

# LimiX: Unleashing Structured-Data Modeling Capability for Generalist Intelligence

LimiX Team

Stable AI & Tsinghua University

-  <https://github.com/limix-ldm/LimiX/>
-  <https://huggingface.co/stableai-org/>
-  <https://modelscope.cn/organization/stable-ai/>

## Abstract

We argue that progress toward general intelligence requires complementary foundation models grounded in language, the physical world, and structured data. This report presents LimiX, the first installment of our large structured-data models (LDMs). LimiX treats structured data as a joint distribution over variables and missingness, thus capable of addressing a wide range of tabular tasks through query-based conditional prediction via a single model. LimiX is pretrained using masked joint-distribution modeling with an episodic, context-conditional objective, where the model predicts for query subsets conditioned on dataset-specific contexts, supporting rapid, training-free adaptation at inference. We evaluate LimiX across 10 large structured-data benchmarks with broad regimes of sample size, feature dimensionality, class number, categorical-to-numerical feature ratio, missingness and sample-to-feature ratios. With a single model and a unified interface, LimiX consistently surpasses strong baselines including gradient-boosting trees, deep tabular networks, recent tabular foundation models, and automated ensembles, as shown in [Figure 1](#) and [Figure 2](#). The superiority holds across a wide range of tasks, such as classification, regression, missing value imputation, and data generation, often by substantial margins, while avoiding task-specific architectures or bespoke training per task. All LimiX models are publicly accessible under Apache 2.0.

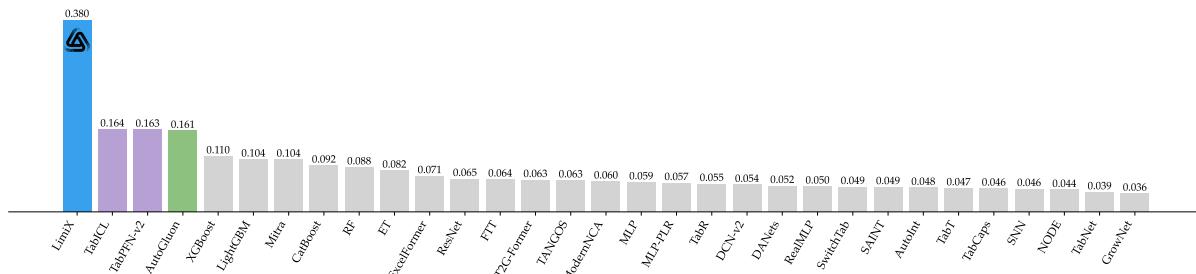


Figure 1: Performance comparison on the averaged reciprocal of the ranks, where the rank is that of the corresponding model on ROC AUC. Higher values indicate stronger average ranking performance across all the classification benchmarks.

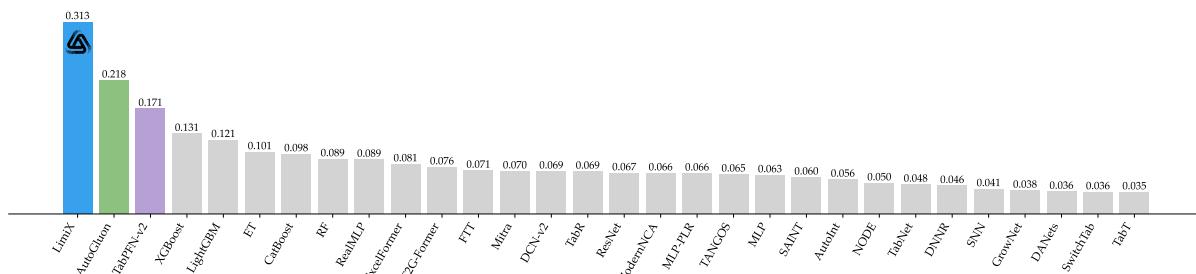


Figure 2: Performance comparison on the averaged reciprocal of the ranks, where the rank is that of the corresponding model on  $R^2$  across all the regression benchmarks.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Architecture</b>	<b>4</b>
2.1	Embedding of Tabular Data . . . . .	4
2.2	Discriminative Feature Encoding . . . . .	4
2.3	Model Structure . . . . .	5
<b>3</b>	<b>Pretraining</b>	<b>5</b>
3.1	Context-Conditional Masked Modeling for Joint Distribution Learning . . . . .	5
3.2	Mask Pattern Design . . . . .	6
3.3	Mask Embedding . . . . .	6
<b>4</b>	<b>Pretraining Data Generation</b>	<b>6</b>
4.1	DAG Generation based on Hierarchical SCMs . . . . .	7
4.2	Data Sampling . . . . .	8
4.3	Task Adaptation . . . . .	8
<b>5</b>	<b>Retrieval-based Ensemble</b>	<b>8</b>
<b>6</b>	<b>Theoretical Analysis on Context-Conditional Masked Modeling</b>	<b>9</b>
6.1	Modeling the Joint Conditional with Random Masks . . . . .	10
6.2	The Choice of Mask Number . . . . .	11
<b>7</b>	<b>Evaluation</b>	<b>11</b>
7.1	Classification . . . . .	11
7.2	Regression . . . . .	26
7.3	Missing Value Imputation . . . . .	34
7.4	Robustness . . . . .	35
7.5	Embedding . . . . .	36
7.6	Fine-tuning . . . . .	37
7.7	Data Generation . . . . .	40
7.8	Out-of-Distribution Generalization . . . . .	41
<b>8</b>	<b>Conclusion</b>	<b>41</b>
<b>9</b>	<b>Contribution</b>	<b>42</b>
<b>A</b>	<b>Experimental Details</b>	<b>49</b>
A.1	Details of Datasets with Distribution Shifts . . . . .	49
A.2	Hyperparameter Search Space for Baselines . . . . .	49
<b>B</b>	<b>Omitted Details in Section 6</b>	<b>50</b>
B.1	Omitted Details in Section 6.1 . . . . .	50
B.2	Omitted Details in Section 6.2 . . . . .	51

---

## 1 Introduction

We posit that progress toward general intelligence is best organized around three complementary spaces: language, physical-world, and structured data, each anchored to a distinct data modality and set of inductive biases. In the language space, large language models (LLMs) provide a universal interface for natural and programming languages and have rapidly advanced instruction following, tool use, and explicit reasoning over token sequences (Achiam et al., 2023; Touvron et al., 2023; Team et al., 2023; Bai et al., 2023). In the physical-world space, recent foundation models ground knowledge in space perception and embodied reasoning. They emphasize spatial intelligence through structured scene understanding, controllable scene generation, and neural radiance field reconstruction (Mildenhall et al., 2020; Ke et al., 2025; Xiang et al., 2025; Li et al., 2024a; Yi et al., 2024). These models also include self-supervised video world models such as V-JEPA, which learn predictive abstractions from large-scale videos and support downstream planning and control (Bardes et al., 2024; Assran et al., 2025).

On the other hand, structured data serves as the foundational bedrock for evidence-based decision-making across a multitude of critical domains, including finance, healthcare, logistics, and public policy (Fuster et al., 2022; Johnson et al., 2016; Yu et al., 2021; Krafft et al., 2020). The structural consistency and inherent order of structured data provide a robust framework for quantitative analysis and reliable operations (Ramakrishnan et al., 2003; Silberschatz et al., 2011; Stonebraker & Çetintemel, 2018), enabling precise prediction, automated reasoning, and rigorous causal inference (Pearl, 2009; Hernán & Robins, 2010; Little & Rubin, 2019). While the emergence of unstructured data has captured considerable attention, the analytical power and operational reliability of structured data remain unparalleled for a vast array of real-world applications (Fang et al., 2024; Van Breugel & Van Der Schaar, 2024). Consequently, advancements in structured-data prediction, analysis, and reasoning are not merely an academic pursuit but a critical enabler for efficiency, innovation, and accuracy in modern data-driven systems. Structured data is not subsumed by language models or embodied intelligence. Converting tables into free text discards metric geometry, physical units, and patterns of missingness that are central to reliable prediction, while models focusing on perception and control in three-dimensional physical environments do not capture discrete interventions, business rules, or causal heterogeneity across environments. Empirical surveys also document current limitations of language models on tabular prediction without bespoke adaptation (Fang et al., 2024; Sui et al., 2024).

Traditionally, practitioners deploy pipelines of specialized models with gradient-boosting trees and automated ensembles such as XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Dorogush et al., 2018), and AutoGluon (Erickson et al., 2020)—that are trained separately for each dataset and task. These systems excel at supervised prediction but require full retraining on every new dataset, prolonging deployment and preventing reuse of knowledge across domains. Recent deep approaches for tables have improved accuracy on mixed-type data (Huang et al., 2020; Yoon et al., 2020; Gorishniy et al., 2021; Somepalli et al., 2022; Bahri et al., 2022). However, they are still typically trained per dataset and do not provide a single model that transfers across objectives and constraints.

These limitations motivate the pursuit of a foundation model for structured data, i.e. models trained on large families of datasets to perform in-context learning (ICL) without per-task fine-tuning. Notably, TabPFN (Hollmann et al., 2022) and its successor TabPFN-v2 (Hollmann et al., 2025) demonstrate state-of-the-art performance and speed on small-to-medium-scale tables via prior-data fitting over diverse generative processes, and community efforts increasingly argue for tabular foundation models as a distinct paradigm (Van Breugel & Van Der Schaar, 2024). Concurrent works such as TabDPT (Ma et al., 2024) and TabICL (Qu et al., 2025) explore scaling these ideas to real and larger datasets, narrowing the gap between tabular foundation models and classical methods under the scenarios of larger sample sizes.

Despite the progress, these are still in the early stages of foundation model development and remain limited in generality and performance. Most models are developed and evaluated primarily for supervised prediction (classification or regression), and typically require per-task models, adapters, or external pipelines to address other objectives. In practice, one still needs to assemble separate components for classification, regression, missing value imputation, data generation, and sample selection for interpretability, with different training losses and hyperparameters, so the resulting system is not a single reusable learner that delivers all of these capabilities end-to-end while maintaining reliable performance. This gap motivates a large structured-data model (LDM) that treats structured data as a joint distribution over variables and missingness, enabling multiple tasks to be posed as queries to one model.

In this work, we introduce LimiX, the first installment of our LDM series. LimiX aims to push generality further: a single model capable of classification, regression, missing value imputation, data generation, and sample selection for interpretability under one training and inference recipe, shifting the paradigm from bespoke pipelines to unified, foundation-style tabular learning. LimiX adopts a lightweight, scalable architecture that represents structured data as a set of sample–feature embeddings and learns dependencies across two dimensions: across features (columns) and across samples (rows). To make

the attention module explicitly column-aware without inflating parameters, we introduce a low-rank discriminative feature encoding that encodes feature identities. Pretraining of LimiX follows a masked joint-distribution objective and an episodic, context-conditional formulation: For each dataset, an in-context subset establishes dataset-specific priors, and the model is trained to predict masked entries in a disjoint query subset, enabling per-dataset adaptation without fine-tuning at inference. The pretraining corpus consists of data synthesized from hierarchical structural causal models (SCMs). Within the synthesis pipeline, we employ graph-aware sampling to obey the causal structure and solvability-aware sampling to accommodate the data quantity of various downstream tasks, improving coverage of local patterns and generalization. At inference, attention-guided retrieval provides an efficient, optional ensemble and fine-tuning mechanism. LimiX retrieves informative samples and features using its own attention scores, aggregates predictions across a handful of lightweight pipelines, and delivers calibrated outputs for various downstream tasks, all through a unified conditional-inference interface and without task-specific architectures or bespoke per-dataset training.

We evaluate LimiX on 10 large structured-data benchmarks with broad regimes of sample size, feature dimensionality, number of classes, categorical-to-numerical feature ratios, missing values, and sample-to-feature ratios. Results show that LimiX demonstrates the strongest predictive performance on all the benchmarks. With a single model and a unified inference interface, LimiX surpasses competitive baselines including gradient-boosting trees, deep tabular networks, recent tabular foundation models, and automated ensemble methods. Across classification, regression, missing value imputation, data generation, and out-of-distribution prediction, LimiX delivers consistent gains, often by large margins, while avoiding task-specific model architectures, customized ensembles, or per-dataset training. On most benchmarks, such as OpenML-CC18 (Bischl et al., 2017), TabArena (Erickson et al., 2025), TALENT-REG (Liu et al., 2024), LimiX is the only model that consistently outperforms AutoGluon, which is considered an outstanding baseline across various tabular-data tasks.

## 2 Architecture

We consider a dataset  $\mathcal{D} = \{(\mathbf{x}^R, \mathbf{y}^R)\}$  of  $d$  features and an outcome variable, where  $\mathbf{x}^R = \{\mathbf{x}_i^R\}_{i=1}^m$ ,  $\mathbf{y}^R = \{y_i^R\}_{i=1}^m$  and the superscript  $R$  indicates the raw input. Here,  $\mathbf{x}_i^R \in \mathbb{R}^d$  and  $y_i^R \in \mathbb{R}$  correspond to the  $i^{th}$  sample, while  $\mathbf{x}^R \in \mathbb{R}^{m \times d}$  and  $\mathbf{y}^R \in \mathbb{R}^m$  correspond to the 2D tabular data. For in-context samples and test samples, we use subscripts to distinguish between them. For example,  $\mathbf{x}_{ct}^R \in \mathbb{R}^{m_{ct} \times d}$  denotes the features of in-context samples, while  $\mathbf{x}_{te}^R \in \mathbb{R}^{m_{te} \times d}$  denotes those of the test samples.

### 2.1 Embedding of Tabular Data

To ensure compatibility with modern architectures such as Transformers (Vaswani et al., 2017), each cell of the 2D tabular input  $x_{i,j}^R \in \mathbb{R}$  is first projected into a latent embedding space  $\mathbb{R}^p$ . Specifically,  $\mathbf{x}^R \in \mathbb{R}^{m \times d}$  is transformed into  $\mathbf{x} \in \mathbb{R}^{m \times d \times p}$  and  $\mathbf{y}^R \in \mathbb{R}^m$  is transformed into  $\mathbf{y} \in \mathbb{R}^{m \times p}$ . All subsequent attention operations are then conducted within this latent embedding space. Such a high-dimensional embedding space could strengthen the expressivity of the model. Concretely, we employ a two-layer MLP with LayerNorm (Ba et al., 2016) as the embedding module, i.e.  $\mathbf{x}_{i,j} = \sigma(LN(\sigma(LN(\mathbf{x}_{i,j}^R \mathbf{W}^{(1)} + \mathbf{b}^{(1)})) \mathbf{W}^{(2)} + \mathbf{b}^{(2)}))$ , where LN is LayerNorm and  $\sigma$  is the GELU activation function (Hendrycks & Gimpel, 2016). Separate embedding modules are used for  $\mathbf{x}^R$  and  $\mathbf{y}^R$ .

### 2.2 Discriminative Feature Encoding

The attention score between feature  $j$  and  $j'$  for sample  $i$  is  $\frac{1}{\sqrt{p}} \langle \mathbf{x}_{i,j} \mathbf{W}_Q, \mathbf{x}_{i,j'} \mathbf{W}_K \rangle$ . This score depends solely on the interactions between the embeddings of cell values and imposes no explicit prior on feature (column) identity. As a result, the model cannot reliably infer the column from which a cell value originates. Inspired by Hollmann et al. (2025), we introduce a learnable low-rank column identifier termed discriminative feature encoding (DFE) whose design philosophy includes two objectives: (i) Encodings of different features should be well separated to ensure discriminability; (ii) Encodings in the embedding space admit a low effective rank so that the model can express column identities compactly and share statistical strength across features.

For implementation, let  $s$  denote the rank of DFE, and  $s \ll p$  compared with the dimension of the embedding space  $p$ . We initialize a matrix  $\mathbf{u} \in \mathbb{R}^{d \times s}$  whose  $j^{th}$  row vector  $\mathbf{u}_j \in \mathbb{R}^s$  is a low-dimensional code of the column identifier corresponding to feature (column)  $j$ . Rows are initialized to be approximately orthogonal and normalized. Then a linear transformation  $\mathbf{E} \in \mathbb{R}^{s \times p}$  lifts the code from the

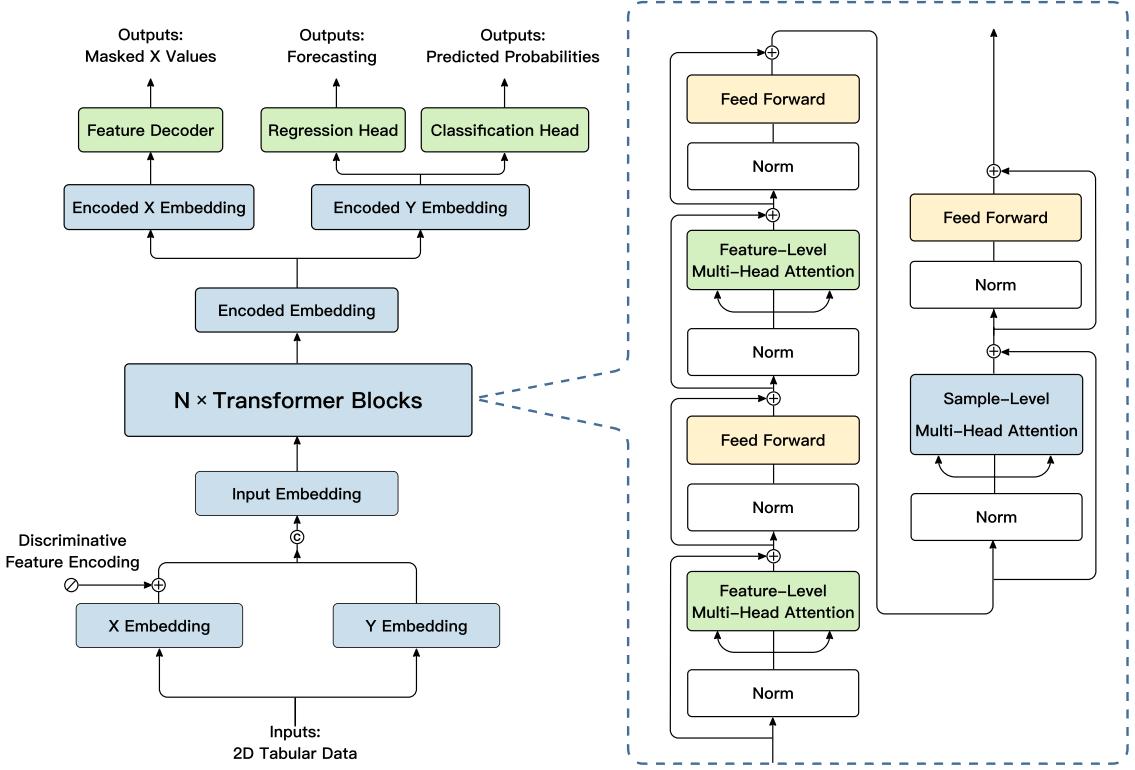


Figure 3: The overall model structure of LimiX.

low-dimensional code space to the high-dimensional embedding space, i.e.  $\mathbf{e}_j = \mathbf{u}_j \mathbf{E} \in \mathbb{R}^p$ , serving as the DFE for feature  $j$ . Finally, the embedding of cell  $(i, j)$  is augmented additively as  $\tilde{\mathbf{x}}_{i,j} = \mathbf{x}_{i,j} + \mathbf{e}_j$ , which is analogous to absolute positional encodings but is applied along the feature axis. Empirically, we set  $s = p/4$  by default.

### 2.3 Model Structure

As shown in Figure 3, LimiX comprises 12 transformer blocks. Each block performs axis-wise self-attention along the feature axis twice and along the sample axis once, and each attention sublayer is followed by a position-wise feed-forward network (FFN). We adopt this asymmetric design with two feature-level passes and one sample-level pass because ablations show that using equal numbers underrepresent feature interactions, whereas doubling attention along the feature axis improves modeling capacity on heterogeneous schemas with minimal overhead. All sublayers use LayerNorm (Ba et al., 2016) in a pre-normalization configuration to stabilize optimization and support scaling in depth.

## 3 Pretraining

### 3.1 Context-Conditional Masked Modeling for Joint Distribution Learning

We pretrain LimiX by randomly masking cells in each row and training the model to recover the hidden entries from the visible context. Exposing the model to various mask patterns forces it to master a wide spectrum of conditional dependencies among variables. When these conditionals are learned properly, they effectively define a single joint model of the data. This joint model can then be queried for diverse tasks with one mechanism: Treat a chosen column as the target to perform regression or classification, fill in missing values by predicting the masked cells from the observed ones, and generate new samples by iteratively masking and refilling subsets of features.

To better align pretraining with inference, we adopt a Context-Conditional Masked Modeling (CCMM) objective. For each dataset, an episode splits rows into a context subset and a query subset. The model encodes the context and conditions predictions for the query rows on this context through attention or

---

feature modulation, learning to answer queries of the form “predict masked entries in the query given the observed query features and the context”. This episodic formulation enables rapid and label-free adaptation to new datasets at inference time: A handful of context rows establish dataset-specific priors such as category frequencies, marginal scales, and cross-feature couplings, while a single parametric model serves all schemas and tasks.

Unlike BERT-style masked modeling (Devlin et al., 2019), where adaptation is implicit in parameters accumulated during pretraining, CCMM considers context as a non-parametric memory that the model can consult during inference. This yields per-dataset calibration, better handling of rare categories, and improved robustness to distribution shift without gradient updates, aligning with principles of in-context learning and meta-learning for mixed-type tabular data.

We also find that CCMM improves modeling of the underlying dependency structure compared with recent tabular foundation models (Hollmann et al., 2025; Qu et al., 2025). By demanding consistency across numerous conditional predictions of all features rather than optimizing the loss of a single conditional prediction of a prefixed feature, the model is compelled to capture stable variable–variable relations instead of brittle decision boundaries. As the coverage of masks becomes richer, these cross-conditional constraints get tightened, yielding more reliable recovery of the joint distribution of all the features and more stable estimates in downstream usages. Please refer to Section 6 for details.

### 3.2 Mask Pattern Design

We employ a heterogeneous schedule that interleaves cell-wise, column-wise, and block masks, enabling the model to practise recovering both isolated entries and coordinated subsets of variables. Cell-wise masks refine local conditional predictions; column-wise masks force the model to treat an entire feature as missing and infer it from the remaining attributes; block masks target higher-order dependencies across semantically related groups (e.g., demographics with outcomes, laboratory panels with diagnoses). Masking rates are stratified by variable type, prevalence, and dispersion to avoid overfitting to common categories and to limit the influence of high-variance continuous features, and we exclude degenerate patterns that remove nearly all informative context. This diversified schedule provides broad coverage of conditional relationships and prevents the model from specializing to a narrow reconstruction regime, yielding a more faithful approximation of the joint distribution. In practice, we sample the mask ratio in [0.1, 0.4] for LimiX.

### 3.3 Mask Embedding

To model masked cells, we introduce learnable mask embeddings that explicitly mark positions to be predicted. For each masked entry, its embedding is replaced by a trainable mask vector that is combined with the column embedding, so the encoder can condition on what is missing as well as where. We align the training masks introduced by the objective with naturally missing/structurally unavailable values observed in real data by using the shared mask embedding and calibrating the masking schedule to match empirical missingness patterns, thereby minimizing distribution shift between synthetic and real scenarios.

The mask embeddings flow through the same attention blocks as observed cells, enabling the model to request information from relevant columns and to produce calibrated distributional predictions at the output head. To reduce the mismatch between pretraining and fine-tuning, we condition the network on the corruption level of each dataset using a mask density feature, defined as the proportion of cells that are masked in the dataset. This scalar is encoded via a lightweight statistics token whose embedding is conditioned on the mask density with a small MLP module. The mask ratio embedding is included alongside the data tokens and participates in self-attention. Providing this cue regularizes the model across a range of masking rates and reduces pretraining to inference mismatch, which improves calibration when the inference pattern, such as masking a single target column, differs from the heavier masks used during pretraining.

## 4 Pretraining Data Generation

The performance of foundation models largely depends on the diversity and quality of pretraining data. To obtain a well-generalized foundation model for tabular data, we generate synthetic datasets using Directed Acyclic Graphs (DAGs), enabling the creation of datasets with diverse characteristics. The data generation process consists of three stages: DAG generation, data sampling, and task adaptation. First, a DAG is constructed to represent complex causal dependencies among variables. Then, a subset of these variables is sampled to define a specific problem, allowing LimiX to develop causal reasoning capabilities. Finally, the sampled data is processed to align with various downstream tasks. Compared with the

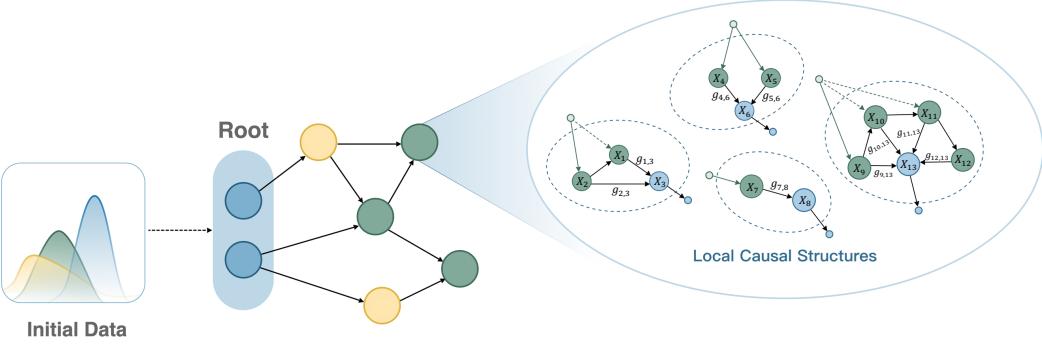


Figure 4: An example of the generated DAG, where  $g_{i,j}$  is an edge function defining the relationship between a parent node  $X_i$  and its child node  $X_j$  in the local causal structures.

data generation methods used in current foundation models (Hollmann et al., 2025; Qu et al., 2025), we adopt a hierarchical generation paradigm that establishes causal dependencies in a more controllable and interpretable manner, while the sampling strategies further enhance the ability to generate datasets with diverse solvability and characteristics.

#### 4.1 DAG Generation based on Hierarchical SCMs

The theoretical foundation of data generation lies in structural causal models (SCMs). As illustrated in Figure 4, within a DAG, initial data for each root node is independently sampled from an assigned distribution, with the distributions themselves chosen from a collection parameterized by randomized hyperparameters. Beginning at the root nodes, the initial data propagates through the DAG along its directed edges, wherein each encountered node represents a distinct local causal structure (LCS). As previously mentioned, the diversity of synthetic data is crucial for effective pretraining. To ensure that DAG generation is both diverse and well-structured, we adopt a hierarchical generation scheme rather than generating DAGs directly, thereby allowing for more fine-grained controls over the generation process. Within the LCS, the data input first propagates to the connected parent nodes, and the data of the child node can be obtained through  $X_i = f(\{g_{i,k}(X_k)_{k \in \text{PA}(X_i)}, \epsilon_i\})$  where  $g_{i,k}(\cdot)$  represents the edge function from  $X_k$  to  $X_i$ ,  $\text{PA}(X_i)$  denotes the set of parent nodes of  $X_i$ ,  $f(\cdot)$  is a parameterized aggregation function, and  $\epsilon_i$  is an observational noise. This framework allows us to capture diverse local causal dependencies. For example, if an LCS contains only one parent node, that node is the direct cause of the child, making the dependency clear and straightforward. However, when multiple parent nodes are included in the LCS, the causal relationship becomes more complex due to the presence of potential confounding variables within the network.

In addition to the structural properties of LCSs, edge functions and the aggregation function also play crucial roles in shaping the dependencies. In this work, we use three types of edge functions:

- **Multilayer perceptrons (MLPs):** To determine the architecture of these neural networks, we uniformly sample properties such as the number of linear layers and their associated activation functions. When multiple layers are included, the MLP introduces complex nonlinear dependencies along the edge, whereas with only a single layer, it degenerates into a simple transformation. For weight initialization, we randomly choose from Xavier initialization (Glorot & Bengio, 2010) and He initialization (He et al., 2015). The activation functions are uniformly sampled from identity mapping, hyperbolic tangent, sigmoid, logarithm, absolute value, sine, squaring, modulo operation, heaviside step function, and GELU.
- **Convolutional layers:** For node-level tabular data, the convolution operation is applied along the sample dimension, serving as a local information mixer. The choices of weight initialization and activation functions follow the same procedure as those employed for MLPs.
- **Decision trees:** These are employed to introduce rule-based mappings and can take the form of either classification or regression trees. Unlike some approaches that fit decision trees on random data, we argue that model fitting is neither necessary nor computationally efficient for the purpose of introducing rule-based dependencies. In this work, decision trees are fixed once constructed, with their hyperparameters sampled independently for each edge.

Once the child node receives the mapped values, they are aggregated using an aggregation function selected from one of three options: simple average, weighted average, or MLP-based aggregation. With

---

respect to observational noise introduced at each edge, rather than adding noise of a fixed magnitude, we generate noise whose magnitude is scaled according to the distribution of each feature. To construct the DAG from all LCSs, we begin with a single-LCS DAG. Each time a new LCS is incorporated, it may connect to one or multiple existing LCSs in the DAG, whereby the child node of the new LCS is linked to the parent nodes of the target LCSs. This process is repeated until no LCSs remain to be added. In the resulting structure, all nodes with an in-degree of zero serve as the root nodes.

