SUFFIKV: LEARNING SUFFICIENT TASK STATES VIA LAYER-WISE KV MEMORIES FOR SCALABLE TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) enable supervised prediction by conditioning a frozen model on a labeled support table, but this makes the table part of every inference computation. As support sets grow, full-context inference becomes a latency and memory bottleneck, and additional raw rows may be redundant, noisy, or poorly aligned with the frozen attention mechanism. We introduce **SuffiKV**, a test-time adaptation framework that compiles a support table into a compact layer-wise key-value task state for a frozen TFM. SuffiKV separates the table used as evidence from the read state used at prediction time: a write stage constructs context-derived KV anchors, a bounded trust-region residual refines them, and a read stage predicts future queries using only the cached KV state. The state is optimized by anchored operator distillation, using full-context predictions and attention effects as early functional guidance while held-out pseudo-query labels refine the state toward supervised task risk. This turns tabular in-context learning into task-state inference: more data can improve the written state without increasing read-time context length. On a long-context tabular benchmark, SuffiKV improves the accuracy–latency–memory Pareto frontier, reducing subsequent inference latency by **89.9** $\times$ and peak memory by **66.2%** while improving AUC by **1.5%**. Code and models will be released.

1 INTRODUCTION

Tabular foundation models (TFMs) promise supervised prediction by conditioning a frozen model on a labeled table, without fitting task-specific parameters (Hollmann et al., 2023; 2025; Ma et al., 2025; Qu et al., 2025; Zhang et al., 2025; 2026; Grinsztajn et al., 2025; Qu et al., 2026; Grinsztajn et al., 2026). This promise carries an unusual deployment cost: the training table remains in the inference computation. Every query repeatedly reads the support context, so larger support sets increase attention cost, memory use, and latency. Worse, the extra rows are not uniformly useful; they may be redundant, noisy, imbalanced, or irrelevant. Full-context inference is therefore not an oracle. It is a raw-context computation performed by a frozen attention mechanism.

We frame this bottleneck as a mismatch between *evidence* and *read state*. The evidence set is the labeled table from which the task should be inferred. The read state is what the model actually consumes at prediction time. Classical fitted tabular models separate the two: data are used to infer a task-specific state, such as trees or parameters, and prediction no longer revisits the raw table (Chen & Guestrin, 2016). Standard tabular in-context learning ties them together: the support table is both the evidence and the read state. As a result, data scale and inference scale grow together.

Existing methods reduce this burden but mostly keep the read state row-like. Retrieval and context selection choose fewer rows (Thomas et al., 2024); TuneTables and related methods learn smaller input-level contexts (Feuer et al., 2024b); scalable attention schemes such as Chunked TabPFN make long contexts cheaper to process (Sergazinov & Yin, 2025). These are important steps, but they still operate at the input or row-attention level. A transformer ultimately reads context through layer-wise keys and values (Vaswani et al., 2017). This suggests a stronger target for adaptation: do not ask which rows or pseudo-rows to show the model; ask which internal KV state the frozen model should read after the evidence has been absorbed.

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

We propose **SuffiKV**, a test-time adaptation framework that turns tabular in-context learning into **task-state inference**. Given a support table, SuffiKV writes the evidence once into compact layer-wise KV task states inside the frozen TFM. Future queries read only these cached states, not the raw table. This write–read separation removes the training table from the query path and decouples the amount of data used to infer the task from the amount of memory read at deployment.

The learned state is not a row, pseudo-row, or prompt embedding. It is a virtual attention memory: keys specialize to routing queries to task-relevant slots, while values carry the predictive payload. SuffiKV initializes this state from a context-derived anchor written by the frozen model, then refines it with bounded low-rank KV residuals. The state is optimized by anchored operator distillation: early full-context predictions and layer-wise attention effects make the memory readable by the frozen model, while later held-out pseudo-query labels refine it toward supervised task risk. Thus, full-context inference is used as privileged guidance, not as a performance ceiling.

This distinction is central. A compact KV state is not merely a lossy summary of the table; it is an optimized read state learned from the table. It can merge redundant examples, suppress noisy rows, allocate capacity to rare classes, represent decision boundaries, and adjust calibration. As the support set grows, full-context TFMs must route attention over a larger raw table, whereas SuffiKV receives richer evidence for writing a fixed-size state. The desired scaling behavior is therefore: more data should produce a better task state, not a longer read path.

We evaluate SuffiKV on 5 tabular benchmarks spanning binary classification, multiclass classification, regression, small-data settings, and long-context regimes. We compare with full-context TFMs, random and class-balanced subsampling, input-level context optimization, coreset selection, retrieval-based contexts, scalable TFM variants, and tuned gradient-boosted tree baselines (Chen & Guestrin, 2016; Prokhorenkova et al., 2018; Erickson et al., 2020). SuffiKV improves the accuracy–latency–memory Pareto frontier: in long-context regimes, compared with full-context inference, it reduces subsequent inference latency by 89.9$\times$ and peak memory by 66.2%, while improving AUC by 1.5%.

Contributions. Our contributions are as follows.

1. We introduce **task-state inference** for frozen TFMs, separating the support table used as evidence from the compact read state used at prediction time.
2. We instantiate this principle with **layer-wise KV task states**, moving tabular context adaptation from rows and prompts to the transformer's internal attention-memory interface.
3. We develop a **write–read adaptation protocol** with **anchored operator distillation**, keeping the deployment read path identical to the training path while using full-context behavior as guidance.
4. We show that SuffiKV learns compact KV states that preserves the directions **accounting for most of full-context predictive behavior**, while additionally exploiting **non-shared, higher-rank directions**.
5. We show that compact KV states can replace full support tables at inference time, improving the accuracy–latency–memory Pareto frontier and changing how frozen TFMs scale with support-set size.

2 Related Work

Tabular foundation models. Recent tabular foundation models (TFMs), such as TabPFN-v2.5 (Grinsztajn et al., 2025), LimiX v1 (Zhang et al., 2025), formulate tabular prediction as in-context inference over a labeled context set, enabling strong zero-shot or few-shot performance without task-specific retraining. In this paradigm, the context acts as an implicit training set that provides task-specific information for test samples. However, as the context size increases, inference becomes increasingly expensive due to the attention cost over context examples. Therefore, how to construct a compact yet informative context is a key problem for scaling TFMs to larger tabular datasets.

Context compression and scalable computation. Existing methods improve TFM efficiency through context reduction and scalable computation. Random and class-balanced subsampling provide simple baselines, while coreset methods select representative instances: MixturePFN (Xu

et al., 2025) partitions training data into candidate contexts and assigns relevant subsets to queries, and LUCoS (Ipas et al., 2026) embeds an unlabeled training pool using an unsupervised PFN and applies K-Medoids in the latent space to identify representative instances. Alternatively, retrieval-based approaches like LocalPFN (Thomas et al., 2024) dynamically fetch nearest-neighbor contexts, though this disrupts PFN-style batching and adds inference overhead. Rather than reducing context size, learned context optimization seeks to retain task-relevant information within a smaller context. TuneTables (Feuer et al., 2024a) learns compact task-specific prompts to replace the original training set, and ICD (Ma et al., 2024) directly optimizes distilled examples through a frozen TabPFN. TACO (Zabërgja et al., 2026) jointly learns a compressor and predictor, mapping training tables into compact latent context rows. Although these methods address context-length limitations in TFMs from different perspectives, they all incur information loss, leading to performance degradation. In contrast, our method learns an optimized "readout state" from the table.

3 METHOD

3.1 CONTEXT–STATE SEPARATION VIA LAYER-WISE KV TASK STATES

Let $D = \{(x_i, y_i)\}_{i=1}^n$ be a labeled support table and let f_θ be a frozen tabular foundation model. Standard full-context inference predicts $p_D(y \mid x) = f_\theta(y \mid x; D)$, so the same table is both the *evidence* used to infer the task and the *read object* consumed at prediction time. SuffiKV separates these roles: it writes D into a compact task state M_D, and all future queries are predicted by a memory-only path $p_{M_D}(y \mid x) = f_\theta^{\mathrm{read}}(y \mid x; M_D)$. The raw table may be used to construct or supervise M_D, but it is absent from the query read path.

The state is placed at the transformer's native attention-memory interface. For layer ℓ and head h, a set of keys and values $(K, V) \in \mathbb{R}^{s\times d_h} \times \mathbb{R}^{s\times d_h}$ defines the read operator

$$\mathcal{A}_{\ell h}(q_{\ell h}; K, V) = \mathrm{softmax}\left(\frac{q_{\ell h}K^\top}{\sqrt{d_h}}\right)V, \qquad q_{\ell h} \in \mathbb{R}^{1\times d_h}. \tag{1}$$

Full-context inference uses keys and values induced by the n support rows. SuffiKV instead learns compact virtual memories with $m_{\ell h} \ll n$.

$$M_D = \left\{(K^M_{\ell h}, V^M_{\ell h}) : K^M_{\ell h}, V^M_{\ell h} \in \mathbb{R}^{m_{\ell h}\times d_h},\ \ell = 1, \ldots, L,\ h = 1, \ldots, H\right\}. \tag{2}$$

The uniform-budget case sets $m_{\ell h} = m$ for all layers and heads. During memory-only inference, the frozen model replaces support-context reads with $\mathcal{A}_{\ell h}(q_{\ell h}; K^M_{\ell h}, V^M_{\ell h})$, while all projection matrices and feed-forward blocks remain fixed.

This parameterization defines a fixed-budget adaptation problem. Let $\mathbf{m} = \{m_{\ell h}\}_{\ell,h}$, $\mathcal{M}_{\mathbf{m}}$ be the admissible class of compact KV read states under this budget, and let $\mathcal{P}_D$ denote the task distribution represented by the support table. The ideal task state is

$$M^\star_D = \operatorname*{arg\,min}_{M\in\mathcal{M}_{\mathbf{m}}} \mathbb{E}_{(x,y)\sim\mathcal{P}_D}\left[\mathcal{L}\left(y, f_\theta^{\mathrm{read}}(\cdot \mid x; M)\right)\right]. \tag{3}$$

In practice, $\mathcal{P}_D$ is accessed through cross-fitted pseudo-query splits from D. Full-context predictions and layer-wise attention effects serve as teacher signals for stable optimization, but they do not define the final target: the learned task state is allowed to improve beyond the full-context path through held-out label supervision.

Layer-wise KV states are more expressive than input-level summaries. They need not be rows, pseudo-rows, or prompt-like embeddings; they are virtual memories inserted directly where the frozen model reads task information. They are also depth-specific: early layers can encode feature-level or neighborhood statistics, middle layers can organize class-conditional structure, and later layers can refine decision boundaries or calibration. Unlike a hidden summary Z, which would tie keys and values through frozen projections such as $K = \mathrm{LN}(Z)W^K_{\ell h}$ and $V = \mathrm{LN}(Z)W^V_{\ell h}$, direct KV states let keys specialize to routing and values specialize to predictive payload. We call such a state *sufficient* when, for a frozen model and memory budget, it supports accurate memory-only prediction, preserves the relevant layer-wise effects of the support table, and removes raw-table access from deployment.

3.2 WRITING, READING, AND TRUST-REGION REFINEMENT

SuffiKV enforces a strict write–read protocol. The support table D is used to write a task state M_D, but every student prediction during both adaptation and deployment is evaluated through the same memory-only path $f_\theta^{\mathrm{read}}(\cdot \mid x; M_D)$. Full-context computation may produce anchors, teacher predictions, or operator targets, but the student query never attends to the raw support table. This path consistency is what prevents summary-token shortcuts from becoming train–inference mismatch.

The initial state is written by the frozen TFM itself. Let $H_\ell^D \in \mathbb{R}^{n\times d}$ be the support-table representations at layer ℓ, and $Z_\ell^S \in \mathbb{R}^{m\times d}$ be m summary slots, corresponding to the uniform-slot case $m_{\ell h} = m$. A frozen writer block lets these slots read the support states,

$$Z_\ell^W = \mathrm{Block}_{\theta,\ell}^{\mathrm{write}}\left(Z_\ell^S,\ H_\ell^D\right). \tag{4}$$

Here, $\mathrm{Block}_{\theta,\ell}^{\mathrm{write}}$ uses the same frozen attention and MLP parameters as the backbone, with summary slots attending to support-table states. The written slots are projected into layer-wise KV anchors

$$K_{\ell h}^0 = \mathrm{LN}(Z_\ell^W)W_{K,\ell h}, \qquad V_{\ell h}^0 = \mathrm{LN}(Z_\ell^W)W_{V,\ell h}. \tag{5}$$

The resulting anchor $M_0(D) = \{(K_{\ell h}^0, V_{\ell h}^0)\}_{\ell,h}$ is compact because $m \ll n$, and model-aligned because it is produced by the backbone's own computation. Crucially, it is written before query prediction; queries never read D during the read stage. The anchor is not the final state, so SuffiKV refines it using a trust-region KV parameterization.

For each layer ℓ and head h, the final tensors are

$$K_{\ell h}^M = K_{\ell h}^0 + g_{\ell h}^K \rho_{\ell h}^K \tanh\left(A_{\ell h}^K B_{\ell h}^K\right), \qquad V_{\ell h}^M = V_{\ell h}^0 + g_{\ell h}^V \rho_{\ell h}^V \tanh\left(A_{\ell h}^V B_{\ell h}^V\right). \tag{6}$$

where $A_{\ell h}^K, A_{\ell h}^V \in \mathbb{R}^{m_{\ell h}\times r}$, $B_{\ell h}^K, B_{\ell h}^V \in \mathbb{R}^{r\times d_h}$, and $r \ll d_h$. The $\tanh(\cdot)$ is elementwise, giving bounded residuals of shape $m_{\ell h} \times d_h$. The scales $\rho_{\ell h}^K, \rho_{\ell h}^V > 0$ are set from empirical full-context KV standard deviations, while gates $g_{\ell h}^K, g_{\ell h}^V \geq 0$ are initialized small so adaptation starts from the written anchor.

This trust-region form gives the method its useful middle ground. The anchor prevents arbitrary out-of-manifold memories; the direct KV residual lets keys specialize to routing and values to predictive payload; the low-rank bounded form limits capacity, controls key norms, and discourages pathological sharp attention. With gates near zero, SuffiKV reduces to the context-written anchor. As the gates grow, the memory can reorganize the evidence set into a better task state while remaining close to a model-aligned read state. The next section describes how anchored operator distillation optimizes this state using early full-context guidance and later held-out label supervision.

3.3 TASK-STATE INFERENCE AS CONSTRAINED FUNCTIONAL DISTILLATION

We now specify what the compact state should do. Since stability is enforced by the trust-region parameterization (Section 3.2), the remaining question is functional: the state should minimize supervised task risk while preserving the internal reading behavior that makes the full-context model usable as guidance. Given a write set $C \subseteq D$, the frozen writer produces an anchor $M_0(C)$, and the learned state is restricted to the trust-region family $\mathcal{M}_{\mathbf{m}}(M_0(C))$. We formulate task-state learning:

$$\min_{M\in\mathcal{M}_{\mathbf{m}}(M_0(C))} \widehat{R}_{\mathrm{sup}}(M) \qquad \text{s.t.} \qquad D_{\mathrm{op}}(M, C) \leq \epsilon_{\mathrm{op}}. \tag{7}$$

Here, $\widehat{R}_{\mathrm{sup}}$ is the supervised risk of the memory-only read path, and D_{op} measures whether the compact state preserves how the full context is read inside the frozen model. Full-context inference is therefore privileged functional supervision, not the final target.

