: critical difference diagrams.

least on this benchmark, does not yet clearly benefit from foundation models. FlexTab does have the advantage of being natively able to support non-numeric data; however, this kind of data is absent in the clustering datasets evaluated here.

B.4 Matching

We provide additional results in Table 11. These include TabPFN and TabICL combined with TF-IDF features obtained for high-cardinality columns via Skrub's `TableVectorizer`, as well as the complete model list of the DeepMatcher evaluation [44]. Finally, we also report results using the `TableVectorizer` embeddings directly for matching.

Overall, the results are in line with the distilled ones presented in the main paper. The matching-native FlexTab-Match performs best across all models considered—both in terms of average F1 and rank. Notably, due to the increased number of evaluated baselines, the rank of the DITTO model has worsened slightly, falling behind our approach.

Lastly, we provide pretraining strategy ablation results in Table 10. We see that, depending on the task, the individual task generation strategies yield different outcomes, with no clear superior strategy. This underlines the different matching notions inherent to these datasets, ranging from simple exact (or fuzzy) column value matching, to a more nuanced semantic matching. Combining the two pretraining strategies, equally weighted, yields a hybrid approach that outperforms the corresponding individual ones on almost all datasets considered, and ranks best overall by a clear margin.

B.5 Relational

To better understand how FlexTab-Multi leverages information from auxiliary tables, we conduct a series of ablation studies across the 12 RelBench classification tasks. We investigate three key factors that control the amount of information available to the model: the encoder context size (number of context rows per table), the decoder context size (number of root-table examples with each example represented by multiple `[ROW]` tokens), and the BFS breadth (the maximum number of related rows retrieved per hop during breadth-first search). Mean AUROC across the 12 tasks is reported in Figure 14.

Table 10: FlexTab-Match ablation: training task generation mix results (%, positive-class F1). The hybrid task generation and pretraining objective performs best, combining the best of each of the individual ones. Each dataset is evaluated via 5-fold cross-validation.

	Benchmark					Average	
Model	Febrl4	F.-Zagat	Bikes	eBooks	Movies	F1	Rk
Hidden-Target (100%)	92.3	58.6	85.9	60.1	49.6	69.3	3.0
Same-Row-Match (100%)	99.2	90.3	**86.9**	84.7	**91.5**	90.5	1.4
SRM 50% + HT 50%	**99.3**	**92.5**	86.8	**87.3**	90.2	**91.2**	**1.2**

Table 11: Matching benchmark F1 scores (%, positive class) obtained as 5-fold cross-validation averages for each dataset. The average per-group rank is denoted as GRk whereas the average rank across all investigated models is denoted as Rk.

	Benchmark					Average		
Model	Febrl4	F.-Zagats	Bikes	eBooks	Movies	F1	GRk	Rk
In-context learners								
TabPFN	97.3	29.4	81.8	67.9	0.0	55.3	3.6	10.0
TabPFN (TF-IDF)	94.8	0.0	79.6	67.7	31.2	54.7	5.0	11.0
TabICL	95.0	36.9	82.4	73.1	0.0	57.5	3.2	9.6
TabICL (TF-IDF)	76.0	4.7	82.4	67.4	5.9	47.3	5.0	11.4
ConTextTab	83.2	23.2	82.3	60.3	79.0	65.6	4.2	9.6
FlexTab (ours)	79.8	12.7	84.3	72.6	61.0	62.1	3.8	9.4
FlexTab-Match (ours)	**99.3**	**92.5**	**86.8**	**87.3**	**90.2**	**91.2**	**1.0**	**2.4**
Embedding similarity								
NGramHash Embeddings	**99.7**	**88.0**	**69.8**	**88.1**	22.9	73.7	**1.6**	**6.4**
Skrub Embeddings	45.0	61.4	47.6	53.1	31.4	47.7	3.4	12.6
TabPFN Embeddings	71.8	41.2	50.8	56.4	24.0	48.9	3.2	12.6
FlexTab Embeddings (ours)	98.5	81.5	54.4	84.9	**78.9**	**79.6**	1.8	7.8
Per-dataset trained								
Deepmatch SIF	99.5	0.0	0.0	35.6	0.0	27.0	4.2	12.4
Deepmatch RNN	99.9	89.5	86.0	88.1	80.2	88.8	2.2	2.8
Deepmatch Attn	99.9	89.5	82.1	84.0	83.4	87.8	2.6	4.0
Deepmatch Hybrid	99.8	87.1	87.7	88.5	**87.6**	**90.2**	2.0	2.6
DITTO	**100.0**	**94.7**	**95.5**	**89.5**	44.9	84.9	**1.6**	**2.4**