## 4.2 Data Sampling

Determining how to sample a subset of the generated data as the training set is also a critical challenge. Although random sampling is possible, the resulting training data often deviates significantly from being truly representative or good for training models. Thus it is necessary to devise an efficient strategy to sample high-quality training data for pretraining. We employ two sampling strategies: graph-aware sampling and solvability-aware sampling. Graph-aware sampling can be regarded as a more advanced variant of random sampling, where the key difference lies in its consideration of the graphical distribution of the sampled training data, which constrains the sampling space and renders it significantly smaller than in the case of random sampling. In contrast to graph-aware sampling, solvability-aware sampling aims to provide training data with varying degrees of solvability, thereby encouraging the model to achieve better generalization. From this point, we divide the subsampled problems into three classes: high-solvability, moderate-solvability and low-solvability problems. We ensure that the sampling ratios of these classes follow a categorical distribution, each of whose parameters is sampled from a distinct Gaussian distribution. In practice, we alternate between the two sampling strategies according to a predefined probability.

## 4.3 Task Adaptation

During the data generation process, the sampled data may be intended for either classification or regression tasks, and the major difference is the processing of the target variable  $y$ , which is initially sampled as a continuous variable. For classification tasks, it is subsequently discretized into categorical variables. For regression tasks, the procedure differs slightly. If the sampled  $y$  is already a discrete variable, a regression task will not be constructed; In cases where  $y$  is continuous but clusters closely around a limited set of distinct values, an in-order transformation can be applied to achieve a more uniform distribution across the magnitude scale.

## 5 Retrieval-based Ensemble

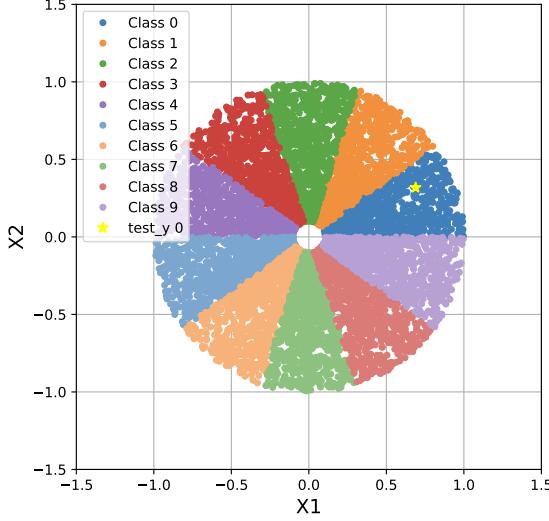
We adopt an inference-time and retrieval-based ensemble strategy that leverages LimiX’s learned attention scores to upweight and select representative in-context samples and features without any additional training, so that we can further improve the performance of LimiX.

In terms of ensemble, we run multiple inference pipelines per dataset and aggregate the results. In each pipeline, we (i) randomly permute the feature columns and reorder the labels for categorical features or outcomes, and (ii) augment a subset of features with simple, schema-preserving transformations like quantile normalization, log-normal transformation, and high-energy SVD components. For classification, the number of inference pipelines is set to 4. For regression, it is set to 8.

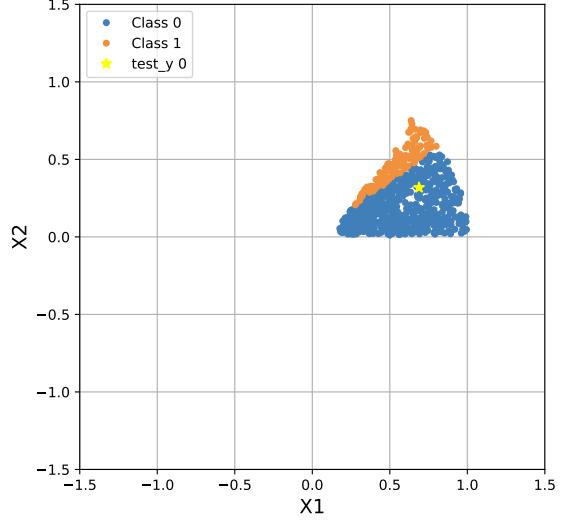
In terms of retrieval, we perform two forward passes of LimiX for each pipeline. The first pass is performed based on all in-context samples and is employed for retrieval. The second pass is performed based on the customized in-context samples retrieved in the first pass.

The procedure of the first pass is as follows. First, last-layer feature-level attention provides importance scores over features, which can be used as feature weights for subsequent sample selection. Concretely, for each test sample, we calculate  $\mathbf{a}_f \in \mathbb{R}^{d+1}$ , which is the feature-level attention score between the outcome  $y_{te}$  and the concatenation of  $F$  features and outcome  $(\mathbf{x}_{te}, y_{te})$ . Then, the module of last-layer sample-level cross-attention between test samples and in-context samples induces importance scores over in-context samples. Concretely, we calculate  $\mathbf{a}_s \in \mathbb{R}^{m_{ct} \times (d+1)}$ , which is the sample-level cross-attention scores between the test sample  $(\mathbf{x}_{te}, y_{te})$  and in-context samples  $(\mathbf{x}_{ct}, \mathbf{y}_{ct})$ . Finally, for a given test sample, we retrieve in-context samples with highest reweighted attention scores as customized in-context samples for this test sample, and perform a forward propagation again for prediction. Concretely, we calculate  $\mathbf{a}_{sf} = \mathbf{a}_s \mathbf{a}_f \in \mathbb{R}^{m_{ct}}$ , a weighted average of  $\mathbf{a}_s$  along the feature dimension with  $\mathbf{a}_f$  as the feature weights.

**Toy examples.** We present the effect of bi-level attention-based retrieval via toy examples.

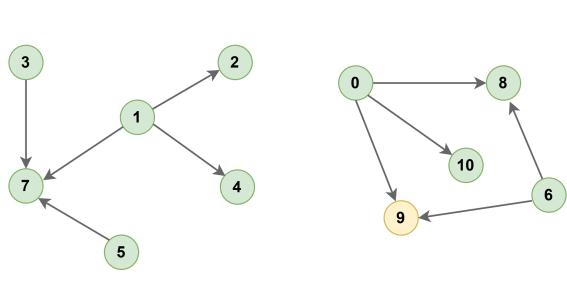


(a) Visualization of synthetic samples. Each sector represents a category of in-context samples. The yellow pentagram represents the test sample.

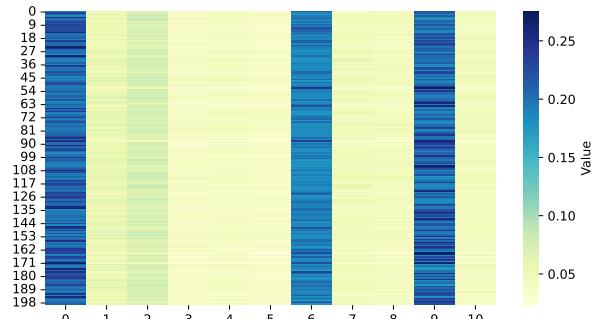


(b) Top 10% of in-context samples sorted by sample-level attention scores.

Figure 5: Toy example of sample-level attention. In-context samples that share the same category as the query sample are assigned higher scores through the attention module of LimiX.



(a) Causal DAG of synthetic samples.



(b) Heatmap of feature-level attention scores of  $Y$ .

Figure 6: Toy example of feature-level attention for the outcome variable. Direct causes of the outcome and the outcome itself are assigned almost all the attention weights in LimiX.

- **Sample-level retrieval.** We generate 2D synthetic data of 10 classes, where each class of data points occupies a sector of a circle in Figure 5a. Since attention depicts sample similarity in the latent embedding space and it also takes the dependency between input features and the category label into account, it is capable of capturing more complex dependencies than naively using Euclidean distance in the original 2D space of features. From Figure 5b, we can see that the attention module predominantly assigns large weights to in-context instances sharing the same category label as the query instance. This indicates that the model leverages class-consistent contextual information from in-context samples to assist its prediction.
- **Feature-level retrieval.** We generate synthetic data via the SCM in Figure 6a, where each green node denotes an observed feature and the yellow node denotes the outcome. Each edge is a two-layer MLP using ReLU as the activation function added with Gaussian noise. From Figure 6b, we can see that the attention module assigns most weights to a subset of features, which is exactly the set of direct causes of the outcome in the SCM. This suggests that feature-level attention could help to focus on causal features and reduce reliance on spurious correlations.

## 6 Theoretical Analysis on Context-Conditional Masked Modeling

We begin by introducing the mathematical formulation underlying our theoretical analysis.

**Notations.** Let  $\Omega$  denote the space for each feature in the table. We represent the  $m$  in-context samples, each with  $d$  features, by the random matrix  $\mathbf{X}^{\text{ct}} \in \Omega^{m \times d}$ . We use the random vector  $\mathbf{X}^{\text{te}} = (X_1^{\text{te}}, \dots, X_d^{\text{te}}) \in \Omega^d$  to denote the test sample<sup>1</sup>. Meanwhile, we do not explicitly introduce a target label  $y$  in this section for simplicity; instead, we treat it as a particular dimension of both the in-context and test samples. Consequently, one dimension of  $\mathbf{X}^{\text{te}}$  is unknown to the model during the test phase.

For any  $\pi \subseteq [d]$ , let  $\mathbf{X}_{\pi}^{\text{te}} = (X_j^{\text{te}})_{j \in \pi}$  and  $\mathbf{X}_{-\pi}^{\text{te}} = (X_j^{\text{te}})_{j \notin \pi}$  denote the subvectors on  $\pi$  and its complement, respectively. In particular, for  $\pi = \{j\}$  we use  $\mathbf{X}_{-j}^{\text{te}}$  as shorthand for  $\mathbf{X}_{-\{j\}}^{\text{te}}$ .

**Training and test procedures.** We are given  $n$  i.i.d. samples  $\{(\mathbf{x}^{\text{ct},(i)}, \mathbf{x}^{\text{te},(i)})\}_{i=1}^n$  drawn from  $p(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}})$ , given in [Section 4](#). The model is trained using CCMM, a masked pre-training method ([Section 3](#)). Let  $\Pi \subseteq 2^{[d]}$  be a set of masks and  $\text{Unif}(\Pi)$  the uniform distribution over  $\Pi$ . In practice, we take  $\Pi = \{\pi \subseteq [d] : |\pi| \in [0.1d, 0.4d]\}$ , while for theory we simplify to masks of fixed size  $k$ :

$$\Pi_k := \{\pi \subseteq [d] : |\pi| = k\}. \quad (1)$$

It is easy to verify that our theoretical insights extend naturally to settings with variable mask sizes.

For each sample  $(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}})$ , we draw  $\pi \sim \text{Unif}(\Pi_k)$  and train  $q_{\theta}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$ , where  $\theta \in \Theta$  and  $\Theta$  denotes the parameter space of the model, to reconstruct the masked features. Let  $\pi_i$  be the mask for  $(\mathbf{x}^{\text{ct},(i)}, \mathbf{x}^{\text{te},(i)})$ . The empirical loss  $\hat{L}_k(\theta)$  and estimator  $\hat{\theta}_{k,n}$  are

$$\hat{L}_k(\theta) = \frac{1}{n} \sum_{i=1}^n -\log q_{\theta}\left(\mathbf{x}_{\pi_i}^{\text{te},(i)} | \mathbf{x}_{-\pi_i}^{\text{te},(i)}, \mathbf{x}^{\text{ct},(i)}\right), \quad \hat{\theta}_{k,n} = \arg \min_{\theta \in \Theta} \hat{L}_k(\theta). \quad (2)$$

As  $n \rightarrow \infty$ , this converges to the population-level loss  $L_k(\theta)$  and the corresponding solution  $\theta_k^*$ :

$$L_k(\theta) = \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k)} [-\log q_{\theta}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})], \quad \theta_k^* = \arg \min_{\theta \in \Theta} L_k(\theta). \quad (3)$$

At test time, all features in  $\mathbf{X}^{\text{ct}}$  are observed, while one feature  $X_j^{\text{te}}$  of  $\mathbf{X}^{\text{te}}$  is missing and must be inferred. The goal is to output  $p(X_j^{\text{te}} | \mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}})$ .

## 6.1 Modeling the Joint Conditional with Random Masks

We now explain the necessity of randomly sampling masks from  $\text{Unif}(\Pi_k)$ . Our result is based on the following proposition.

**Proposition 6.1** (Informal; See [Proposition B.1](#)). *Under mild assumptions, for any  $k \in [d]$ , there is a one-to-one correspondence between the distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  and the family of conditionals  $\{p(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \forall \pi \in \Pi_k\}$ .*

At test time, the target variable  $y$  may correspond to any feature  $X_j^{\text{te}}$  of the test sample  $\mathbf{X}^{\text{te}}$ , so the model must be able to estimate  $p(X_j^{\text{te}} | \mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for every  $j \in [d]$ . As shown in [Proposition 6.1](#), this requirement is equivalent to learning the joint conditional distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$ . Hence, a model can achieve strong predictive performance at test time if and only if it learns  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$ . Moreover, if each  $p(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  can be well approximated by  $q_{\theta}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for all  $\pi \in \Pi_k$ , then the model can recover the joint conditional distribution and thereby generalize effectively. By contrast, [Example B.1](#) shows that knowledge of  $p(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for only a subset of  $\pi \in \Pi_k$  may be insufficient to recover  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$ , preventing the model from generalizing effectively at test time.

Moreover, the target distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  is closely connected to the underlying structural causal models (SCMs). Let  $S$  denote a random variable corresponding to a structural causal model (SCM). By the data-generating process, we have  $\mathbf{X}^{\text{ct}} \perp \mathbf{X}^{\text{te}} | S$ , which yields

$$p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}}) = \int_S p(\mathbf{X}^{\text{te}}, S | \mathbf{X}^{\text{ct}}) dS = \int_S p(S | \mathbf{X}^{\text{ct}}) p(\mathbf{X}^{\text{te}} | S, \mathbf{X}^{\text{ct}}) dS = \int_S p(S | \mathbf{X}^{\text{ct}}) p(\mathbf{X}^{\text{te}} | S) dS. \quad (4)$$

Intuitively, [Equation \(4\)](#) suggests that one efficient way to learn  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  is through two components: (i)  $p(S | \mathbf{X}^{\text{ct}})$ , the posterior distribution over SCMs given the context, and (ii)  $p(\mathbf{X}^{\text{te}} | S)$ , the likelihood of the test sample under a given SCM.

<sup>1</sup>For simplicity, we assume (i) all features share the same space  $\Omega$ , though the results generalize to distinct spaces; (ii) we use  $m$  instead of  $m_{ct}$  in [Section 2](#); and (iii) there is only one test sample, i.e.,  $m_{te} = 1$  in [Section 2](#).

## 6.2 The Choice of Mask Number

In this subsection, we demonstrate that the choice  $k$  in [Equation \(1\)](#) has great impact on models' performances and we choose  $k > 1$  due to both sample efficiency and generalization considerations. Our result is an extension from the analysis by [Li et al. \(2024c\)](#) on the properties of masked sequence prediction.

Based on [Proposition 6.1](#), we will henceforth use  $q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  to denote the distributions induced by the learned family of conditional probabilities  $\{q_\theta(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \pi \in \Pi_k\}$ .

**Sample efficiency.** Theorem below shows that larger  $k$  yields lower estimation uncertainty.

**Theorem 6.2** (Informal; see [Theorem B.2](#)). *Suppose there exists  $\theta^* \in \Theta$  such that  $q_{\theta^*}(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) = p(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for all  $\mathbf{X}^{\text{ct}} \in \Omega^{m \times d}$ ,  $\mathbf{X}^{\text{te}} \in \Omega^d$ , and  $\pi \subseteq [d]$ , and that the minimizer of  $L_k(\theta)$  is unique for every  $k$ . Then, under mild regularity conditions, as  $n \rightarrow \infty$ ,*

$$\sqrt{n} (\hat{\theta}_{k,n} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \Gamma_k),$$

where  $\Gamma_k$  does not depend on  $n$  and satisfies  $\Gamma_{k+1} \preceq \Gamma_k$ .

The assumption on  $\theta^*$  states that the optimal solution of the model can approximate any conditional distribution, which is reasonable given the expressive power of transformer-based architectures. This theorem further shows that for sufficiently large and fixed  $n$ , if  $k_1 > k_2$  then  $\Gamma_{k_1} \preceq \Gamma_{k_2}$ , implying that  $\hat{\theta}_{k_1,n}$  has lower estimation uncertainty than  $\hat{\theta}_{k_2,n}$ . Equivalently, when  $k$  is larger, fewer samples are required to achieve the same uncertainty, leading to greater sample efficiency.

**Generalization for joint distribution learning.** Masked pretraining empowers the model to infer the joint distribution of data from the context of training samples. We further show that an increase in the density of random masks enables a more accurate reconstruction of the joint distribution.

**Theorem 6.3** (Informal; See [Theorem B.5](#)). *Under regularity conditions on  $q_\theta$ , with high probability, for any  $\theta \in \Theta$ , the expected total variation between  $q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  and the true conditional distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  is at most:*

$$\sqrt{\frac{1}{2} C_k(q_\theta)(\hat{L}_k(\theta) + \text{complexity terms}) + \mathcal{O}(n^{-1/2})},$$

where constants  $C_k(q_\theta)$  depend only on  $q_\theta$  and  $k$ . Furthermore,  $C_{k+1}(q_\theta) \leq C_k(q_\theta)$  for any  $\theta \in \Theta$ .

The theorem establishes an upper bound for the generalization error of the estimated conditional joint distribution  $q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  compared to the true joint distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$ , and shows that the upper bound decreases monotonically with respect to the number of masked cells.

## 7 Evaluation

### 7.1 Classification

**Benchmarks.** For the quantitative evaluation of classification performance, we utilize multiple benchmarks, including TALENT-CLS ([Liu et al., 2024](#)), OpenML-CC18 ([Bischl et al., 2017](#)), PFN-CLS ([Hollmann et al., 2025](#)), TabZilla ([McElfresh et al., 2023](#)), and TabArena ([Erickson et al., 2025](#)). Among these benchmarks, the datasets containing more than 50,000 training samples (The number of testing samples is not constrained), 10,000 features, or 10 target categories were excluded. This selection process resulted in a final collection of 179 datasets from TALENT-CLS, 62 from OpenML-CC18, 29 from PFN-CLS, 27 from TabZilla, and 33 from TabArena.

Furthermore, we introduce **Balanced Comprehensive Challenging Omni-domain (BCCO) Benchmark**, comprising BCCO-CLS and BCCO-REG, for the evaluation of LimiX and baseline models. The BCCO benchmark is constructed from extensive open-source structured-data corpora, meticulously deduplicated and cleaned. It presents a significant challenge due to several intrinsic characteristics: the distribution of dataset attributes (such as the ratio of categorical features), and the diversity of real-world prediction targets. Unlike previous benchmarks, nearly one-third of the datasets in our BCCO benchmark contain missing values.

The BCCO-CLS benchmark comprises 106 datasets. The collection spans diverse sources, domains, and scales, and is designed to cover a wide range of problem characteristics, including the number of samples, number of features, number of classes, categorical-to-numerical feature ratio, sample-to-feature ratio, and proportion of missing values. This diversity allows for uniform binning along these dimensions,

---

enabling results to be reported within bins and macro-averaged across bins, thereby providing a nearly unbiased assessment of model performance across heterogeneous task regimes. Additionally, datasets containing more than 50,000 training samples (The number of testing samples is not constrained), 10,000 features, or 10 target categories were excluded.

For each dataset in these benchmarks, we use the provided train-test split when available. If no predefined test set exists, the data are partitioned into a 70% training set and a 30% test set using stratified sampling to preserve the label distribution.

**Baselines.** We compare LimiX with a range of state-of-the-art baseline models, categorized into tree-based models, neural networks (NN), and recent ICL-based approaches.

- **Tree-based approaches.** We include XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), CatBoost (Dorogush et al., 2018), Random Forest (RF) (Breiman, 2001), and Extra Trees (ET) (Geurts et al., 2006). All models are optimized using the Optuna (Akiba et al., 2019) framework via 5-fold stratified cross-validation, with hyperparameters sampled from the ranges specified in Appendix A.2. Additionally, for AutoGluon-Tabular (Erickson et al., 2020), which automates workflows of model searching and ensemble, we use the default search space and set a default 600s time constraint for hyperparameter searching for each dataset.
- **NN-based approaches.** We evaluate against SNN (Klambauer et al., 2017), AutoInt (Song et al., 2019), NODE (Popov et al., 2020), TabTransformer (Huang et al., 2020), GrowNet (Badirli et al., 2020), TabNet (Arik & Pfister, 2021), DCN-v2 (Wang et al., 2021), FT-Transformer (Gorishniy et al., 2021), MLP (Goodfellow et al., 2016; Gorishniy et al., 2021), ResNet (He et al., 2016; Gorishniy et al., 2021), SAINT (Somepalli et al., 2022), MLP-PLR (Gorishniy et al., 2022), DANets (Chen et al., 2022), TANGOS (Jeffares et al., 2023), T2G-Former (Yan et al., 2023), ExcelFormer (Chen et al., 2023b), Trompt (Chen et al., 2023c), TabR (Gorishniy et al., 2024), TabCaps (Chen et al., 2023a), RealMLP (Holzmüller et al., 2024), SwitchTab (Wu et al., 2024), and ModernNCA (Ye et al., 2025). These NN-based models are trained using the TALENT (Liu et al., 2024) Toolbox.
- **ICL-based models.** Recent baselines includes TabPFN-v2 (Hollmann et al., 2025), TabICL (Qu et al., 2025), and Mitra (Zhang & Danielle, 2025).

Since TabDPT (Ma et al., 2024), another recent ICL-based model, is pretrained on real data that has a large overlap with datasets in the benchmarks, we do not adopt it as a baseline in the evaluation of standard classification and regression for a fair comparison. However, we include it in our experiments of fine-tuning in Section 7.6 to show the superiority of LimiX.

**Metrics.** To evaluate model performance, we employ ROC AUC (area under the receiver operating characteristic curve), accuracy, and F1 score for classification tasks. For multi-class classification, the One-vs-One strategy is applied to both ROC AUC and F1 score calculations.

The critical difference diagram is also employed to compare the performance differences between LimiX and the baseline models. We conducted a Friedman test followed by a post-hoc Wilcoxon-Holm test, using a significance level of 0.05. In the diagram, the horizontal line indicates the range of ranks among which differences are not statistically significant.

**Results.** Figure 7 shows that LimiX achieves the best performance among all methods on most datasets in BCCO-CLS. The most competitive baselines include other ICL-based models TabICL and TabPFN-v2, and the AutoML ensemble framework AutoGluon. In all critical difference diagrams, i.e. Figures 8 to 13, we can see that LimiX achieves the highest ranking in terms of all three evaluation metrics on BCCO-CLS, OpenML-CC18, and TALENT-CLS, where critical differences can be observed in most cases, indicating a significant margin.

For detailed quantitative results listed in Tables 1, 3, 5, 7, 9 and 11, LimiX outperforms all baselines on every benchmark in terms of both mean and rank of all three metrics. All these results clearly demonstrate that LimiX achieves state-of-the-art performance in terms of tabular classification tasks, surpassing not only traditional ensemble methods but also advanced ICL-based models.

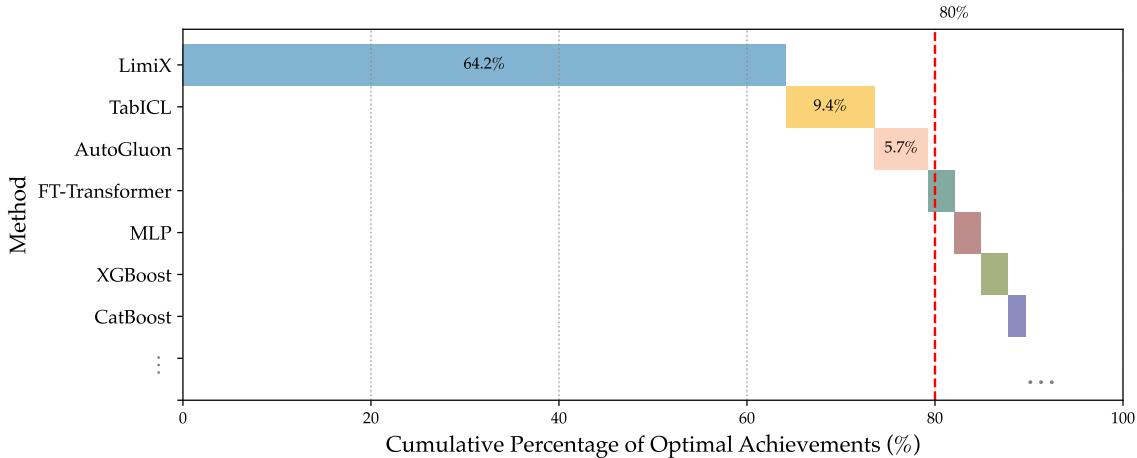


Figure 7: The proportion of models achieving the best AUC. The length of each bar represents the proportion of the 106 datasets in BCCO-CLS where a given method achieves the highest AUC.

**Subgroup analysis.** We use the sample subgroups when building BCCO-CLS to perform stratified analyses. Subgroups are defined based on the following criteria: the type of classification (binary or multi-class), the number of training samples, the ratio between the number of samples and features (length-to-width ratio), the proportion of categorical features, and the presence or absence of missing values. The number of training samples, categorical feature ratio, and length-to-width ratio are discretized into terciles (equal-frequency bins), ensuring that approximately the same number of datasets is allocated to each stratum and that the distribution of the dataset attribute which is not considered during analysis remains nearly uniform across strata.

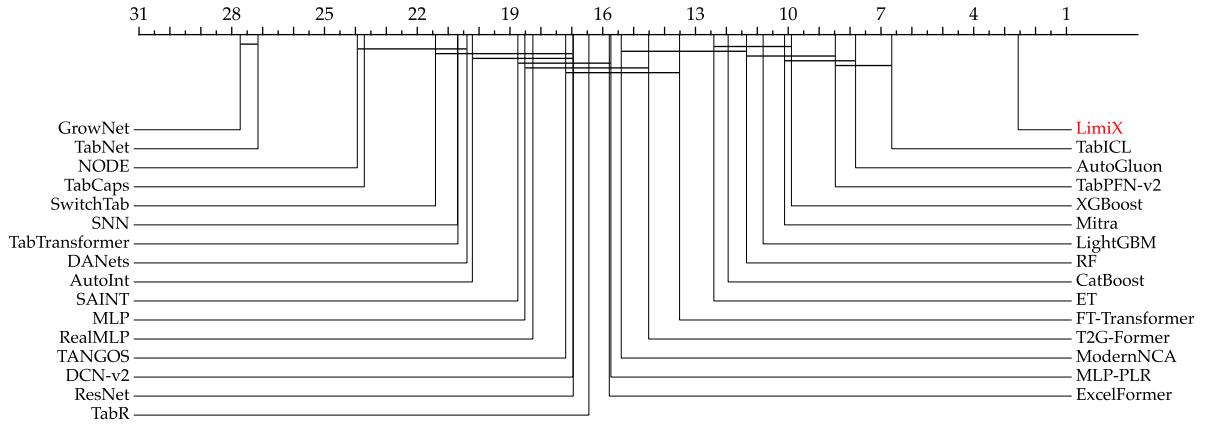
From [Figure 19](#), we observe that LimiX exhibits leading performance across all subgroups compared with other methods. Notably, after stratification, under some subgroups like the third subgroup in [Figure 19c](#) that indicates a larger training sample size, AutoGluon proves to be a strong competitor to ICL-based models. In these cases, LimiX is the only ICL-based model that outperforms AutoGluon. Notably, [Figure 19e](#) shows that, as the proportion of categorical features increases, performance for most baselines drops rapidly, reflecting the sparsity and high-cardinality challenges. In contrast, LimiX exhibits only modest degradation and remains comparatively stable across these regimes.

Table 1: Classification results on the BCCO-CLS benchmark. The best scores are shown in bold.