In practice, we optimize the scheduled Lagrangian

$$\mathcal{L}_{\mathrm{SuffiKV}}^{(t)}(M) = \mathcal{R}_{\mathrm{task}}^{(t)}(M) + \lambda_{\mathrm{op}}^{(t)} D_{\mathrm{op}}(M, C). \tag{8}$$

For a pseudo-query (x, y), we have $p_C(y \mid x) = f_\theta(y \mid x; C)$, $\quad p_M(y \mid x) = f_\theta^{\mathrm{read}}(y \mid x; M)$. The task term is teacher-guided early and label-refined later:

$$\mathcal{R}_{\mathrm{task}}^{(t)}(x, y; M) = (1-\beta_t)\tau^2 \mathrm{KL}(p_C^\tau(\cdot \mid x) \,\|\, p_M^\tau(\cdot \mid x)) + \beta_t \mathcal{L}(y, p_M(\cdot \mid x)). \tag{9}$$

It is averaged over pseudo-queries. Here, p^τ denotes temperature-scaled predictions, $\tau > 0$, and $\beta_t \in [0, 1]$ increases. For classification, $\mathcal{L}$ is cross-entropy; for regression, we use the analogous interpolation between matching the full-context prediction and fitting the pseudo-query target.

The operator term matches how the frozen model reads context internally. For layer ℓ and head h:

$$\mathcal{A}^C_{\ell h}(q) = \mathcal{A}_{\ell h}(q; K^C_{\ell h}, V^C_{\ell h}), \qquad \mathcal{A}^M_{\ell h}(q) = \mathcal{A}_{\ell h}(q; K^M_{\ell h}, V^M_{\ell h}). \tag{10}$$

Here $\mathcal{A}_{\ell h}$ is the attention operator from Section 3.1. We define

$$D_{\mathrm{op}}(M, C) = \sum_{\ell,h} w_{\ell h} \mathbb{E}_{q\sim\mathcal{Q}_{\ell h}} \left[\frac{\left\| \mathcal{A}^C_{\ell h}(q) - \mathcal{A}^M_{\ell h}(q) \right\|_2^2}{\left\| \mathcal{A}^C_{\ell h}(q) \right\|_2^2 + \epsilon} \right]. \tag{11}$$

with $w_{\ell h} \geq 0$ and $\epsilon > 0$. Early in adaptation, $\mathcal{Q}_{\ell h}$ is induced by query states from the full-context teacher forward pass, giving stable operator targets; later it can be evaluated on the memory-only read path. This enforces operator sufficiency: the compact state should not only produce good outputs, but should remain readable by the frozen model in the way the full context is read.

The schedule implements a transition from imitation to improvement.

$$\beta_t \uparrow, \qquad \lambda^{(t)}_{\mathrm{op}} \downarrow . \tag{12}$$

Early adaptation inherits predictions and internal reading behavior from the full-context model. Later, held-out pseudo-query labels dominate, while the operator term keeps the state aligned with the frozen backbone. Thus, SuffiKV uses full-context inference as a guide for task-state learning, not as a ceiling on performance. Additional details on cross-fitted state learning, adaptation and inference, complexity, and why compact KV states can outperform raw full context are given in Appendix A.

4 Experiment

We designed our experiments to answer the following questions: (1) How does SuffiKV compare with existing context compression methods, and full-context TFMs in predictive performance across classification and regression tasks? (2) To what extent does SuffiKV reduce inference latency and peak GPU memory usage relative to full-context inference? (3) Can SuffiKV mitigate the adverse effects of noise in long contexts? (4) What properties of the learned layer-wise KV representations and which design choices explain the predictive gains of SuffiKV?

4.1 Experimental Setup

Benchmarks. To facilitate a rigorous quantitative assessment of performance, we evaluate our approach on several established benchmark suites, specifically TabArena-CLS, TabArena-Reg (Erickson et al., 2025), and TabZilla (McElfresh et al., 2023). Among these benchmarks, the datasets containing more than 50,000 training samples (the number of testing samples is not constrained), 10000 features, or 10 target categories were excluded, following LimiX (Zhang et al., 2025). This selection process resulted in a final collection of 27 from TabZilla-Cls, 33 from TabArena-Cls and 12 from TabArena-Reg. Moreover, we curated two benchmarks from Kaggle, named TabMacro-Cls and TabMacro-Reg. These benchmarks comprise 29 large-scale classification datasets and 8 regression datasets, respectively, each containing more than 50,000 instances. Detailed descriptions of the evaluated benchmark datasets are provided in the Appendix C.3.

Baselines. We integrate our proposed method into TFMs within the In-Context Learning (ICL) framework and conduct extensive evaluations against representative baselines on both standard and large-scale datasets. To ensure a comprehensive comparison, our baseline suite encompasses full-context TFMs (Grinsztajn et al., 2025; Zhang et al., 2025); heuristic sampling strategies, including random and class-balanced subsampling; input-level context optimization techniques such as TuneTables and TACO (Feuer et al., 2024a; Zabërgja et al., 2026); the coreset selection algorithm LUCos (Ipas et al., 2026); retrieval-based context approaches like LoCALPFN (Thomas et al., 2024); and well-tuned tree-based and AutoML baselines, namely XGBoost, CatBoost, and AutoGluon (Chen & Guestrin, 2016; Prokhorenkova et al., 2018; Erickson et al., 2020).

Table 1: Benchmark-level mean performance and efficiency of the classification benchmark. Bold indicates the best results for TFM and the corresponding context-optimized variants.

Method	TabArena-Cls				TabZilla-Cls				TabMacro-Cls			
	Performance		Efficiency		Performance		Efficiency		Performance		Efficiency	
	AUC ↑	ACC ↑	Speedup↑	Mem↓	AUC ↑	ACC ↑	Speedup↑	Mem↓	AUC ↑	ACC ↑	Speedup↑	Mem↓
LimiX v1	0.835	0.859	1×	100%	0.937	0.866	1×	100%	0.777	0.735	1×	100%
LimiX v1 + TuneTables	0.754	0.733	0.5×	72.2%	0.839	0.682	1.0×	82.0%	0.718	0.594	2.3×	41.7%
LimiX v1 + SuffiKV	**0.851**	**0.871**	**4.5**×	**45.5%**	**0.946**	**0.884**	**8.4**×	**48.2%**	**0.804**	**0.755**	**101.9**×	**34.6%**
TabPFN v2.6	0.849	**0.875**	1×	100%	0.937	0.878	1×	100%	0.799	**0.749**	1×	100%
TabPFN v2.6 + TuneTables	0.731	0.701	0.2×	**66.2%**	0.792	0.646	0.5×	75.6%	0.701	0.566	1.4×	**23.1%**
TabPFN v2.6 + SuffiKV	**0.858**	0.871	**3.7**×	72.6%	**0.946**	**0.886**	**6.61**×	**67.1%**	**0.801**	0.746	**77.9**×	33.1%

Table 2: Benchmark-level mean performance and efficiency of the regression benchmark. Bold indicates the best results for TFM and the corresponding context-optimized variants.

Method	TabArena-Reg				TabMacro-Reg			
	Performance		Efficiency		Performance		Efficiency	
	R^2 ↑	RMSE ↓	Speedup↑	Mem↓	R^2 ↑	RMSE ↓	Speedup↑	Mem↓
LimiX v1	0.781	0.422	1×	100%	0.521	0.601	1×	100%
LimiX v1 + TuneTables	0.337	0.794	0.3×	64.5%	0.156	0.905	0.9×	**43.9%**
LimiX v1 + SuffiKV	**0.785**	**0.416**	**4.2**×	**59.0%**	**0.568**	**0.556**	**24.8**×	53.3%
TabPFN v2.6	0.802	0.398	**1**×	100%	**0.541**	0.586	1×	100%
TabPFN v2.6 + TuneTables	0.345	0.797	0.3×	168.0%	0.108	0.942	0.6×	239.0%
TabPFN v2.6 + SuffiKV	**0.802**	**0.397**	0.5×	**58.2%**	**0.541**	**0.580**	**2.0**×	**33.7%**

Metrics. To quantify model performance, we utilize the Area Under the Receiver Operating Characteristic curve (ROC AUC) and Accuracy for classification tasks. In multi-class scenarios, we employ the One-vs-One (OvO) strategy to calculate the ROC AUC. For regression objectives, we evaluate predictive performance using standard metrics, specifically the Root Mean Square Error (RMSE) and the coefficient of determination (R^2).

4.2 Main Results.

We conduct comprehensive experiments across five benchmarks, spanning binary classification, multiclass classification, regression, small-data settings, and long-context regimes. For classification tasks, as demonstrated in Table 1, our method successfully reduces inference latency and peak memory while simultaneously enhancing model performance. Across all three classification benchmarks, SuffiKV yields an AUC that significantly surpasses both the full context TFM and TuneTable. Notably, LimiX v1 equipped with SuffiKV achieves an AUC improvement of 2.7% over the standard full context LimiX v1 baseline. Furthermore, SuffiKV delivers more substantial improvements in inference speed and memory efficiency compared to TuneTable. We evaluate single-pass inference latency (prediction phase), alongside peak GPU memory consumption, comparing SuffiKV with full-context TFMs and Tunetables across five benchmarks. Specifically on the Large Cls benchmark, SuffiKV accelerates inference by 101.9× and 77.9× for LimiX v1 and TabPFN v2.6, respectively. This exceeds the speedup provided by TuneTable by an average of 17.4× across all TFMs and benchmarks. Concurrently, it reduces inference memory consumption by 65.4% and 66.9% for the respective models. Consequently, rather than operating as a lossy distillation mechanism like TuneTable, SuffiKV functions as a highly predictive and sufficient KV state. Table 2 presents the results across 2 regression benchmarks. Consistently, SuffiKV successfully reduces both inference latency and memory footprint while simultaneously improving overall performance. This advantage is particularly pronounced on the large-scale benchmark.

Furthermore, we conduct comprehensive comparisons against tree-based models as well as various context compression and selection methods. As illustrated in Table 3, our method consistently outperforms these diverse baselines across both classification and regression benchmarks. Notably, on the TabArena Cls benchmark, our approach achieves a 2.5% AUC improvement over AutoGluon, which serves as the strongest tree-based baseline. Although existing context compression and selection baselines (such as LocalPFN, LUCos, and TACO) successfully scale to massive datasets,

Table 3: Benchmark-level mean performance for tree-based models, context selection methods, and sampling baselines. Bold indicates the best results for TFM and the corresponding context-optimized variants. Regression tasks were excluded from the class-balanced sampling experiments.

	Method	TabArena-Cls		TabZilla-Cls		TabMacro-Cls		TabArena-Reg		TabMacro-Reg	
		AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	R2 ↑	RMSE ↓	R2 ↑	RMSE ↓
Tree-based	XGBoost	0.820	0.851	0.893	0.808	0.774	0.731	0.741	0.459	0.514	0.594
	CatBoost	0.783	0.838	0.870	0.765	0.748	0.718	0.772	0.429	0.526	0.587
	AutoGluon	0.830	0.863	0.927	0.868	0.803	0.761	0.798	0.402	0.586	0.546
TabPFN	TabPFN v2	0.842	0.870	0.925	0.856	0.769	0.715	0.768	0.434	0.482	0.624
	LoCalPFN	0.832	0.856	0.918	0.862	0.798	0.754	0.576	0.587	0.414	0.658
	LUCoS	0.750	0.825	0.849	0.780	0.681	0.694	0.738	0.459	0.430	0.667
	TACO	0.832	0.858	0.910	0.824	0.766	0.721	0.696	0.520	0.396	0.713
LimiX v1 Sampling	Random (5% / 2048)	0.789	0.832	0.871	0.796	0.777	0.735	0.659	0.546	0.457	0.636
	Random (10% / 4096)	0.814	0.845	0.885	0.810	0.782	0.738	0.717	0.491	0.482	0.620
	Random (20% / 8192)	0.829	0.853	0.913	0.836	0.788	0.740	0.750	0.457	0.500	0.610
	Random (50% / 30000)	0.843	0.867	0.931	0.857	0.790	0.741	0.779	0.428	0.518	0.601
	KNN-based	0.794	0.854	0.882	0.797	0.737	0.715	0.664	0.542	0.356	0.764
	Class-balanced	0.768	0.737	0.845	0.726	0.740	0.577	-	-	-	-
	SuffiKV	**0.851**	**0.871**	**0.946**	**0.884**	**0.804**	**0.755**	**0.785**	**0.416**	**0.568**	**0.556**
TabPFN v2.6 Sampling	Random (5% / 2048)	0.791	0.831	0.883	0.797	0.775	0.733	0.682	0.524	0.454	0.643
	Random (10% / 4096)	0.822	0.848	0.890	0.816	0.780	0.736	0.721	0.486	0.479	0.620
	Random (20% / 8192)	0.836	0.855	0.903	0.840	0.790	0.740	0.749	0.457	0.495	0.613
	Random (50% / 30000)	0.849	0.870	0.932	0.867	0.800	**0.746**	0.781	0.424	0.526	0.593
	KNN-based	0.789	0.854	0.872	0.797	0.728	0.713	0.686	0.511	0.375	0.751
	Class-balanced	0.784	0.692	0.857	0.727	0.738	0.567	-	-	-	-
	SuffiKV	**0.858**	**0.871**	**0.946**	**0.886**	**0.801**	**0.746**	**0.802**	**0.397**	**0.541**	**0.580**

they inevitably suffer from varying degrees of performance degradation when compared to their corresponding full context TFM counterparts. In contrast, SuffiKV not only enables TFMs to scale efficiently to massive datasets but also actively enhances their predictive performance. We provide the complete experimental results in Appendix C.4.

4.3 Inference Efficiency.

When evaluated on a single 24GB RTX 5090 GPU, the full context TFMs encounter out-of-memory (OOM) errors on certain datasets. In contrast, SuffiKV completely avoids OOM errors across both small-scale and large-scale benchmarks. For datasets where full-context TFMs encounter OOM errors, we apply an outer-batch inference method to obtain their baseline metrics on the test set. Section 4.2 details our preceding analysis regarding the substantial improvements in inference speed and memory footprint achieved by SuffiKV on empirical benchmarks.

We further investigate inference efficiency on synthetic datasets. We construct a grid of synthetic datasets varying across sample sizes $n \in [5000, 100000]$ and feature dimensions $F \in [5, 1000]$. For each configuration in the grid, we evaluate both our SuffiKV-augmented models and their corresponding full-context TFMs by measuring their fitting and prediction times, where fitting time denotes the preprocessing overhead. For consistency, we fix the test set to 1,000 rows. Figure 1 presents a heatmap of inference speed as the context size and number of features vary.

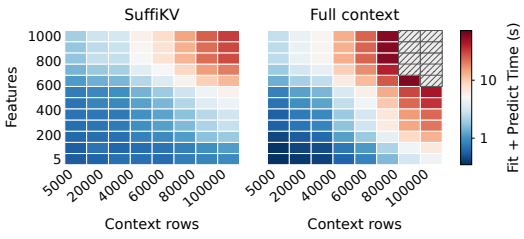

Figure 1: Fit + Predict time heatmap of TabPFN v2.6+SuffiKV and full-context TabPFN v2.6.

Across all grid points, SuffiKV consistently achieves faster inference and encounters no OOM errors, as reflected by the blue and light-red regions in the heatmap. In contrast, full-context TFMs encounter OOM errors on 9 of the 96 dataset configurations in the grid (Gray hatched area). These results further demonstrate that SuffiKV can scale to larger tabular workloads under stricter memory constraints.

4.4 SUFFIKV ACTS AS A ROBUST TASK MEMORY.

As illustrated by the solid line in Figure 2, model performance initially improves with the incorporation of additional contextual information. However, beyond a specific threshold, marginal performance gains gradually saturate. This suggests that irrelevant and redundant information or noise within excessively long contexts may degrade the efficacy of in-context learning. Conversely, as depicted by the dashed line, TFMs equipped with SuffiKV maintain a consistent upward performance trajectory. This demonstrates that SuffiKV not only facilitates efficient inference on massive datasets but also alleviates this performance bottleneck by filtering uninformative signals.

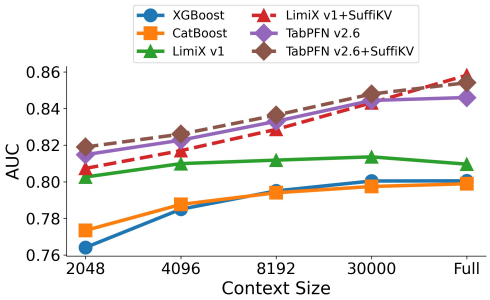

Figure 2: As the context size increases, the performance of tree-based models and TFMs eventually saturates and reaches a bottleneck.