Encoder context size: As shown in the top-left plot of Figure 14, performance improves as the encoder context size grows from 32 to 32 768 rows while keeping the decoder context size fixed at 16 384 and BFS breadth fixed at 16. This indicates that the encoder effectively leverages larger intra-table context to produce richer row embeddings. Although we observe gains as the context size is increased, 8192 is used as the default value in our experiments to obtain a good trade-off between predictive performance and computational cost, since every table in the schema must be independently embedded by the encoder and the cost of this step grows with the context size.

Decoder Context Size: The top-right plot shows a similar monotonic trend as the decoder context is expanded from 32 to 32 768 rows, with mean AUROC rising steadily up to 16 384 rows. In this experiment, the encoder context size is fixed at 8192 and the BFS breadth to 16. This confirms that the decoder can successfully aggregate signal from a growing set of `[ROW]` embeddings up to a saturation point where additional context does not provide further improvement.

BFS Breadth: The bottom plot examines the effect of capping the number of related rows retrieved per hop during BFS across the database. Increasing the BFS breadth from 0 (no auxiliary rows, i.e., root-table-only prediction) to 16 yields a substantial improvement from ~61 AUROC at breadth 0 to ~71 AUROC at breadth 16. This clearly demonstrates that the FlexTab-Multi decoder effectively leverages the additional `[ROW]` tokens from auxiliary tables retrieved via BFS, validating our design choice of decoupling per-table encoding from cross-table relational reasoning at the decoder level.

Overall, these ablations confirm that FlexTab-Multi's performance scales favorably with both the intra-table context provided to the encoder and the number of auxiliary rows aggregated by the decoder. Notably, the monotonic improvement with BFS breadth highlights that, despite being

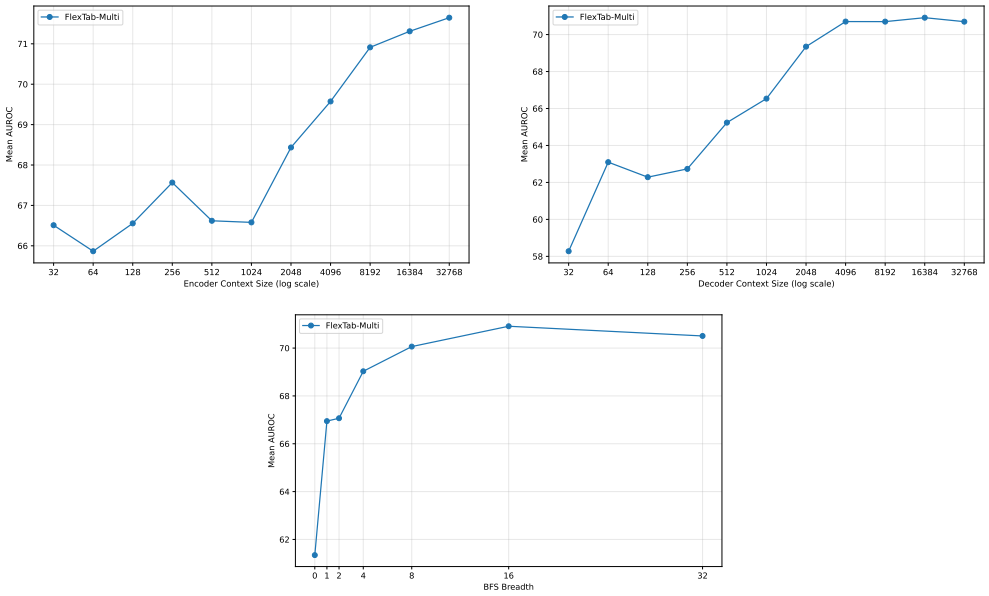


Figure 14: FlexTab-Multi ablation across 12 RelBench tasks showing how scaling input data affects mean AUROC. (Top left) Effect of encoder context size. (Top right) Effect of decoder context size. (Bottom) Effect of BFS breadth. X-axes in the top left and top right plots are shown in log scale.