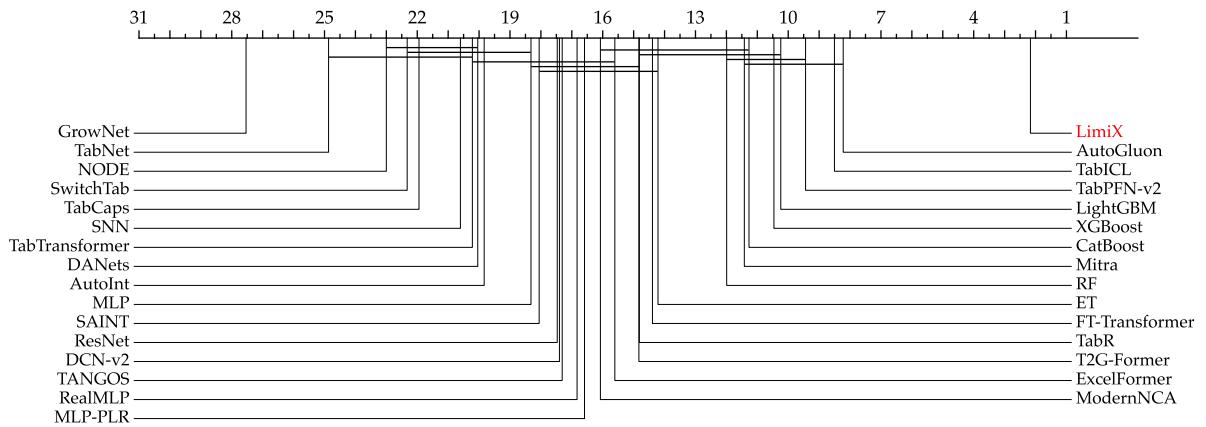
Model	BCCO-CLS					
	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
Limix	<b>0.871</b> $\pm$ 0.008	<b>0.804</b> $\pm$ 0.010	<b>0.731</b> $\pm$ 0.020	<b>2.547</b>	<b>2.151</b>	<b>2.764</b>
TabICL	0.847 $\pm$ 0.008	0.768 $\pm$ 0.010	0.672 $\pm$ 0.016	6.642	8.142	10.132
AutoGluon	0.846 $\pm$ 0.000	0.771 $\pm$ 0.001	0.677 $\pm$ 0.001	7.811	7.934	9.302
TabPFN-v2	0.843 $\pm$ 0.005	0.772 $\pm$ 0.007	0.679 $\pm$ 0.009	8.462	9.094	10.698
Mitra	0.836 $\pm$ 0.017	0.764 $\pm$ 0.018	0.664 $\pm$ 0.035	10.104	11.038	12.849
XGBoost	0.834 $\pm$ 0.005	0.762 $\pm$ 0.007	0.674 $\pm$ 0.012	9.887	10.208	10.887
LightGBM	0.832 $\pm$ 0.005	0.763 $\pm$ 0.006	0.678 $\pm$ 0.011	10.802	10.075	10.698
RF	0.829 $\pm$ 0.007	0.756 $\pm$ 0.008	0.652 $\pm$ 0.014	11.349	11.717	13.047
CatBoost	0.829 $\pm$ 0.008	0.757 $\pm$ 0.010	0.664 $\pm$ 0.014	11.943	11.000	11.792
ET	0.825 $\pm$ 0.005	0.745 $\pm$ 0.006	0.618 $\pm$ 0.014	12.406	13.925	16.783
ModernNCA	0.815 $\pm$ 0.017	0.752 $\pm$ 0.016	0.658 $\pm$ 0.027	15.377	15.689	15.321
FT-Transformer	0.813 $\pm$ 0.014	<b>0.744</b> $\pm$ 0.015	0.642 $\pm$ 0.030	13.509	13.953	14.377
ExcelFormer	0.810 $\pm$ 0.018	<b>0.742</b> $\pm$ 0.015	0.655 $\pm$ 0.023	15.783	15.217	13.840
TabR	0.809 $\pm$ 0.019	0.750 $\pm$ 0.019	0.657 $\pm$ 0.035	16.443	14.415	13.443
T2G-Former	0.808 $\pm$ 0.021	0.742 $\pm$ 0.020	0.646 $\pm$ 0.034	14.500	14.491	13.670
MLP-PLR	0.804 $\pm$ 0.007	0.733 $\pm$ 0.011	0.635 $\pm$ 0.018	15.717	16.179	15.377
ResNet	0.800 $\pm$ 0.017	0.728 $\pm$ 0.016	0.641 $\pm$ 0.022	16.943	17.160	15.896
TANGOS	0.799 $\pm$ 0.015	0.731 $\pm$ 0.016	0.641 $\pm$ 0.022	17.179	16.943	15.406
DCN-v2	0.794 $\pm$ 0.020	0.725 $\pm$ 0.018	0.618 $\pm$ 0.035	16.953	16.925	16.698
RealMLP	0.794 $\pm$ 0.020	0.738 $\pm$ 0.014	0.644 $\pm$ 0.021	18.255	16.462	15.208
SAINT	0.791 $\pm$ 0.017	0.726 $\pm$ 0.013	0.623 $\pm$ 0.021	18.736	17.642	16.311
MLP	0.787 $\pm$ 0.011	0.720 $\pm$ 0.013	0.614 $\pm$ 0.018	18.509	17.925	18.783
AutoInt	0.779 $\pm$ 0.028	0.718 $\pm$ 0.017	0.601 $\pm$ 0.037	20.208	19.453	19.849
SNN	0.773 $\pm$ 0.024	0.708 $\pm$ 0.019	0.584 $\pm$ 0.049	20.698	20.321	20.594
DANets	0.771 $\pm$ 0.029	0.705 $\pm$ 0.027	0.601 $\pm$ 0.043	20.377	19.575	19.868
SwitchTab	0.766 $\pm$ 0.021	0.700 $\pm$ 0.018	0.590 $\pm$ 0.036	21.396	21.991	20.868
TabTransformer	0.762 $\pm$ 0.018	0.699 $\pm$ 0.023	0.566 $\pm$ 0.047	20.670	19.849	20.321
NODE	0.754 $\pm$ 0.046	0.695 $\pm$ 0.031	0.531 $\pm$ 0.078	23.925	22.594	25.255
TabCaps	0.744 $\pm$ 0.027	0.701 $\pm$ 0.019	0.580 $\pm$ 0.026	23.708	21.670	22.113
TabNet	0.712 $\pm$ 0.047	0.685 $\pm$ 0.028	0.561 $\pm$ 0.041	27.151	24.660	24.774
GrowNet	0.682 $\pm$ 0.033	0.641 $\pm$ 0.028	0.522 $\pm$ 0.040	27.726	27.368	26.387

Table 2: Statistical profile of the benchmark BCCO-CLS, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

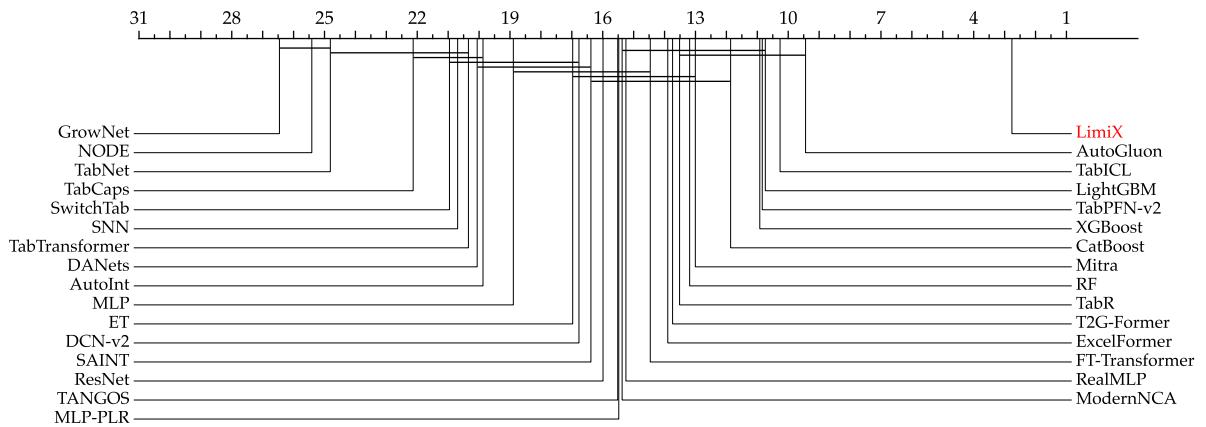
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	4	12	48	23	33	1	259
# Classes	2	2	5	3	2	2	10
Missing Values (Ratio)	0	0	0.069	0.024	0.068	0	0.403
Categorical Features (Ratio)	0	0.384	0.929	0.399	0.337	0	1
Features w/ Missing Values (Ratio)	0	0	0.523	0.126	0.241	0	0.909



(a) AUC on the BCCO-CLS benchmark.



(b) Accuracy on the BCCO-CLS benchmark.



(c) F1-score on the BCCO-CLS benchmark.

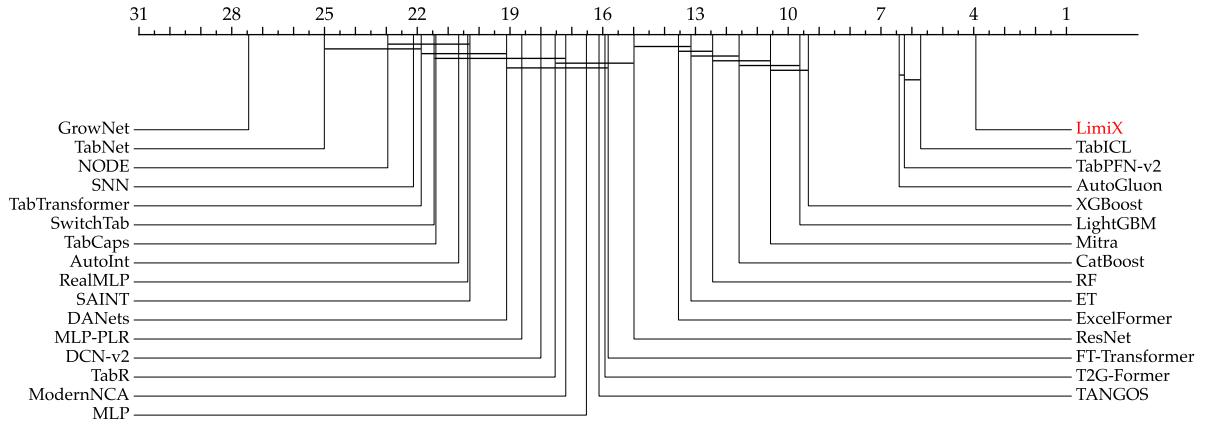
Figure 8: Critical difference diagrams on BCCO-CLS benchmark.

Table 3: Classification results on the TALENT-CLS benchmark. The best scores are shown in bold.

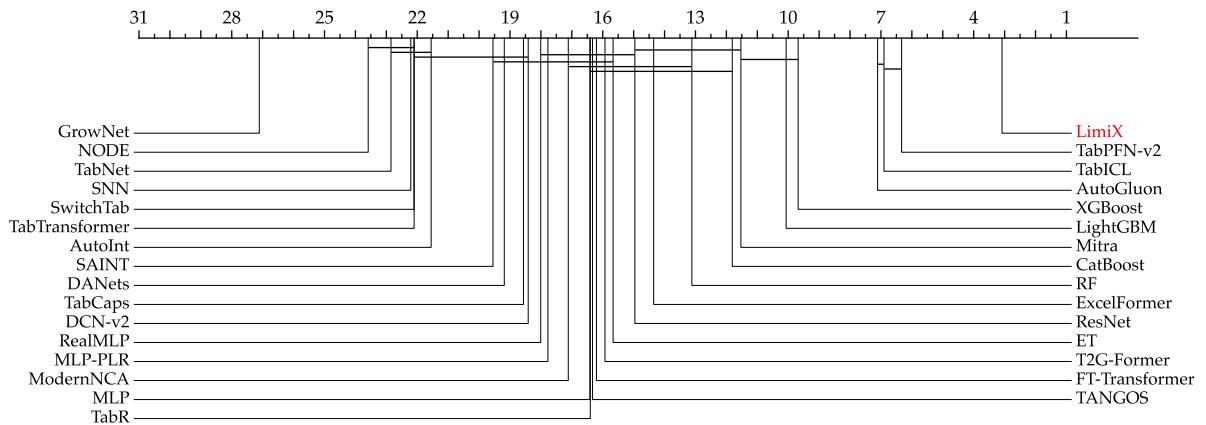
	TALENT-CLS					
Model	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
LimiX	<b>0.903<math>\pm</math>0.004</b>	<b>0.861<math>\pm</math>0.005</b>	<b>0.752<math>\pm</math>0.016</b>	<b>3.894</b>	<b>3.000</b>	<b>4.028</b>
TabPFN-v2	0.895 $\pm$ 0.002	0.850 $\pm$ 0.003	0.727 $\pm$ 0.008	6.056	5.860	7.670
TabICL	0.894 $\pm$ 0.003	0.845 $\pm$ 0.005	0.715 $\pm$ 0.008	5.536	6.480	7.955
AutoGluon	0.891 $\pm$ 0.000	0.845 $\pm$ 0.000	0.719 $\pm$ 0.001	6.229	6.642	7.425
Mitra	0.882 $\pm$ 0.000	0.834 $\pm$ 0.001	0.689 $\pm$ 0.001	10.391	11.095	13.168
XGBoost	0.881 $\pm$ 0.003	0.837 $\pm$ 0.004	0.713 $\pm$ 0.009	9.235	9.380	9.715
LightGBM	0.880 $\pm$ 0.003	0.836 $\pm$ 0.004	0.713 $\pm$ 0.009	9.486	9.721	10.067
RF	0.877 $\pm$ 0.003	0.828 $\pm$ 0.004	0.691 $\pm$ 0.010	12.307	12.793	13.927
CatBoost	0.876 $\pm$ 0.005	0.828 $\pm$ 0.006	0.704 $\pm$ 0.014	11.453	11.497	11.743
ET	0.875 $\pm$ 0.003	0.821 $\pm$ 0.006	0.662 $\pm$ 0.017	12.994	15.380	16.933
ExcelFormer	0.870 $\pm$ 0.012	0.826 $\pm$ 0.009	0.699 $\pm$ 0.019	13.358	13.989	12.961
ResNet	0.866 $\pm$ 0.009	0.825 $\pm$ 0.006	0.695 $\pm$ 0.014	14.933	14.721	13.207
MLP	0.862 $\pm$ 0.008	0.817 $\pm$ 0.008	0.675 $\pm$ 0.017	16.419	16.140	16.346
TANGOS	0.861 $\pm$ 0.009	0.818 $\pm$ 0.007	0.684 $\pm$ 0.020	16.011	15.978	14.754
ModernNCA	0.861 $\pm$ 0.013	0.825 $\pm$ 0.010	0.683 $\pm$ 0.022	17.101	16.877	16.151
FT-Transformer	0.859 $\pm$ 0.010	0.822 $\pm$ 0.007	0.678 $\pm$ 0.017	15.693	15.905	15.609
TabR	0.858 $\pm$ 0.015	0.824 $\pm$ 0.011	0.680 $\pm$ 0.026	17.413	16.034	15.179
T2G-Former	0.858 $\pm$ 0.016	0.823 $\pm$ 0.010	0.683 $\pm$ 0.019	15.777	15.536	15.134
DCN-v2	0.854 $\pm$ 0.016	0.815 $\pm$ 0.013	0.662 $\pm$ 0.027	17.849	18.039	17.899
MLP-PLR	0.849 $\pm$ 0.007	0.816 $\pm$ 0.009	0.663 $\pm$ 0.019	18.492	17.346	17.391
DANets	0.848 $\pm$ 0.018	0.805 $\pm$ 0.018	0.654 $\pm$ 0.038	19.039	18.838	18.497
AutoInt	0.842 $\pm$ 0.011	0.803 $\pm$ 0.008	0.646 $\pm$ 0.022	20.587	21.246	20.721
SwitchTab	0.842 $\pm$ 0.013	0.795 $\pm$ 0.011	0.637 $\pm$ 0.029	21.346	21.894	21.089
RealMLP	0.839 $\pm$ 0.017	0.820 $\pm$ 0.007	0.678 $\pm$ 0.017	20.212	17.687	15.994
SNN	0.836 $\pm$ 0.020	0.796 $\pm$ 0.016	0.625 $\pm$ 0.041	22.011	21.955	22.089
TabCaps	0.834 $\pm$ 0.014	0.813 $\pm$ 0.008	0.654 $\pm$ 0.017	21.324	18.223	18.687
TabTransformer	0.832 $\pm$ 0.014	0.790 $\pm$ 0.016	0.627 $\pm$ 0.037	21.743	21.816	20.933
NODE	0.830 $\pm$ 0.043	0.779 $\pm$ 0.040	0.570 $\pm$ 0.104	22.883	23.240	24.816
TabNet	0.818 $\pm$ 0.035	0.794 $\pm$ 0.021	0.630 $\pm$ 0.041	25.006	22.603	23.039
SAINT	0.813 $\pm$ 0.012	0.781 $\pm$ 0.007	0.630 $\pm$ 0.020	20.279	19.402	18.888
GrowNet	0.743 $\pm$ 0.019	0.704 $\pm$ 0.011	0.542 $\pm$ 0.024	27.453	27.000	25.028

Table 4: Statistical profile of the benchmark TALENT-CLS, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

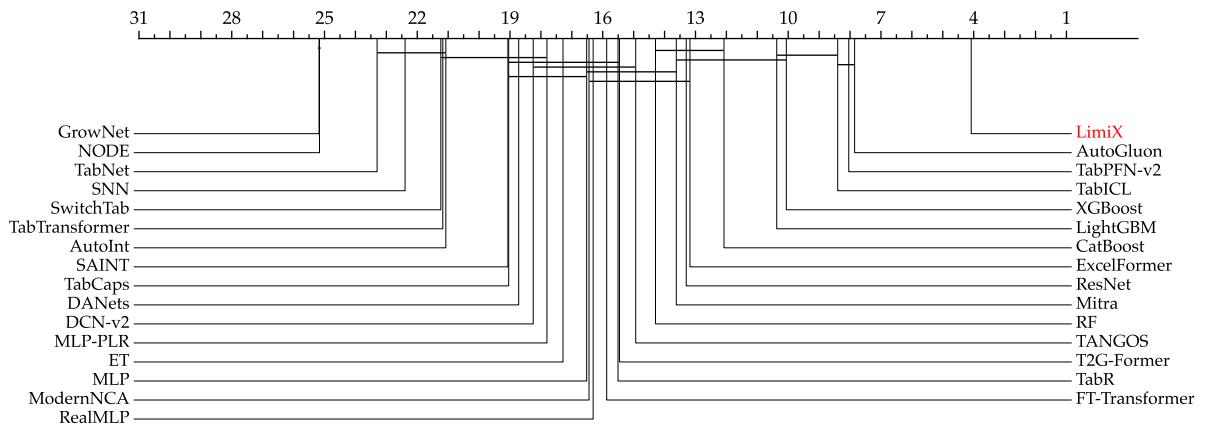
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	7	19	70	33	47	3	308
# Classes	2	2	10	6	14	2	100
Missing Values (Ratio)	0	0	0	0.001	0.008	0	0.1
Categorical Features (Ratio)	0	0.121	0.972	0.306	0.365	0	1
Features w/ Missing Values (Ratio)	0	0	0	0.025	0.121	0	0.979



(a) AUC on the TALENT-CLS benchmark.



(b) Accuracy on the TALENT-CLS benchmark.



(c) F1-score on the TALENT-CLS benchmark.

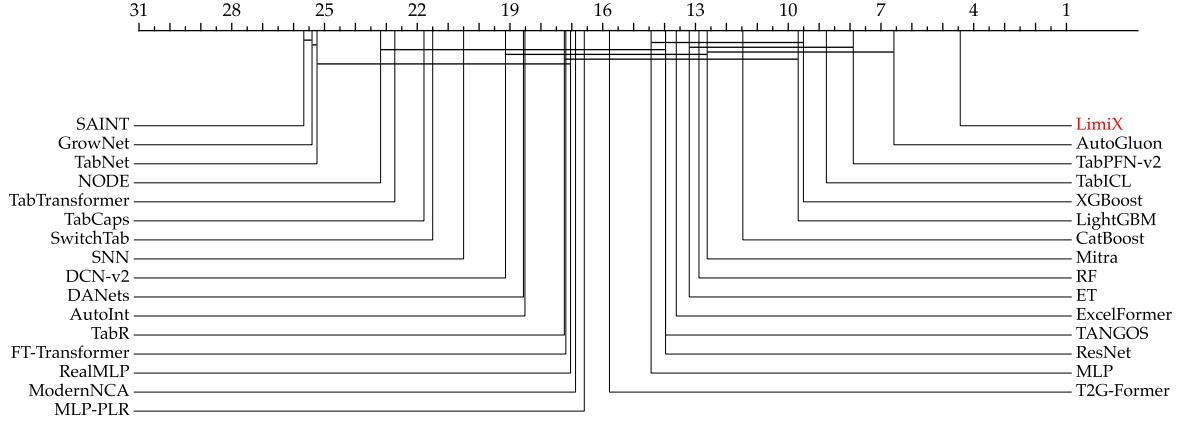
Figure 9: Critical difference diagram on the TALENT-CLS benchmark

Table 5: Classification results on the OpenML-CC18 benchmark. The best scores are shown in bold.

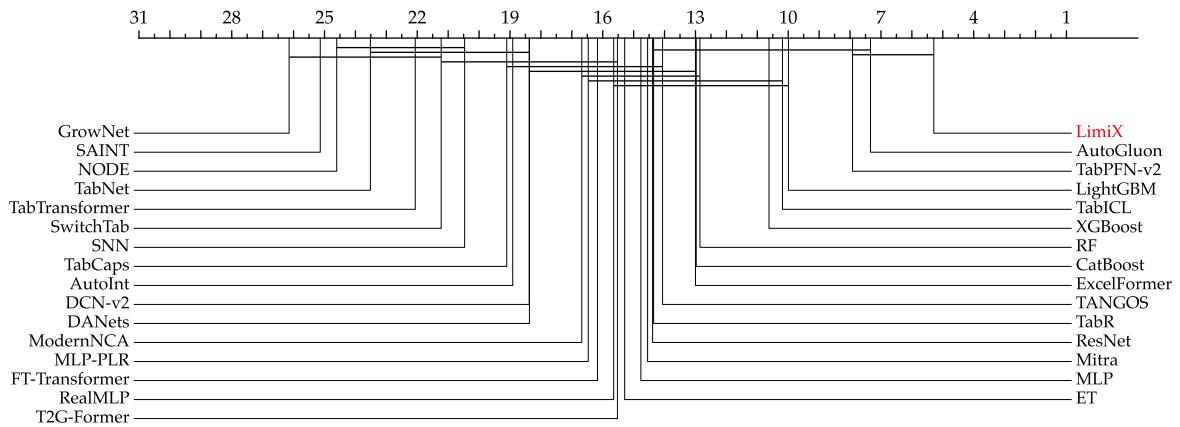
	OpenML-cc18					
Model	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
Limix	<b>0.939<math>\pm</math>0.004</b>	<b>0.893<math>\pm</math>0.008</b>	<b>0.811<math>\pm</math>0.014</b>	<b>4.048</b>	<b>4.952</b>	<b>4.258</b>
AutoGluon	0.932 $\pm$ 0.000	0.885 $\pm$ 0.001	0.790 $\pm$ 0.003	5.823	6.694	7.048
TabPFN-v2	0.929 $\pm$ 0.002	0.886 $\pm$ 0.005	0.790 $\pm$ 0.006	7.177	7.177	7.629
XGBoost	0.929 $\pm$ 0.002	0.879 $\pm$ 0.005	0.775 $\pm$ 0.012	8.952	10.145	10.000
TabICL	0.927 $\pm$ 0.004	0.875 $\pm$ 0.008	0.782 $\pm$ 0.011	8.242	9.677	9.613
LightGBM	0.927 $\pm$ 0.003	0.879 $\pm$ 0.005	0.775 $\pm$ 0.015	9.323	9.532	9.565
CatBoost	0.926 $\pm$ 0.005	0.870 $\pm$ 0.008	0.770 $\pm$ 0.016	11.258	12.726	12.613
RF	0.925 $\pm$ 0.003	0.871 $\pm$ 0.006	0.762 $\pm$ 0.016	12.871	12.452	13.145
ET	0.922 $\pm$ 0.003	0.861 $\pm$ 0.006	0.721 $\pm$ 0.017	12.903	15.016	16.839
Mitra	0.920 $\pm$ 0.001	0.866 $\pm$ 0.004	0.743 $\pm$ 0.004	12.177	13.758	15.226
ExcelFormer	0.918 $\pm$ 0.014	0.870 $\pm$ 0.014	0.773 $\pm$ 0.022	13.065	12.210	12.048
ResNet	0.913 $\pm$ 0.009	0.860 $\pm$ 0.011	0.764 $\pm$ 0.019	13.435	13.613	12.806
TANGOS	0.910 $\pm$ 0.007	0.863 $\pm$ 0.010	0.759 $\pm$ 0.025	13.355	13.339	13.484
MLP	0.908 $\pm$ 0.007	0.857 $\pm$ 0.008	0.743 $\pm$ 0.019	13.887	13.984	15.016
T2G-Former	0.908 $\pm$ 0.012	0.859 $\pm$ 0.011	0.748 $\pm$ 0.020	15.145	14.677	14.710
ModernNCA	0.906 $\pm$ 0.009	0.858 $\pm$ 0.013	0.747 $\pm$ 0.021	16.661	16.129	15.532
FT-Transformer	0.904 $\pm$ 0.015	0.856 $\pm$ 0.012	0.739 $\pm$ 0.019	16.532	15.306	16.194
TabR	0.900 $\pm$ 0.015	0.863 $\pm$ 0.011	0.757 $\pm$ 0.025	16.661	13.484	13.113
DANets	0.900 $\pm$ 0.021	0.840 $\pm$ 0.029	0.715 $\pm$ 0.053	18.145	17.613	18.581
DCN-v2	0.899 $\pm$ 0.017	0.851 $\pm$ 0.015	0.729 $\pm$ 0.046	18.710	17.516	18.129
SNN	0.896 $\pm$ 0.014	0.835 $\pm$ 0.023	0.698 $\pm$ 0.055	20.161	20.000	20.774
MLP-PLR	0.896 $\pm$ 0.010	0.858 $\pm$ 0.008	0.734 $\pm$ 0.020	15.919	15.645	16.000
NODE	0.889 $\pm$ 0.042	0.809 $\pm$ 0.060	0.626 $\pm$ 0.126	23.177	24.242	26.194
RealMLP	0.889 $\pm$ 0.016	0.858 $\pm$ 0.013	0.742 $\pm$ 0.026	16.274	14.839	14.065
AutoInt	0.889 $\pm$ 0.010	0.835 $\pm$ 0.015	0.717 $\pm$ 0.033	18.145	18.161	18.290
SwitchTab	0.881 $\pm$ 0.031	0.819 $\pm$ 0.039	0.671 $\pm$ 0.065	21.403	20.790	21.516
TabCaps	0.875 $\pm$ 0.024	0.848 $\pm$ 0.017	0.692 $\pm$ 0.042	21.597	18.613	20.161
TabNet	0.863 $\pm$ 0.039	0.816 $\pm$ 0.044	0.666 $\pm$ 0.076	25.226	23.177	24.839
TabTransformer	0.829 $\pm$ 0.021	0.776 $\pm$ 0.022	0.616 $\pm$ 0.051	22.565	21.645	22.387
GrowNet	0.816 $\pm$ 0.027	0.748 $\pm$ 0.027	0.581 $\pm$ 0.050	25.403	25.855	24.935
SAINT	0.512 $\pm$ 0.000	0.492 $\pm$ 0.000	0.441 $\pm$ 0.000	25.210	24.597	23.290

Table 6: Statistical profile of the benchmark OpenML-CC18, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

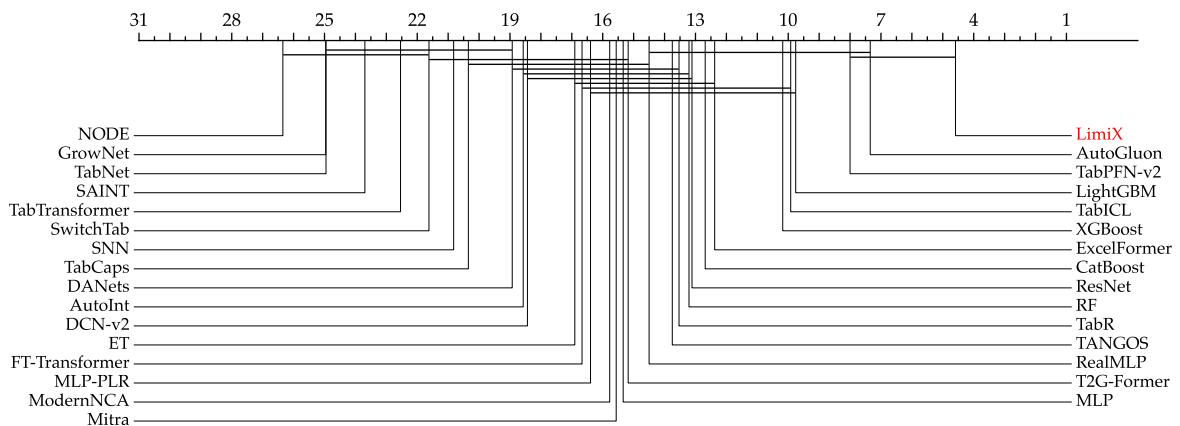
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	30	611	336	1344	4	10935
# Classes	2	3	10	6	7	2	46
Missing Values (Ratio)	0	0	0.002	0.004	0.019	0	0.139
Categorical Features (Ratio)	0	0.091	1	0.327	0.406	0	1
Features w/ Missing Values (Ratio)	0	0	0.204	0.054	0.165	0	0.757



(a) AUC on the OpenML-CC18 benchmark.