Consequently, the compact KV states generated during the SuffiKV write phase do not merely constitute a lossy summary of the table. Instead, they represent an optimized read state learned from the tabular data. This optimized state is capable of merging redundant samples, eliminating noisy rows, allocating capacity for rare classes, delineating decision boundaries, and refining model calibration. Our empirical evaluations confirm that SuffiKV extends beyond simple lossy summarization; it effectively mitigates the adverse effects of noisy information within ultra-long context windows.

Specifically, we construct a noisy context by perturbing the labels of selected high-confidence samples, utilizing this as a teacher-side context to optimize a noisy SuffiKV variant. Figure 3 presents a comparative analysis of inference performance on a large-scale classification test set. The results demonstrate that full-context TFMs conditioned on noisy contexts suffer severe performance degradation compared to their clean baselines. In contrast, SuffiKV exhibits remarkable robustness, incurring only a negligible performance drop relative to its clean-context counterpart.

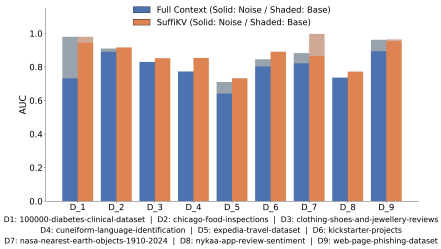

Figure 3: Performance degradation magnitude of full context TFMs (LimiX v1) and SuffiKV in the presence of context noise.

4.5 SUFFIKV ACTS AS AN EXPRESSIVE TASK MEMORY.

We conduct a diagnostic analysis to understand why SuffiKV improves in-context tabular inference. For each transformer layer, we collect the key and value activations used by query examples when attending to either the full-context or SuffiKV. Given a key/value matrix $X \in \mathbb{R}^{n \times d}$, we center it and compute its right singular subspace with basis $Q_\rho = [v_1, \ldots, v_{r_\rho}]$: $\bar{X} = X - \mathbf{1}\mu^\top, \quad \bar{X} = U\Sigma V^\top$.

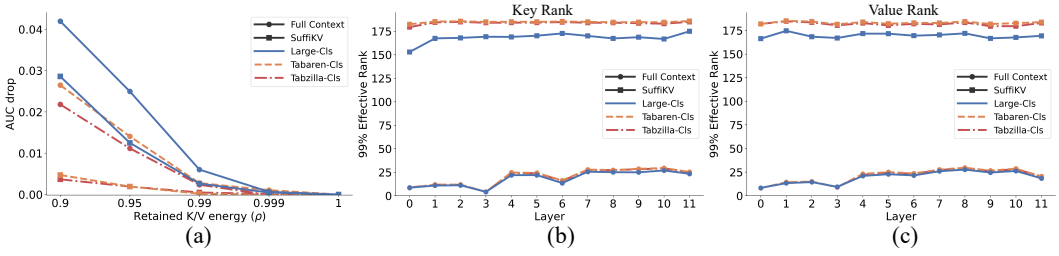

Figure 4: **(a):** AUC drop under different energy thresholds. **(b) and (c):** Layer-wise 99% energy rank for Key and Value of SuffiKV and full-context states. All are based on the LimiX v1 and SuffiKV.

We define the ρ-energy subspace by the smallest rank

$$r_\rho = \min\left\{ r : \frac{\sum_{i=1}^{r} \sigma_i^2}{\sum_{i=1}^{d} \sigma_i^2} \geq \rho \right\}. \tag{13}$$

To calibrate this diagnostic, we project the layer-wise KV states onto their ρ-energy subspace. As shown in Figure 4 (a), this projection with $\rho = 0.99$ causes only a marginal drop in AUC, suggesting that the subspace retained at this threshold captures most of the task-relevant information within the sequence-attention states. Under this calibrated threshold $\rho = 0.99$, Figure 4 (b) and (c) demonstrate that the SuffiKV states occupy a substantially higher-dimensional subspace compared to the full-context states. Averaged across all benchmarks and KV states, the 99% energy rank is 178.6 for SuffiKV, compared to 19.8 for full-context inference. Thus, SuffiKV does not simply compress training rows into a low-rank memory; rather, it forms richer directions for routing task information.

We further compare the two subspaces using principal angles. Let Q_F and Q_S be the ρ-energy bases of full-context and KV states. We compute $Q_F^\top Q_S = L\Gamma R^\top$, where γ_i are the cosines of the principal angles, and treat directions with $\gamma_i^2 \geq \tau$ as shared directions. We use $\tau = 0.7$, with similar conclusions for nearby thresholds. The subspaces are estimated from KV states in the unmodified forward pass; during ablation, we re-run inference and apply the resulting projectors to KV states at each sequence-attention layer. As shown in Figure 5, projecting full-context states onto the shared directions leaves inference nearly unchanged, suggesting that SuffiKV preserves most directions needed by full-context inference. In contrast, applying the same shared-only projection to SuffiKV causes a much larger drop: the mean AUC drop over 3 benchmarks is 0.024 for full-context inference but 0.325 for SuffiKV. This suggests that SuffiKV also learns additional predictive directions beyond those shared with full-context inference.

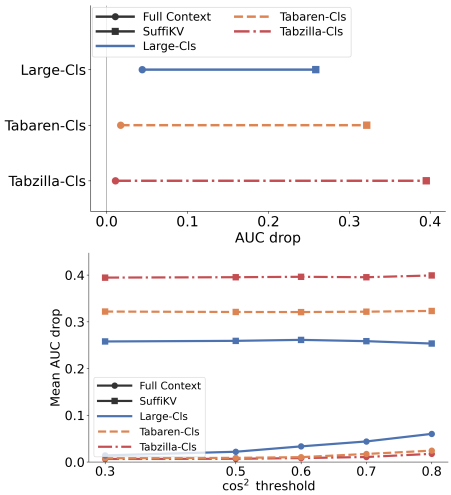

Figure 5: **Top:** Shared-threshold sensitivity on LimiX v1 and SuffiKV. **Bottom:** Shared-direction projection ablation on LimiX v1 and SuffiKV.

Overall, this diagnostic supports the view that SuffiKV acts as an expressive task memory: it retains directions needed by full-context inference while introducing additional predictive directions that improve performance.

5 Conclusion

We propose SuffiKV, a test-time adaptation framework that reformulates in-context learning on tabular data as a compact task-state inference process. Instead of reading the full support table for each query, SuffiKV distills the contextual information into a hierarchical KV state, which is directly applicable to frozen tabular foundation models. Through context-derived KV anchors, bounded low-rank refinement, and anchored operator distillation, SuffiKV learns an optimized read state that preserves task-relevant predictive information while decoupling the amount of evidence used for task inference from the memory required at deployment. On the large-scale benchmark, SuffiKV improves the accuracy–latency–memory trade-off, reducing inference time by 89.9× and peak memory by 66.2%, while concurrently improving AUC by 1.5%. These results indicate that a compact KV state can serve as a robust and expressive task memory rather than merely a lossy summary of the original table. Nevertheless, SuffiKV introduces an additional write-time optimization phase. Furthermore, it predominantly targets task-specific adaptation on frozen backbones. Future research will focus on developing more efficient state construction techniques and exploring joint backbone optimization.

AI USE STATEMENT

In this work, we have not used generative AI tools for any of the tasks that require mandatory disclosure, such as generating synthetic datasets, developing theoretical models, formulating proofs, designing experiments, implementing methods, or interpreting results. We utilized generative AI tools solely for tasks with recommended disclosure: specifically, to correct grammatical errors. We have carefully reviewed all AI-assisted language revisions to ensure that our original scientific meaning, logic, and technical accuracy remain completely unaltered. We take full responsibility for the final content of this work, including all text and claims.

REPRODUCIBILITY STATEMENT

We have included the core source code in the supplementary materials. The appendix provides explicit clarifications of all theoretical assumptions alongside complete mathematical proofs. Additionally, comprehensive details regarding the datasets, baseline methods, and implementation procedures are thoroughly documented in the appendix.

REFERENCES

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.

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. In *NeurIPS 2025-The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Benjamin Feuer, Robin T Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024a.

Benjamin Feuer, Robin Tibor Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. TuneTables: Context optimization for scalable prior-data fitted networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024b. doi: 10.52202/079017-2654. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/97dc07f1253ab33ee514f395a82fa7cc-Abstract-Conference.html.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, Mihir Manium, Rosen Yu, Felix Jablonski, Shi Bin Hoo, Anurag Garg, Jake Robertson, Magnus Bühler, Vladyslav Moroshan, Lennart Purucker, Clara Cornu, Lilly Charlotte Wehrhahn, Alessandro Bonetto, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025. doi: 10.48550/arXiv.2511.08667. URL https://arxiv.org/abs/2511.08667.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. Tabpfn-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

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*, 637:319–326, 2025. doi: 10.1038/s41586-024-08328-6. URL https://doi.org/10.1038/s41586-024-08328-6.

Oroel Ipas, Guillermo Gomez-Trenado, Rocío Romero-Zaliz, and Isaac Triguero. Lucos: Latent unsupervised context selection for tabular foundation models. *arXiv preprint arXiv:2605.27254*, 2026.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with tabpfn. *arXiv preprint arXiv:2402.06971*, 2024.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2410.18164.

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

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

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.05564.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026. doi: 10.48550/arXiv.2602.11139. URL https://arxiv.org/abs/2602.11139.

Renat Sergazinov and Shao-An Yin. Chunked TabPFN: Exact training-free in-context learning for long-context tabular data. *arXiv preprint arXiv:2509.00326*, 2025. doi: 10.48550/arXiv.2509.00326. URL https://arxiv.org/abs/2509.00326.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Derek Xu, Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular pfns. In *International Conference on Learning Representations*, volume 2025, pp. 24630–24665, 2025.

Guri Zabërgja, Rafiq Kamel, Arlind Kadra, Christian MM Frey, and Josif Grabocka. End-to-end compression for tabular foundation models. *arXiv preprint arXiv:2602.05649*, 2026.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, et al. Limix-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026.

A Additional Method Details

A.1 Cross-fitted Task-State Learning

SuffiKV learns a task state from the support table without allowing evaluated labels to leak into the state being tested. During adaptation, we repeatedly split the training table into two disjoint parts,

$$D = C_t \cup B_t, \qquad C_t \cap B_t = \emptyset,$$

where C_t is the *write set* and B_t is the *pseudo-query set*. The write set constructs the task state $M_t = M(C_t)$, while the pseudo-query set is used only to compute the loss. For each $(x, y) \in B_t$, the prediction is made through the memory-only read path

$$p_{M_t}(y \mid x) = f_\theta^{\text{read}}(y \mid x; M_t).$$

The label y is never available to the writer that constructs M_t. Thus, the state must summarize C_t in a way that generalizes to held-out rows in B_t, rather than memorizing the examples on which it is evaluated.

The same split produces full-context teacher signals. For $x \in B_t$, the teacher prediction conditions the frozen model on the write set only,

$$p_{C_t}(y \mid x) = f_\theta(y \mid x; C_t),$$

with the pseudo-query label masked from the context. This yields teacher predictions and layer-wise operator targets without exposing labels from B_t to the write stage. In practice, C_t and B_t are sampled repeatedly, or formed through K-fold cross-fitting, so every training row can serve both as evidence and as a held-out pseudo-query across adaptation steps.

Cross-fitting makes the optimization match the intended deployment problem: a state written from labeled support data must predict unseen queries. It also explains why label-refined training can improve beyond the full-context teacher without collapsing into label memorization. The state receives supervised gradients only through held-out pseudo-queries, while the write path only observes the corresponding write set.

After adaptation, the final state is written from the full available training table,

$$M^\star = M(D_{\text{train}}),$$

and cached. All test examples are then predicted by

$$p(y \mid x_{\text{test}}) = f_\theta^{\text{read}}(y \mid x_{\text{test}}; M^\star),$$

with no access to the raw training table in the query path.

A.2 Optimization and Algorithm

SuffiKV optimizes only the task-state variables; the TFM parameters θ remain frozen throughout. The learnable variables include the write slots, the bounded residual factors, and optional layer/head gates. The optimization follows the schedule in Section 3.3: early steps emphasize full-context teacher guidance and operator distillation, while later steps increase the weight on pseudo-query labels.

We use smaller learning rates for keys than for values, since keys control attention routing and are more sensitive to small changes. The trust-region parameterization in Section 3.2 and the operator constraint in Section 3.3 stabilize optimization. In implementation, we warm up the state with teacher and operator signals before increasing the label-supervision weight.

This algorithm separates adaptation cost from deployment cost. State construction can be performed once per dataset, after which all future predictions use the cached memory.

A.3 Inference, Complexity, and Data-size Scaling

Let n be the number of support rows, m the number of memory slots, q the query batch size, L the number of layers, H the number of heads, and d_h the head dimension. In a full-context read path, each query attends to the support table, giving query-to-context cost

$$O(LHqnd_h).$$

Algorithm 1 SuffiKV Adaptation and Inference

Input: Frozen TFM f_θ, support table D_{train}, memory budget m, adaptation steps T
Output: Cached KV task state $M^\star$

```
Initialize writer slots and trust-region residual parameters.
for t = 1, ..., T do
    Sample disjoint write set C_t and pseudo-query set B_t from D_train.
    Run the frozen TFM on C_t to obtain context activations.
    Write C_t into a context-derived anchor state M_0(C_t).
    Form the current KV task state M_t = M_0(C_t) + ΔM_t.
    Compute full-context teacher predictions and operator targets on B_t.
    Compute memory-only student predictions on B_t.
    Update task-state variables using L^(t)_SuffiKV.
end for
Select the state checkpoint using an internal validation criterion.
Recompute M* = M(D_train) using all training rows.
Cache M*.
Predict future queries using only f^read_θ(· | ·; M*).
return M*
```

For architectures that jointly process support and query rows with full sample attention, the cost can be higher, for example

$$O(LH(n+q)^2 d_h).$$

SuffiKV replaces the n-row read state with m cached KV slots. The memory-only read cost is

$$O(LHqmd_h), \qquad m \ll n,$$

and the cached memory size is

$$O(2LHmd_h).$$

Thus, after the state has been written, inference cost depends on m, not on n. This is the main computational advantage: the amount of data used to infer the task is decoupled from the amount of memory read by each query.

The one-time adaptation cost is amortized over future predictions. For a deployment setting with N_{query} queries, the total cost is

$$\text{Cost}_{\text{write}} + N_{\text{query}}\text{Cost}_{\text{read}}.$$

When many queries are served from the same support table, the write cost becomes negligible relative to repeated full-context inference.

This separation also changes how performance scales with dataset size. In full-context inference, increasing n gives the model more evidence, but also forces every query to attend to a larger raw table. In SuffiKV, larger datasets provide richer evidence for writing the task state, while the read-time memory budget remains fixed. The intended scaling behavior is

$$\text{more data} \Rightarrow \text{better state}, \qquad \text{not} \qquad \text{more data} \Rightarrow \text{longer read path}.$$

This explains why SuffiKV can become increasingly advantageous in long-context regimes: the evidence set grows, but the deployed read state remains compact.

A.4 Why Compact KV States Can Outperform Full Context

A compact task state is not merely a lossy summary of the support table. It is an optimized read state learned from the support table. This distinction is central to SuffiKV.

Full-context inference exposes the frozen model to raw rows at prediction time. The raw table may contain useful evidence, but it is not organized for the frozen model's attention mechanism. It can include redundant rows, noisy labels, imbalanced classes, irrelevant examples, and regions of the feature space that should receive different predictive weight. The full-context path processes all of this through fixed attention without explicitly fitting a task-specific state.