Table 12: Retrieval influence on classification and regression performance, depicting mean accuracy (Acc) and (soft-clipped) R^2 score, in percent, as well as mean per-task rank.

Model Name	All	CARTE			TabArena			TALENT			TextTab		
	Rank	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2	Rank	Acc	R2
FlexTab [k=1024, retrieval]	**2.0**	**1.9**	**77.3**	**75.5**	**1.7**	**87.6**	**80.4**	2.7	87.7	**86.3**	**1.9**	**84.2**	70.0
FlexTab [k=8192, random]	2.7	2.8	77.0	73.6	2.4	87.0	79.1	**2.2**	**87.9**	85.5	3.5	83.7	61.4
FlexTab [k=512, retrieval]	3.0	2.8	77.0	74.9	3.3	87.2	79.9	3.2	87.1	85.0	2.2	83.9	72.1
FlexTab [k=4096, random]	3.3	4.1	76.5	72.1	2.7	87.3	78.5	2.7	87.7	85.0	3.7	83.5	60.0
FlexTab [k=256, retrieval]	3.7	4.0	77.0	74.1	3.5	87.4	79.6	3.7	86.7	84.4	3.8	83.4	**72.7**
FlexTab [k=2048, random]	4.6	5.9	75.7	69.8	3.8	87.1	76.9	4.1	86.5	84.2	4.3	82.6	59.3
FlexTab [k=128, retrieval]	4.9	5.2	76.4	72.9	4.6	87.3	78.6	5.2	86.4	82.4	4.6	82.6	71.2
FlexTab [k=1024, random]	6.0	7.5	74.3	66.8	4.7	86.7	75.0	6.0	85.4	83.2	5.5	81.3	57.1

pretrained only on chunks derived from single tables, the decoder generalizes to multi-table relational reasoning at inference time.

B.6 Retrieval

To further examine the meaningfulness of embeddings, in Table 12 we show that retrieval with FlexTab embeddings outperforms random retrieval across various context sizes.

B.7 Runtime Analysis

We conducted a runtime analysis comparing FlexTab with TabPFN, TabICL, ConTextTab, and two trainable baselines, XGBoost and RealMLP, with results presented in Figure 15. To support this analysis, we generated a synthetic dataset with 100 columns split evenly between categorical/textual and numerical features, while varying the number of training/context rows from 1000 to 10 000. Each test was performed on a single query row at a time across different context sizes, averaging over 10 repetitions for each table size. All experiments ran on a compute node equipped with 40 CPU cores, 320 GB of RAM, and an H100 GPU with 96 GB of VRAM.

Depending on the table size, FlexTab runtimes range from ~0.1s to ~4s. Our findings indicate that FlexTab is approximately twice as fast as TabPFNv2.6, which is mostly attributable to TabPFNv2.6

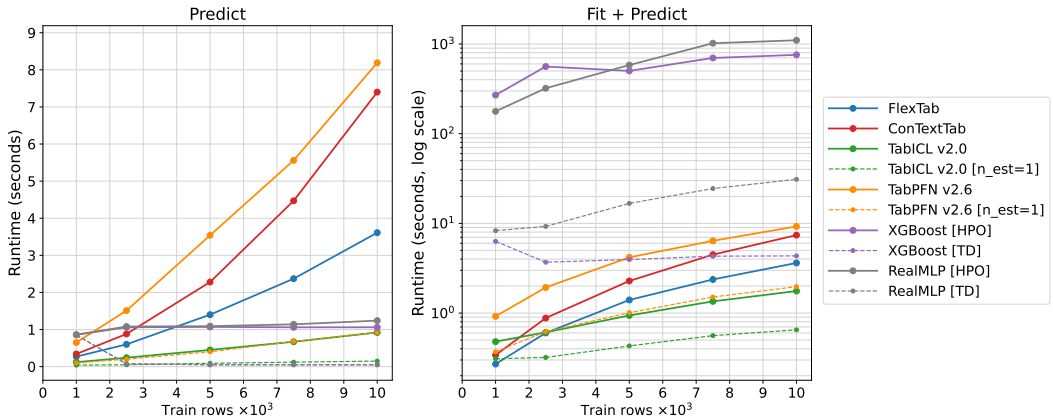


Figure 15: Runtime comparison of different models across varying numbers of training rows. (Left) Runtime for model prediction. (Right) Runtime for model prediction, including fitting. Note the linear y-axis scale on the left side as opposed to the log-scale on the right.