(b) Accuracy on the OpenML-CC18 benchmark.



(c) F1-score on the OpenML-CC18 benchmark.

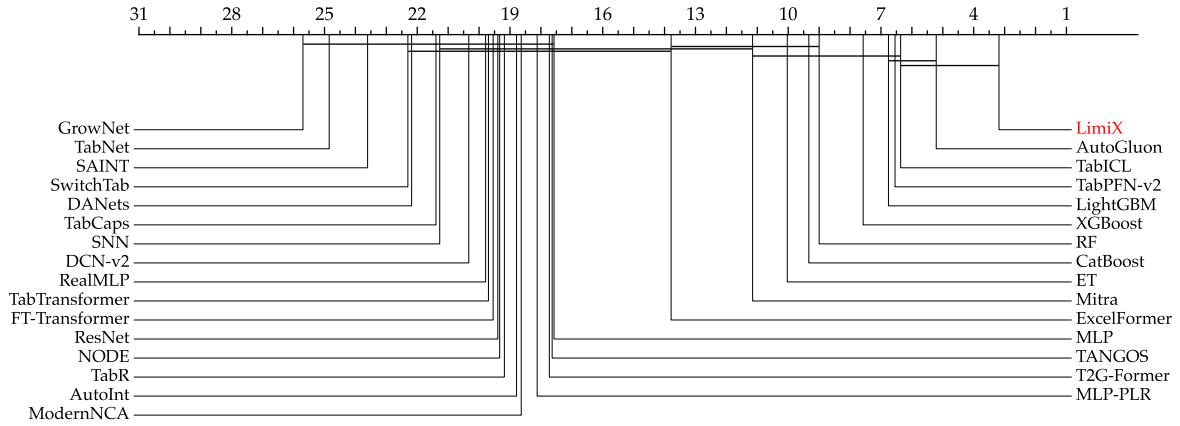
Figure 10: Critical difference diagrams on the OpenML-CC18 benchmark.

Table 7: Classification results on the TabArena benchmark. The best scores are shown in bold.

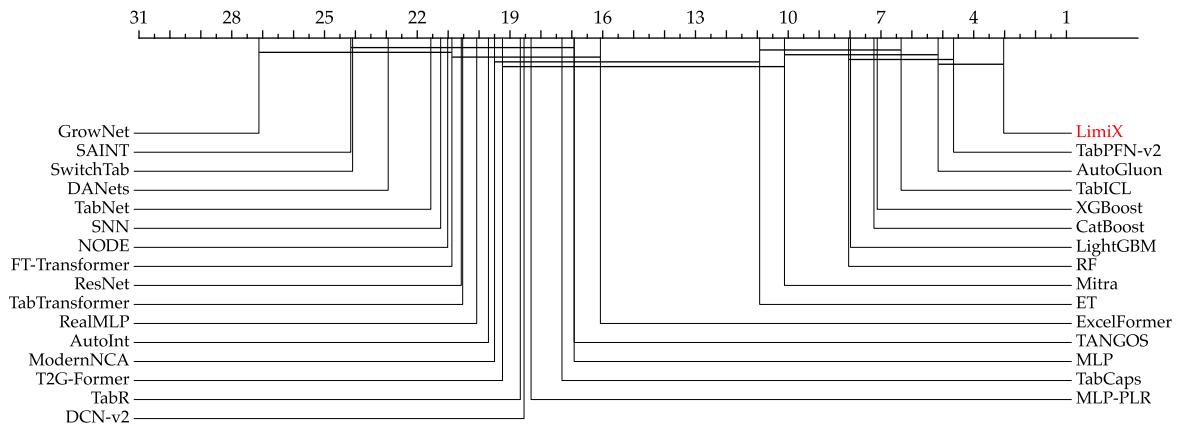
	TabArena					
Model	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
LimiX	<b>0.849<math>\pm</math>0.005</b>	<b>0.877<math>\pm</math>0.004</b>	<b>0.597<math>\pm</math>0.014</b>	<b>3.182</b>	<b>3.030</b>	<b>6.606</b>
AutoGluon	0.844 $\pm$ 0.001	0.870 $\pm$ 0.000	0.574 $\pm$ 0.001	5.212	4.636	8.364
LightGBM	0.841 $\pm$ 0.002	0.868 $\pm$ 0.002	0.574 $\pm$ 0.011	6.758	7.667	9.848
TabICL	0.840 $\pm$ 0.006	0.870 $\pm$ 0.002	0.553 $\pm$ 0.007	6.364	5.727	9.970
XGBoost	0.838 $\pm$ 0.002	0.867 $\pm$ 0.003	0.567 $\pm$ 0.018	7.576	6.939	10.182
TabPFN-v2	0.838 $\pm$ 0.002	0.872 $\pm$ 0.002	0.589 $\pm$ 0.004	6.545	4.000	8.242
RF	0.837 $\pm$ 0.002	0.864 $\pm$ 0.003	0.558 $\pm$ 0.011	9.000	7.455	11.303
CatBoost	0.835 $\pm$ 0.005	0.867 $\pm$ 0.004	0.574 $\pm$ 0.015	9.333	6.758	9.273
ET	0.833 $\pm$ 0.003	0.857 $\pm$ 0.003	0.505 $\pm$ 0.019	10.030	10.333	15.515
Mitra	0.815 $\pm$ 0.025	0.862 $\pm$ 0.008	0.533 $\pm$ 0.049	11.152	9.485	14.152
ExcelFormer	0.810 $\pm$ 0.014	0.849 $\pm$ 0.015	0.555 $\pm$ 0.027	13.788	15.939	14.152
TANGOS	0.791 $\pm$ 0.019	0.844 $\pm$ 0.009	0.522 $\pm$ 0.047	17.636	16.848	15.212
TabR	0.785 $\pm$ 0.012	0.842 $\pm$ 0.013	0.510 $\pm$ 0.049	19.182	18.606	15.303
ModernNCA	0.783 $\pm$ 0.020	0.846 $\pm$ 0.006	0.511 $\pm$ 0.025	18.636	19.424	16.152
ResNet	0.781 $\pm$ 0.029	0.824 $\pm$ 0.034	0.532 $\pm$ 0.045	19.394	20.515	15.515
MLP-PLR	0.781 $\pm$ 0.007	0.836 $\pm$ 0.004	0.460 $\pm$ 0.013	18.121	18.091	18.818
T2G-Former	0.779 $\pm$ 0.037	0.822 $\pm$ 0.047	0.482 $\pm$ 0.092	17.727	19.091	15.061
RealMLP	0.777 $\pm$ 0.020	0.822 $\pm$ 0.015	0.512 $\pm$ 0.032	19.788	20.030	13.697
MLP	0.772 $\pm$ 0.016	0.822 $\pm$ 0.014	0.459 $\pm$ 0.019	17.576	16.697	18.727
FT-Transformer	0.770 $\pm$ 0.054	0.803 $\pm$ 0.067	0.468 $\pm$ 0.089	19.545	20.667	17.788
DCN-v2	0.769 $\pm$ 0.037	0.833 $\pm$ 0.019	0.482 $\pm$ 0.051	20.333	18.424	17.394
AutoInt	0.769 $\pm$ 0.042	0.826 $\pm$ 0.032	0.474 $\pm$ 0.060	18.788	19.636	17.788
NODE	0.769 $\pm$ 0.073	0.792 $\pm$ 0.070	0.352 $\pm$ 0.145	19.333	20.515	23.485
SNN	0.755 $\pm$ 0.054	0.818 $\pm$ 0.033	0.442 $\pm$ 0.054	21.273	21.091	21.152
SwitchTab	0.754 $\pm$ 0.040	0.799 $\pm$ 0.030	0.409 $\pm$ 0.082	22.303	23.939	21.485
DANets	0.749 $\pm$ 0.050	0.776 $\pm$ 0.084	0.453 $\pm$ 0.073	22.182	22.848	17.545
TabCaps	0.742 $\pm$ 0.019	0.837 $\pm$ 0.007	0.471 $\pm$ 0.020	21.394	17.000	18.818
TabTransformer	0.739 $\pm$ 0.021	0.781 $\pm$ 0.022	0.438 $\pm$ 0.040	19.667	20.424	18.303
TabNet	0.709 $\pm$ 0.089	0.789 $\pm$ 0.070	0.438 $\pm$ 0.081	24.848	21.364	22.303
SAINT	0.694 $\pm$ 0.021	0.739 $\pm$ 0.018	0.437 $\pm$ 0.029	23.576	24.061	20.606
GrowNet	0.646 $\pm$ 0.052	0.674 $\pm$ 0.083	0.361 $\pm$ 0.072	25.697	27.091	22.394

Table 8: Statistical profile of the benchmark TabArena, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

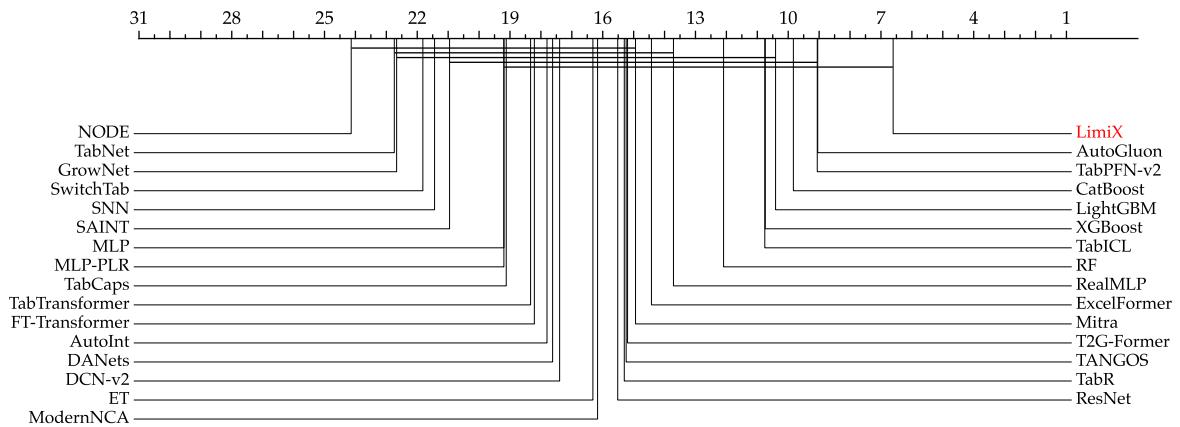
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	9	21	129	125	374	4	1776
# Classes	2	2	3	262	1577	2	9856
Missing Values (Ratio)	0	0	0.056	0.027	0.109	0	0.672
Categorical Features (Ratio)	0	0.47	1	0.45	0.366	0	1
Features w/ Missing Values (Ratio)	0	0	0.696	0.139	0.294	0	0.994



(a) AUC on the TabArena benchmark.



(b) Accuracy on the TabArena benchmark.



(c) F1-score on the TabArena benchmark.

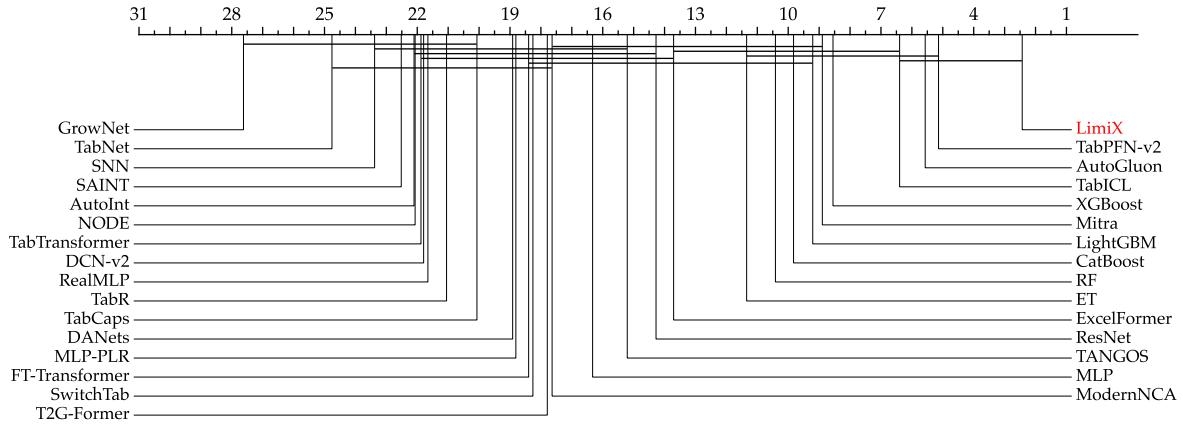
Figure 11: Critical difference diagrams on the TabArena benchmark.

Table 9: Classification results on the PFN-CLS benchmark. The best scores are shown in bold.

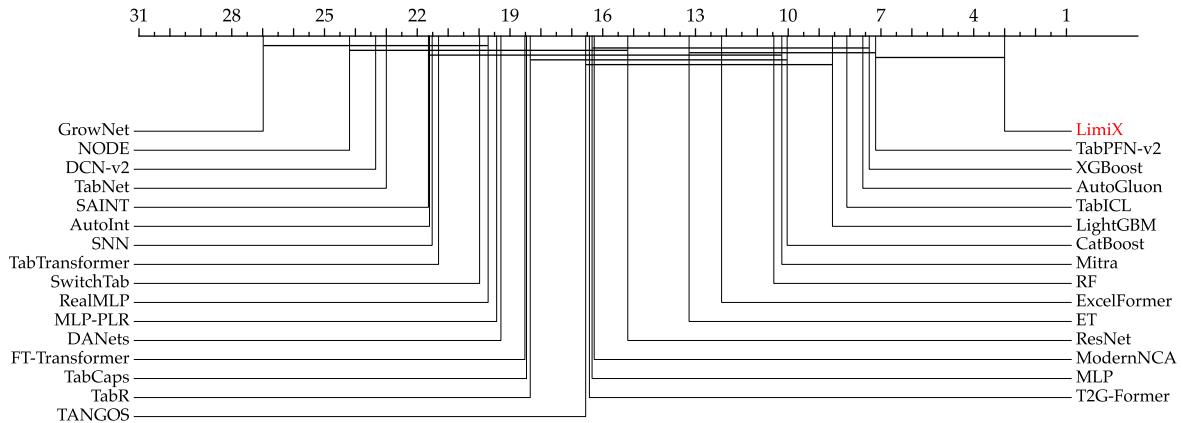
	PFN-CLS					
Model	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
LimiX	<b>0.923</b> $\pm$ 0.005	<b>0.862</b> $\pm$ 0.009	<b>0.786</b> $\pm$ 0.015	<b>2.034</b>	<b>3.000</b>	<b>2.862</b>
TabPFN-v2	0.910 $\pm$ 0.002	0.845 $\pm$ 0.007	0.756 $\pm$ 0.010	4.862	6.793	6.483
AutoGluon	0.906 $\pm$ 0.001	0.835 $\pm$ 0.001	0.738 $\pm$ 0.002	5.172	7.069	6.793
TabICL	0.903 $\pm$ 0.005	0.832 $\pm$ 0.008	0.742 $\pm$ 0.014	6.276	8.034	8.379
XGBoost	0.898 $\pm$ 0.003	0.831 $\pm$ 0.007	0.733 $\pm$ 0.023	8.241	6.862	7.759
Mitra	0.897 $\pm$ 0.010	0.826 $\pm$ 0.017	0.719 $\pm$ 0.032	8.621	9.552	11.276
RF	0.896 $\pm$ 0.006	0.822 $\pm$ 0.008	0.721 $\pm$ 0.019	10.414	10.172	11.310
CatBoost	0.895 $\pm$ 0.005	0.819 $\pm$ 0.009	0.720 $\pm$ 0.015	9.552	9.448	10.621
LightGBM	0.893 $\pm$ 0.004	0.826 $\pm$ 0.006	0.725 $\pm$ 0.016	8.793	8.103	8.655
ET	0.893 $\pm$ 0.003	0.809 $\pm$ 0.007	0.675 $\pm$ 0.018	11.345	12.759	14.586
ExcelFormer	0.883 $\pm$ 0.008	0.812 $\pm$ 0.012	0.713 $\pm$ 0.023	13.379	11.621	12.897
ResNet	0.869 $\pm$ 0.011	0.801 $\pm$ 0.010	0.700 $\pm$ 0.021	14.276	14.621	13.931
MLP	0.866 $\pm$ 0.007	0.795 $\pm$ 0.009	0.695 $\pm$ 0.019	16.310	15.759	15.310
TANGOS	0.865 $\pm$ 0.010	0.797 $\pm$ 0.009	0.698 $\pm$ 0.019	14.931	15.931	14.310
ModernNCA	0.860 $\pm$ 0.011	0.798 $\pm$ 0.010	0.692 $\pm$ 0.020	17.345	15.759	14.724
SwitchTab	0.858 $\pm$ 0.014	0.776 $\pm$ 0.013	0.626 $\pm$ 0.044	18.241	19.448	19.069
FT-Transformer	0.849 $\pm$ 0.018	0.789 $\pm$ 0.012	0.683 $\pm$ 0.017	18.103	17.897	16.586
T2G-Former	0.848 $\pm$ 0.024	0.792 $\pm$ 0.016	0.688 $\pm$ 0.031	17.379	15.897	15.897
MLP-PLR	0.848 $\pm$ 0.006	0.786 $\pm$ 0.007	0.650 $\pm$ 0.020	18.414	18.828	19.931
DCN-v2	0.846 $\pm$ 0.021	0.771 $\pm$ 0.020	0.633 $\pm$ 0.058	21.793	23.000	23.103
DANets	0.844 $\pm$ 0.039	0.770 $\pm$ 0.040	0.631 $\pm$ 0.077	18.897	18.862	19.966
NODE	0.844 $\pm$ 0.058	0.754 $\pm$ 0.060	0.535 $\pm$ 0.175	22.069	23.862	26.000
TabR	0.842 $\pm$ 0.028	0.789 $\pm$ 0.013	0.688 $\pm$ 0.024	20.759	17.931	16.483
AutoInt	0.838 $\pm$ 0.024	0.772 $\pm$ 0.017	0.640 $\pm$ 0.049	22.103	21.241	21.069
TabCaps	0.834 $\pm$ 0.022	0.788 $\pm$ 0.014	0.636 $\pm$ 0.031	20.069	18.103	19.207
SNN	0.831 $\pm$ 0.027	0.762 $\pm$ 0.023	0.595 $\pm$ 0.063	23.379	21.138	23.034
RealMLP	0.829 $\pm$ 0.016	0.780 $\pm$ 0.008	0.678 $\pm$ 0.014	21.241	19.138	18.241
TabNet	0.825 $\pm$ 0.043	0.768 $\pm$ 0.027	0.631 $\pm$ 0.059	24.759	22.586	23.793
TabTransformer	0.821 $\pm$ 0.025	0.761 $\pm$ 0.015	0.604 $\pm$ 0.057	21.483	20.724	22.379
GrowNet	0.756 $\pm$ 0.024	0.689 $\pm$ 0.019	0.497 $\pm$ 0.034	27.621	26.655	26.138
SAINT	0.708 $\pm$ 0.033	0.669 $\pm$ 0.020	0.563 $\pm$ 0.037	22.241	21.379	20.276

Table 10: Statistical profile of the benchmark PFN-CLS, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

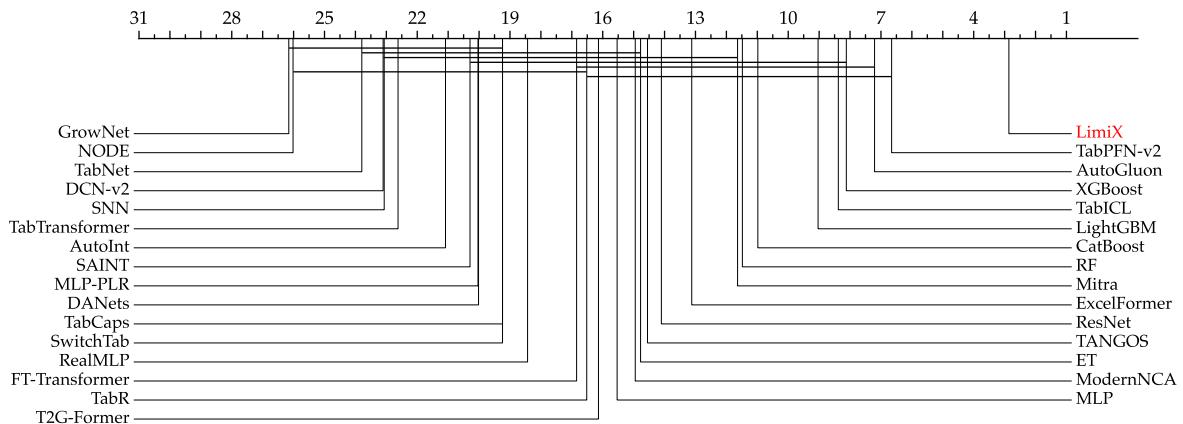
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	21	187	58	80	4	308
# Classes	2	2	7	4	2	2	10
Missing Values (Ratio)	0	0	0	0.001	0.006	0	0.032
Categorical Features (Ratio)	0	0.081	0.956	0.291	0.381	0	1
Features w/ Missing Values (Ratio)	0	0	0	0.016	0.086	0	0.474



(a) AUC on the PFN-CLS benchmark



(b) Accuracy on the PFN-CLS benchmark



(c) F1-score on the PFN-CLS benchmark

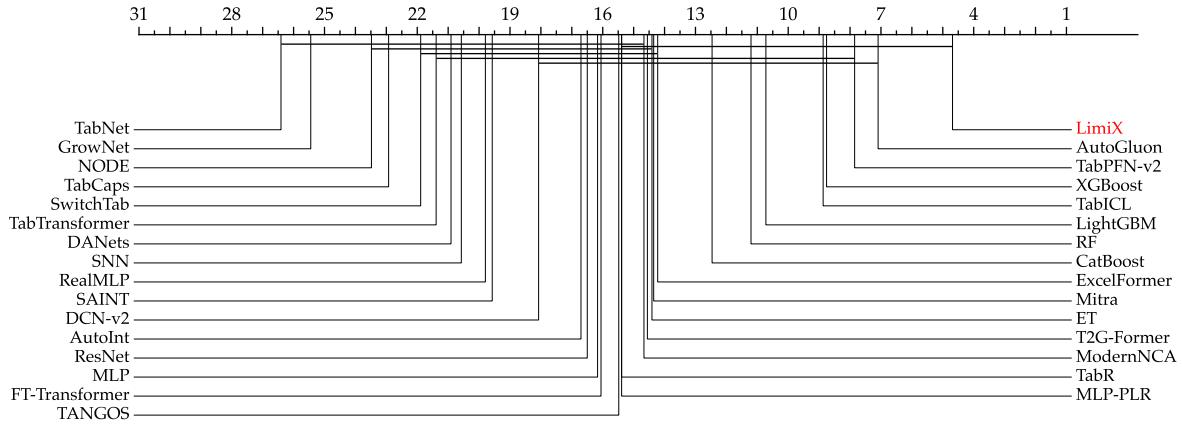
Figure 12: Critical difference diagrams on PFN-CLS benchmark

Table 11: Classification results on the TabZilla benchmark. The best scores are shown in bold.

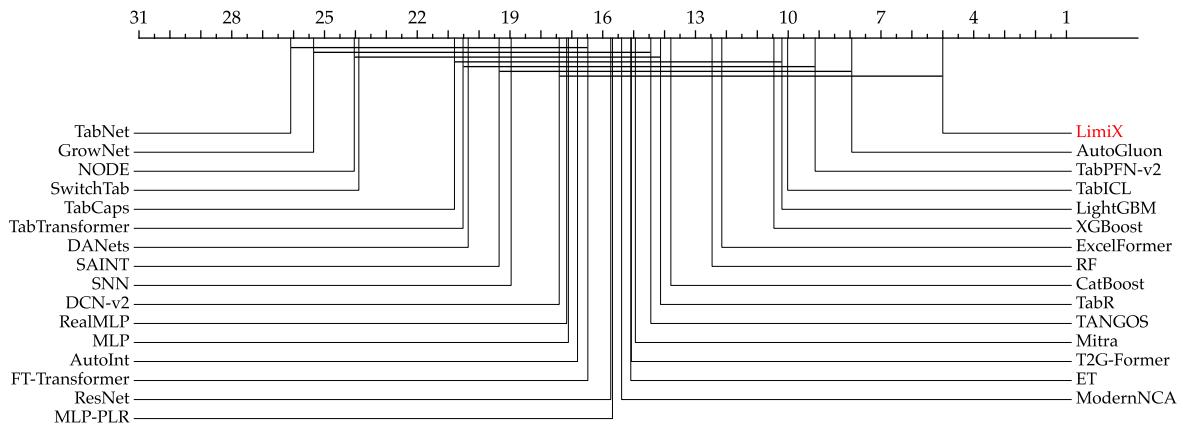
	TabZilla					
Model	Mean			Rank		
	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC ( $\downarrow$ )	Acc. ( $\downarrow$ )	F1 ( $\downarrow$ )
Limix	<b>0.943</b> $\pm$ 0.005	<b>0.885</b> $\pm$ 0.018	<b>0.836</b> $\pm$ 0.022	<b>4.556</b>	<b>4.593</b>	<b>5.741</b>
AutoGluon	0.933 $\pm$ 0.001	0.871 $\pm$ 0.002	0.803 $\pm$ 0.008	6.778	7.148	8.593
TabICL	0.933 $\pm$ 0.006	0.864 $\pm$ 0.016	0.803 $\pm$ 0.023	8.444	9.185	10.148
TabPFN-v2	0.929 $\pm$ 0.002	0.863 $\pm$ 0.008	0.797 $\pm$ 0.012	7.333	8.481	8.889
XGBoost	0.929 $\pm$ 0.003	0.863 $\pm$ 0.005	0.789 $\pm$ 0.011	8.741	10.074	11.185
LightGBM	0.927 $\pm$ 0.003	0.863 $\pm$ 0.007	0.796 $\pm$ 0.010	10.444	9.667	10.185
RF	0.924 $\pm$ 0.004	0.852 $\pm$ 0.006	0.773 $\pm$ 0.012	10.926	11.852	12.481
CatBoost	0.922 $\pm$ 0.006	0.848 $\pm$ 0.011	0.780 $\pm$ 0.020	12.185	13.296	13.185
Mitra	0.915 $\pm$ 0.010	0.841 $\pm$ 0.015	0.758 $\pm$ 0.037	14.074	14.074	15.037
ExcelFormer	0.915 $\pm$ 0.012	0.861 $\pm$ 0.014	0.802 $\pm$ 0.029	13.815	11.444	11.444
ET	0.912 $\pm$ 0.004	0.837 $\pm$ 0.008	0.745 $\pm$ 0.014	14.407	14.926	16.222
T2G-Former	0.909 $\pm$ 0.009	0.852 $\pm$ 0.012	0.790 $\pm$ 0.014	14.148	14.333	14.407
TANGOS	0.909 $\pm$ 0.006	0.841 $\pm$ 0.014	0.776 $\pm$ 0.032	15.148	13.704	14.148
ResNet	0.908 $\pm$ 0.008	0.834 $\pm$ 0.017	0.769 $\pm$ 0.028	16.444	15.148	14.741
ModernNCA	0.907 $\pm$ 0.007	0.850 $\pm$ 0.012	0.794 $\pm$ 0.017	14.222	14.556	13.926
MLP-PLR	0.906 $\pm$ 0.005	0.847 $\pm$ 0.008	0.773 $\pm$ 0.016	14.963	14.889	15.926
TabR	0.904 $\pm$ 0.014	0.853 $\pm$ 0.014	0.793 $\pm$ 0.029	14.963	13.185	13.630
DCN-v2	0.904 $\pm$ 0.008	0.844 $\pm$ 0.013	0.781 $\pm$ 0.018	17.556	16.519	17.111
MLP	0.903 $\pm$ 0.013	0.825 $\pm$ 0.021	0.747 $\pm$ 0.042	16.148	16.556	17.000
FT-Transformer	0.903 $\pm$ 0.023	0.842 $\pm$ 0.027	0.769 $\pm$ 0.058	15.926	15.926	15.370
AutoInt	0.896 $\pm$ 0.020	0.833 $\pm$ 0.030	0.748 $\pm$ 0.069	16.704	16.259	16.519
RealMLP	0.892 $\pm$ 0.012	0.846 $\pm$ 0.009	0.781 $\pm$ 0.018	19.370	16.407	15.741
TabCaps	0.887 $\pm$ 0.014	0.816 $\pm$ 0.016	0.729 $\pm$ 0.032	22.926	20.296	20.630
DANets	0.881 $\pm$ 0.039	0.800 $\pm$ 0.050	0.712 $\pm$ 0.091	20.889	19.963	20.556
SNN	0.874 $\pm$ 0.021	0.816 $\pm$ 0.020	0.706 $\pm$ 0.038	20.444	18.704	19.741
NODE	0.869 $\pm$ 0.046	0.784 $\pm$ 0.046	0.633 $\pm$ 0.101	23.481	23.593	25.556
SwitchTab	0.860 $\pm$ 0.030	0.764 $\pm$ 0.045	0.660 $\pm$ 0.073	21.889	23.519	23.148
TabNet	0.860 $\pm$ 0.035	0.771 $\pm$ 0.051	0.668 $\pm$ 0.070	26.407	25.778	25.407
GrowNet	0.829 $\pm$ 0.019	0.732 $\pm$ 0.017	0.651 $\pm$ 0.032	25.444	25.037	24.037
SAINT	0.824 $\pm$ 0.013	0.764 $\pm$ 0.017	0.680 $\pm$ 0.046	19.148	18.852	18.556
TabTransformer	0.814 $\pm$ 0.020	0.759 $\pm$ 0.024	0.659 $\pm$ 0.051	21.333	19.852	20.778

Table 12: Statistical profile of the benchmark TabZilla, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

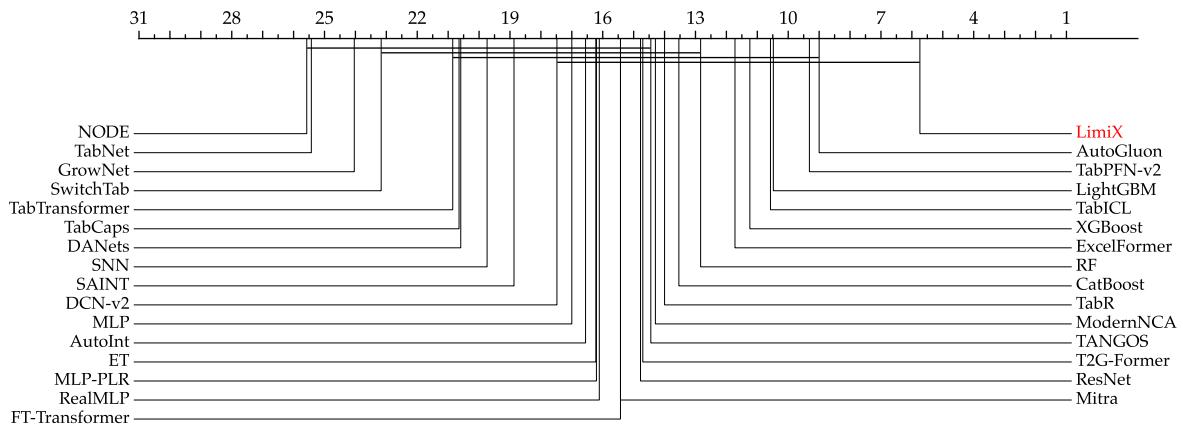
Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	24	132	226	756	4	4296
# Classes	2	2	10	7	16	2	100
Missing Values (Ratio)	0	0	0.078	0.022	0.059	0	0.205
Categorical Features (Ratio)	0	0.528	1	0.474	0.39	0	1
Features w/ Missing Values (Ratio)	0	0	0.483	0.102	0.218	0	0.808



(a) AUC on the TabZilla benchmark.