SuffiKV instead transforms the evidence set into a compact KV state optimized for the frozen model to read. The state can use its limited memory budget to represent what matters for prediction rather than what appears as raw rows. In particular, it can merge redundant examples into shared slots, reduce the influence of noisy or irrelevant rows, allocate capacity to rare classes, create slots for decision-boundary regions, encode calibration or class-prior information, and correct full-context teacher errors through label-refined adaptation.

The training schedule is crucial for this behavior. Early teacher and operator signals make the state inherit useful full-context behavior and remain readable by the frozen model. Later pseudo-query labels refine the state toward the supervised task objective. As a result, the full-context path acts as privileged supervision for learning the state, not as an upper bound on performance.

This provides the conceptual reason behind the empirical gains. Full-context inference reads raw evidence repeatedly. SuffiKV writes that evidence once into an optimized task state and then reads the state directly. When the dataset grows, the raw-context path becomes harder to read, while the learned-state path receives more evidence for constructing a better fixed-size memory. This allows SuffiKV to improve both efficiency and predictive performance, especially in large-support regimes.

A.5 What Exactly Does the Compact KV State Learn?

In Section 4.5, we analyzed LimiX v1 and its SuffiKV variant by comparing the principal angles between the keys and values derived from full context sequence attention and the corresponding KV states produced by SuffiKV. In this section, we provide supplementary experimental results and analyses for TabPFN v2.6 and its SuffiKV variant. The empirical observations for TabPFN v2.6 are strictly consistent with those reported for LimiX v1. Under calibrated threshold $\rho = 0.99$, SuffiKV states occupy a substantially higher-dimensional subspace than full-context states, shown as Figure 6 (b) and (c). Averaged over benchmarks, layers, and KV states, the 99% energy rank is 57.2 for SuffiKV and 30.7 for full-context inference. Thus, SuffiKV does not simply compress training rows into a low-rank memory; rather, it forms richer directions for routing task information.

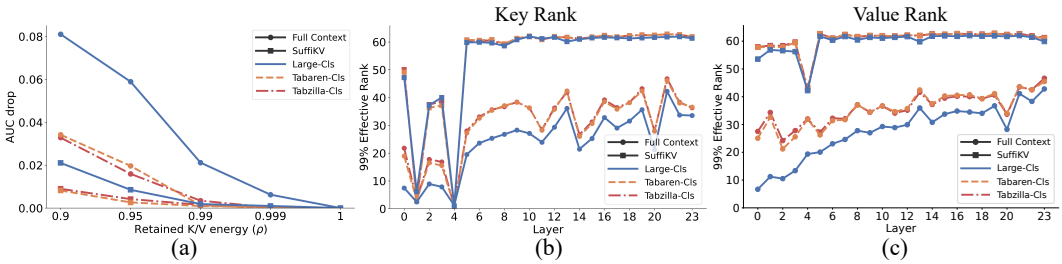

Figure 6: **(a):** AUC drop under different energy thresholds. **(b) and (c):** Layer-wise 99% energy rank for Key and Value of SuffiKV and full-context states. All are based on TabPFN v2.6. and SuffiKV.

Similarly, following the procedure detailed in Section 4.5 to extract the shared directions, we observe that projecting the full context states onto these shared directions yields negligible changes in inference performance. Conversely, applying this identical projection to the SuffiKV states incurs a substantially larger performance degradation. Specifically, as depicted in Figure 7, the average reduction in AUC across the three benchmarks is a mere 0.010 for full context inference, compared to 0.041 for SuffiKV. This indicates that SuffiKV learns additional predictive directions beyond those shared with the full context baseline.

We further investigate the query embeddings, which are intrinsically tied to the KV states. For both the TabArena-Cls and TabZilla-Cls benchmarks, we uniformly sample up to 1,000 test instances without replacement from each dataset, strictly preserving the sequence of the test rows. We conduct standard forward passes using both the full context TFM and the TFM SuffiKV variant to extract their respective query embeddings prior to the decoder, denoted as H^D and H^M. Specifically, we extract the query embeddings immediately preceding the classification or regression decoder for LimiX v1, and just prior to the final output projection layer for TabPFN v2.6. Subsequently, we apply Principal Component Analysis to reduce their dimensionality within a shared representational space. Our observations reveal that the feature distributions of the full context representations and the SuffiKV

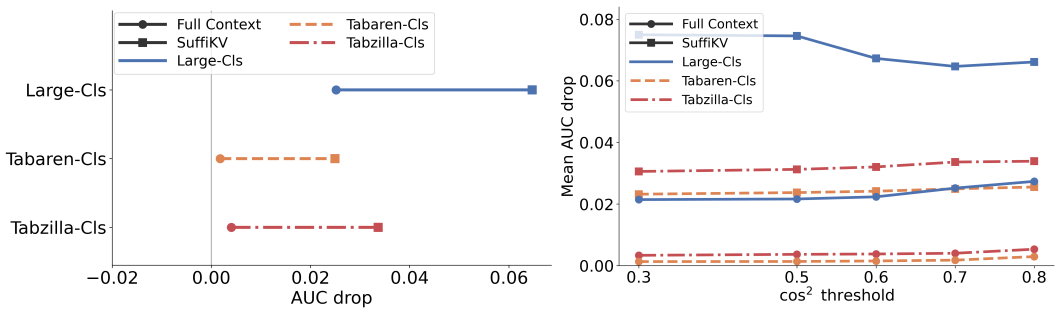

Figure 7: **Left:** Shared-threshold sensitivity on TabPFN v2.6 and SuffiKV variant. **Right:** Shared-direction projection ablation on TabPFN v2.6 and SuffiKV.

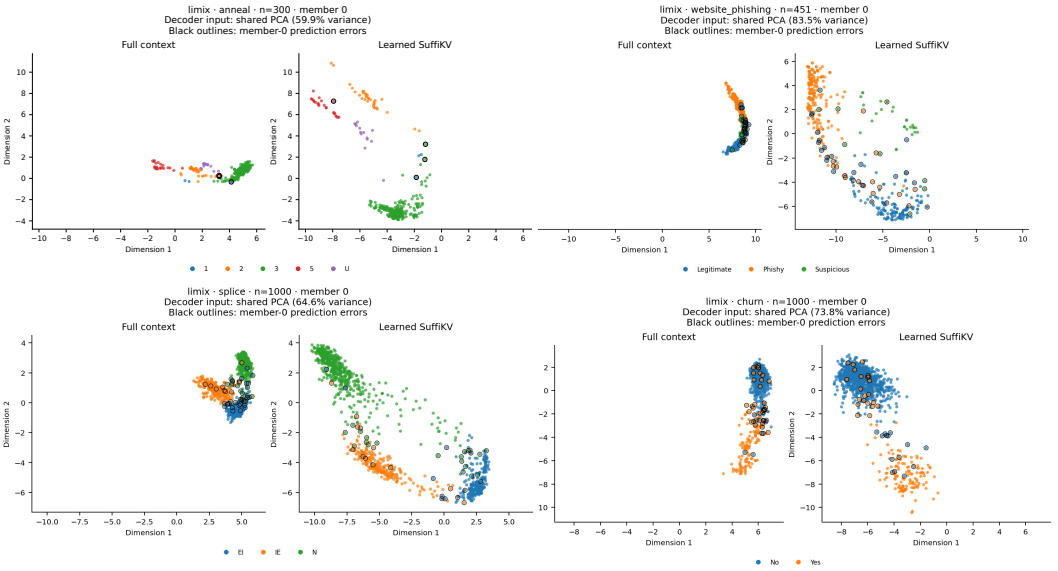

Figure 8: Principal Component Analysis visualization of query embeddings for LimiX v1 on the TabArena-Cls benchmark.

states do not closely align. Notably, after dimensionality reduction, SuffiKV exhibits a pronounced reduction in local class entanglement, a structural advantage that directly corroborates the empirical performance improvements demonstrated by the model. Figure 8, 9, 10, and 11 provides case studies illustrating these Principal Component Analysis projections across various models and benchmarks.

We employ class-proportional stratified sampling to extract 10% of the original training dataset, which serves as the queries to train two independent linear probes. The remaining 90% of the data consistently functions as the full context throughout both the training and testing stages. During the forward passes of both the full context TFMs and the SuffiKV variant, we extract the query embeddings immediately preceding the final linear layer within the decoder, which act as the inputs to the ultimate output projection. These probes are trained using a cross entropy objective optimized via Stochastic Gradient Descent for 300 epochs. Subsequently, we process the test samples in an identical sequence through both the full context and SuffiKV configurations to extract the corresponding query embeddings at this exact network depth. We then evaluate the capacity of the linear probes to predict labels directly from these embeddings. As shown in Table 4, probing results demonstrate that the SuffiKV representations achieve superior AUC and accuracy metrics compared to the full context baseline. This confirms that the label information encoded within the SuffiKV representations possesses enhanced linear decodability.

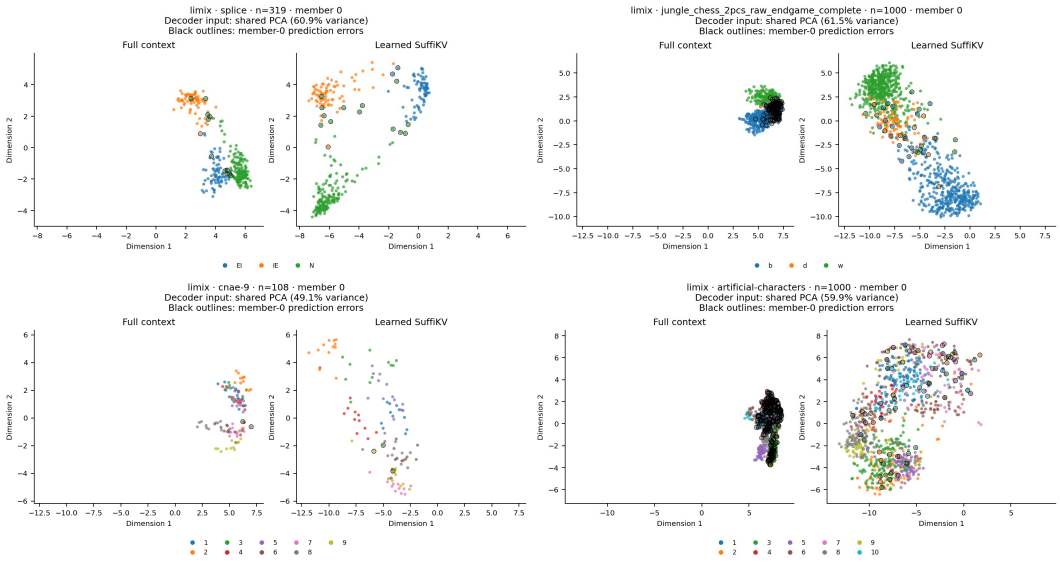

Figure 9: Principal Component Analysis visualization of query embeddings for LimiX v1 on the TabZilla-Cls benchmark.

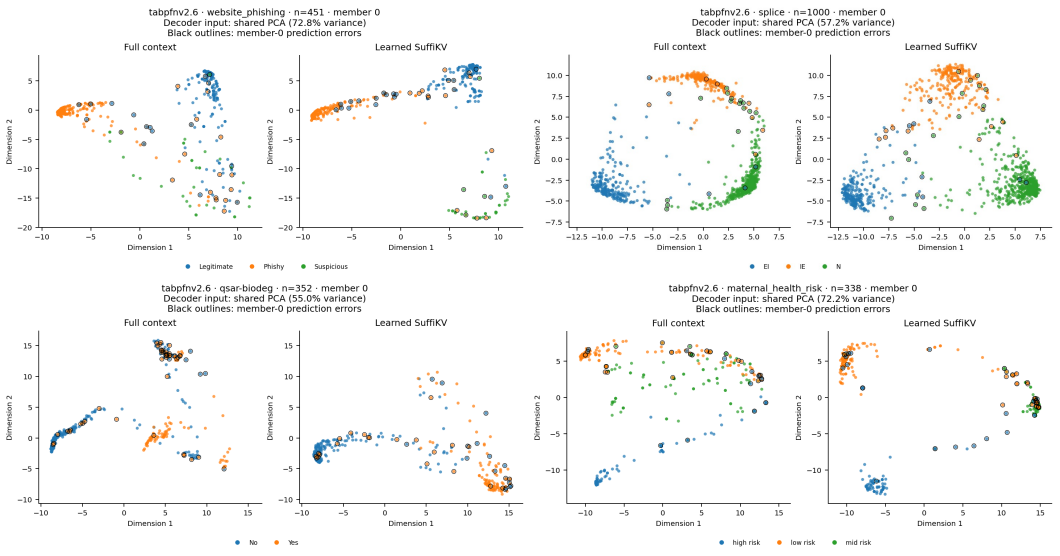

Figure 10: Principal Component Analysis visualization of query embeddings for TabPFN v2.6 on the TabZilla-Cls benchmark.

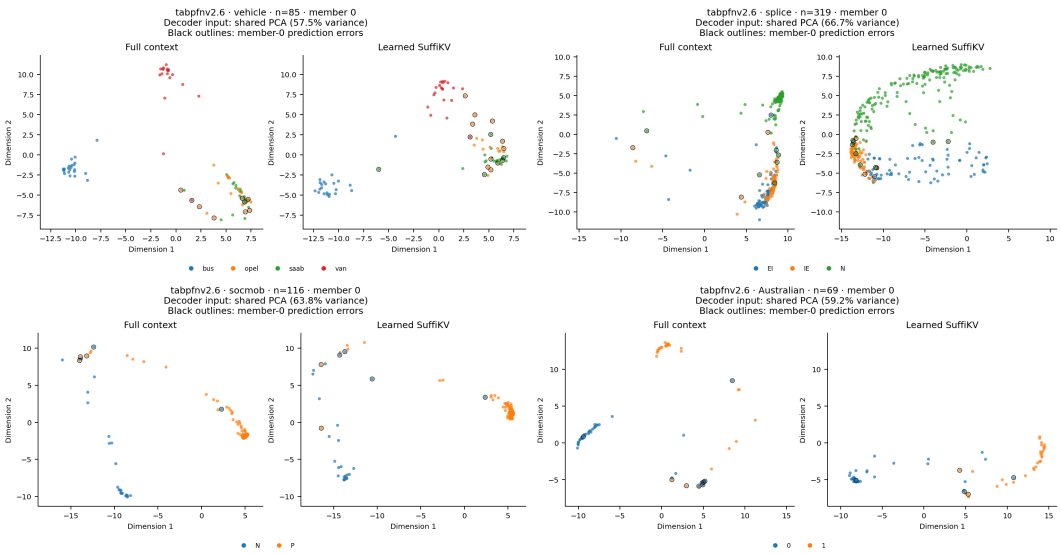

Figure 11: Principal Component Analysis visualization of query embeddings for TabPFN v2.6 on the TabArena-Cls benchmark.

Table 4: Probing results on TabArena-Cls benchmark.

Model	Full-context		SuffiKV	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑
LimiX v1	0.851	0.815	0.855	0.835
TabPFN v2.6	0.857	0.838	0.873	0.838

B More Details on Implementation of SuffiKV

B.1 The effect of the capacity of the KV States

We investigate the impact of the capacity of the task states, m_ℓ, on model performance. Initially, we apply a uniform m_ℓ across all layers, evaluating capacities of 24, 48, 96, 192, and 384 on the TabPFN v2.6 and LimiX v1 models using the TabArena-Cls benchmark. As demonstrated in Table 5, an insufficiently small $m_{\ell,h}$ restricts SuffiKV from retaining effective information, resulting in a marginal performance degradation. Conversely, increasing the capacity effectively enhances overall model performance. Notably, at $m_\ell = 384$, both models achieve higher AUC compared to full-context inference.