using 24 layers with 8 estimators, while our model uses a 12-layer encoder with a larger hidden dimension of 768 and bagging of 1. We also achieve a ~2× speedup over ConTextTab for a larger context size of 10k, which is a consequence of our use of flash attention, whereas ConTextTab relies on memory-efficient attention combined with a custom attention mask. As a result, ConTextTab exhibits quadratic scaling, while this trend is far less pronounced for the other models. TabICL is clearly the fastest ICL model in this comparison, as it uses a different architecture. Our experiment reveals that most of FlexTab's runtime is spent in the encoder, which processes all cells in the table, whereas the decoder is notably faster since it operates only on `[ROW]` tokens and the target column.

We additionally report runtimes for TabPFN and TabICL using a single estimator instead of the default of 8. While this yields an almost linear reduction in runtime, we cannot guarantee comparable predictive performance in this configuration. In contrast, FlexTab employs bagging with 8 estimators only for larger contexts exceeding 8192 rows, and in this experiment, for none of the reported runtimes. Inference runtime could be further reduced through complementary techniques such as KV caching or model compilation.

Note that, while non-ICL models offer significantly faster prediction times, they require extensive training. In particular, the hyperparameter-optimized (HPO) versions of XGBoost and RealMLP exhibit fit+predict times that are two orders of magnitude longer.

B.8 Pretraining Ablations

In this section we briefly list some experiments we ran to evaluate the impact of our curriculum-learning approach against more direct approaches that do not reuse a frozen encoder pre-trained for classification and regression tasks.

No encoder As a first experiment, we replace the encoder part of the architecture by a simple average of the tokens in each row. Therefore, only the embedding and decoder layers (cross-attending to the average of the embeddings) are maintained. This training run was quickly aborted: after around 4 million micro-steps (15 thousand optimizer steps) the training collapsed and was unable to recover.

Shuffled weights As a further sanity check that learned encoder weights are needed, we tried shuffling them (while maintaining embedding and decoder weights frozen), to see if the inductive bias of the architecture, plus a learned decoder, could be enough to retrieve some prediction quality. Unsurprisingly, the answer is negative, with the overall model collapsing to random-level predictions (or worse).

Shuffled weights and fine-tune In the same setup of the previous point, we then tried fine-tuning the decoder, to see if the performance could be retrieved. This training failed even faster than the first one, with performance collapsing before 1 million micro-steps, never to recover.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Through thorough experimentation covering a wide range of tabular prediction tasks (classification, regression, anomaly detection, matching, and relational prediction), including analysis of statistical significance in the core areas, we show the efficacy of our flexible architecture. In addition, we provide ablations on key architectural ingredients such as the row embedding aggregation and investigation of asymmetric encoder-decoder layouts.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide discussion of limitations and future works in the conclusion section, as well as in more detail in the appendix – most importantly regarding potential information/compression bottlenecks of our architecture as well as potential data contamination of our pretraining data with respect to the evaluation datasets used.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: Our paper does not include theoretical results

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in the appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our architecture and training strategies, both in the main paper and include extensive details in the appendix. Further, we provide all details on the datasets and baselines investigated. We will make the inference code and model checkpoints publicly available, which will enable the community to verify our results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in the supplemental material?

   Answer: [Yes]

   Justification: The inference code and model checkpoints will be made publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all necessary details regarding the data and hyperparameter optimization strategies used for the affected baseline models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide, where applicable, error bars obtained via bootstrapping, as well as critical-difference diagrams for our core results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide compute details and time of execution details on all training setups. We do not provide inference-time estimates as we did not track them.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we reviewed and comply with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: Beyond the societal impact that any tabular prediction model may have, our approach does not yield any specific potential societal impact unique to our work. Hence, we do not discuss general broader impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we report all data and code sources used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Does not apply.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.