(b) Accuracy on the TabZilla benchmark.



(c) F1-score on the TabZilla benchmark.

Figure 13: Critical difference diagrams on the TabZilla benchmark.

## 7.2 Regression

**Benchmark.** For the quantitative evaluation of regression performance, we leverage four benchmarks, including three open-source benchmark, TALENT-REG (Liu et al., 2024), PFN-REG (Hollmann et al., 2025), and CTR23 (Fischer et al., 2023). After applying filters to exclude datasets containing more than 50,000 training samples or 10,000 features, we have 33 datasets from CTR23, 28 from PFN-REG, and 99 from TALENT-REG.

Similar to BCCO-CLS, we introduce BCCO-REG, a balanced regression benchmark comprising 50 datasets. For each dataset, we adopt the provided train-test split when available. If no predefined test set is available, we partition the data into a 70% training set and a 30% testing set randomly. Similar to the protocol employed in classification tasks, regression datasets with more than 50,000 training samples or 10,000 features are excluded.

**Baselines.** For comparison, we include tree-based methods, NN-based methods, AutoML frameworks, and ICL-based models, which are basically consistent with baselines employed in classification tasks described above. TabICL and TabCaps are excluded because it cannot handle regression tasks. We add another baseline, DNNR (Nader et al., 2022), which is specifically designed for regression. The model training procedure remains aligned with that employed in the classification experiments.

**Metrics.** To evaluate regression performance properly, we employ normalized RMSE and  $R^2$  as the two evaluation metrics. We also analyze the ranks of models with respect to both metrics.

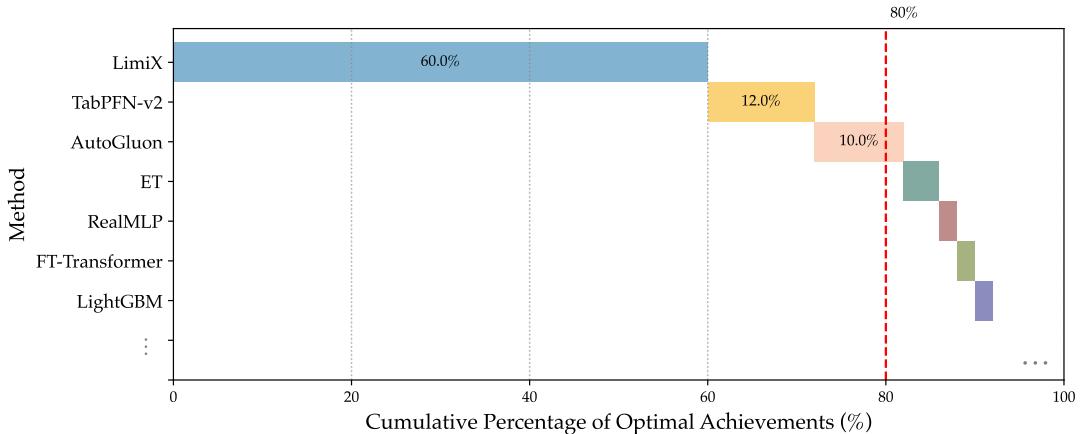
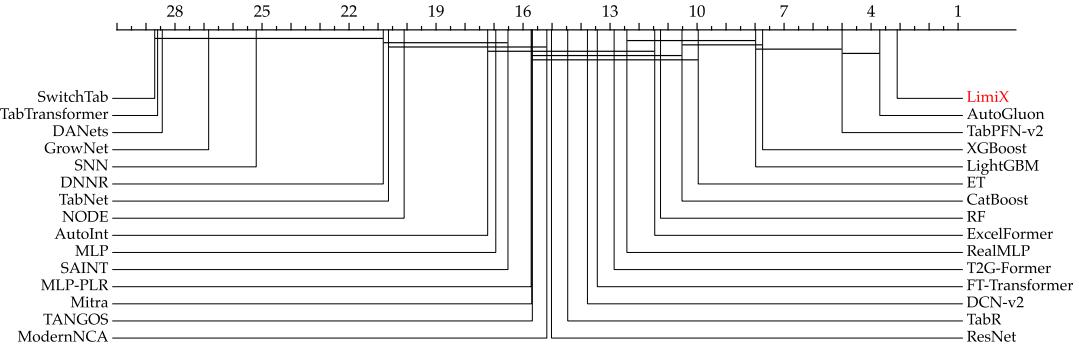


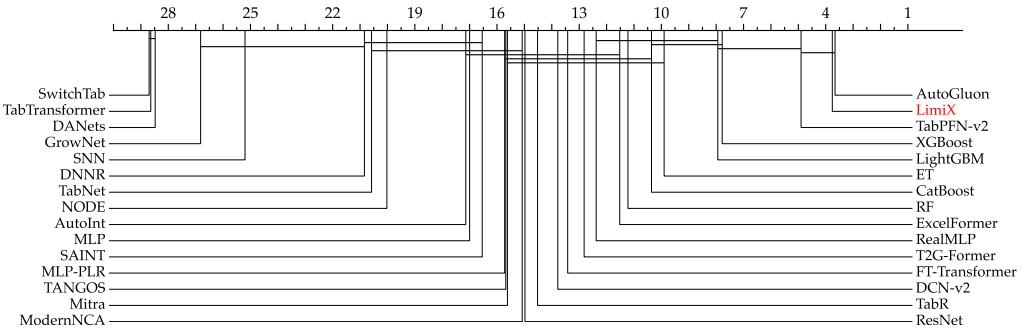
Figure 14: The proportion of models achieving the best  $R^2$ . The length of each bar represents the proportion of 106 datasets in which a given method achieved the highest  $R^2$  on the BCCO-REG benchmark.

**Results.** As shown by critical difference diagrams, i.e. Figures 15 to 18, LimiX always achieves the best performance, outperforming strong baselines like AutoGluon, XGBoost, and TabPFN-v2. Quantitative results in Tables 13, 15, 17 and 19 also confirms that LimiX achieves state-of-the-art regression performance, leading in both normalized RMSE and  $R^2$ .

**Subgroup analysis.** Similar to the subgroup analysis in classification, for regression, we also use the sample subgroups when building BCCO-REG to perform stratified analyses, whose criteria include the number of training samples, the ratio between the number of samples and features (length-to-width ratio), and the proportion of categorical features. Figure 20 shows that LimiX outperforms other methods across all subgroups compared with other methods. For Figures 20b to 20d, we can see that performance of Mitra substantially drops for the second subgroup, while LimiX do not exhibit a large drop.

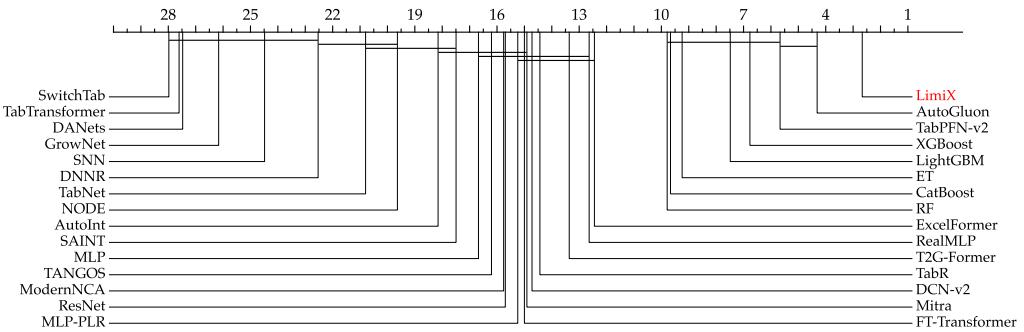


(a)  $R^2$  on the BCCO-REG benchmark

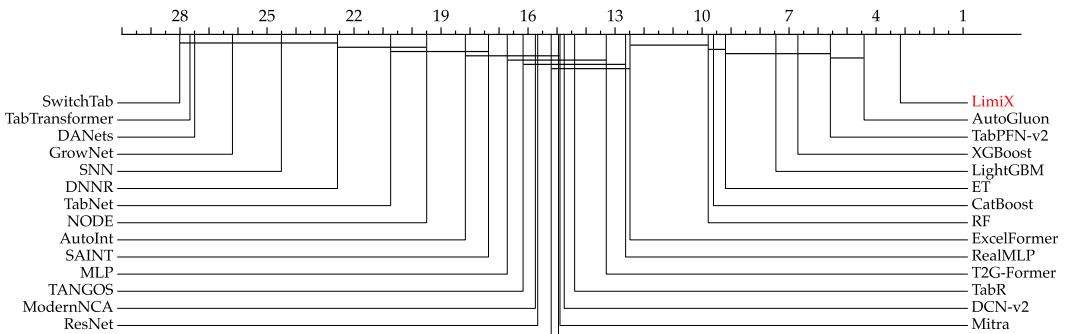


(b) RMSE on the BCCO-REG benchmark

Figure 15: Critical difference diagram on the BCCO-REG benchmark

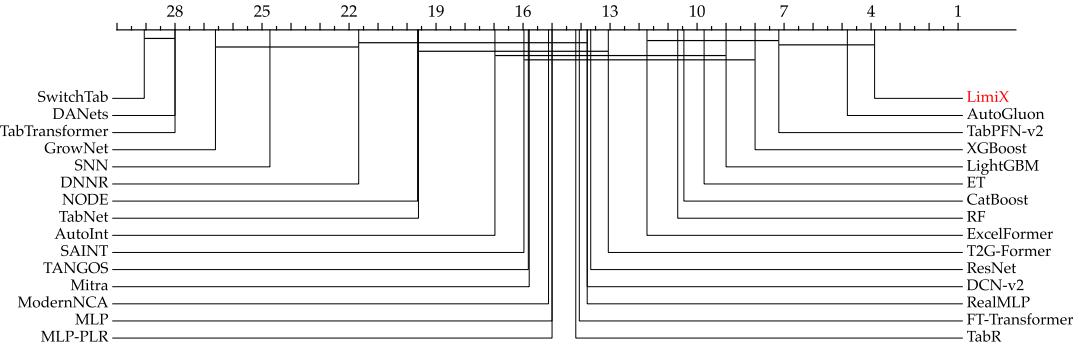


(a)  $R^2$  on the TALENT-REG benchmark

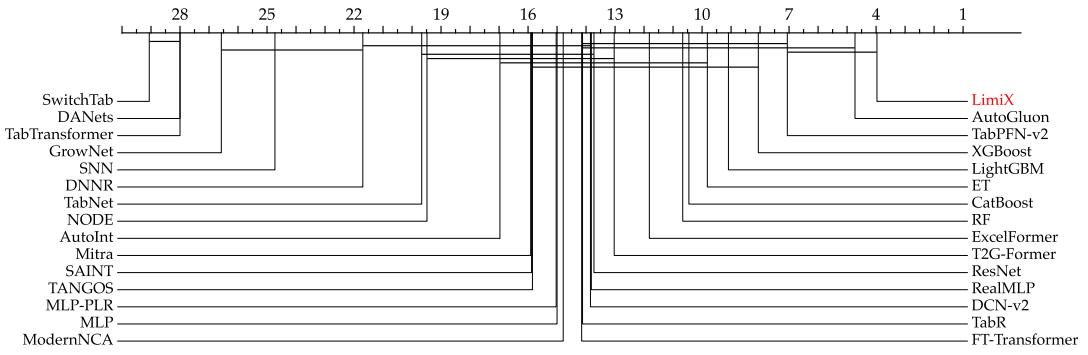


(b) RMSE on TALENT-REG benchmark

Figure 16: Critical difference diagram on the TALENT-REG benchmark

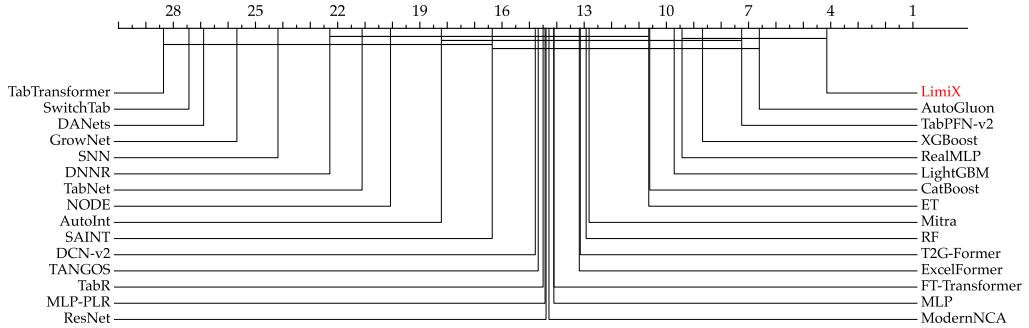


(a)  $R^2$  on the CTR23 benchmark

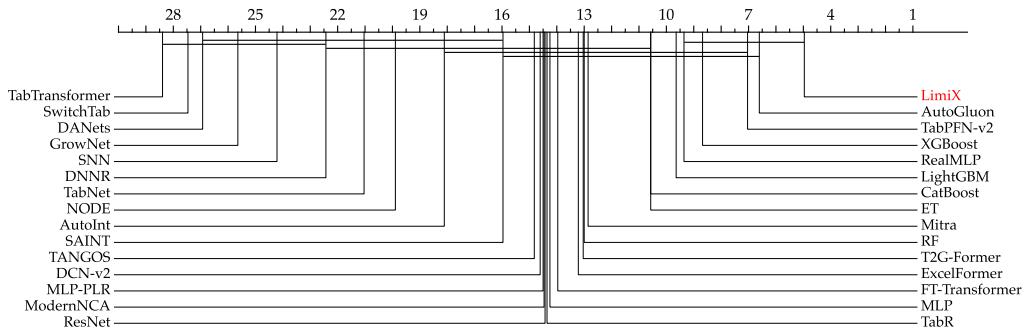


(b) RMSE on the CTR23 benchmark

Figure 17: Critical difference diagram on the CTR23 benchmark



(a)  $R^2$  on the PFN-REG benchmark



(b) RMSE on PFN-REG benchmark

Figure 18: Critical difference diagram on the PFN-REG benchmark

Table 13: Regression results on the BCCO-REG benchmark. The best scores are shown in bold.

Model	BCCO-REG			
	Mean		Rank	
	R <sup>2</sup> ( $\uparrow$ )	RMSE ( $\downarrow$ )	R <sup>2</sup> ( $\downarrow$ )	RMSE ( $\downarrow$ )
Limix	<b>0.794</b> $\pm$ 0.004	<b>0.386</b> $\pm$ 0.004	<b>3.100</b>	3.760
AutoGluon	0.781 $\pm$ 0.001	0.398 $\pm$ 0.000	3.700	<b>3.660</b>
TabPFN-v2	0.772 $\pm$ 0.001	0.404 $\pm$ 0.002	5.000	4.900
XGBoost	0.764 $\pm$ 0.006	0.415 $\pm$ 0.005	7.740	7.780
ET	0.757 $\pm$ 0.005	0.431 $\pm$ 0.006	9.960	9.900
RF	0.752 $\pm$ 0.007	0.438 $\pm$ 0.009	11.260	11.220
T2G-Former	0.743 $\pm$ 0.011	0.442 $\pm$ 0.012	12.860	12.820
ExcelFormer	0.743 $\pm$ 0.015	0.443 $\pm$ 0.012	11.460	11.520
CatBoost	0.741 $\pm$ 0.026	0.427 $\pm$ 0.020	10.520	10.360
DCN-v2	0.739 $\pm$ 0.015	0.448 $\pm$ 0.014	13.780	13.780
FT-Transformer	0.737 $\pm$ 0.009	0.448 $\pm$ 0.010	13.440	13.420
MLP-PLR	0.734 $\pm$ 0.012	0.453 $\pm$ 0.009	15.720	15.720
TabR	0.733 $\pm$ 0.017	0.448 $\pm$ 0.011	14.460	14.520
RealMLP	0.725 $\pm$ 0.029	0.441 $\pm$ 0.010	12.420	12.380
AutoInt	0.724 $\pm$ 0.015	0.465 $\pm$ 0.018	17.220	17.140
ResNet	0.720 $\pm$ 0.009	0.468 $\pm$ 0.010	15.020	14.980
TANGOS	0.719 $\pm$ 0.010	0.468 $\pm$ 0.009	15.680	15.680
LightGBM	0.715 $\pm$ 0.019	0.423 $\pm$ 0.005	7.980	7.940
MLP	0.701 $\pm$ 0.013	0.487 $\pm$ 0.010	16.940	17.000
SAINT	0.701 $\pm$ 0.022	0.481 $\pm$ 0.019	16.520	16.540
TabNet	0.670 $\pm$ 0.024	0.516 $\pm$ 0.024	20.640	20.580
Mitra	0.667 $\pm$ 0.027	0.474 $\pm$ 0.015	15.700	15.620
NODE	0.643 $\pm$ 0.099	0.543 $\pm$ 0.097	20.100	20.020
ModernNCA	0.598 $\pm$ 0.195	0.471 $\pm$ 0.038	15.180	15.080
SNN	0.434 $\pm$ 0.111	0.720 $\pm$ 0.072	25.200	25.200
GrowNet	0.201 $\pm$ 0.182	0.864 $\pm$ 0.110	26.840	26.820
DANets	0.005 $\pm$ 0.008	0.979 $\pm$ 0.004	28.440	28.480
SwitchTab	0.001 $\pm$ 0.032	0.981 $\pm$ 0.016	28.700	28.700
TabTransformer	0.000 $\pm$ 0.016	0.981 $\pm$ 0.008	28.600	28.640
DNNR	-2.152 $\pm$ 0.150	1.329 $\pm$ 0.027	20.820	20.840

Table 14: Statistical profile of the benchmark BCCO-REG, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	12	33	16	14	4	81
Missing Values (Ratio)	0	0	0	0	0	0	0.001
Categorical Features (Ratio)	0	0.16	0.65	0.242	0.264	0	0.967
Features w/ Missing Values (Ratio)	0	0	0	0.008	0.051	0	0.333

Table 15: Regression results on the TALENT-REG benchmark. The best scores are shown in bold.

Model	TALENT-REG			
	Mean		Rank	
	R <sup>2</sup> ( $\uparrow$ )	RMSE ( $\downarrow$ )	R <sup>2</sup> ( $\downarrow$ )	RMSE ( $\downarrow$ )
LimiX	<b>0.735</b> $\pm$ 0.009	<b>0.433</b> $\pm$ 0.009	<b>2.667</b>	<b>3.162</b>
AutoGluon	0.722 $\pm$ 0.006	0.448 $\pm$ 0.009	4.313	4.414
XGBoost	0.710 $\pm$ 0.006	0.462 $\pm$ 0.005	6.768	6.697
LightGBM	0.707 $\pm$ 0.004	0.461 $\pm$ 0.004	7.485	7.455
CatBoost	0.700 $\pm$ 0.015	0.471 $\pm$ 0.018	9.667	9.606
RF	0.697 $\pm$ 0.008	0.474 $\pm$ 0.009	9.788	9.788
ET	0.696 $\pm$ 0.008	0.476 $\pm$ 0.007	9.242	9.192
TabPFN-v2	0.695 $\pm$ 0.003	0.465 $\pm$ 0.002	5.667	5.576
RealMLP	0.656 $\pm$ 0.013	0.510 $\pm$ 0.010	12.636	12.636
T2G-Former	0.656 $\pm$ 0.019	0.512 $\pm$ 0.017	13.364	13.303
MLP-PLR	0.653 $\pm$ 0.017	0.521 $\pm$ 0.017	15.242	15.202
ExcelFormer	0.653 $\pm$ 0.016	0.512 $\pm$ 0.012	12.444	12.485
TabR	0.651 $\pm$ 0.023	0.516 $\pm$ 0.021	14.434	14.394
FT-Transformer	0.648 $\pm$ 0.019	0.519 $\pm$ 0.018	15.000	14.949
AutoInt	0.636 $\pm$ 0.020	0.538 $\pm$ 0.021	18.152	18.162
ModernNCA	0.633 $\pm$ 0.033	0.530 $\pm$ 0.029	15.758	15.747
Mitra	0.602 $\pm$ 0.053	0.547 $\pm$ 0.022	14.909	14.899
TANGOS	0.592 $\pm$ 0.027	0.547 $\pm$ 0.013	16.202	16.172
TabNet	0.576 $\pm$ 0.042	0.586 $\pm$ 0.032	20.798	20.737
NODE	0.568 $\pm$ 0.096	0.600 $\pm$ 0.091	19.636	19.495
ResNet	0.562 $\pm$ 0.095	0.550 $\pm$ 0.023	15.697	15.667
MLP	0.556 $\pm$ 0.025	0.564 $\pm$ 0.008	16.677	16.717
SNN	0.344 $\pm$ 0.138	0.777 $\pm$ 0.069	24.485	24.505
DANets	0.005 $\pm$ 0.009	0.998 $\pm$ 0.004	27.475	27.495
TabTransformer	0.001 $\pm$ 0.017	1.001 $\pm$ 0.008	27.606	27.657
SwitchTab	-0.002 $\pm$ 0.027	1.002 $\pm$ 0.013	27.980	28.010
GrowNet	-0.182 $\pm$ 0.319	0.920 $\pm$ 0.120	26.162	26.192
DCN-v2	-0.361 $\pm$ 0.161	0.818 $\pm$ 0.182	14.717	14.747
SAINT	-1.541 $\pm$ 0.014	0.571 $\pm$ 0.010	17.495	17.364
DNNR	-9.172 $\pm$ 5.278	2.528 $\pm$ 0.814	22.525	22.566

Table 16: Statistical profile of the benchmark TALENT-REG, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	11	79	30	49	4	266
Missing Values (Ratio)	0	0	0	0.007	0.034	0	0.283
Categorical Features (Ratio)	0	0.131	0.667	0.238	0.283	0	1
Features w/ Missing Values (Ratio)	0	0	0.013	0.043	0.161	0	0.875

Table 17: Regression results on the CTR23 benchmark. The best scores are shown in bold.

Model	CTR23			
	Mean		Rank	
	R <sup>2</sup> ( $\uparrow$ )	RMSE ( $\downarrow$ )	R <sup>2</sup> ( $\downarrow$ )	RMSE ( $\downarrow$ )
LimiX	<b>0.745</b> $\pm$ 0.008	<b>0.477</b> $\pm$ 0.007	<b>3.879</b>	<b>3.970</b>
AutoGluon	0.725 $\pm$ 0.001	0.497 $\pm$ 0.001	4.818	4.727
TabPFN-v2	0.716 $\pm$ 0.002	0.503 $\pm$ 0.002	7.182	7.061
XGBoost	0.712 $\pm$ 0.008	0.511 $\pm$ 0.007	8.000	8.061
LightGBM	0.706 $\pm$ 0.004	0.516 $\pm$ 0.005	9.000	9.091
CatBoost	0.700 $\pm$ 0.022	0.528 $\pm$ 0.030	10.455	10.455
ET	0.697 $\pm$ 0.007	0.535 $\pm$ 0.008	9.758	9.818
RF	0.694 $\pm$ 0.010	0.539 $\pm$ 0.012	10.667	10.667
T2G-Former	0.674 $\pm$ 0.013	0.544 $\pm$ 0.014	13.061	13.030
MLP-PLR	0.672 $\pm$ 0.013	0.553 $\pm$ 0.014	15.000	15.030
TabR	0.671 $\pm$ 0.014	0.543 $\pm$ 0.014	14.182	14.121
DCN-v2	0.670 $\pm$ 0.015	0.545 $\pm$ 0.015	13.788	13.848
FT-Transformer	0.667 $\pm$ 0.011	0.549 $\pm$ 0.011	14.061	14.152
ModernNCA	0.667 $\pm$ 0.027	0.550 $\pm$ 0.027	15.121	14.788
ExcelFormer	0.665 $\pm$ 0.014	0.556 $\pm$ 0.013	11.727	11.818
RealMLP	0.661 $\pm$ 0.013	0.549 $\pm$ 0.011	13.788	13.818
AutoInt	0.655 $\pm$ 0.015	0.568 $\pm$ 0.018	16.970	16.970
SAINT	0.654 $\pm$ 0.015	0.561 $\pm$ 0.017	15.970	15.879
ResNet	0.645 $\pm$ 0.018	0.587 $\pm$ 0.017	13.667	13.727
TANGOS	0.642 $\pm$ 0.014	0.586 $\pm$ 0.013	15.818	15.848
Mitra	0.624 $\pm$ 0.017	0.583 $\pm$ 0.016	15.788	15.909
MLP	0.608 $\pm$ 0.008	0.623 $\pm$ 0.008	15.000	15.000
TabNet	0.605 $\pm$ 0.046	0.623 $\pm$ 0.041	19.606	19.667
NODE	0.568 $\pm$ 0.105	0.666 $\pm$ 0.096	19.636	19.485
SNN	0.369 $\pm$ 0.073	0.834 $\pm$ 0.048	24.727	24.727
GrowNet	0.185 $\pm$ 0.184	0.944 $\pm$ 0.113	26.606	26.576
DANets	0.001 $\pm$ 0.010	1.052 $\pm$ 0.005	28.000	28.000
TabTransformer	0.000 $\pm$ 0.016	1.053 $\pm$ 0.008	28.000	28.000
SwitchTab	-0.006 $\pm$ 0.025	1.057 $\pm$ 0.013	29.061	29.061
DNNR	-2.969 $\pm$ 0.262	1.651 $\pm$ 0.040	21.667	21.697

Table 18: Statistical profile of the benchmark CTR23, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	11	39	20	22	5	116
Missing Values (Ratio)	0	0	0	0.008	0.036	0	0.209
Categorical Features (Ratio)	0	0.143	0.842	0.312	0.345	0	1
Features w/ Missing Values (Ratio)	0	0	0	0.011	0.05	0	0.286

Table 19: Regression results on the PFN-REG benchmark. The best scores are shown in bold.