Furthermore, we explore layer-specific capacity allocations for LimiX v1 on the TabMacro-Cls benchmark. We evaluate three structural variations: monotonically decreasing, monotonically increasing, and convex. With the variation rate set to 4, the monotonically decreasing m_ℓ sequence is [192, 178, 165, 152, 139, 126, 113, 100, 87, 74, 61, 48]; the monotonically increasing sequence is [48, 61, 74, 87, 100, 113, 126, 139, 152, 165, 178, 192], and the convex configuration is [48, 77, 106, 134, 163, 192, 192, 163, 134, 106, 77, 48]. As shown in Table 6, we observe that a monotonically decreasing allocation risks discarding critical information. Meanwhile, both the increasing and convex configurations suffer from an overly constrained initial capacity m_0, which bottlenecks early-stage information processing. Because the source information is fundamentally impaired, allocating larger capacities in subsequent layers fails to increase the efficacy. Therefore, to strike an optimal trade-off among performance enhancement, inference speedup, and memory footprint reduction, we adopt a uniform capacity across all layers, setting $m_\ell = 192$ for LimiX v1 and $m_\ell = 384$ for TabPFN v2.6.

Table 5: AUC of TabArena-Cls across different m_ℓ capacities.

Model	Full-context TFMs	SuffiKV-$m_\ell = 24$	SuffiKV-$m_\ell = 48$	SuffiKV-$m_\ell = 96$	SuffiKV-$m_\ell = 192$	SuffiKV-$m_\ell = 384$
LimiX v1	0.835	0.846	0.849	0.849	0.851	0.852
TabPFN v2.6	0.849	0.848	0.849	0.851	0.853	0.858

Table 6: AUC of TabMacro-Cls benchmark under different layer-specific capacity allocation schemes with the variation rate set to 4.

Model	Full-context TFMs	SuffiKV-fixed $m_\ell = 192$	SuffiKV-decay $m_0 = 192$	SuffiKV-increase $m_0 = 48$	SuffiKV-convex $m_l \in [48, 192]$
LimiX v1	0.777	0.804	0.799	0.799	0.786

B.2 Ablation Study of SuffiKV's Loss Components

We conduct an ablation study on the proposed loss function (Eq. 8) to evaluate the contribution of each component during KV state optimization. Specifically, we train the KV states on the TabMacro-Cls benchmark using three configurations: label refinement only ($\mathcal{L}_{\text{label}}$), label refinement with task-term anchored distillation, and the full objective including both label refinement, task-term, and operator-term anchored distillation ($\mathcal{L}^{(t)}_{\text{SuffiKV}}$).

As shown in Table 7, each component steadily contributes to the overall performance. Optimizing the KV state solely with label refinement yields an AUC of 0.792, which already significantly outperforms the full-context TFMs (0.777). Incorporating the task-term anchored distillation marginally improves the AUC to 0.794. Most notably, integrating both the task-term and operator-term anchored distillations leads to the most substantial performance leap, achieving a peak AUC of 0.804. This demonstrates that aligning the intermediate operator states during the self-attention process is crucial for learning highly expressive KV states.

Table 7: AUC of TabMacro-Cls benchmark for the loss components' ablation study.

Model	Full-context TFMs	Label refinement $\mathcal{L}_{\text{label}}$	Label refinement + Task-term anchored distillation:$\mathcal{R}^{(t)}_{\text{task}}$	Label refinement + Task-term and Operator-term anchored distillation:$\mathcal{L}^{(t)}_{\text{SuffiKV}}$
LimiX v1	0.777	0.792	0.794	**0.804**

B.3 Specific implementation details of SuffiKV

Initialization of KV State. SuffiKV initializes the KV state from an anchor derived from the context and generated by the frozen model. Specifically, we utilize the frozen Tabular Foundation Model (TFM) to capture the projected K^D_ℓ and V^D_ℓ representations at each layer during the standard forward pass of the full context. We designate these representations as the Teacher K^D_ℓ and V^D_ℓ. We then generate stochastic values based on statistics of the Teacher K^D_ℓ and V^D_ℓ. Initially, we compute the mean and standard deviation of K^D_ℓ and V^D_ℓ along the sample dimension for each layer. Following standardization, we calculate the correlation coefficient $\rho_{f,e}$ as follows:

$$\rho_{f,e} = E_n \left[\frac{K_{n,f,e} - \mu^K_{f,e}}{\sigma^K_{f,e}} \frac{V_{n,f,e} - \mu^V_{f,e}}{\sigma^V_{f,e}} \right]$$

where n, f, and e represent the index of the teacher sample, the feature token, and the embedding dimension, respectively. Subsequently, we sample standard Gaussian variables that maintain this exact correlation. By generating two independent standard normal variables z_K and z_ϵ, we formulate the initialization anchor K^0_ℓ for K^M_ℓ as $K^0_\ell = \mu_K + \sigma_K z_K$. To model the correlated standard normal component for V^0_ℓ, we apply the transformation $z_V = \rho z_K + \sqrt{1-\rho^2} z_\epsilon$. This mathematical formulation allows us to derive the final initialization anchor for V^M_ℓ as $V^0_\ell = \mu_V + \sigma_V z_V$.

Guided by the analysis presented in Appendix B.1, we set the slot capacity m of the KV state at each layer to 192 for LimiX v1 and 384 for TabPFN v2.6. Consequently, the resulting anchor

$M_0(D) = \{(K_\ell^0, V_\ell^0)\}_\ell$ is aligned with the model since it is generated directly by the intrinsic computations of the backbone network. Furthermore, it remains highly compact given that $m_\ell \ll n$, where m equals either 192 or 384.

Refinement of KV State. The initialization anchor is not the final state, so SuffiKV refines it using a trust-region KV parameterization. Section 3.3 describes anchored operator distillation to optimize KV state using early full-context guidance and later held-out label supervision. During each epoch, we randomly partition the full context into multiple chunks. We then divide each chunk into a context split and a test split (held-out pseudo-query) for training, ensuring that the model traverses all chunks within a single epoch. When partitioning the data, we allocate distinct chunk sizes based on dataset scale. An excessively small chunk size may introduce bias into the guidance context, which complicates optimization convergence and prevents the model from reaching an optimal solution. Conversely, an overly large chunk size might degrade model generalization and predispose the optimization trajectory to sharp local optima. Consequently, for datasets where the total number of full context samples n is less than or equal to 10000, we set the chunk size to $\lfloor n/10 \rfloor$. For $10000 < n \leq 50000$, we allocate a chunk size of 2000. For $n > 50000$, we define the chunk size as 5000.

During the initial full context guidance phase, the loss function is primarily steered by the distillation anchors generated by the full context TFMs and their respective query states obtained during the forward pass. In the later stages, the held-out pseudo labels from the test split dominate the supervision process. We regulate this transition using the parameters β_t and λ_{op}^t. We apply a sigmoid decay schedule with a constant phase of w steps to $1 - \beta_t$. Similarly, we employ a cosine decay schedule with a constant phase of w steps for λ_{op}^t. We configure these two parameters as follows:

$$\beta_t = 1 - \sigma\left(-k[\max(t - w, 0) - \tau]\right), \qquad k = 0.05, \quad w = \lfloor T/10 \rfloor, \quad \tau = T/4.$$

$$\lambda_{op}^t = \lambda_{op_{\min}} + \frac{\lambda_{op_{\max}} - \lambda_{op_{\min}}}{2}\left[1 + \cos\left(\pi \,\mathrm{clip}\left(\frac{t - w}{T - w}, 0, 1\right)\right)\right], \quad w = \lfloor T/10 \rfloor.$$

This configuration maintains parameter constancy for the initial w steps, followed by a sigmoid or cosine decay. By adopting this scheduling strategy, we accomplish a seamless transition from imitation to optimization. Throughout the optimization process, we utilize the AdamW optimizer and train for $T = 1000$ epochs across all datasets. For LimiX v1, we assign a learning rate of 1×10^{-4} and a weight decay of 5×10^{-6}. For TabPFN v2.6, we specify a learning rate of 1×10^{-2} and a weight decay of 1×10^{-3}.

When optimizing the KV state, we must account for the fact that each pipeline within the ensemble mechanism of the full context TFMs applies distinct data preprocessing routines. For instance, different pipelines may employ varying feature shuffling permutations and inconsistent label mappings. Consequently, we allocate an individual subsidiary KV state to each pipeline. These individual states are ultimately aggregated to form the complete KV state, thereby materializing the ensemble mechanism of SuffiKV. Furthermore, during both training and inference, all data preprocessing operations strictly adhere to the default parameters and configuration files officially provided by the respective TFMs (LimiX v1 and TabPFN v2.6).

B.4 Implementation of baselines

For random sampling in classification benchmarks, we apply stratified sampling based on the training set classes. We allocate the sample sizes proportionally to the original class distributions while guaranteeing a minimum selection of one instance per class. For regression tasks, we partition the entire training set into a maximum of ten quantile strata based on the ranked target values y. Subsequently, we perform proportional random sampling across these strata to ensure comprehensive coverage of diverse target intervals. This procedure degenerates into single stratum random sampling if all target values are identical. For datasets containing fewer than 30000 context samples, we extract stratified context subsets at ratios of 5%, 10%, 20%, and 50% for inference. For datasets exceeding 30000 context samples, we sample fixed stratified context quantities of 2048, 4096, 8192, and 30000 instances for inference.

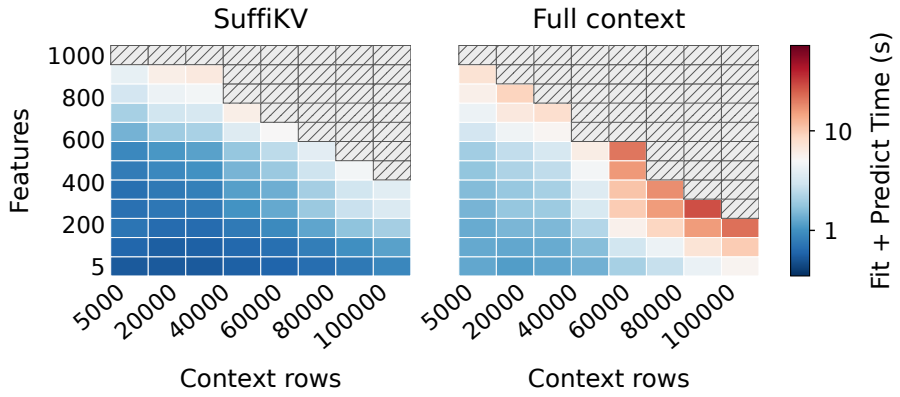

Figure 12: Fit + Predict time heatmap of LimiX v1+SuffiKV and full-context LimiX v1.

Regarding class-balanced subsampling, assuming all categories possess abundant samples, we configure the retained sample size to 5% of the original training set when the total context count n is strictly less than 5000. Under this condition, we distribute the sample quotas equally across all classes. When n exceeds 5000, we fix the retained sample size to either 192 or 384, similarly maintaining an equal allocation among classes. In scenarios where certain categories exhibit sample scarcity, we establish a baseline using the minority class count and randomly sample an equivalent quantity for all other categories.

For KNN based sampling, we cluster the test samples using the KMeans algorithm, allocating a maximum of 256 test instances per group. We then utilize the mean feature vector of each group to retrieve context neighbors. Consequently, all test samples within a given group share these retrieved neighbors as their unified context. Concerning the selection of K, we set K to 5% of the total data volume when the context quantity is below 5000. For context sizes exceeding 5000, we configure K as 192 for the LimiX v1 model and 384 for the TabPFN v2.6 model.

Because the LoCalPFN architecture exclusively supports datasets with fewer than 100 features, we apply Principal Component Analysis to reduce the dimensionality to 100. We fit this transformation solely on the training subset and subsequently apply it to transform the validation and test sets. For regression tasks, we retain the original local context standardization metrics. We adopt default configurations for all other parameters: we approximate K as $\min(\lfloor 10\sqrt{N_{fit}} \rfloor, 1000)$; we conduct training for 21 epochs with 30 steps per epoch; and we assign a validation ratio of 20%.

Following the default configuration of LUCoS, we set the context compression target to $C \times 32$ for classification tasks and 1024 for regression tasks (C is the number of classes), strictly utilizing the Euclidean distance metric. We maintain consistency with the officially provided settings for the remaining parameters: a candidate pool of 10000, 5 repetitions, and an ensemble size of 8, ultimately averaging the final outputs.

For TACO, we retain approximately 4% of the rows, achieving a compression factor of roughly 25. The original publication indicates that this specific ratio yields a favorable balance between performance and efficiency. We set the chunk size to 9500, establish a valid context upper bound of 9000. Although TACO does not natively support regression tasks, we extend its functionality to regression via a bucketing mechanism, constructing up to 10 quantile buckets for the continuous targets.

C More Experiment Results.

C.1 Inference Time and Memory Consumption

Section 4.3 presents the inference latency heatmaps for TabPFN v2.6 and its SuffiKV variant on synthetic datasets; we provide supplementary heatmaps evaluated on the LimiX v1 here. As depicted in Figure 12, although LimiX v1 augmented with SuffiKV still occasionally encounters Out of

Table 8: Average ranks over all classification and regression dataset entries. The classification ranking uses AUC over 89 dataset entries, while the regression ranking uses R^2 over 21 dataset entries. Lower is better.

Classification: AUC ↑

Rank	Method	Avg. Rank ↓
1	TabPFN v2.6 + SuffiKV	4.365
2	LimiX v1 + SuffiKV	4.556
3	TabPFN v2.6	5.152
4	TabPFN v2.6 + Random sampling (50%/30000)	6.309
5	LimiX v1 + Random sampling (50%/30000)	8.000
6	LimiX v1	8.837
7	AutoGluon	9.146
8	TabPFN v2.6 + Random sampling (20%/8192)	10.303
9	TabPFN v2	10.466
10	LoCalPFN	10.635
11	LimiX v1 + Random sampling (20%/8192)	10.955
12	TACO	13.056
13	TabPFN v2.6 + Random sampling (10%/4096)	13.590
14	LimiX v1 + Random sampling (10%/4096)	14.253
15	XGBoost	14.933
16	TabPFN v2.6 + Random sampling (5%/2048)	16.174
17	LimiX v1 + kNN-based sampling	16.551
18	LimiX v1 + Random sampling (5%/2048)	16.635
19	TabPFN v2.6 + kNN-based sampling	17.213
20	LimiX v1 + TuneTables	17.961
21	CatBoost	17.983
22	TabPFN v2.6 + TuneTables	18.028
23	TabPFN v2.6 + Class-balanced sampling	19.096
24	LimiX v1 + Class-balanced sampling	19.938
25	LUCoS	20.865

Regression: R^2 ↑

Rank	Method	Avg. Rank ↓
1	TabPFN v2.6	3.881
2	AutoGluon	3.976
3	TabPFN v2.6 + SuffiKV	4.333
4	LimiX v1 + SuffiKV	4.857
5	TabPFN v2.6 + Random sampling (50%/30000)	7.024
6	CatBoost	7.619
7	LimiX v1 + Random sampling (50%/30000)	7.714
8	LimiX v1	7.810
9	TabPFN v2	9.595
10	LimiX v1 + Random sampling (20%/8192)	11.119
11	TabPFN v2.6 + Random sampling (20%/8192)	11.238
12	XGBoost	11.262
13	TabPFN v2.6 + Random sampling (10%/4096)	13.071
14	LUCoS	13.214
15	LimiX v1 + Random sampling (10%/4096)	13.667
16	TabPFN v2.6 + kNN-based sampling	15.952
17	TabPFN v2.6 + Random sampling (5%/2048)	16.000
18	LimiX v1 + Random sampling (5%/2048)	16.452
19	TACO	17.167
20	LoCalPFN	18.262
21	LimiX v1 + kNN-based sampling	18.310
22	LimiX v1 + TuneTables	21.262
23	TabPFN v2.6 + TuneTables	22.214

Memory (OOM) errors (denoted by the gray hatched regions), its overall failure rate is substantially lower than that of the full context LimiX v1. Furthermore, across all viable grid points where inference completes successfully, SuffiKV achieves significantly reduced inference latency compared to the full context configuration, as evidenced by the predominantly blue areas.

C.2 Method comparison under AUC and R2

We extends the evaluation in Section 4 (Table 1-3) by incorporating average rank metrics alongside AUC and R2. As observed in Table 8, SuffiKV variants constructed on both base models consistently achieve the best average rank on classification tasks. For regression tasks, SuffiKV variants likewise outperform virtually all baseline methods in average ranking.