Model	PFN-REG			
	Mean		Rank	
	R <sup>2</sup> ( $\uparrow$ )	RMSE ( $\downarrow$ )	R <sup>2</sup> ( $\downarrow$ )	RMSE ( $\downarrow$ )
LimiX	<b>0.692</b> $\pm$ 0.012	<b>0.461</b> $\pm$ 0.011	<b>4.143</b>	<b>4.964</b>
AutoGluon	0.677 $\pm$ 0.000	0.482 $\pm$ 0.001	6.607	6.607
TabPFN-v2	0.676 $\pm$ 0.004	0.475 $\pm$ 0.002	7.250	7.036
XGBoost	0.671 $\pm$ 0.007	0.487 $\pm$ 0.007	8.679	8.679
RealMLP	0.664 $\pm$ 0.012	0.484 $\pm$ 0.010	9.429	9.357
LightGBM	0.664 $\pm$ 0.008	0.496 $\pm$ 0.008	9.714	9.643
CatBoost	0.661 $\pm$ 0.014	0.498 $\pm$ 0.014	10.607	10.571
ET	0.652 $\pm$ 0.007	0.515 $\pm$ 0.008	10.643	10.571
ExcelFormer	0.646 $\pm$ 0.013	0.513 $\pm$ 0.013	13.179	13.214
ModernNCA	0.645 $\pm$ 0.021	0.508 $\pm$ 0.015	14.286	14.464
RF	0.643 $\pm$ 0.010	0.524 $\pm$ 0.011	12.929	13.000
T2G-Former	0.640 $\pm$ 0.027	0.503 $\pm$ 0.022	13.143	13.036
MLP-PLR	0.638 $\pm$ 0.017	0.513 $\pm$ 0.019	14.429	14.500
DCN-v2	0.638 $\pm$ 0.024	0.511 $\pm$ 0.022	14.786	14.607
TabR	0.636 $\pm$ 0.036	0.509 $\pm$ 0.030	14.500	14.357
FT-Transformer	0.636 $\pm$ 0.033	0.509 $\pm$ 0.027	14.107	13.964
Mitra	0.630 $\pm$ 0.014	0.531 $\pm$ 0.016	12.821	12.857
AutoInt	0.610 $\pm$ 0.031	0.539 $\pm$ 0.033	18.214	18.107
ResNet	0.599 $\pm$ 0.017	0.553 $\pm$ 0.013	14.393	14.429
TANGOS	0.587 $\pm$ 0.024	0.562 $\pm$ 0.025	14.679	14.821
MLP	0.577 $\pm$ 0.023	0.576 $\pm$ 0.016	14.107	14.250
NODE	0.496 $\pm$ 0.135	0.666 $\pm$ 0.117	20.071	19.893
TabNet	0.430 $\pm$ 0.216	0.643 $\pm$ 0.106	21.107	21.036
SNN	0.378 $\pm$ 0.079	0.756 $\pm$ 0.056	24.179	24.214
GrowNet	0.096 $\pm$ 0.141	0.942 $\pm$ 0.080	25.679	25.643
DANets	0.001 $\pm$ 0.012	0.997 $\pm$ 0.006	26.893	26.929
TabTransformer	-0.021 $\pm$ 0.029	1.007 $\pm$ 0.014	28.357	28.393
SwitchTab	-0.026 $\pm$ 0.034	1.011 $\pm$ 0.017	27.429	27.464
DNNR	-7.850 $\pm$ 0.583	2.224 $\pm$ 1.125	22.286	22.429
SAINT	-8.088 $\pm$ 0.020	0.612 $\pm$ 0.014	16.357	15.964

Table 20: Statistical profile of the benchmark PFN-REG, where Q10, Q50, and Q90 correspond to the 10%, 50%, and 90% quantiles, respectively; for categorical feature statistics, we only consider features that are either string-typed or have fewer than 10 unique values.

Metric	Statistics						
	Q10	Q50	Q90	Mean	Std	Min	Max
# Features	6	16	176	66	93	3	376
Missing Values (Ratio)	0	0	0.089	0.029	0.076	0	0.335
Categorical Features (Ratio)	0	0.209	0.949	0.355	0.381	0	1
Features w/ Missing Values (Ratio)	0	0	0.211	0.059	0.169	0	0.841

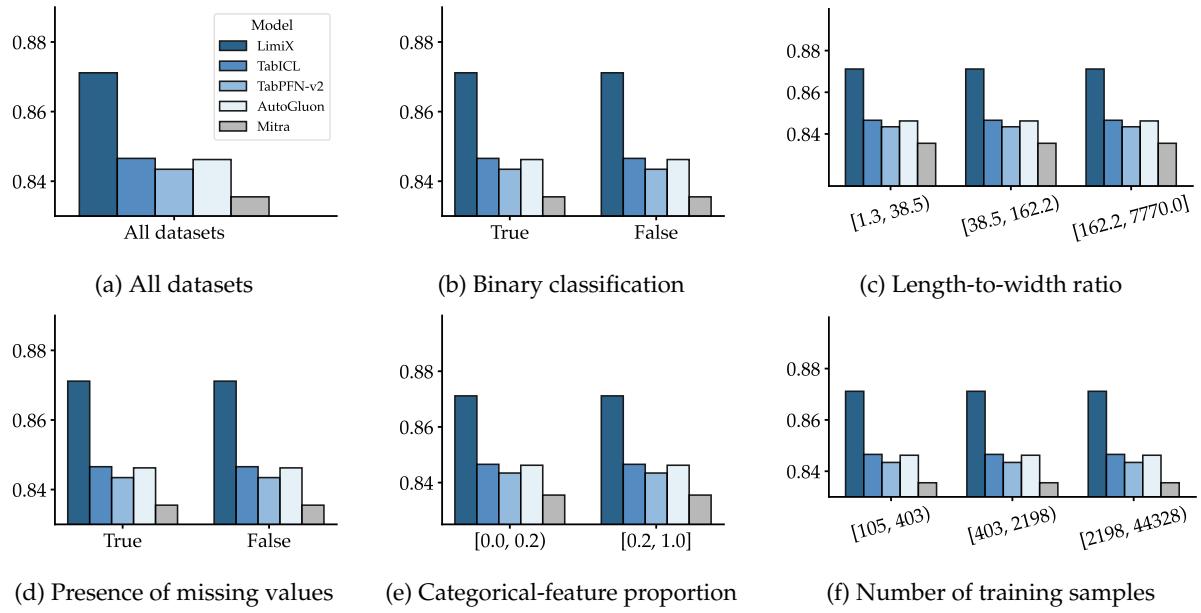


Figure 19: AUC on subsets with various sample size, number of classes, categorical–to-numerical feature ratios, missing values, and sample-to-feature ratios.

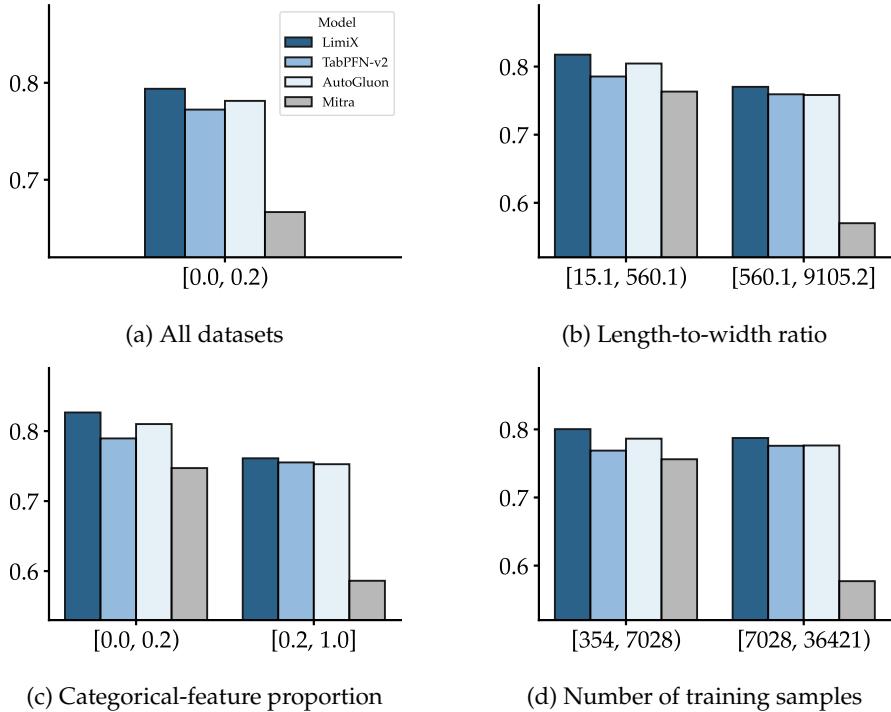


Figure 20:  $R^2$  on subsets with various sample size, categorical–to-numerical feature ratios, and sample-to-feature ratios.

### 7.3 Missing Value Imputation

The ubiquity of missing values in tabular data is harmful to downstream tasks or statistical analyses (Lin & Tsai, 2020). Meanwhile, the problem of missing value imputation poses a bigger challenge compared with standard classification and regression tasks since it requires modeling of the joint distribution of  $P(X, Y)$  instead of  $P(Y|X)$  only. A noteworthy capability of LimiX is missing value imputation, which naturally stems from the mask modeling during pretraining. Although there are existing deep learning approaches for missing value imputation (Jarrett et al., 2022; Zhang et al., 2025), they all require additional training on unseen datasets. In contrast, LimiX is the first model to be capable of missing value imputation on unseen datasets via in-context learning without additional training. This brings great convenience to the usage in downstream tasks.

We conduct experiments on the following datasets: Analcatdata BroadwayMult (Simonoff, 2003; Van-schoren, 2014a), Early Stage Diabetes 2020 (Islam et al., 2019; 2020), Forty Soybean Cultivars from Subsequent Harvests (de Oliveira et al., 2023; Rodrigues de Oliveira & Mario Zuffo, 2023), HCC survival (Santos et al., 2015b;a), Vehicle (Siebert, 1987; Mowforth & Shepherd, 1987), Eucalyptus (van Rijn, 2014), and Z-Alizadeh Sani (Alizadehsani et al., 2013). To simulate the scenario of missing value imputation, we randomly mask a fixed proportion  $\alpha$  of cells in  $x_{te}$  and try to recover values of these cells. In our experiments, we set  $\alpha = 0.05$ . Baselines include: Mean/Mode (use mean for continuous features and mode for categorical features), k-nearest neighbors (KNN), MICE (Van Buuren & Groothuis-Oudshoorn, 2011), MissForest (Stekhoven & Bühlmann, 2012), GAIN (Yoon et al., 2018), MIWAE (Mattei & Frellsen, 2019), and HyperImpute (Jarrett et al., 2022). For continuous features, we normalize each feature using MinMaxScaler and calculate RMSE between imputed values and ground truth values after normalization as the evaluation metric. For categorical features, the error rate is used as the evaluation metric. Table 21 shows that LimiX consistently outperforms previous baselines, all of which require additional training or fitting.

Table 21: Evaluation of missing value imputation. For continuous features, we calculate RMSE. For categorical features, we calculate classification error. Our method outperforms previous missing value imputation methods.

Metric	Regression RMSE ( $\downarrow$ )						Classification Error ( $\downarrow$ )	
Method	Analcatdata	Early Diabetes	Harvests	Hcc Survival	Vehicle	Eucalyptus	Z-Alizadeh Sani	
Mean/Mode	0.321	0.235	0.179	0.351	0.209	0.804	0.200	
KNN	0.358	0.205	0.158	0.246	0.083	0.275	0.206	
MICE	0.294	0.244	0.155	0.234	0.102	0.627	0.181	
MissForest	0.203	0.223	0.136	0.215	0.107	0.137	0.156	
GAIN	0.299	0.328	0.196	0.413	0.102	0.647	0.175	
MIWAE	0.561	0.478	0.322	0.639	0.415	0.706	0.294	
HyperImpute	0.272	0.270	0.152	0.297	0.086	0.647	0.194	
LimiX	<b>0.194</b>	<b>0.161</b>	<b>0.104</b>	<b>0.188</b>	<b>0.064</b>	<b>0.118</b>	<b>0.131</b>	

## 7.4 Robustness

Neural networks have been found to be vulnerable to various types of perturbations and attacks (Carlini & Wagner, 2017; Akhtar & Mian, 2018). In order to investigate the robustness of LimiX, following Hollmann et al. (2025), we conduct experiments under two types of controlled perturbations: adding uninformative features and outliers.

For uninformative features, we randomly select columns from the original dataset, shuffle each column, and concatenate them to the original dataset. For outliers, we multiply an outlier coefficient to each cell value in the original dataset with a probability of 2%. The outlier coefficient is randomly chosen between 0 and the outlier factor.

In Figure 21, we show Normalized AUC in classification tasks under the two types of perturbations. We compare LimiX with TabPFN-v2, TabICL, and CatBoost. The left figure shows that even when adding up to 90% uninformative features, the normalized AUC of LimiX remains nearly unchanged. In contrast, the performance of TabICL and CatBoost drops significantly. This indicates that LimiX is much more robust than TabICL and CatBoost. The right figure shows that LimiX consistently outperforms other methods regardless of the outlier factor. In Figure 22, we show RMSE in regression tasks under perturbations. In the left figure of adding uninformative features, a similar phenomenon is observed as that in classification tasks. In the right figure of adding outliers, we find that RMSE of TabPFN-v2 rapidly increases when the outlier factor grows from 100 to 10000, while LimiX do not. This demonstrates the superior robustness of LimiX compared with TabPFN-v2 in regression tasks. Overall, LimiX exhibits superior robustness relative to the baselines.

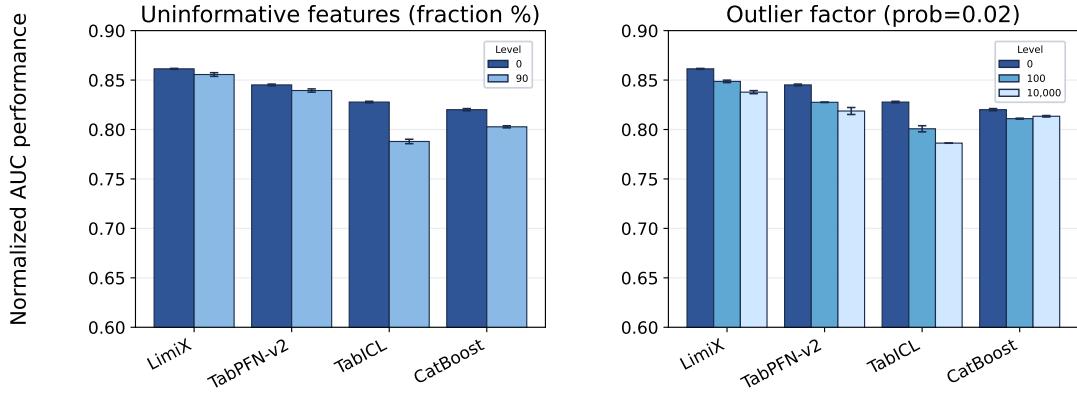


Figure 21: Robustness analysis in classification Tasks. LimiX consistently exhibits the best performance and superior robustness under perturbations.

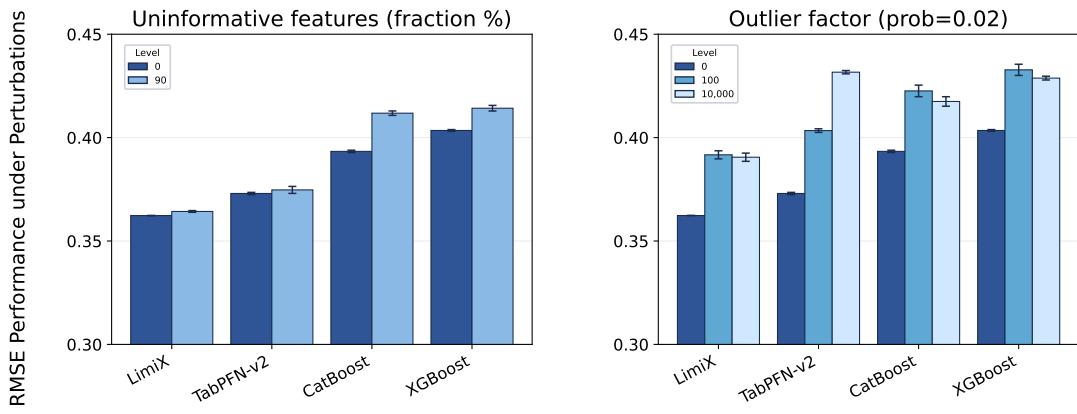


Figure 22: Robustness analysis in regression tasks. LimiX consistently exhibits the best performance and superior robustness under perturbations.

## 7.5 Embedding

Pretrained models (Devlin et al., 2019; Oquab et al., 2024) are generally expected to provide effective and transferable feature representations that can be leveraged for downstream tasks. To assess the quality of embeddings extracted by LimiX, we conduct experiments on six datasets of various sample sizes and numbers of categories. We compare embeddings of LimiX with those of MLP (Goodfellow et al., 2016; Gorishniy et al., 2021), ResNet (He et al., 2016; Gorishniy et al., 2021), ModernNCA (Ye et al., 2025), TabPFN-v2 (Hollmann et al., 2025), and TabICL (Qu et al., 2025). For each model, we treat representations prior to the classification head as embeddings. Figure 23 shows t-SNE visualization of embeddings. We can see that embeddings of different categories extracted by LimiX are more separated than those extracted by other models.

To further evaluate the quality of embeddings extracted by LimiX, we additionally conduct linear probing experiments, which are widely adopted in the analysis of feature representations (Kumar et al., 2022). The experiments are conducted on BCCO-CLS. Table 22 shows that embeddings of LimiX achieve the highest average AUC and rank among the three ICL-based models. Overall, LimiX consistently outperforms baselines in both qualitative and quantitative experiments.

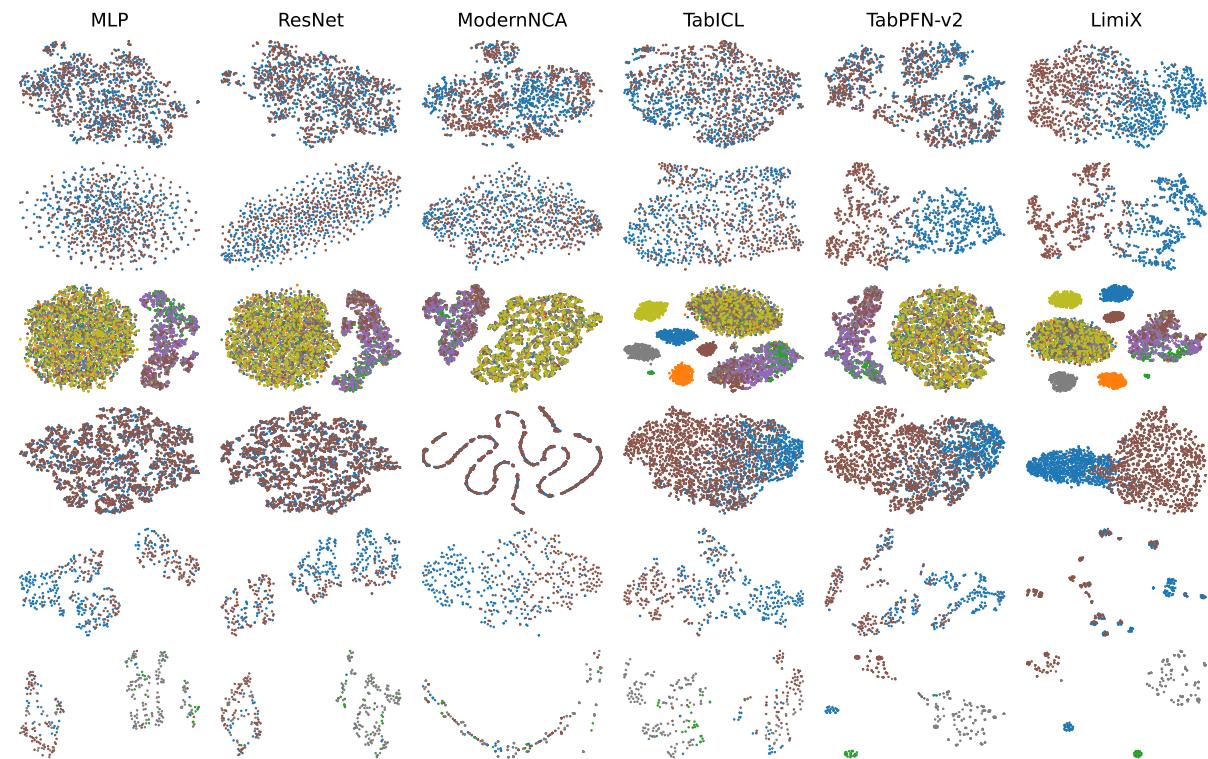


Figure 23: t-SNE visualization of embeddings. Different colors represent different categories. Embeddings extracted by LimiX are better separated between categories than other models.

Table 22: Results of linear probing on BCCO-CLS. LimiX outperforms the other two ICL-based models, indicating the strongest embedding capability.

Model	AUC( $\uparrow$ )	Rank( $\downarrow$ )
TabPFN-v2	0.832	2.189
TabICL	0.838	1.981
LimiX	<b>0.850</b>	<b>1.792</b>

## 7.6 Fine-tuning

Long contexts increase memory cost and can degrade optimization during fine-tuning. Therefore, it is common practice to subsample the training corpus to a manageable budget, usually via random or KNN selection (Thomas et al., 2024; Xu et al., 2025). Based on our attention-based retrieval strategy, we adopt a retrieval-guided downsampling scheme that concentrates on locally most relevant patterns within each dataset, thus improving sample efficiency and prediction performance.

For each dataset, we split the original training set into a retrieval pool and a query set. For each instance in the query set, we select samples from the retrieval pool via the strategy proposed in Section 5 as in-context samples and construct an episode of in-context learning. Fine-tuning then proceeds over these episodes rather than over full, unfiltered contexts. This strategy substantially reduces the number of epochs required to achieve good performance. Note that the retrieved in-context sets may contain duplicated samples, which leads to a trade-off between relevance and diversity. We empirically find that the retrieved context size controls the balance between computation efficiency and prediction performance.

We compare LimiX with some representative baselines that are trained or fine-tuned on real datasets, including TabDPT (Ma et al., 2024). For the fine-tuning of TabPFN-v2, we adopt the strategy and hyper-parameter configurations from its official repository<sup>2</sup>. As shown in Tables 23 and 24, LimiX significantly outperforms the others across various metrics in most benchmarks after fine-tuning. Meanwhile, we illustrate the change of AUC for LimiX on some datasets of BCCO-CLS before and after fine-tuning. From Figure 24, a significant improvement can be observed in most cases.

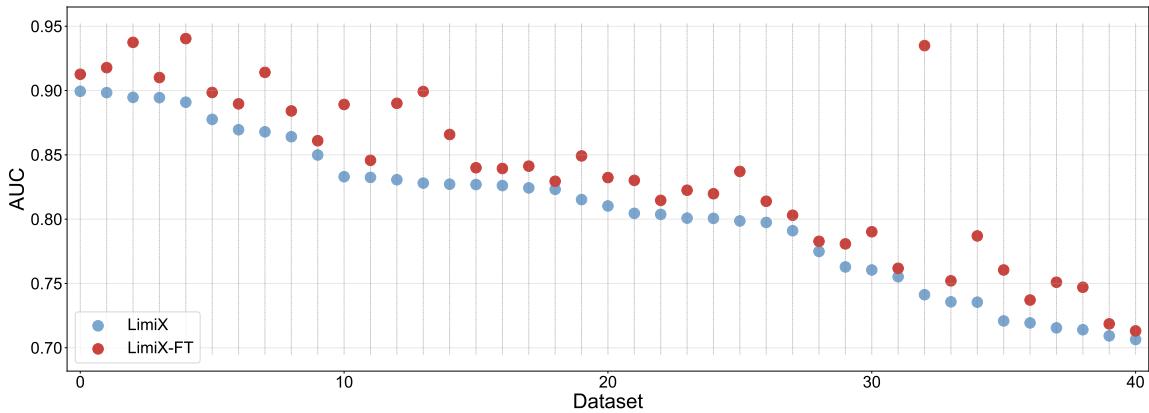


Figure 24: The improvement of fine-tuning in AUC on the BCCO-CLS benchmark.

<sup>2</sup><https://github.com/PriorLabs/TabPFN>

Table 23: Performance comparison of representative methods on benchmarks of classification. "FT" indicates the model performance after fine-tuning. TabDPT-NE represents the performance of TabDPT without ensemble while its default version has an ensemble size of 8. The best scores are shown in bold.

Benchmark	Method	AUC ( $\uparrow$ )	Acc. ( $\uparrow$ )	F1 ( $\uparrow$ )	AUC-Rank ( $\downarrow$ )	Acc.-Rank ( $\downarrow$ )	F1-Rank ( $\downarrow$ )
BCCO-CLS	XGBoost	0.834	0.762	0.674	5.811	5.660	5.509
	CatBoost	0.829	0.757	0.664	6.264	6.000	5.802
	TabDPT-NE	0.841	0.774	0.685	5.085	4.679	4.783
	TabDPT	0.846	0.777	0.687	4.453	4.358	4.604
	TabPFN-v2	0.843	0.772	0.679	5.170	5.208	5.274
	TabPFN-v2-FT	0.842	0.773	0.678	5.198	4.925	5.330
	LimiX	0.871	0.804	0.731	2.538	2.575	2.651
	LimiX-FT	<b>0.873</b>	<b>0.806</b>	<b>0.733</b>	<b>1.396</b>	<b>1.434</b>	<b>1.481</b>
TALENT-CLS	XGBoost	0.881	0.837	0.713	5.603	5.564	5.402
	CatBoost	0.876	0.828	0.704	6.318	6.229	6.000
	TabDPT-NE	0.891	0.846	0.719	4.955	4.754	4.933
	TabDPT	0.893	0.849	0.723	4.179	4.207	4.413
	TabPFN-v2	0.895	0.850	0.727	4.436	4.464	4.570
	TabPFN-v2-FT	0.889	0.842	0.718	5.212	5.156	5.106
	LimiX	0.903	0.861	0.752	3.061	2.860	2.933
	LimiX-FT	<b>0.904</b>	<b>0.863</b>	<b>0.755</b>	<b>1.726</b>	<b>1.598</b>	<b>1.777</b>
OpenML-CC18	XGBoost	0.929	0.879	0.775	5.177	5.194	5.500
	CatBoost	0.926	0.870	0.770	6.548	6.032	6.274
	TabDPT-NE	0.927	0.881	0.786	4.677	4.758	4.984
	TabDPT	0.930	0.885	0.799	3.919	3.919	4.161
	TabPFN-v2	0.929	0.886	0.790	5.065	4.468	4.613
	TabPFN-v2-FT	0.933	0.889	<b>0.883</b>	4.161	3.823	3.032
	LimiX	0.939	0.893	0.811	2.952	3.677	3.613
	LimiX-FT	<b>0.941</b>	<b>0.894</b>	0.813	<b>1.500</b>	<b>2.306</b>	<b>2.435</b>
TabArena	XGBoost	0.838	0.867	0.567	5.424	6.030	5.788
	CatBoost	0.835	0.867	0.574	6.242	5.545	5.121
	TabDPT-NE	0.891	0.846	0.719	4.950	4.749	4.927
	TabDPT	0.839	0.870	0.580	4.970	4.636	4.667
	TabPFN-v2	0.838	0.872	0.589	4.939	4.303	4.818
	TabPFN-v2-FT	0.850	0.874	<b>0.713</b>	3.697	4.273	<b>2.682</b>
	LimiX	0.849	0.877	0.597	3.242	3.030	4.212
	LimiX-FT	<b>0.851</b>	<b>0.878</b>	0.600	<b>2.030</b>	<b>1.697</b>	2.939
TabZilla	XGBoost	0.929	0.863	0.789	5.185	5.148	5.481
	CatBoost	0.922	0.848	0.780	6.556	6.037	6.074
	TabDPT-NE	0.932	0.880	0.824	4.333	3.963	4.037
	TabDPT	0.934	0.882	0.824	3.815	3.815	4.148
	TabPFN-v2	0.929	0.863	0.797	5.111	4.815	5.185
	TabPFN-v2-FT	0.932	0.876	<b>0.847</b>	4.185	4.556	3.318
	LimiX	0.943	0.885	0.836	3.556	3.444	3.889
	LimiX-FT	<b>0.944</b>	<b>0.887</b>	0.838	<b>2.074</b>	<b>2.407</b>	<b>3.136</b>
PFN-CLS	XGBoost	0.898	0.831	0.733	6.000	5.138	5.103
	CatBoost	0.895	0.819	0.720	6.310	5.828	6.138
	TabDPT-NE	0.891	0.822	0.727	5.034	5.483	5.828
	TabDPT	0.896	0.833	0.742	4.655	4.103	4.448
	TabPFN-v2	0.910	0.845	0.756	4.138	4.655	4.552
	TabPFN-v2-FT	0.900	0.837	0.759	5.379	5.310	4.828
	LimiX	0.923	0.862	0.786	2.241	2.690	2.759
	LimiX-FT	<b>0.924</b>	<b>0.864</b>	<b>0.788</b>	<b>1.276</b>	<b>1.517</b>	<b>1.724</b>

Table 24: Performance comparison of representative methods on benchmarks of regression. "FT" indicates the model performance after fine-tuning. TabDPT-NE represents the performance of TabDPT without ensemble while its default version has an ensemble size of 8. The best scores are shown in bold.

Benchmark	Method	R <sup>2</sup> ( $\uparrow$ )	RMSE ( $\downarrow$ )	R <sup>2</sup> -Rank ( $\downarrow$ )	RMSE-Rank ( $\downarrow$ )
BCCO-REG	XGBoost	0.764	0.415	5.820	5.740
	CatBoost	0.741	0.427	6.600	6.460
	TabDPT-NE	0.769	0.410	5.220	5.060
	TabDPT	0.772	0.406	4.460	4.260
	TabPFN-v2	0.772	0.404	4.600	4.280
	TabPFN-v2-FT	0.777	0.399	4.020	3.720
	LimiX	0.794	0.386	3.320	<b>2.860</b>
	LimiX-FT	<b>0.796</b>	<b>0.385</b>	<b>1.960</b>	2.880
TALENT-REG	XGBoost	0.710	0.462	5.545	5.404
	CatBoost	0.700	0.471	6.485	6.434
	TabDPT-NE	0.709	0.461	5.061	4.929
	TabDPT	0.711	0.458	4.263	4.081
	TabPFN-v2	0.695	0.465	4.980	4.747
	TabPFN-v2-FT	0.702	0.459	4.879	4.657
	LimiX	0.735	0.433	2.970	2.566
	LimiX-FT	<b>0.737</b>	<b>0.417</b>	<b>1.818</b>	<b>2.177</b>
CTR23	XGBoost	0.712	0.511	5.636	5.606
	CatBoost	0.700	0.528	6.273	6.212
	TabDPT-NE	0.728	0.500	4.909	4.818
	TabDPT	0.731	0.498	4.273	4.121
	TabPFN-v2	0.716	0.503	4.545	4.394
	TabPFN-v2-FT	0.722	0.498	4.636	4.545
	LimiX	0.745	0.477	3.576	2.818
	LimiX-FT	<b>0.748</b>	<b>0.473</b>	<b>2.152</b>	<b>2.437</b>
PFN-REG	XGBoost	0.671	0.487	4.929	4.714
	CatBoost	0.661	0.498	6.214	6.071
	TabDPT-NE	0.672	0.491	5.393	5.071
	TabDPT	0.675	0.484	4.607	4.500
	TabPFN-v2	0.676	0.475	4.393	4.143
	TabPFN-v2-FT	0.687	0.466	4.179	3.857
	LimiX	0.692	<b>0.461</b>	3.857	<b>3.571</b>
	LimiX-FT	<b>0.695</b>	0.466	<b>2.429</b>	4.071

## 7.7 Data Generation

Data generation is a challenging and meaningful task across multiple modalities since it needs to capture the joint distribution of  $P(X, Y)$  compared with prediction tasks that focus on modeling  $P(Y|X)$ . While there is abundant literature on the generation of images (Croitoru et al., 2023), videos (Xing et al., 2024), and text (Li et al., 2024b), relatively less attention has been paid to tabular data generation. However, data generation is even more critical for tabular data due to its scarcity and possible privacy issues (Hernandez et al., 2022). As a foundation model, LimiX has the ability of tabular data generation given an unseen real dataset. A significant advantage is that it does not require additional training of generative models. Following TabPFN-v2 (Hollmann et al., 2025), firstly we conduct data generation in an iterative style. For the first column, we sample from the empirical categorical distribution if it is a categorical feature, or we use the original first column with added random noise if it is a continuous feature. For other columns, we generate  $j^{th}$  column based on the  $j - 1$  columns of real data (treated as  $x^{ct}$ ) and generated data (treated as  $x^{te}$ ). We iterate this process until the last column is generated. Then for LimiX, we randomly mask a proportion of cell values of generated data and conduct missing value imputation multiple times so that we can leverage LimiX’s capability of modeling the joint distribution. We conduct experiments on the following datasets: Early Stage Diabetes 2020 (Islam et al., 2019; 2020), Vertebral Column (Barreto & Neto, 2005), Seeds (Gomes Mantovani, 2015), Wine (Hardik, 2021), and Grub Damage (Vanschoren, 2014b). For evaluation, we use Trend and Shape to evaluate fidelity, which are proposed by SDMetrics<sup>3</sup>. We also calculate the AUC of XGBoost on a hold-out test dataset using generated data or real data. From Table 25, we find that LimiX outperforms TabPFN-v2 in terms of all metrics on the five classification tasks. On the dataset of Grub Damage, the prediction performance using data generated by LimiX is even higher than using real data. The results show that LimiX is capable of capturing dependencies between features of  $X$ , which is brought by the pretraining strategy of mask prediction. In contrast, TabPFN-v2 is only capable of modeling  $P(Y|X)$ .

Table 25: Evaluation of data generation on five classification tasks. “Trend” and “Shape” are two fidelity metrics. “AUC” measures the classification performance of XGBoost using generated (or real) data. LimiX consistently outperforms TabPFN-v2 in all three metrics.

Metric	Method	Early Diabetes	Vertebra	Seeds	Wine	Grub Damage
Trend ( $\uparrow$ )	TabPFN-v2	0.797	0.580	0.696	0.686	0.486
	LimiX	0.804	0.591	0.699	0.688	0.673
Shape ( $\uparrow$ )	TabPFN-v2	0.889	0.754	0.739	0.622	0.635
	LimiX	0.902	0.763	0.768	0.646	0.762
AUC ( $\uparrow$ )	TabPFN-v2	0.839	0.652	0.861	0.670	0.695
	LimiX	0.879	0.783	0.932	0.912	0.727
	Real	0.969	0.915	0.982	0.996	0.710

<sup>3</sup><https://docs.sdv.dev/sdmetrics>

## 7.8 Out-of-Distribution Generalization

In real-world applications, tabular data is often subject to distribution shifts between training data and the test data encountered during deployment. For instance, a machine learning model may be trained on patient records collected from one hospital but deployed on data from a different hospital. Such distribution shifts arising from domain variation can lead to substantial degradation in model performance. This challenge is commonly referred to as the Out-of-Distribution (OOD) generalization problem, and has been the focus of extensive research in the machine learning community (Liu et al., 2021; Yu et al., 2024).

We evaluate the OOD generalization performance of various tabular models on 10 public datasets drawn from the TableShift (Gardner et al., 2023), which is a benchmark for distribution shifts in tabular data. It contains 15 binary classification tasks in total, covering a wide range of data sources, including finance, medical diagnosis, policy, etc. More details can be found in Appendix A.1. To ensure fair comparisons, for each dataset, if the number of training or test samples exceeds 10,000, we randomly subsample 10,000 instances. Each experiment is repeated five times and we report the average.

From Table 26, we can see that LimiX achieves state-of-the-art performance, securing top ranks in both ID (In-Distribution) and OOD evaluations. This could be attributed to LimiX’s integration of causal data with a Context-Conditional Masked Modeling framework. This strategy enables the model to capture robust causal relationships rather than superficial correlations, leading to significantly enhanced generalization on OOD data. As for baselines, models that are also based on in-context learning (ICL), notably TabICL and TabPFN-v2, also deliver competitive results. This indicates that the ICL mechanism could better capture latent invariant patterns in data, thereby endowing models with strong generalization potential.

Table 26: Average AUC and ranks on TableShift. We can see that LimiX consistently outperforms all baselines in terms of ID or OOD performance.

Model	ID_AUC ( $\uparrow$ )	OOD_AUC ( $\uparrow$ )	ID_Rank ( $\downarrow$ )	OOD_Rank ( $\downarrow$ )
LimiX	<b>0.848</b>	<b>0.806</b>	<b>2.5</b>	<b>1.3</b>
TabICL	0.847	0.799	4.1	3.9
AutoGluon	0.842	0.797	3.5	4.0
TabPFN-v2	0.841	0.797	6.4	5.2
CatBoost	0.840	0.793	4.1	5.9
LightGBM	0.836	0.790	6.1	7.3
MLP	0.839	0.792	7.5	7.6
FT-Transformer	0.840	0.789	7.8	8.3
ResNet	0.837	0.789	8.6	8.3
NODE	0.836	0.789	9.2	9.2
XGBoost	0.830	0.783	9.9	9.5
TabDPT	0.822	0.763	12.6	12.7
TabR	0.820	0.767	13.2	13.0
TANGOS	0.801	0.752	13.8	13.6
ModernNCA	0.801	0.757	15.4	14.6

## 8 Conclusion

In this report, we introduce LimiX, an LDM that treats structured-data inputs as a joint distribution over variables and missingness, so that classification, regression, missing value imputation, data generation, and sample selection for interpretability, can all be expressed as conditional queries to a single model. Methodologically, LimiX adopts a lightweight, scalable architecture that models causal relations among variables while jointly capturing dependencies across features and samples. Meanwhile, LimiX combines masked joint-distribution pretraining with an episodic, context-conditional objective of per-dataset adaptation for the versatility in downstream tasks. The pretraining data for LimiX is generated with hierarchical SCMs via graph-aware and solvability-aware sampling. Attention-guided retrieval of LimiX further supports efficient inference-time ensemble and fine-tuning if desired. Empirically, experiments across 10 large structured-data benchmarks, spanning wide ranges of sample sizes, feature dimensions, number of classes, categorical-to-numerical feature ratios, missingness, and sample-to-feature ratios, confirm the effectiveness of LimiX. As a single model, LimiX consistently surpasses competitive baselines on various downstream tasks.

---

## 9 Contribution

### Project Design and Lead

Xingxuan Zhang, Peng Cui

### Core Contributors

Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu

### Contributors

Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang

---

## References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.
- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.
- Roohallah Alizadehsani, Mohamad Roshanzamir, and Zahra Sani. Z-alizadeh sani. UCI Machine Learning Repository, 2013. DOI: <https://doi.org/10.24432/C5Q31T>.
- Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.
- Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, Khoa Doan, and Sathiya S Keerthi. Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*, 2020.
- Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *Transactions on Machine Learning Research*, 2024.
- Guilherme Barreto and Ajalmar Neto. Vertebral column. UCI Machine Learning Repository, 2005. DOI: <https://doi.org/10.24432/C5K89B>.
- Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2017.
- Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- Pietro Caputo and Daniel Parisi. Block factorization of the relative entropy via spatial mixing. *Communications in Mathematical Physics*, 388(2):793–818, 2021.
- Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. Ieee, 2017.
- Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3930–3938, 2022.
- Jintai Chen, KuanLun Liao, Yanwen Fang, Danny Chen, and Jian Wu. Tabcaps: A capsule neural network for tabular data classification with bow routing. In *The Eleventh International Conference on Learning Representations*, 2023a.
- Jintai Chen, Jiahuan Yan, Qiyuan Chen, Danny Ziyi Chen, Jian Wu, and Jimeng Sun. Excelformer: A neural network surpassing gbdts on tabular data. *arXiv preprint arXiv:2301.02819*, 2023b.
- Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Darby Tien-Hao Chang. Trompt: towards a better deep neural network for tabular data. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 4392–4434, 2023c.

- 
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 785–794, 2016.
- Florinel-Alin Croitoru, Vlad Hondu, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(9):10850–10869, 2023.
- Bruno Rodrigues de Oliveira, Alan Mario Zuffo, Francisco Charles dos Santos Silva, Ricardo Mezzomo, Leandra Matos Barrozo, Tatiane Scilewski da Costa Zanatta, Joel Cabral dos Santos, Carlos Henrique Conceição Sousa, and Yago Pinto Coelho. Dataset: Forty soybean cultivars from subsequent harvests. *Trends in Agricultural and Environmental Sciences*, pp. e230005–e230005, 2023.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.
- Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.
- Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.
- Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.
- Xi Fang, Weijie Xu, Fiona Anting Tan, Ziqing Hu, Jian Zhang, Yanjun Qi, Srinivasan H Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding-a survey. *Transactions on Machine Learning Research*, 2024.
- Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. Openml-ctr23—a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.
- Andreas Fuster, Paul Goldsmith-Pinkham, Tarun Ramadorai, and Ansgar Walther. Predictably unequal? the effects of machine learning on credit markets. *The Journal of Finance*, 77(1):5–47, 2022.
- Josh Gardner, Zoran Popovic, and Ludwig Schmidt. Benchmarking distribution shift in tabular data with tableshift. *Advances in Neural Information Processing Systems*, 36:53385–53432, 2023.
- Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.
- Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.
- Rafael Gomes Mantovani. Seeds. OpenML, 2015. URL <https://api.openml.org/d/1499>.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.
- Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.
- Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.
- Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors. In *The Twelfth International Conference on Learning Representations*, 2024.
- Hardik. Wine. Kaggle, 2021. URL <https://www.kaggle.com/datasets/hrdkcodes/wine-data>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

- 
- Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- Miguel A Hernán and James M Robins. Causal inference, 2010.
- Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022.
- Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2022.
- Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.
- David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658, 2024.
- Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.
- MM Faniqul Islam, Rahatara Ferdousi, Sadikur Rahman, and Humayra Yasmin Bushra. Likelihood prediction of diabetes at early stage using data mining techniques. In *Computer Vision and Machine Intelligence in Medical Image Analysis: International Symposium, ISCM 2019*, pp. 113–125. Springer, 2019.
- MM Faniqul Islam, Rahatara Ferdousi, Sadikur Rahman, and Humayra Yasmin Bushra. Early stage diabetes risk prediction. UCI Machine Learning Repository, 2020. DOI: <https://doi.org/10.24432/C5VG8H>.
- Daniel Jarrett, Bogdan C Cebere, Tennison Liu, Alicia Curth, and Mihaela van der Schaar. Hyperimpute: Generalized iterative imputation with automatic model selection. In *International Conference on Machine Learning*, pp. 9916–9937. PMLR, 2022.
- Alan Jeffares, Tennison Liu, Jonathan Crabbé, Fergus Imrie, and Mihaela van der Schaar. Tangos: Regularizing tabular neural networks through gradient orthogonalization and specialization. In *The Eleventh International Conference on Learning Representations*, 2023.
- Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.
- Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *Conference on Robot Learning*, pp. 1949–1974. PMLR, 2025.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.
- Peter M Krafft, Meg Young, Michael Katell, Karen Huang, and Ghislain Buggingo. Defining ai in policy versus practice. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 72–78, 2020.
- Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.
- Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, et al. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024a.
- Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Pre-trained language models for text generation: A survey. *ACM Computing Surveys*, 56(9):1–39, 2024b.

---

Yuchen Li, Alexandre Kirchmeyer, Aashay Mehta, Yilong Qin, Boris Dadachev, Kishore Papineni, Sanjiv Kumar, and Andrej Risteski. Promises and pitfalls of generative masked language modeling: Theoretical framework and practical guidelines. In *International Conference on Machine Learning*, pp. 27969–28017. PMLR, 2024c.

Wei-Chao Lin and Chih-Fong Tsai. Missing value imputation: a review and analysis of the literature (2006–2017). *Artificial Intelligence Review*, 53(2):1487–1509, 2020.

Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2019.

Jiashuo Liu, Zheyuan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and Han-Jia Ye. Talent: A tabular analytics and learning toolbox. *arXiv preprint arXiv:2407.04057*, 2024.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L Caterini. Tabdpt: Scaling tabular foundation models. *arXiv preprint arXiv:2410.18164*, 2024.

Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International conference on machine learning*, pp. 4413–4423. PMLR, 2019.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369, 2023.

Edward Metz. Assistments: From research to practice at scale in education. <https://ies.ed.gov/logs/research/post/assistments-from-research-to-practice-at-scale-in-education>, 2020. Accessed: 2023-06-01.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pp. 405–421. Springer, 2020.

Pete Mowforth and Barry Shepherd. Statlog (vehicle silhouettes). UCI Machine Learning Repository, 1987. DOI: <https://doi.org/10.24432/C5HG6N>.

Youssef Nader, Leon Sixt, and Tim Landgraf. Dnnr: Differential nearest neighbors regression. In *International Conference on Machine Learning*, pp. 16296–16317. PMLR, 2022.

Maxime Oquab, Timothée Darcret, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.

Yilong Qin and Andrej Risteski. Fit like you sample: sample-efficient generalized score matching from fast mixing diffusions. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 4413–4457. PMLR, 2024.

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

Raghu Ramakrishnan, Johannes Gehrke, and Johannes Gehrke. *Database management systems*, volume 3. McGraw-Hill New York, 2003.

Matthew A Reyna, Christopher S Josef, Russell Jeter, Supreeth P Shashikumar, M Brandon Westover, Shamim Nemati, Gari D Clifford, and Ashish Sharma. Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. *Critical care medicine*, 48(2):210–217, 2020.

Bruno Rodrigues de Oliveira and Alan Mario Zuffo. Forty soybean cultivars from subsequent harvests. UCI Machine Learning Repository, 2023. DOI: <https://doi.org/10.46420/TAES.e230005>.

---

Miriam Seoane Santos, Pedro Henriques Abreu, Pedro J García-Laencina, Adélia Simão, and Armando Carvalho. Hcc survival. UCI Machine Learning Repository, 2015a. DOI: <https://doi.org/10.24432/C5TS4S>.

Miriam Seoane Santos, Pedro Henriques Abreu, Pedro J García-Laencina, Adélia Simão, and Armando Carvalho. A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. *Journal of biomedical informatics*, 58:49–59, 2015b.

J Paul Siebert. Vehicle recognition using rule based methods. 1987.

Abraham Silberschatz, Henry F Korth, and Shashank Sudarshan. Database system concepts. 2011.

Jeffrey S Simonoff. *Analyzing categorical data*, volume 496. Springer, 2003.

Gowthami Somepalli, Avi Schwarzschild, Micah Goldblum, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1161–1170, 2019.

Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

Michael Stonebraker and Uğur Çetintemel. "one size fits all" an idea whose time has come and gone. In *Making databases work: the pragmatic wisdom of Michael Stonebraker*, pp. 441–462. 2018.

Beata Strack, Jonathan P DeShazo, Chris Gennings, Juan L Olmo, Sebastian Ventura, Krzysztof J Cios, and John N Clore. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international*, 2014(1):781670, 2014.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 645–654, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maks Volkovs, and Anthony L Caterini. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Boris Van Breugel and Mihaela Van Der Schaar. Position: Why tabular foundation models should be a research priority. In *International Conference on Machine Learning*, pp. 48976–48993. PMLR, 2024.

Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011.

Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

Jan van Rijn. Eucalyptus. OpenML, 2014. URL <https://api.openml.org/d/188>.

Joaquin Vanschoren. Analcatdata broadwaymult. OpenML, 2014a. URL <https://api.openml.org/d/961>.

Joaquin Vanschoren. Grub damage. OpenML, 2014b. URL <https://api.openml.org/d/338>.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

- 
- Jing Wu, Suiyao Chen, Qi Zhao, Renat Sergazinov, Chen Li, Shengjie Liu, Chongchao Zhao, Tianpei Xie, Hanqing Guo, Cheng Ji, et al. Switchtab: Switched autoencoders are effective tabular learners. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 15924–15933, 2024.
- Tiange Xiang, Kai Li, Chengjiang Long, Christian Häne, Peihong Guo, Scott Delp, Ehsan Adeli, and Li Fei-Fei. Repurposing 2d diffusion models with gaussian atlas for 3d generation. *arXiv preprint arXiv:2503.15877*, 2025.
- Zhen Xing, Qijun Feng, Haoran Chen, Qi Dai, Han Hu, Hang Xu, Zuxuan Wu, and Yu-Gang Jiang. A survey on video diffusion models. *ACM Computing Surveys*, 57(2):1–42, 2024.
- Derek Qiang Xu, F Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular pfns. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z Chen, and Jian Wu. T2g-former: organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10720–10728, 2023.
- Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6796–6807, 2024.
- Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pp. 5689–5698. PMLR, 2018.
- Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela Van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in neural information processing systems*, 33:11033–11043, 2020.
- Han Yu, Jiashuo Liu, Xingxuan Zhang, Jiayun Wu, and Peng Cui. A survey on evaluation of out-of-distribution generalization. *arXiv preprint arXiv:2403.01874*, 2024.
- Wantao Yu, Chee Yew Wong, Roberto Chavez, and Mark A Jacobs. Integrating big data analytics into supply chain finance: The roles of information processing and data-driven culture. *International journal of production economics*, 236:108135, 2021.
- Hengrui Zhang, Liancheng Fang, Qitian Wu, and Philip S Yu. Diffputer: Empowering diffusion models for missing data imputation. In *The Thirteenth International Conference on Learning Representations*, 2025.
- Xiyuan Zhang and Maddix Robinson Danielle. Mitra: Mixed synthetic priors for enhancing tabular foundation models, 2025. URL <https://www.amazon.science/blog/mitra-mixed-synthetic-priors-for-enhancing-tabular-foundation-models>.

---

## A Experimental Details

### A.1 Details of Datasets with Distribution Shifts

The 10 public datasets we adopt from TableShift include Voting<sup>4</sup>, Unemployment<sup>5</sup>, Sepsis (Reyna et al., 2020), Public Health Insurance<sup>5</sup>, Income<sup>5</sup>, Hospital Readmission (Strack et al., 2014)<sup>6</sup>, Food Stamps<sup>5</sup>, Diabetes<sup>7</sup>, College Scorecard<sup>8</sup>, and ASSISTments (Metz, 2020)<sup>9</sup>. Detailed information of the benchmark TableShift we adopt in Section 7.8 can be found in Table 27.

Table 27: Statistics of datasets in TableShift. "#Features" indicates the number of features. "#Train" indicates the number of training samples. "#ID\_test" indicates the number of ID (In-Distribution) test samples. "#OOD\_test" indicates the number of OOD (Out-of-Distribution) test samples. "Shift Type" indicates how the datasets are split into multiple domains to create distribution shifts. "#Domains" indicates the numbers of training and test domains.

Dataset	#Features	#Train	#ID_test	#OOD_test	Shift type	#Domains
Voting	365	34,796	4,350	21,231	Geographic Region	4/1
ASSISTments	26	2,132,526	266,566	1,906	School	386/10
Diabetes	142	969,229	121,154	209,375	Race	1/5
Food Stamps	239	629,018	78,628	48,878	Geographic Region	9/1
Hospital Readmission	183	34,288	4,287	50,968	Admission Source	15/1
Income	232	1,264,123	158,016	75,911	Geographic Region	9/1
Health Insurance	135	4,006,249	500,782	817,877	Disability Status	1/1
Sepsis	41	1,122,299	140,288	134,402	Length of Stay	47/2
Unemployment	223	1,290,914	161,365	163,611	Education Level	9/15
College ScoreCard	118	98,556	12,320	1,352	Institution Type	26/8

### A.2 Hyperparameter Search Space for Baselines

The hyperparameter search space for the baseline models is summarized in Table 28.

<sup>4</sup><https://electionstudies.org/>

<sup>5</sup><https://www.census.gov/programs-surveys/acs>

<sup>6</sup><https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008>

<sup>7</sup><https://www.cdc.gov/brfss/index.html>

<sup>8</sup><https://collegescorecard.ed.gov/>

<sup>9</sup><https://new.assistments.org>

Table 28: The hyperparameter search space for tree-based models is utilized by Optuna, which provides automatic suggestions via Bayesian optimization. The only difference between classification and regression configurations lies in the optimization target. All other hyperparameter settings remain identical.

Baseline	Hyperparameter	Data Type	Log	Search Space
RF	n_estimators	int	no	[100, 500]
	max_depth	int	no	[3, 20]
	min_samples_split	int	no	[2, 20]
	min_samples_leaf	int	no	[1, 20]
	max_features	categorical	no	{Sqrt, Log2, None}
	bootstrap	categorical	no	{True, False}
ET	n_estimators	int	no	[100, 500]
	max_depth	int	no	[3, 20]
	min_samples_split	int	no	[2, 20]
	min_samples_leaf	int	no	[1, 20]
XGBoost	n_estimators	int	no	[100, 300]
	max_depth	int	no	[3, 9]
	learning_rate	float	yes	[0.01, 0.3]
	subsample	float	no	[0.5, 1.0]
	colsample_bytree	float	no	[0.5, 1.0]
LightGBM	n_estimators	int	no	[100, 300]
	learning_rate	float	yes	[0.01, 0.3]
	max_depth	int	no	[-1, 20]
	subsample	float	no	[0.5, 1.0]
	colsample_bytree	float	no	[0.5, 1.0]
CatBoost	iterations	int	no	[100, 300]
	depth	int	no	[4, 10]
	learning_rate	float	yes	[0.01, 0.3]

## B Omitted Details in Section 6

### B.1 Omitted Details in Section 6.1

**Proposition B.1** (Formal Version of Proposition 6.1). *Suppose that the distribution  $p(\mathbf{X}^{\text{te}}|\mathbf{X}^{\text{ct}})$  has a strictly positive probability density function. Then there is a one-to-one correspondence between the distribution  $p(\mathbf{X}^{\text{te}}|\mathbf{X}^{\text{ct}})$  and the family of conditionals  $\{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \forall \pi \in \Pi_k\}$ .*

*Proof.* It is clear that  $p(\mathbf{X}^{\text{te}}|\mathbf{X}^{\text{ct}})$  directly yields all conditionals  $\{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \pi \in \Pi_k\}$ . We now show the converse:  $p(\mathbf{X}^{\text{te}}|\mathbf{X}^{\text{ct}})$  can be recovered from this family of conditionals.

As the first step, we derive  $\{p(X_j^{\text{te}}|\mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}}) : j \in [d]\}$ . For any  $j \in [d]$ , select a mask  $\pi \in \Pi_k$  with  $j \in \pi$ . Then

$$p(X_j^{\text{te}}|\mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}}) = \frac{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})}{p(\mathbf{X}_{\pi \setminus \{j\}}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})} = \frac{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})}{\int_{\Omega} p(\mathbf{X}_{\pi \setminus \{j\}}^{\text{te}}, X_j^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) dX_j^{\text{te}}}.$$

Each term on the right-hand side belongs to  $\{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \pi \in \Pi_k\}$ , hence  $p(X_j^{\text{te}}|\mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}})$  can indeed be recovered.

We now use induction to show that the knowledge of  $\{p(X_j^{\text{te}}|\mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}}) : j \in [d]\}$  suffices to recover all conditionals in

$$\mathcal{P}_k = \{p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : |\pi| = k\}.$$

Clearly,  $\mathcal{P}_1$  coincides with  $\{p(X_j^{\text{te}}|\mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}}) : j \in [d]\}$ . Suppose  $\mathcal{P}_{k'}$  is obtainable for all  $k' \leq k$ . Consider  $p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) \in \mathcal{P}_{k+1}$  with  $|\pi| = k+1$ . Pick  $j \in \pi$  and set  $\pi' = \pi \setminus \{j\}$ . Then

$$\int_{\Omega} \frac{p(X_j^{\text{te}}|\mathbf{X}_{\pi'}^{\text{te}}, \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})}{p(\mathbf{X}_{\pi'}^{\text{te}}|\mathbf{X}_j^{\text{te}}, \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})} dX_j^{\text{te}} = \int_{\Omega} \frac{p(X_j^{\text{te}}|\mathbf{X}_{\pi'}^{\text{te}}, \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})}{p(\mathbf{X}_{\pi'}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})} dX_j^{\text{te}} = \frac{1}{p(\mathbf{X}_{\pi'}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})}.$$

Each term on the left-hand side belongs to  $\mathcal{P}_1$  or  $\mathcal{P}_k$ , so  $p(\mathbf{X}_{\pi'}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  is obtainable. Finally,

$$p(\mathbf{X}_{\pi}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) = p(\mathbf{X}_{\pi'}^{\text{te}}|\mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) \cdot p(X_j^{\text{te}}|\mathbf{X}_{\pi'}^{\text{te}}, \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}),$$

showing that  $p(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  can be obtained as well. Hence, any element in  $\mathcal{P}_{k+1}$  can be obtained. By induction, the claim follows.  $\square$

Now we show that the knowledge of a single conditional distribution  $p(X_j^{\text{te}} | \mathbf{X}_{-j}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for some  $j \in [d]$  (i.e., when  $\Pi = \{j\}$ ) is insufficient to recover the full conditional distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$ .

**Example B.1.** Consider the case  $d = 2$  and  $m = 0$ , i.e., a setting with no in-context samples and a feature dimension of 2. In this case, the target distribution  $p(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  reduces to  $p(\mathbf{X}^{\text{te}})$ . Suppose further that  $\mathbf{X}^{\text{te}} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ , where  $\boldsymbol{\mu} = (\mu_1, \mu_2) \in \mathbb{R}^2$  and  $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \succeq 0$ . Then the conditional distribution of  $X_1^{\text{te}}$  given  $X_2^{\text{te}}$  is

$$X_1^{\text{te}} | X_2^{\text{te}} \sim \mathcal{N}\left(\Sigma_{12}\Sigma_{22}^{-1}X_2^{\text{te}} + \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}\mu_2, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right).$$

Consequently, knowledge of  $p(X_1^{\text{te}} | X_2^{\text{te}})$  alone provides access only to the quantities  $\Sigma_{12}\Sigma_{22}^{-1}$ ,  $\mu_1 - \Sigma_{12}\Sigma_{22}^{-1}\mu_2$ , and  $\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ . These are insufficient to uniquely determine the full parameter set  $(\boldsymbol{\mu}, \Sigma)$ , and therefore  $p(\mathbf{X}^{\text{te}})$  cannot be fully recovered from  $p(X_1^{\text{te}} | X_2^{\text{te}})$  alone. By symmetry,  $p(\mathbf{X}^{\text{te}})$  also cannot be recovered solely from  $p(X_2^{\text{te}} | X_1^{\text{te}})$ .