C.3 More Details on Benchmark

We present detailed information for each dataset in 5 benchmarks, including the number of training samples, the number of test samples, the number of features, and the number of classes.

TABARENA-CLS BENCHMARK

ID	Dataset	Train	Test	Feat.	Cls.
A01	Amazon_employee_access	21,846	10,923	9	2
A02	Bank_Customer_Churn	6,666	3,334	10	2
A03	E-CommereShippingData	7,332	3,667	10	2
A04	Fitness_Club	1,000	500	6	2
A05	Is-this-a-good-customer	1,148	575	13	2
A06	MIC	1,132	567	111	8
A07	Marketing_Campaign	1,493	747	25	2
A08	NATICUSdroid	4,994	2,497	86	2
A09	anneal	598	300	38	5
A10	bank-marketing	30,140	15,071	13	2
A11	blood-transfusion-service-center	498	250	4	2
A12	churn	3,333	1,667	19	2
A13	coil2000_insurance_policies	6,548	3,274	85	2
A14	credit_card_clients_default	20,000	10,000	23	2
A15	diabetes	512	256	8	2
A16	hazelnut-spread-contaminant-detection	1,600	800	30	2
A17	heloc	6,972	3,487	23	2
A18	jm1	7,256	3,629	21	2
A19	maternal_health_risk	676	338	6	3
A20	online_shoppers_intention	8,220	4,110	17	2
A21	polish_companies_bankruptcy	3,940	1,970	64	2
A22	qsar-biodeg	702	352	41	2
A23	seismic-bumps	1,722	862	15	2
A24	splice	2,126	1,064	60	3
A25	students_dropout_and_academic_success	2,949	1,475	36	3
A26	website_phishing	902	451	9	3
A27	Bioresponse	2,500	1,251	1,776	2
A28	HR_Analytics_Job_Change_of_Data$scientists$	12,772	6,386	12	2
A29	credit-g	666	334	20	2
A30	hiva_agnostic	2,563	1,282	1,617	3
A31	in_vehicle_coupon_recommendation	8,456	4,228	24	2
A32	kddcup09_appetency	33,333	16,667	212	2
A32	taiwanese_bankruptcy_prediction	4,546	2,273	94	2

TABZILLA-CLS DATASET INDEX

ID	Dataset	Train	Test	Feat.	Cls.
Z01	Australian	621	69	14	2
Z02	GesturePhaseSegmentationProcessed	8,885	988	32	5
Z03	Bioresponse	3,375	376	1,776	2
Z04	ada_agnostic	4,105	457	48	2
Z05	SpeedDating	7,540	838	120	2
Z06	artificial-characters	9,196	1,022	7	10
Z07	balance-scale	562	63	4	3
Z08	cnae-9	972	108	856	9
Z09	colic	331	37	26	2
Z10	credit-approval	621	69	15	2
Z11	electricity	40,780	4,532	8	2
Z12	elevators	14,939	1,660	18	2
Z13	heart-h	264	30	13	2
Z14	credit-g	900	100	20	2
Z15	jasmine	2,685	299	144	2
Z16	jungle_chess_2pcs_raw_endgame_complete	40,337	4,482	6	3
Z17	kc1	1,898	211	21	2
Z18	mfeat-fourier	1,800	200	76	10
Z19	mfeat-zernike	1,800	200	47	10
Z20	monks-problems-2	540	61	6	2

ID	Dataset	Train	Test	Feat.	Cls.
Z21	nomao	31,018	3,447	118	2
Z22	phoneme	4,863	541	5	2
Z23	profb	604	68	9	2
Z24	qsar-biodeg	949	106	41	2
Z25	socmob	1,040	116	5	2
Z26	splice	2,871	319	60	3
Z27	vehicle	761	85	18	4

TabMacro-Cls dataset index

ID	Dataset	Train	Test	Feat.	Cls.
L01	100000-diabetes-clinical-dataset	80,000	20,000	15	2
L02	90000-cars-data-from-1970-to-2024	78,169	19,543	9	5
L03	Trustpilot-Reviews-123k	98,544	24,637	3	5
L04	airlines-dataset-to-predict-a-delay	431,506	107,877	7	2
L05	airlines-delay	431,505	107,877	7	2
L06	campaign-data	80,000	20,000	11	2
L07	cardio-vascular-disease-detection	55,440	13,861	11	2
L08	chicago-food-inspections	177,174	44,294	15	7
L09	classification-dataset-10-classes	1,200,000	300,000	3	10
L10	clothing-shoes-and-jewellery-reviews	633,600	158,400	1	5
L11	credit-Card-fraud	800,000	200,000	4	2
L12	cuneiform-language-identification	111,536	27,885	1	7
L13	diabetes-clinical-dataset100k-rows	80,000	20,000	16	2
L14	diabetes-prediction-dataset	80,000	20,000	8	2
L15	ecommerce-customer-service-satisfaction	68,725	17,182	18	5
L16	employee-attrition-dataset	47,678	11,920	22	2
L17	enhanced-gait-biomechanics-dataset	142,560	35,640	5	3
L18	expedia-travel-dataset	80,000	20,000	22	2
L19	health-insurance-lead-prediction-raw-data	40,705	10,177	12	2
L20	kickstarter-projects	299,882	74,971	7	5
L21	medium-app-reviews-from-google-play-store	49,012	12,254	8	4
L22	nasa-nearest-earth-objects-1910-2024	270,559	67,640	6	2
L23	numerical-images	57,272	14,318	2	10
L24	nykaa-app-review-sentiment	124,435	31,109	1	3
L25	roadside-noise-level-dataset-with-labels	333,600	83,400	1	2
L26	rsmc-tokyo-typhoon-center-best-track-data	55,000	13,750	14	7
L27	stellar-classification-dataset-sdss17	80,000	20,000	8	3
L28	travel-insurance	50,660	12,666	10	2
L29	web-page-phishing-dataset	80,061	20,016	19	2

TabArena-Reg dataset index

ID	Dataset	Train	Test	Feat.
R01	Another-Dataset-on-used-Fiat-500	1,025	513	7
R02	Food_Delivery_Time	30,300	15,151	9
R03	QSAR_fish_toxicity	604	303	6
R04	airfoil_self_noise	1,002	501	5
R05	concrete_compressive_strength	686	344	8
R06	diamonds	35,960	17,980	9
R07	healthcare_insurance_expenses	892	446	6
R08	houses	13,760	6,880	8
R09	miami_housing	9,184	4,592	15
R10	physiochemical_protein	30,486	15,244	9
R11	superconductivity	14,175	7,088	81
R12	wine_quality	4,331	2,166	12
R13	QSAR-TID-11	3,828	1,914	1024

TABMACRO-REG DATASET INDEX

ID	Dataset	Train	Test	Feat.
G01	Car-Price-Prediction	43,425	10,848	11
G02	advanced-soybean-agricultural-dataset-2025	44,360	11,090	14
G03	atp-tennis-2000-2023daily-pull	51,685	12,861	16
G04	eshop-clothing-dataset	132,379	33,095	13
G05	global-superstore-dataset	41,167	10,123	25
G06	indian-moviesimdb	40,526	10,076	6
G07	spotify-dataset-114k-songs	91,200	22,800	19
G08	spotify-tracks	91,200	22,800	19

C.4 DETAILED RESULTS ACROSS DATASETS

In this section, we present detailed experimental results for all datasets across the evaluated benchmarks.

Table 14: Detailed AUC and ACC results on TabArena-Cls for TFM backbones and adaptation methods. Dataset IDs follow Appendix C.3.

Dataset	LX		LX+TT		LX+SKV		TP2.6		TP2.6+TT		TP2.6+SKV	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
A01	0.847	0.948	0.604	0.939	0.838	0.947	0.849	0.948	0.574	0.549	0.831	0.949
A02	0.875	0.867	0.779	0.550	0.875	0.868	0.878	0.868	0.880	0.870	0.869	0.848
A03	0.744	0.689	0.598	0.516	0.752	0.668	0.742	0.689	0.752	0.745	0.755	0.676
A04	0.807	0.782	0.783	0.626	0.814	0.778	0.809	0.774	0.715	0.286	0.812	0.764
A05	0.680	0.885	0.753	0.812	0.689	0.885	0.732	0.885	0.736	0.604	0.747	0.885
A06	0.646	0.866	0.716	0.450	0.698	0.885	0.670	0.891	0.652	0.889	0.743	0.880
A07	0.926	0.904	0.842	0.772	0.929	0.892	0.936	0.902	0.699	0.788	0.941	0.892
A08	0.987	0.944	0.973	0.873	0.987	0.942	0.986	0.943	0.916	0.833	0.987	0.944
A09	0.999	0.997	0.984	0.850	0.999	0.995	0.999	0.997	0.691	0.614	0.999	0.997
A10	0.775	0.894	0.734	0.379	0.775	0.895	0.774	0.894	0.808	0.698	0.768	0.891
A11	0.728	0.792	0.675	0.752	0.741	0.760	0.728	0.784	0.757	0.909	0.771	0.796
A12	0.934	0.960	0.710	0.855	0.945	0.968	0.940	0.964	0.930	0.849	0.946	0.958
A13	0.767	0.940	0.676	0.713	0.755	0.940	0.762	0.940	0.704	0.640	0.759	0.940
A14	0.790	0.821	0.746	0.719	0.793	0.823	0.793	0.823	0.724	0.666	0.790	0.823
A15	0.837	0.758	0.807	0.723	0.841	0.754	0.838	0.758	0.865	0.829	0.853	0.766
A16	0.988	0.956	0.925	0.851	0.974	0.914	0.990	0.960	0.814	0.642	0.990	0.960
A17	0.800	0.736	0.776	0.709	0.800	0.736	0.800	0.733	0.598	0.042	0.799	0.732
A18	0.761	0.815	0.707	0.764	0.761	0.814	0.772	0.809	0.785	0.823	0.765	0.804
A19	0.963	0.893	0.839	0.675	0.971	0.873	0.965	0.882	0.822	0.385	0.961	0.843
A20	0.931	0.899	0.863	0.867	0.929	0.899	0.933	0.902	0.930	0.896	0.930	0.899
A21	0.973	0.956	0.768	0.708	0.974	0.969	0.987	0.983	0.884	0.807	0.982	0.978
A22	0.935	0.895	0.893	0.798	0.937	0.892	0.926	0.878	0.939	0.406	0.946	0.898
A23	0.793	0.934	0.774	0.647	0.810	0.934	0.795	0.934	0.738	0.716	0.814	0.940
A24	0.996	0.973	0.941	0.790	0.995	0.966	0.997	0.978	0.975	0.865	0.997	0.977
A25	0.871	0.782	0.832	0.613	0.870	0.766	0.872	0.782	0.814	0.511	0.862	0.742
A26	0.977	0.931	0.925	0.807	0.978	0.905	0.977	0.933	0.842	0.800	0.983	0.925
A27	0.879	0.800	0.528	0.536	0.905	0.824	0.889	0.803	0.526	0.535	0.879	0.794
A28	0.811	0.805	0.592	0.751	0.818	0.802	0.814	0.806	0.630	0.751	0.812	0.798
A29	0.788	0.761	0.578	0.698	0.810	0.775	0.785	0.760	0.539	0.701	0.801	0.746
A30	0.245	0.482	0.514	0.965	0.536	0.965	0.484	0.965	0.473	0.965	0.617	0.965
A31	0.799	0.737	0.497	0.535	0.838	0.767	0.852	0.766	0.465	0.566	0.850	0.776
A32	0.757	0.982	0.757	0.982	0.810	0.982	0.807	0.982	0.473	0.982	0.820	0.982
A33	0.937	0.969	0.785	0.966	0.951	0.971	0.948	0.971	0.482	0.968	0.951	0.971

Method abbreviations: LX: LimiX v1; LX+TT: LimiX v1 + TuneTables; LX+SKV: LimiX v1 + SuffiKV; TP2.6: TabPFN v2.6; TP2.6+TT: TabPFN v2.6 + TuneTables; TP2.6+SKV: TabPFN v2.6 + SuffiKV.

Table 15: Detailed AUC and ACC results on TabArena-Cls for standard and context-selection baselines. Dataset IDs follow Appendix C.3.

Dataset	XGB		CatB		AG		TPv2		LoCal		LUCoS		TACO	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
A01	0.710	0.942	0.569	0.942	0.817	0.943	0.669	0.942	0.761	0.945	0.551	0.934	0.669	0.942
A02	0.851	0.851	0.840	0.852	0.863	0.864	0.876	0.868	0.864	0.861	0.778	0.816	0.853	0.855
A03	0.742	0.641	0.737	0.676	0.744	0.676	0.742	0.688	0.751	0.675	0.724	0.633	0.743	0.674
A04	0.777	0.764	0.784	0.762	0.790	0.766	0.812	0.776	0.802	0.770	0.787	0.758	0.803	0.772
A05	0.714	0.883	0.504	0.885	0.610	0.861	0.741	0.885	0.698	0.885	0.609	0.880	0.721	0.880
A06	0.776	0.875	0.746	0.885	0.675	0.854	0.815	0.884	0.815	0.884	0.659	0.882	0.810	0.878
A07	0.871	0.859	0.814	0.869	0.862	0.866	0.926	0.896	0.911	0.886	0.791	0.857	0.920	0.892
A08	0.971	0.922	0.972	0.920	0.985	0.939	0.984	0.937	0.983	0.944	0.967	0.904	0.985	0.942
A09	0.998	0.990	0.759	0.873	0.926	0.987	0.999	0.997	0.993	0.963	0.995	0.983	0.969	0.993
A10	0.751	0.887	0.702	0.884	0.766	0.893	0.767	0.893	0.767	0.892	0.612	0.874	0.754	0.889
A11	0.663	0.760	0.692	0.772	0.661	0.780	0.716	0.792	0.674	0.716	0.699	0.765	0.714	0.780
A12	0.927	0.938	0.901	0.918	0.931	0.958	0.940	0.957	0.921	0.937	0.730	0.867	0.928	0.936
A13	0.715	0.940	0.628	0.940	0.690	0.940	0.753	0.940	0.691	0.940	0.619	0.930	0.735	0.940
A14	0.777	0.807	0.763	0.822	0.790	0.823	0.784	0.823	0.782	0.821	0.700	0.791	0.774	0.816
A15	0.773	0.695	0.774	0.727	0.805	0.723	0.843	0.746	0.844	0.785	0.821	0.748	0.831	0.762
A16	0.894	0.843	0.870	0.789	0.971	0.921	0.986	0.943	0.989	0.956	0.896	0.811	0.970	0.918
A17	0.779	0.709	0.779	0.718	0.784	0.716	0.797	0.734	0.789	0.712	0.722	0.668	0.788	0.734
A18	0.739	0.807	0.709	0.811	0.740	0.813	0.750	0.815	0.765	0.814	0.672	0.802	0.739	0.812
A19	0.895	0.751	0.857	0.686	0.932	0.820	0.959	0.864	0.931	0.775	0.845	0.678	0.891	0.713
A20	0.919	0.893	0.876	0.891	0.916	0.896	0.929	0.901	0.904	0.892	0.872	0.875	0.927	0.896
A21	0.906	0.958	0.920	0.954	0.958	0.963	0.961	0.964	0.938	0.951	0.799	0.918	0.925	0.939
A22	0.867	0.818	0.837	0.793	0.921	0.852	0.921	0.878	0.938	0.884	0.868	0.801	0.932	0.878
A23	0.735	0.934	0.704	0.933	0.782	0.934	0.791	0.934	0.763	0.934	0.685	0.926	0.784	0.934
A24	0.991	0.957	0.889	0.743	0.994	0.960	0.997	0.975	0.973	0.881	0.990	0.932	0.996	0.972
A25	0.851	0.736	0.837	0.740	0.875	0.766	0.884	0.780	0.856	0.757	0.828	0.733	0.875	0.765
A26	0.972	0.918	0.815	0.812	0.972	0.909	0.978	0.920	0.961	0.887	0.840	0.846	0.958	0.851
A27	0.884	0.799	0.886	0.815	0.892	0.808	0.859	0.791	0.859	0.795	0.718	0.652	0.861	0.770
A28	0.791	0.782	0.812	0.801	0.813	0.799	0.805	0.801	0.778	0.775	0.678	0.757	0.793	0.769
A29	0.767	0.737	0.782	0.751	0.780	0.772	0.769	0.743	0.720	0.710	0.696	0.709	0.769	0.781
A30	0.495	0.963	0.468	0.965	0.524	0.965	0.527	0.965	0.520	0.965	0.512	0.963	0.513	0.965
A31	0.841	0.770	0.834	0.765	0.844	0.771	0.799	0.735	0.771	0.714	0.588	0.576	0.777	0.725
A32	0.787	0.982	0.834	0.982	0.829	0.982	0.777	0.982	0.801	0.982	0.595	0.982	0.794	0.982
A33	0.942	0.971	0.948	0.972	0.942	0.972	0.942	0.971	0.933	0.970	0.896	0.959	0.944	0.970

Method abbreviations: XGB: XGBoost; CatB: CatBoost; AG: AutoGluon; TPv2: TabPFN v2; LoCal: LoCalPFN; LUCoS: LUCoS; TACO: TACO.

Table 16: Detailed AUC and ACC results on TabArena-Cls for TabPFN v2.6 sampling. Dataset IDs follow Appendix C.3.

TabPFN v2.6 sampling

Dataset	TP-R5		TP-R10		TP-R20		TP-R50		TP-kNN		TP-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
A01	0.625	0.942	0.719	0.942	0.787	0.945	0.854	0.949	0.705	0.938	0.611	0.568
A02	0.863	0.861	0.873	0.867	0.878	0.868	0.879	0.868	0.791	0.829	0.826	0.688
A03	0.743	0.689	0.741	0.686	0.742	0.688	0.742	0.689	0.711	0.643	0.736	0.672
A04	0.797	0.774	0.799	0.768	0.793	0.770	0.808	0.778	0.766	0.766	0.772	0.772
A05	0.608	0.885	0.680	0.885	0.733	0.885	0.732	0.883	0.678	0.875	0.564	0.583
A06	0.735	0.857	0.778	0.880	0.827	0.885	0.861	0.889	0.690	0.866	0.652	0.427
A07	0.777	0.825	0.875	0.874	0.826	0.779	0.900	0.898	0.841	0.873	0.859	0.720
A08	0.978	0.930	0.981	0.938	0.982	0.933	0.984	0.939	0.985	0.944	0.978	0.918
A09	0.929	0.863	0.957	0.933	0.978	0.973	0.994	0.997	0.965	0.990	0.975	0.737
A10	0.752	0.893	0.765	0.894	0.768	0.894	0.775	0.894	0.684	0.886	0.722	0.663
A11	0.643	0.708	0.640	0.764	0.718	0.784	0.723	0.792	0.689	0.792	0.684	0.500
A12	0.900	0.912	0.902	0.893	0.921	0.931	0.937	0.951	0.927	0.956	0.896	0.830
A13	0.749	0.941	0.755	0.940	0.762	0.940	0.760	0.940	0.607	0.939	0.735	0.682
A14	0.780	0.821	0.784	0.821	0.788	0.822	0.793	0.823	0.706	0.802	0.743	0.732
A15	0.735	0.672	0.813	0.746	0.803	0.738	0.821	0.766	0.824	0.781	0.636	0.590
A16	0.907	0.829	0.938	0.863	0.950	0.889	0.984	0.948	0.978	0.910	0.905	0.813
A17	0.793	0.736	0.799	0.736	0.800	0.733	0.800	0.733	0.766	0.698	0.768	0.699
A18	0.745	0.815	0.770	0.816	0.782	0.819	0.782	0.819	0.707	0.813	0.708	0.671
A19	0.751	0.612	0.849	0.686	0.833	0.701	0.897	0.787	0.955	0.864	0.741	0.559
A20	0.922	0.896	0.929	0.901	0.933	0.903	0.933	0.902	0.887	0.883	0.917	0.816
A21	0.879	0.940	0.960	0.961	0.971	0.970	0.984	0.980	0.954	0.963	0.928	0.840
A22	0.843	0.801	0.876	0.804	0.889	0.838	0.915	0.858	0.916	0.878	0.887	0.795
A23	0.739	0.934	0.798	0.934	0.788	0.934	0.776	0.934	0.746	0.935	0.781	0.643
A24	0.992	0.943	0.994	0.958	0.996	0.973	0.997	0.974	0.995	0.962	0.988	0.928
A25	0.847	0.727	0.854	0.740	0.867	0.764	0.878	0.765	0.877	0.781	0.832	0.664
A26	0.831	0.805	0.881	0.874	0.908	0.845	0.974	0.896	0.970	0.918	0.859	0.705
A27	0.721	0.679	0.800	0.712	0.837	0.758	0.872	0.800	0.754	0.695	0.776	0.703
A28	0.803	0.800	0.813	0.803	0.813	0.803	0.814	0.807	0.775	0.774	0.785	0.771
A29	0.715	0.704	0.698	0.689	0.760	0.743	0.756	0.731	0.559	0.695	0.706	0.599
A30	0.531	0.965	0.556	0.965	0.543	0.965	0.479	0.965	0.480	0.899	0.490	0.346
A31	0.779	0.722	0.821	0.754	0.857	0.784	0.859	0.786	0.728	0.682	0.714	0.663
A32	0.784	0.982	0.806	0.982	0.815	0.982	0.821	0.982	0.573	0.977	0.779	0.687
A33	0.919	0.967	0.919	0.968	0.930	0.972	0.944	0.971	0.857	0.967	0.934	0.852

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 17: Detailed AUC and ACC results on TabArena-Cls for LimiX v1 sampling. Dataset IDs follow Appendix C.3.

	LimiX v1 sampling											
Dataset	LX-R5		LX-R10		LX-R20		LX-R50		LX-kNN		LX-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
A01	0.656	0.942	0.729	0.943	0.789	0.945	0.851	0.948	0.709	0.941	0.634	0.569
A02	0.867	0.864	0.872	0.865	0.877	0.868	0.877	0.868	0.796	0.824	0.821	0.712
A03	0.743	0.690	0.741	0.687	0.738	0.689	0.738	0.689	0.730	0.666	0.740	0.674
A04	0.743	0.766	0.780	0.742	0.787	0.766	0.809	0.784	0.760	0.740	0.710	0.744
A05	0.589	0.885	0.604	0.885	0.670	0.885	0.698	0.885	0.632	0.882	0.564	0.605
A06	0.698	0.836	0.778	0.861	0.767	0.857	0.792	0.864	0.679	0.852	0.655	0.471
A07	0.809	0.858	0.885	0.880	0.882	0.878	0.910	0.900	0.892	0.881	0.854	0.764
A08	0.978	0.928	0.979	0.931	0.981	0.933	0.985	0.938	0.986	0.948	0.978	0.920
A09	0.949	0.863	0.988	0.953	0.990	0.967	0.998	0.997	0.964	0.987	0.971	0.727
A10	0.755	0.891	0.765	0.894	0.771	0.894	0.774	0.894	0.712	0.887	0.742	0.699
A11	0.568	0.668	0.611	0.732	0.730	0.772	0.720	0.792	0.678	0.788	0.655	0.564
A12	0.893	0.893	0.880	0.872	0.907	0.900	0.924	0.944	0.924	0.949	0.906	0.839
A13	0.749	0.941	0.763	0.940	0.769	0.940	0.769	0.940	0.641	0.940	0.727	0.673
A14	0.777	0.821	0.784	0.822	0.787	0.821	0.790	0.820	0.707	0.804	0.734	0.708
A15	0.775	0.727	0.828	0.762	0.800	0.742	0.825	0.746	0.828	0.789	0.644	0.609
A16	0.909	0.830	0.941	0.874	0.963	0.899	0.984	0.943	0.982	0.933	0.909	0.813
A17	0.790	0.726	0.799	0.731	0.801	0.733	0.801	0.733	0.758	0.692	0.764	0.694
A18	0.745	0.818	0.759	0.816	0.765	0.818	0.765	0.818	0.715	0.810	0.702	0.724
A19	0.751	0.601	0.851	0.725	0.824	0.678	0.899	0.790	0.958	0.861	0.712	0.562
A20	0.920	0.893	0.926	0.898	0.932	0.900	0.931	0.901	0.888	0.889	0.914	0.831
A21	0.829	0.932	0.896	0.938	0.919	0.939	0.952	0.951	0.919	0.948	0.924	0.869
A22	0.869	0.807	0.877	0.804	0.891	0.841	0.930	0.889	0.918	0.872	0.909	0.827
A23	0.742	0.934	0.796	0.932	0.782	0.934	0.753	0.932	0.750	0.932	0.759	0.664
A24	0.985	0.916	0.991	0.943	0.994	0.966	0.995	0.965	0.992	0.957	0.983	0.914
A25	0.852	0.732	0.858	0.750	0.869	0.761	0.878	0.761	0.878	0.774	0.822	0.671
A26	0.850	0.843	0.868	0.851	0.904	0.856	0.975	0.889	0.970	0.896	0.875	0.789
A27	0.743	0.686	0.793	0.699	0.841	0.752	0.873	0.801	0.733	0.654	0.754	0.695
A28	0.805	0.797	0.814	0.800	0.816	0.805	0.817	0.806	0.789	0.789	0.775	0.774
A29	0.725	0.725	0.681	0.704	0.767	0.760	0.779	0.749	0.663	0.713	0.559	0.695
A30	0.531	0.965	0.539	0.965	0.564	0.965	0.526	0.965	0.472	0.964	0.480	0.899
A31	0.768	0.711	0.795	0.732	0.811	0.746	0.811	0.746	0.720	0.673	0.728	0.682
A32	0.757	0.982	0.757	0.982	0.757	0.982	0.757	0.982	0.606	0.982	0.573	0.977
A33	0.920	0.971	0.932	0.970	0.927	0.971	0.949	0.969	0.863	0.967	0.857	0.967

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 18: Detailed AUC and ACC results on TabZilla-Cls for TFM backbones and adaptation methods. Dataset IDs follow Appendix C.3.

Dataset	LX		LX+TT		LX+SKV		TP2.6		TP2.6+TT		TP2.6+SKV	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
Z01	0.942	0.884	0.971	0.913	0.964	0.913	0.958	0.870	0.958	0.826	0.979	0.913
Z02	0.945	0.770	0.709	0.429	0.926	0.735	0.964	0.817	0.623	0.221	0.934	0.761
Z04	0.910	0.869	0.868	0.790	0.907	0.871	0.896	0.856	0.829	0.302	0.905	0.858
Z06	0.975	0.736	0.614	0.147	0.998	0.953	0.995	0.911	0.593	0.157	0.998	0.957
Z07	1.000	1.000	0.998	0.968	0.999	0.984	1.000	1.000	0.982	0.905	1.000	1.000
Z08	0.999	0.954	0.908	0.481	0.996	0.949	0.999	0.963	0.999	0.963	1.000	0.981
Z09	0.920	0.784	0.968	0.892	0.962	0.838	0.888	0.784	0.944	0.784	0.913	0.784
Z10	0.926	0.855	0.951	0.870	0.938	0.870	0.931	0.913	0.967	0.913	0.951	0.928
Z11	0.981	0.926	0.789	0.425	0.982	0.936	0.987	0.943	0.789	0.711	0.963	0.896
Z12	0.942	0.896	0.874	0.781	0.945	0.899	0.947	0.900	0.743	0.688	0.938	0.895
Z13	0.828	0.767	0.866	0.767	0.876	0.800	0.847	0.767	0.000	0.000	0.900	0.767
Z15	0.880	0.839	0.863	0.789	0.898	0.843	0.884	0.833	0.845	0.769	0.902	0.829
Z16	0.974	0.880	0.825	0.697	1.000	0.995	0.985	0.897	0.783	0.506	1.000	0.998
Z17	0.877	0.882	0.823	0.152	0.872	0.896	0.860	0.872	0.791	0.758	0.852	0.858
Z18	0.988	0.855	0.875	0.140	0.990	0.840	0.990	0.850	0.810	0.325	0.993	0.880
Z19	0.981	0.840	0.801	0.345	0.980	0.765	0.983	0.835	0.768	0.215	0.996	0.930
Z20	1.000	1.000	1.000	0.984	1.000	0.984	1.000	1.000	0.996	0.967	1.000	1.000
Z21	0.994	0.946	0.991	0.957	0.995	0.966	0.997	0.973	0.997	0.973	0.996	0.970
Z22	0.980	0.930	0.805	0.713	0.974	0.926	0.980	0.930	0.838	0.590	0.968	0.906
Z23	0.784	0.721	0.776	0.691	0.773	0.779	0.685	0.662	0.696	0.618	0.752	0.676
Z24	0.952	0.868	0.963	0.830	0.966	0.906	0.944	0.877	0.962	0.896	0.973	0.887
Z25	0.992	0.966	0.986	0.948	0.995	0.966	0.991	0.957	0.989	0.940	0.998	0.974
Z26	0.997	0.966	0.969	0.853	0.997	0.957	0.996	0.972	0.919	0.708	0.997	0.972
Z27	0.969	0.835	0.963	0.765	0.955	0.835	0.975	0.871	0.909	0.635	0.973	0.847
Z03	0.879	0.800	0.574	0.543	0.905	0.824	0.889	0.801	0.494	0.543	0.910	0.832
Z05	0.850	0.852	0.483	0.835	0.882	0.888	0.887	0.889	0.513	0.835	0.887	0.879
Z14	0.844	0.773	0.436	0.700	0.871	0.740	0.846	0.770	0.641	0.700	0.870	0.750

Method abbreviations: LX: LimiX v1; LX+TT: LimiX v1 + TuneTables; LX+SKV: LimiX v1 + SuffiKV; TP2.6: TabPFN v2.6; TP2.6+TT: TabPFN v2.6 + TuneTables; TP2.6+SKV: TabPFN v2.6 + SuffiKV.

Table 19: Detailed AUC and ACC results on TabZilla-Cls for standard and context-selection baselines. Dataset IDs follow Appendix C.3.