## B.2 Omitted Details in Section 6.2

We denote by  $q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}}), \theta \in \Theta$  the distribution induced by the learned family of conditional probabilities  $\{q_\theta(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) : \pi \in \Pi_k\}$ . We overload the notation by writing  $q_\theta(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) := q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})p(\mathbf{X}^{\text{ct}})$ .

### B.2.1 Sample Efficiency

**Theorem B.2** (Formal Version of Theorem 6.2). *Suppose there exists  $\theta^* \in \Theta$  such that  $q_{\theta^*}(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) = p(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})$  for all  $\mathbf{X}^{\text{ct}} \in \Omega^{m \times d}$ ,  $\mathbf{X}^{\text{te}} \in \Omega^d$ , and  $\pi \subseteq [d]$ , and that the minimizer of  $L_k(\theta)$  is unique for every  $k$ . Assume that for all  $\theta \in \Theta$ ,  $\mathbf{X}^{\text{ct}} \in \Omega^{m \times d}$ ,  $\mathbf{X}^{\text{te}} \in \Omega^d$ , and  $\pi \subseteq [d]$ , the gradient norm  $\|\nabla_\theta q_\theta(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})\|_2$  and the Hessian norm  $\|\nabla_\theta^2 q_\theta(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})\|_F$  exist and are finite. Assume  $\nabla_\theta^2 L_k(\theta^*) \succ 0$  for all  $k \in [d]$ . Then, for every sufficiently small neighborhood  $\mathcal{B}$  of  $\theta^*$ , there exists a sufficiently large  $n$  such that  $\hat{\theta}_{k,n}$  is the unique minimizer of  $\hat{L}_k(\theta)$  in  $\mathcal{B}$ . Moreover,*

$$\sqrt{n}(\hat{\theta}_{k,n} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \Gamma_k),$$

where  $\Gamma_k$  does not depend on  $n$  and satisfies  $\Gamma_{k+1} \preceq \Gamma_k$ .

The proof of Theorem B.2 is based on the following lemma.

**Lemma B.3** (Van der Vaart (2000), Theorem 5.23; statement adapted from Qin & Risteski (2024); Li et al. (2024c)). *Consider a loss  $L : \Theta \rightarrow \mathbb{R}$ , such that  $L(\theta) = \mathbb{E}_{\mathbf{X} \sim p}[\ell_\theta(\mathbf{X})]$  for  $\ell_\theta : \mathcal{X} \rightarrow \mathbb{R}$ . Let  $\Theta^*$  be the set of global minima of  $L$ , i.e.,*

$$\Theta^* = \{\theta^* : L(\theta^*) = \min_{\theta \in \Theta} L(\theta)\}.$$

Suppose the following conditions are met:

- (Gradient bounds on  $\ell_\theta$ ) The map  $\theta \mapsto \ell_\theta$  is measurable and differentiable at every  $\theta^* \in \Theta^*$  for  $p$ -almost every  $\mathbf{X}$ . Furthermore, there exists a function  $B(\mathbf{X})$ , s.t.  $\mathbb{E}[B(\mathbf{X})^2] < \infty$  and for every  $\theta_1, \theta_2$  near  $\theta^*$ , we have

$$|\ell_{\theta_1}(\mathbf{X}) - \ell_{\theta_2}(\mathbf{X})| < B(\mathbf{X}) \|\theta_1 - \theta_2\|_2$$

- (Twice-differentiability of  $L$ )  $L(\theta)$  is twice-differentiable at every  $\theta^* \in \Theta^*$  with Hessian  $\nabla_\theta^2 L(\theta^*)$ , and furthermore  $\nabla_\theta^2 L(\theta^*) \succ 0$ .
- (Uniform law of large numbers) The loss  $L$  satisfies a uniform law of large numbers, that is

$$\sup_{\theta \in \Theta} |\hat{\mathbb{E}}[\ell_\theta(\mathbf{X})] - L(\theta)| \xrightarrow{p} 0.$$

- (Realizability) The data distribution  $p$  satisfies:  $\exists \theta^* \in \Theta$  such that  $p_{\theta^*} = p$ .

Then for every  $\theta^* \in \Theta^*$ , and every sufficiently small neighborhood  $S$  of  $\theta^*$ , there exists a sufficiently large  $n$ , such that there is a unique minimizer  $\hat{\theta}_n$  of  $\hat{\mathbb{E}}[\ell_\theta(\mathbf{X})]$  in  $S$ . Furthermore,  $\hat{\theta}_n$  satisfies:

$$\sqrt{n}(\hat{\theta}_n - \theta^*) \xrightarrow{d} \mathcal{N}\left(0, \left(\nabla_\theta^2 L(\theta^*)\right)^{-1} \text{Cov}(\nabla_\theta \ell_{\theta^*}(\mathbf{X})) \left(\nabla_\theta^2 L(\theta^*)\right)^{-1}\right).$$

Similar to Lemma 2 in Li et al. (2024c), we have the following lemma.

**Lemma B.4.** *Under the same assumptions as in Theorem B.2, we have*

$$\nabla_\theta^2 L_k(\theta^*) = \text{Cov}_{(\mathbf{X}^{ct}, \mathbf{X}^{te}) \sim p, \pi \sim \text{Unif}(\Pi_k)} (-\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})). \quad (5)$$

*Proof.* It is easy to verify that  $\theta^*$  is the minimizer of  $L_k(\theta)$  for any  $k \in [d]$  and hence  $\theta_k^* = \theta^*$  by assumption.

Rewrite  $\nabla_\theta^2 L_k(\theta^*)$  and we get that

$$\begin{aligned} & \nabla_\theta^2 L_k(\theta^*) \\ &= \nabla_\theta^2 \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} [-\log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} [-\nabla_\theta^2 \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} \left[ (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}))^\top - \frac{\nabla_\theta^2 q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \right]. \end{aligned} \quad (6)$$

In addition, we have

$$\begin{aligned} & \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} \left[ \frac{\nabla_\theta^2 q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \right] = \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \mathbb{E}_{\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}} \left[ \frac{\nabla_\theta^2 q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \int_{\Omega^k} \frac{\nabla_\theta^2 q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \cdot p(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \int_{\Omega^k} \nabla_\theta^2 q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] \quad (\text{Due to the assumption } q_{\theta^*} = p) \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \nabla_\theta^2 \int_{\Omega^k} q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] = \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} [\nabla_\theta^2 1] = 0. \end{aligned}$$

Combined with Equation (6), we can get that

$$\nabla_\theta^2 L_k(\theta^*) = \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} \left[ (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}))^\top \right]. \quad (7)$$

Now rewrite the right-hand side of Equation (5) and we get

$$\begin{aligned} & \text{Cov}(-\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} \left[ (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}))^\top \right] \\ &\quad - \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} [\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})] \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} [\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})]^\top. \end{aligned} \quad (8)$$

Note that

$$\begin{aligned} & \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} [\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})] = \mathbb{E}_{\mathbf{X}^{ct}, \mathbf{X}^{te}, \pi} \left[ \frac{\nabla_\theta q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \mathbb{E}_{\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}} \left[ \frac{\nabla_\theta q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \int_{\Omega^k} \frac{\nabla_\theta q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})}{q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})} \cdot p(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \int_{\Omega^k} \nabla_\theta q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] \quad (\text{Due to the assumption } q_{\theta^*} = p) \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} \left[ \nabla_\theta \int_{\Omega^k} q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}) d\mathbf{X}_\pi^{te} \right] \\ &= \mathbb{E}_{\mathbf{X}^{ct}, \pi} \mathbb{E}_{\mathbf{X}_{-\pi}^{te} | \mathbf{X}^{ct}} [\nabla_\theta 1] = 0. \end{aligned}$$

Combined with Equation (8), we can get that

$$\text{Cov}(-\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) = \mathbb{E} \left[ (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct})) (\nabla_\theta \log q_{\theta^*}(\mathbf{X}_\pi^{te} | \mathbf{X}_{-\pi}^{te}, \mathbf{X}^{ct}))^\top \right]. \quad (9)$$

Now the claim follows from Equations (7) and (9).  $\square$

Based on the above lemmas, we can now prove [Theorem B.2](#). The proof follows a similar idea to that of Theorem 1 in [Li et al. \(2024c\)](#).

*Proof of Theorem B.2.* It is easy to verify that  $\theta^*$  is the minimizer of  $L_k(\theta)$  for any  $k \in [d]$  and hence  $\theta_k^* = \theta^*$  by assumption. According to [Lemma B.3](#), it holds that for every sufficiently small neighborhood  $S$  of  $\theta^*$ , there exists a sufficiently large  $n$ , such that there is a unique minimizer  $\hat{\theta}_{k,n}$  in  $S$ . Furthermore,  $\hat{\theta}_{k,n}$  satisfies:

$$\sqrt{n} (\hat{\theta}_{k,n} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \Gamma_k).$$

Here due to [Lemma B.4](#), it holds that

$$\begin{aligned} \Gamma_k &= \left( \nabla_{\theta}^2 L_k(\theta^*) \right)^{-1} \text{Cov}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k)} (-\nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})) \left( \nabla_{\theta}^2 L_k(\theta^*) \right)^{-1} \\ &= \left( \nabla_{\theta}^2 L_k(\theta^*) \right)^{-1}. \end{aligned} \quad (10)$$

Fix a  $k \in [d-1]$ . Then for every  $\pi \in \Pi_{k+1}$  and  $j \in \pi$ , let  $\gamma = \pi \setminus \{j\}$  and we have

$$\begin{aligned} \log q_{\theta}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}) &= \log q_{\theta}(\mathbf{X}_{\gamma}^{\text{te}}, X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \\ &= \log q_{\theta}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) + \log q_{\theta}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}). \end{aligned} \quad (11)$$

As a result, we have

$$\begin{aligned} &\nabla_{\theta}^2 L_{k+1}(\theta^*) \\ &= \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_{k+1})} \left[ (\nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})) (\nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}))^{\top} \right] \\ &\quad \text{(Due to Equation (7))} \\ &= \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \gamma \sim \text{Unif}(\Pi_k), j \in \text{Unif}([d] \setminus \gamma)} \left[ (\nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}})) (\nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\pi}^{\text{te}} | \mathbf{X}_{-\pi}^{\text{te}}, \mathbf{X}^{\text{ct}}))^{\top} \right] \\ &\quad \text{(Letting } \pi = \gamma \cup \{j\}\text{)} \\ &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}, \gamma, j} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) + \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \times \right. \\ &\quad \left. \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) + \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right]. \quad \text{(By Equation (11))} \end{aligned}$$

Define

$$\begin{aligned} A &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}, \gamma, j} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right]. \\ B &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}, \gamma, j} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \left( \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right]. \\ C &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}, \gamma, j} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \left( \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right]. \end{aligned}$$

Then we have

$$\nabla_{\theta}^2 L_{k+1}(\theta^*) = A + B + B^{\top} + C.$$

It is easy to verify that  $C = \nabla_{\theta}^2 L_k(\theta^*)$ . Now consider  $B$ . It holds that

$$\begin{aligned} B &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}, \gamma, j} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \left( \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right] \\ &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \gamma, j, \mathbf{X}_{-\gamma}^{\text{te}}} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \cdot \mathbb{E}_{\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(\mathbf{X}_{\gamma}^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right)^{\top} \right] \right] \\ &= \mathbb{E}_{\mathbf{X}^{\text{ct}}, \gamma, j, \mathbf{X}_{-\gamma}^{\text{te}}} \left[ \left( \nabla_{\theta} \log q_{\theta^*}(X_j^{\text{te}} | \mathbf{X}_{-(\gamma \cup \{j\})}^{\text{te}}, \mathbf{X}^{\text{ct}}) \right) \cdot 0 \right] \quad \text{(See Equation (12) below)} \\ &= 0. \end{aligned}$$

Here the third equation is due to:

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}} [\nabla_\theta \log q_{\theta^*} (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}})] &= \int_{\Omega^k} \frac{\nabla_\theta q_{\theta^*} (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}})}{q_{\theta^*} (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}})} \cdot p (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) d\mathbf{X}_\gamma^{\text{te}} \\
&= \int_{\Omega^k} \nabla_\theta q_{\theta^*} (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) d\mathbf{X}_\gamma^{\text{te}} \\
&= \nabla_\theta \int_{\Omega^k} q_{\theta^*} (\mathbf{X}_\gamma^{\text{te}} | \mathbf{X}_{-\gamma}^{\text{te}}, \mathbf{X}^{\text{ct}}) d\mathbf{X}_\gamma^{\text{te}} = \nabla_\theta 1 = 0.
\end{aligned} \tag{12}$$

In addition, since  $A \succeq 0$ , we have

$$\nabla_\theta^2 L_{k+1}(\theta^*) = A + B + B^\top + C = A + \nabla_\theta^2 L_k(\theta^*) \succeq \nabla_\theta^2 L_k(\theta^*).$$

Noting that  $\Gamma_k = (\nabla_\theta^2 L_k(\theta^*))^{-1}$  by [Equation \(10\)](#), it follows that  $\Gamma_{k+1} \preceq \Gamma_k$ . Now the claim follows.  $\square$

## B.2.2 Generalization for Joint Distribution Learning

The generalization error for joint distribution learning is characterized by a key concept termed *approximate tensorization of entropy*. It measures the “complexity” of the distribution over  $(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  by evaluating how easily an algorithm can generate samples from the joint distribution  $q(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  with access to local conditional distributions  $q(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}})$ . Technically, approximate tensorization of entropy is associated with the mixing time of Gibbs sampling dynamics, which is the sample generation algorithm to be considered.

**Definition B.1** (Approximate Tensorization of Entropy ([Caputo & Parisi, 2021](#))). For a distribution  $q(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  and the set of  $k$ -cell masks  $\Pi_k$ , if there exists a constant  $C_k(q)$  depending on  $q$  and  $k$ , such that for any distribution  $r$  over  $(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$ ,

$$D_{\text{KL}}(r \| q) \leq C_k(q) \cdot \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}}) \| q(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}}))],$$

then  $q$  satisfies approximate tensorization of entropy with respect to the constant  $C_k$  and the mask set  $\Pi_k$ . Let  $\underline{C}_k(q)$  be the minimum of all possible constants  $C_k(q)$  such that  $q$  satisfies approximate tensorization of entropy.

Before presenting the main result, we introduce a few regularity conditions in the parametric class  $q_\theta(\mathbf{X}^{\text{te}} | \mathbf{X}^{\text{ct}})$  that defines the model.

**Assumption B.1** (Regularity Conditions in the Parametric Class ([Li et al., 2024c](#))).

1. There exists  $\beta \in (0, 1)$  such that  $\forall 1 \leq k \leq d, \forall \pi \in \Pi_k$  and  $\forall \theta \in \Theta$ ,  $p(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}}) > 0$  implies  $q_\theta(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}}) > \beta$ .
2. For any  $\epsilon > 0$ , there exists a partition  $\text{Par}(\Theta) = \{\Theta_1, \dots, \Theta_{|\text{Par}(\Theta)|}\}$  of  $\Theta$ , such that  $\forall 1 \leq k \leq d, \forall \pi \in \Pi_k, \forall \Theta_i \in \text{Par}(\Theta), \forall \theta_1, \theta_2 \in \Theta_i$ , and any  $(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$ ,

$$|\log q_{\theta_1}(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}}) - \log q_{\theta_2}(\mathbf{X}_\pi^{\text{te}} | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}})| \leq \frac{\epsilon}{2}.$$

Let  $N(\Theta, \epsilon)$  be the cardinality of the smallest partition  $\text{Par}(\Theta)$  that satisfies the condition above.

The first assumption implies that the true distribution  $p(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}})$  supports the parametric distribution  $q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}^{\text{te}})$ . The second assumption specifies the covering number of the parameter space  $\Theta$  and the lipschitz continuity of the log-likelihood loss function.

**Theorem B.5** (Formal Version of [Theorem 6.3](#)). For  $\theta \in \Theta$ , assume  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_1(q_\theta)$  and the mask set  $\Pi_1$ . Then for any  $1 \leq k \leq d$ ,  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_k(q_\theta)$  and the mask set  $\Pi_k$ . Furthermore,  $\underline{C}_{k+1}(q_\theta) \leq \underline{C}_k(q_\theta)$ .

Under [Assumption B.1](#), for any  $\epsilon > 0$  and any  $\delta \in (0, \frac{1}{d})$ , with probability at least  $1 - d\delta$ , for any  $1 \leq k \leq d$  and any  $\theta \in \Theta$ ,

$$\mathbb{E}_{\mathbf{X}^{\text{ct}} \sim p} [D_{\text{TV}}(q_\theta(\cdot | \mathbf{X}^{\text{ct}}) \| p(\cdot | \mathbf{X}^{\text{ct}}))] < \sqrt{\frac{1}{2} \underline{C}_k(q_\theta) \left( \hat{L}_k(\theta) + B \log \frac{1}{\beta} + \epsilon \right) + C},$$

where  $B = \sqrt{\frac{1}{\delta} \cdot (8|\Omega|)^{d(m+1)} N(\Theta, \epsilon)} + \sqrt{\frac{1}{2n} \cdot \log \frac{8N(\Theta, \epsilon)}{\delta}}$ , and  $C = \sqrt{\frac{|\Omega|^{3d(m+1)}}{8\delta n}}$ .

With a sufficiently large sample size  $n$  and a sufficiently small loss value  $\hat{L}_k(\theta)$ , the upper bound is dominated by the term  $\sqrt{\frac{1}{2}C_k(q_\theta)(B \log \frac{1}{\beta} + \epsilon)}$ , which is scaled by  $C_k(q_\theta)$ , the constant for the approximate tensorization of entropy. The theorem implies a reduced upper bound for the estimation error of the joint distribution with an increasing number of masked cells.

We prove the monotonicity of  $C_k(q_\theta)$  with respect to  $k$  in [Proposition B.6](#), and prove the upper bound for generalization error in [Proposition B.7](#).

**Proposition B.6.** *For  $\theta \in \Theta$ , assume  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_1(q_\theta)$  and the mask set  $\Pi_1$ . Then for any  $1 \leq k \leq d$ ,  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_k(q_\theta)$  and the mask set  $\Pi_k$ . Furthermore,  $C_{k+1}(q_\theta) \leq C_k(q_\theta)$ .*

*Proof of Proposition B.6.* Since  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_1(q_\theta)$  and the mask set  $\Pi_1$ , for any distribution  $r$  over  $(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$ ,

$$D_{\text{KL}}(r \parallel q_\theta) \leq C_1(q_\theta) \cdot \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_1)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))].$$

By [Definition B.1](#),

$$D_{\text{KL}}(r \parallel q_\theta) \leq C_1(q_\theta) \cdot \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_1)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))].$$

We prove the proposition by deduction. Assume for some  $1 \leq k < d$ ,  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_k(q_\theta)$  and the mask set  $\Pi_k$ . It follows that

$$D_{\text{KL}}(r \parallel q_\theta) \leq C_k(q_\theta) \cdot \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))]. \quad (13)$$

We have

$$\begin{aligned} & \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_{k+1})} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))] \\ &= \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k), a \sim \text{Unif}([d] \setminus \pi)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi \cup \{a\}}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi \cup \{a\}}))] \\ &\geq \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k), a \sim \text{Unif}([d] \setminus \pi)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))] \\ &= \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_k)} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))]. \end{aligned} \quad (14)$$

The inequality follows from the data processing inequality:

$$D_{\text{KL}}(p(x, y) \parallel q(x, y)) = \mathbb{E}_x [D_{\text{KL}}(p(y|x) \parallel q(y|x))] + D_{\text{KL}}(p(x) \parallel q(x)) \geq \mathbb{E}_x [D_{\text{KL}}(p(y|x) \parallel q(y|x))].$$

Combining [Equations \(13\)](#) and [\(14\)](#),

$$D_{\text{KL}}(r \parallel q_\theta) \leq C_k(q_\theta) \cdot \mathbb{E}_{(\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}}) \sim p, \pi \sim \text{Unif}(\Pi_{k+1})} [D_{\text{KL}}(r(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}) \parallel q_\theta(\cdot | \mathbf{X}^{\text{ct}}, \mathbf{X}_{-\pi}))].$$

Therefore,  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to  $C_k(q_\theta)$  and the mask set  $\Pi_{k+1}$ . It follows that  $C_{k+1}(q_\theta) \leq C_k(q_\theta)$ . By deduction, for any  $1 \leq k \leq d$ ,  $q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})$  satisfies approximate tensorization of entropy with respect to some constant  $C_k(q_\theta)$  and the mask set  $\Pi_k$ .  $\square$

**Proposition B.7.** *Under [Assumption B.1](#) and the condition in [Proposition B.6](#), for any  $\epsilon > 0$  and any  $\delta \in (0, \frac{1}{d})$ , with probability at least  $1 - d\delta$ , for any  $1 \leq k \leq d$  and any  $\theta \in \Theta$ ,*

$$\mathbb{E}_{\mathbf{X}^{\text{ct}} \sim p} [D_{\text{TV}}(q_\theta(\cdot | \mathbf{X}^{\text{ct}}) \parallel p(\cdot | \mathbf{X}^{\text{ct}}))] < \sqrt{\frac{1}{2}C_k(q_\theta) \left( \hat{L}_k(\theta) + B \log \frac{1}{\beta} + \epsilon \right)} + C,$$

where  $B = \sqrt{\frac{1}{\delta} \cdot (8|\Omega|)^{d(m+1)} N(\Theta, \epsilon)} + \sqrt{\frac{1}{2n} \cdot \log \frac{8N(\Theta, \epsilon)}{\delta}}$ , and  $C = \sqrt{\frac{|\Omega|^{3d(m+1)}}{8\delta n}}$ .

The proposition is a corollary from Theorem 4 in [Li et al. \(2024c\)](#), which provides an upper bound for  $D_{\text{TV}}(q_\theta \parallel p)$  in our setting. The remaining gap is an extension of the result to total variation between conditional distributions. We present [Li et al. \(2024c\)](#)'s result as a lemma.

**Lemma B.8** ([Li et al. \(2024c\)](#), Theorem 4). *Consider random variables  $\mathbf{X} \in \Omega^d$  and  $\pi \subset [d]$ . We are given  $n$  i.i.d. samples  $\{\mathbf{x}^{(i)}\}_{i=1}^n$  drawn from  $p(\mathbf{X})$ . For each sample  $\mathbf{x}^{(i)}$ , we observe  $l$  i.i.d. masks  $\{\pi_j^{(i)}\}_{j=1}^l$  drawn from  $p(\pi | \mathbf{x}^{(i)})$ . Consider the empirical loss function:*

$$\hat{L}(\theta) = \frac{1}{nl} \sum_{i=1}^n \sum_{j=1}^l -\log q_\theta \left( \mathbf{x}_{\pi_j^{(i)}}^{(i)} | \mathbf{x}_{-\pi_j^{(i)}}^{(i)}, \pi_j^{(i)} \right).$$

Suppose the following conditions are met:

1. There exists a constant  $C(q_\theta)$  depending on  $q_\theta$ , such that for any distribution  $r$  over  $\mathbf{X}$ ,

$$D_{\text{KL}}(r \parallel q) \leq C(q_\theta) \cdot \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}), \pi \sim p(\pi|\mathbf{x})} [D_{\text{KL}}(r(\cdot|\mathbf{x}_{-\pi}, \pi) \parallel q(\cdot|\mathbf{x}_{-\pi}, \pi)].$$

2. There exists  $\beta \in (0, 1)$  such that  $\forall \pi \subset [d]$  and  $\forall \theta \in \Theta$ ,  $p(\mathbf{x}_\pi | \mathbf{x}_{-\pi}, \pi) > 0$  implies  $q_\theta(\mathbf{x}_\pi | \mathbf{x}_{-\pi}, \pi) > \beta$ .

3. For any  $\epsilon > 0$ , there exists a partition  $\text{Par}(\Theta) = \{\Theta_1, \dots, \Theta_{|\text{Par}(\Theta)|}\}$  of  $\Theta$ , such that  $\forall \pi \subset [d]$ ,  $\forall \Theta_i \in \text{Par}(\Theta)$ ,  $\forall \theta_1, \theta_2 \in \Theta_i$ , and any  $\mathbf{x}$ ,

$$|\log q_{\theta_1}(\mathbf{x}_\pi | \mathbf{x}_{-\pi}, \pi) - \log q_{\theta_2}(\mathbf{x}_\pi | \mathbf{x}_{-\pi}, \pi)| \leq \frac{\epsilon}{2}.$$

Let  $N(\Theta, \epsilon)$  be the cardinality of the smallest partition  $\text{Par}(\Theta)$  that satisfies the condition above.

Then for any  $\epsilon > 0$  and any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , for any  $\theta \in \Theta$ ,

$$D_{\text{TV}}(q_\theta(\mathbf{X}) \parallel p(\mathbf{X})) < \sqrt{\frac{1}{2}C(q_\theta)\left(\hat{L}(\theta) + B \log \frac{1}{\beta} + \epsilon\right)} + C,$$

where  $B = \sqrt{\frac{1}{\delta} \cdot (8|\Omega|)^d N(\Theta, \epsilon)} + \sqrt{\frac{1}{2n} \cdot \log \frac{8N(\Theta, \epsilon)}{\delta}}$ , and  $C = \sqrt{\frac{|\Omega|^{3d}}{8\delta n}}$ .

*Remark B.1.* Theorem 4 in [Li et al. \(2024c\)](#) is specified for  $\hat{\theta}$  as the minimizer of the empirical loss function. In fact, the proof applies uniformly to arbitrarily  $\theta \in \Theta$ .

*Proof of Proposition B.7.* For each pair of  $(\mathbf{x}^{\text{ct},(i)}, \mathbf{x}^{\text{te},(i)})$ , exactly one mask  $\pi_i$  is drawn independently from  $\text{Unif}(\Pi_k)$ . Therefore,  $\pi \perp\!\!\!\perp (\mathbf{X}^{\text{ct}}, \mathbf{X}^{\text{te}})$ . It follows from [Lemma B.8](#) that for each  $1 \leq k \leq d$ , for any  $\epsilon > 0$  and any  $\delta \in (0, 1)$ , with probability at least  $1 - \delta$ , for any  $\theta \in \Theta$ ,

$$D_{\text{TV}}(q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}}) \parallel p(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})) < \sqrt{\frac{1}{2}C_k(q_\theta)\left(\hat{L}_k(\theta) + B \log \frac{1}{\beta} + \epsilon\right)} + C, \quad (15)$$

where  $B = \sqrt{\frac{1}{\delta} \cdot (8|\Omega|)^{d(m+1)} N(\Theta, \epsilon)} + \sqrt{\frac{1}{2n} \cdot \log \frac{8N(\Theta, \epsilon)}{\delta}}$ , and  $C = \sqrt{\frac{|\Omega|^{3d(m+1)}}{8\delta n}}$ .

By union bound, for any  $\epsilon > 0$  and any  $\delta \in (0, \frac{1}{d})$ , with probability at least  $1 - d\delta$ , [Equation \(15\)](#) is satisfied for any  $1 \leq k \leq d$  and any  $\theta \in \Theta$ .

Furthermore, we have

$$\begin{aligned} \mathbb{E}_{\mathbf{X}^{\text{ct}} \sim p}[D_{\text{TV}}(q_\theta(\cdot | \mathbf{X}^{\text{ct}}) \parallel p(\cdot | \mathbf{X}^{\text{ct}}))] &= \sum_{\mathbf{x}^{\text{ct}} \in \Omega^{m \times d}} p(\mathbf{x}^{\text{ct}}) D_{\text{TV}}(q_\theta(\cdot | \mathbf{x}^{\text{ct}}) \parallel p(\cdot | \mathbf{x}^{\text{ct}})) \\ &= \sum_{\mathbf{x}^{\text{ct}} \in \Omega^{m \times d}} p(\mathbf{x}^{\text{ct}}) \cdot \frac{1}{2} \sum_{\mathbf{x}^{\text{te}} \in \Omega^d} |q_\theta(\mathbf{x}^{\text{te}} | \mathbf{x}^{\text{ct}}) - p(\mathbf{x}^{\text{te}} | \mathbf{x}^{\text{ct}})| \\ &= \frac{1}{2} \sum_{\mathbf{x}^{\text{ct}} \in \Omega^{m \times d}, \mathbf{x}^{\text{te}} \in \Omega^d} |p(\mathbf{x}^{\text{ct}}) q_\theta(\mathbf{x}^{\text{te}} | \mathbf{x}^{\text{ct}}) - p(\mathbf{x}^{\text{ct}}) p(\mathbf{x}^{\text{te}} | \mathbf{x}^{\text{ct}})| \quad (16) \\ &= \frac{1}{2} \sum_{\mathbf{x}^{\text{ct}} \in \Omega^{m \times d}, \mathbf{x}^{\text{te}} \in \Omega^d} |q_\theta(\mathbf{x}^{\text{te}}, \mathbf{x}^{\text{ct}}) - p(\mathbf{x}^{\text{te}}, \mathbf{x}^{\text{ct}})| \\ &= D_{\text{TV}}(q_\theta(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}}) \parallel p(\mathbf{X}^{\text{te}}, \mathbf{X}^{\text{ct}})). \end{aligned}$$

The proof is complete by combining [Equations \(15\)](#) and [\(16\)](#).  $\square$