Dataset	XGB		CatB		AG		TPv2		LoCal		LUCoS		TACO	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
Z01	0.932	0.870	0.938	0.855	0.954	0.870	0.947	0.870	0.963	0.913	0.932	0.852	0.953	0.870
Z02	0.809	0.529	0.756	0.487	0.920	0.721	0.922	0.705	0.921	0.726	0.717	0.478	0.860	0.618
Z04	0.868	0.814	0.873	0.840	0.899	0.858	0.902	0.851	0.890	0.840	0.840	0.811	0.890	0.851
Z06	0.941	0.607	0.888	0.429	0.997	0.947	0.978	0.748	0.980	0.798	0.943	0.575	0.946	0.573
Z07	0.920	0.889	0.892	0.857	1.000	1.000	1.000	1.000	1.000	1.000	0.981	0.927	1.000	0.984
Z08	0.992	0.880	0.841	0.528	0.997	0.926	0.999	0.944	0.999	0.944	0.995	0.926	0.999	0.944
Z09	0.813	0.730	0.861	0.838	0.846	0.757	0.865	0.784	0.804	0.784	0.844	0.784	0.817	0.703
Z10	0.923	0.841	0.938	0.855	0.938	0.928	0.925	0.899	0.899	0.899	0.928	0.867	0.926	0.899
Z11	0.882	0.795	0.860	0.784	0.987	0.945	0.970	0.902	0.972	0.913	0.790	0.735	0.900	0.816
Z12	0.854	0.818	0.837	0.811	0.942	0.897	0.940	0.893	0.940	0.898	0.890	0.858	0.914	0.878
Z13	0.801	0.700	0.778	0.800	0.849	0.667	0.828	0.733	0.775	0.733	0.839	0.760	0.828	0.733
Z15	0.848	0.799	0.859	0.803	0.862	0.803	0.852	0.816	0.852	0.816	0.815	0.742	0.862	0.816
Z16	0.921	0.791	0.873	0.749	0.999	0.972	0.978	0.873	0.995	0.955	0.795	0.684	0.919	0.786
Z17	0.801	0.853	0.780	0.867	0.769	0.872	0.824	0.882	0.842	0.877	0.763	0.844	0.830	0.863
Z18	0.962	0.775	0.926	0.680	0.981	0.830	0.988	0.850	0.986	0.835	0.987	0.842	0.980	0.780
Z19	0.959	0.735	0.908	0.505	0.978	0.835	0.985	0.845	0.990	0.870	0.985	0.855	0.976	0.805
Z20	0.990	0.934	0.948	0.852	1.000	1.000	0.996	0.951	0.999	0.967	0.611	0.669	0.902	0.803
Z21	0.981	0.941	0.980	0.935	0.996	0.975	0.993	0.963	0.993	0.963	0.959	0.884	0.991	0.951
Z22	0.903	0.802	0.881	0.800	0.967	0.915	0.967	0.904	0.970	0.911	0.819	0.763	0.944	0.878
Z23	0.614	0.750	0.625	0.662	0.664	0.662	0.686	0.676	0.598	0.632	0.615	0.665	0.695	0.676
Z24	0.908	0.849	0.898	0.811	0.956	0.868	0.941	0.877	0.935	0.858	0.885	0.813	0.954	0.858
Z25	0.978	0.948	0.973	0.940	0.987	0.957	0.985	0.931	0.985	0.940	0.980	0.931	0.981	0.948
Z26	0.992	0.931	0.879	0.730	0.993	0.937	0.995	0.962	0.977	0.878	0.985	0.908	0.993	0.953
Z27	0.906	0.718	0.909	0.741	0.954	0.812	0.972	0.847	0.972	0.894	0.949	0.772	0.945	0.800
Z03	0.862	0.803	0.870	0.806	0.878	0.803	0.847	0.769	0.863	0.798	0.635	0.584	0.867	0.798
Z05	0.879	0.881	0.873	0.874	0.869	0.876	0.859	0.866	0.857	0.874	0.725	0.838	0.861	0.875
Z14	0.863	0.830	0.859	0.810	0.847	0.790	0.830	0.760	0.827	0.760	0.721	0.706	0.849	0.790

Method abbreviations: XGB: XGBoost; CatB: CatBoost; AG: AutoGluon; TPv2: TabPFN v2; LoCal: LoCalPFN; LUCoS: LUCoS; TACO: TACO.

Table 20: Detailed AUC and ACC results on TabZilla-Cls for TabPFN v2.6 sampling. Dataset IDs follow Appendix C.3.

	TabPFN v2.6 sampling											
Dataset	TP-R5		TP-R10		TP-R20		TP-R50		TP-kNN		TP-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
Z01	0.941	0.884	0.942	0.841	0.928	0.870	0.951	0.870	0.947	0.899	0.930	0.826
Z02	0.874	0.645	0.927	0.736	0.960	0.811	0.965	0.816	0.709	0.465	0.763	0.451
Z04	0.874	0.838	0.892	0.849	0.886	0.851	0.893	0.847	0.895	0.858	0.870	0.746
Z06	0.967	0.677	0.975	0.744	0.992	0.886	0.995	0.923	0.733	0.320	0.928	0.504
Z07	0.916	0.905	0.836	0.889	0.979	0.873	0.998	0.984	0.973	0.921	0.925	0.778
Z08	0.946	0.731	0.985	0.889	0.993	0.898	0.999	0.954	0.997	0.944	0.936	0.694
Z09	0.821	0.676	0.795	0.730	0.862	0.757	0.926	0.811	0.827	0.703	0.737	0.622
Z10	0.917	0.870	0.926	0.899	0.922	0.855	0.912	0.855	0.921	0.884	0.932	0.870
Z11	0.920	0.845	0.942	0.869	0.959	0.887	0.983	0.935	0.843	0.761	0.851	0.769
Z12	0.936	0.896	0.940	0.899	0.942	0.899	0.947	0.901	0.855	0.826	0.906	0.857
Z13	0.766	0.800	0.770	0.767	0.847	0.800	0.847	0.767	0.754	0.700	0.766	0.733
Z15	0.846	0.796	0.836	0.773	0.852	0.799	0.857	0.826	0.868	0.803	0.842	0.779
Z16	0.960	0.838	0.966	0.849	0.971	0.856	0.984	0.899	0.862	0.732	0.885	0.711
Z17	0.766	0.844	0.805	0.834	0.810	0.853	0.849	0.848	0.862	0.858	0.837	0.735
Z18	0.966	0.715	0.984	0.860	0.985	0.810	0.987	0.830	0.986	0.825	0.980	0.810
Z19	0.969	0.735	0.978	0.795	0.980	0.805	0.986	0.875	0.985	0.835	0.962	0.695
Z20	0.781	0.639	0.817	0.721	0.563	0.672	0.973	0.918	0.988	0.934	0.651	0.623
Z21	0.991	0.957	0.993	0.963	0.993	0.966	0.996	0.971	0.947	0.904	0.974	0.910
Z22	0.892	0.828	0.912	0.835	0.939	0.882	0.968	0.915	0.975	0.922	0.888	0.774
Z23	0.635	0.618	0.607	0.618	0.709	0.735	0.699	0.676	0.613	0.662	0.477	0.441
Z24	0.888	0.830	0.869	0.802	0.939	0.887	0.941	0.887	0.923	0.858	0.851	0.726
Z25	0.976	0.922	0.973	0.940	0.979	0.948	0.986	0.948	0.987	0.940	0.971	0.879
Z26	0.992	0.931	0.991	0.931	0.993	0.953	0.995	0.956	0.996	0.947	0.986	0.912
Z27	0.905	0.753	0.920	0.718	0.952	0.788	0.965	0.812	0.954	0.835	0.905	0.694
Z03	0.804	0.726	0.789	0.721	0.830	0.739	0.852	0.777	0.756	0.665	0.747	0.662
Z05	0.853	0.868	0.872	0.866	0.888	0.889	0.888	0.889	0.726	0.837	0.830	0.696
Z14	0.750	0.750	0.778	0.700	0.731	0.700	0.829	0.720	0.663	0.670	0.805	0.730

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 21: Detailed AUC and ACC results on TabZilla-Cls for LimiX v1 sampling. Dataset IDs follow Appendix C.3.

Dataset	*LimiX v1 sampling*											
	LX-R5		LX-R10		LX-R20		LX-R50		LX-kNN		LX-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
Z01	0.944	0.884	0.921	0.855	0.919	0.884	0.952	0.884	0.947	0.913	0.918	0.870
Z02	0.872	0.633	0.912	0.700	0.938	0.748	0.942	0.766	0.702	0.465	0.766	0.464
Z04	0.880	0.836	0.889	0.845	0.892	0.856	0.899	0.864	0.901	0.853	0.868	0.759
Z06	0.969	0.683	0.979	0.763	0.980	0.777	0.981	0.786	0.758	0.333	0.930	0.542
Z07	0.858	0.889	0.825	0.873	0.982	0.889	1.000	0.984	0.932	0.873	0.772	0.603
Z08	0.910	0.750	0.980	0.852	0.992	0.898	0.999	0.954	0.989	0.907	0.911	0.685
Z09	0.833	0.784	0.849	0.730	0.821	0.703	0.926	0.784	0.843	0.703	0.702	0.622
Z10	0.926	0.841	0.902	0.812	0.914	0.855	0.909	0.855	0.926	0.899	0.915	0.855
Z11	0.928	0.858	0.949	0.878	0.964	0.896	0.976	0.918	0.850	0.772	0.853	0.773
Z12	0.936	0.894	0.939	0.899	0.942	0.896	0.941	0.895	0.888	0.847	0.913	0.863
Z13	0.761	0.733	0.756	0.767	0.871	0.800	0.837	0.800	0.856	0.733	0.766	0.700
Z15	0.839	0.776	0.833	0.779	0.853	0.813	0.857	0.816	0.868	0.806	0.846	0.796
Z16	0.960	0.842	0.965	0.852	0.975	0.876	0.982	0.895	0.882	0.752	0.883	0.725
Z17	0.790	0.872	0.771	0.829	0.815	0.863	0.856	0.839	0.872	0.872	0.845	0.768
Z18	0.965	0.735	0.984	0.835	0.981	0.795	0.987	0.875	0.987	0.825	0.980	0.830
Z19	0.961	0.715	0.981	0.810	0.982	0.835	0.983	0.820	0.979	0.795	0.961	0.745
Z20	0.513	0.607	0.748	0.705	0.888	0.738	0.990	0.902	1.000	1.000	0.615	0.525
Z21	0.990	0.956	0.993	0.965	0.994	0.966	0.996	0.971	0.946	0.914	0.973	0.907
Z22	0.902	0.817	0.913	0.830	0.939	0.891	0.967	0.906	0.979	0.928	0.897	0.789
Z23	0.588	0.618	0.589	0.529	0.693	0.662	0.712	0.632	0.611	0.603	0.520	0.529
Z24	0.890	0.858	0.888	0.774	0.928	0.849	0.944	0.877	0.920	0.877	0.874	0.774
Z25	0.973	0.940	0.974	0.948	0.986	0.940	0.984	0.957	0.988	0.948	0.960	0.871
Z26	0.991	0.922	0.988	0.928	0.993	0.944	0.995	0.953	0.994	0.937	0.985	0.915
Z27	0.875	0.671	0.920	0.765	0.949	0.847	0.963	0.824	0.944	0.812	0.848	0.624
Z03	0.822	0.763	0.810	0.745	0.851	0.753	0.853	0.763	0.707	0.609	0.757	0.689
Z05	0.853	0.872	0.872	0.870	0.877	0.877	0.877	0.877	0.775	0.848	0.841	0.734
Z14	0.784	0.750	0.753	0.730	0.732	0.730	0.842	0.750	0.765	0.700	0.706	0.650

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 22: Detailed AUC and ACC results on TabMacro-Cls for TFM backbones and adaptation methods. Dataset IDs follow Appendix C.3.

Dataset	LX		LX+TT		LX+SKV		TP2.6		TP2.6+TT		TP2.6+SKV	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
L01	0.980	0.972	0.956	0.932	0.980	0.972	0.980	0.972	0.925	0.919	0.979	0.971
L02	0.944	0.992	0.869	0.846	0.955	0.994	0.958	0.994	0.721	0.398	0.958	0.989
L03	0.520	0.261	0.518	0.228	0.550	0.230	0.583	0.262	0.516	0.215	0.595	0.263
L04	0.685	0.624	0.668	0.634	0.725	0.671	0.717	0.665	0.625	0.596	0.712	0.663
L05	0.681	0.620	0.663	0.628	0.726	0.672	0.721	0.667	0.651	0.612	0.719	0.664
L06	0.498	0.494	0.512	0.504	0.514	0.510	0.505	0.503	0.516	0.501	0.523	0.505
L07	0.798	0.730	0.790	0.714	0.802	0.735	0.801	0.734	0.769	0.714	0.800	0.733
L08	0.886	0.773	0.711	0.211	0.919	0.843	0.931	0.834	0.638	0.128	0.929	0.795
L09	0.960	0.663	0.813	0.100	0.974	0.737	0.962	0.677	0.925	0.417	0.968	0.700
L10	0.834	0.537	0.726	0.438	0.859	0.582	0.831	0.555	0.660	0.343	0.822	0.548
L11	0.720	0.913	0.734	0.913	0.735	0.913	0.735	0.913	0.733	0.245	0.766	0.558
L12	0.768	0.578	0.614	0.326	0.858	0.663	0.806	0.601	0.557	0.296	0.809	0.913
L13	0.978	0.971	0.943	0.920	0.979	0.971	0.979	0.971	0.912	0.915	0.978	0.968
L14	0.979	0.971	0.902	0.889	0.979	0.971	0.979	0.971	0.926	0.805	0.978	0.966
L15	0.608	0.706	0.523	0.531	0.620	0.710	0.622	0.708	0.544	0.615	0.616	0.696
L16	0.848	0.759	0.812	0.732	0.849	0.762	0.849	0.760	0.767	0.661	0.847	0.757
L17	0.498	0.333	0.504	0.333	0.504	0.333	0.508	0.339	0.507	0.335	0.514	0.342
L18	0.717	0.920	0.697	0.665	0.750	0.920	0.748	0.920	0.674	0.920	0.741	0.920
L19	0.652	0.760	0.563	0.748	0.680	0.760	0.674	0.759	0.550	0.739	0.675	0.760
L20	0.839	0.885	0.717	0.627	0.879	0.886	0.883	0.890	0.723	0.522	0.888	0.888
L21	0.822	0.660	0.739	0.530	0.837	0.671	0.823	0.654	0.686	0.547	0.820	0.655
L22	0.870	0.872	0.858	0.855	1.000	0.996	0.979	0.943	0.861	0.831	0.981	0.944
L23	0.500	0.110	0.505	0.101	0.505	0.110	0.499	0.108	0.505	0.098	0.505	0.100
L24	0.684	0.710	0.566	0.705	0.781	0.736	0.745	0.723	0.549	0.710	0.759	0.728
L25	0.637	0.963	0.626	0.467	0.675	0.963	0.673	0.963	0.667	0.963	0.674	0.963
L26	0.997	0.942	0.917	0.489	0.994	0.936	0.999	0.976	0.819	0.318	0.999	0.976
L27	0.822	0.726	0.673	0.292	0.856	0.770	0.858	0.772	0.633	0.225	0.859	0.785
L28	0.839	0.985	0.773	0.985	0.855	0.985	0.848	0.985	0.825	0.977	0.857	0.985
L29	0.963	0.897	0.943	0.869	0.965	0.900	0.965	0.900	0.932	0.858	0.963	0.897

Method abbreviations: LX: LimiX v1; LX+TT: LimiX v1 + TuneTables; LX+SKV: LimiX v1 + SuffiKV; TP2.6: TabPFN v2.6; TP2.6+TT: TabPFN v2.6 + TuneTables; TP2.6+SKV: TabPFN v2.6 + SuffiKV.

Table 23: Detailed AUC and ACC results on TabMacro-Cls for standard and context-selection baselines. Dataset IDs follow Appendix C.3.

Dataset	XGB		CatB		AG		TPv2		LoCal		LUCoS		TACO	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
L01	0.970	0.972	0.948	0.972	0.977	0.972	0.978	0.972	0.977	0.972	0.910	0.939	0.972	0.972
L02	0.940	0.952	0.945	0.903	0.919	0.993	0.983	0.981	0.953	0.991	0.763	0.935	0.987	0.960
L03	0.526	0.253	0.524	0.259	0.525	0.261	0.517	0.261	0.606	0.276	0.507	0.240	0.520	0.261
L04	0.683	0.635	0.668	0.633	0.722	0.668	0.674	0.635	0.710	0.660	0.556	0.557	0.678	0.638
L05	0.684	0.634	0.671	0.633	0.726	0.671	0.673	0.637	0.713	0.661	0.544	0.555	0.676	0.639
L06	0.503	0.507	0.494	0.497	0.498	0.498	0.495	0.501	0.498	0.502	0.498	0.499	0.501	0.503
L07	0.792	0.729	0.788	0.728	0.798	0.733	0.797	0.735	0.799	0.733	0.749	0.689	0.764	0.719
L08	0.913	0.820	0.886	0.785	0.962	0.839	0.951	0.813	0.894	0.795	0.839	0.758	0.902	0.707
L09	0.959	0.674	0.937	0.626	0.974	0.742	0.961	0.672	0.966	0.686	0.958	0.663	0.958	0.660
L10	0.817	0.551	0.772	0.527	0.876	0.612	0.819	0.548	0.879	0.609	0.629	0.333	0.819	0.548
L11	0.735	0.913	0.734	0.913	0.735	0.913	0.733	0.913	0.734	0.913	0.640	0.913	0.733	0.913
L12	0.780	0.568	0.678	0.460	0.925	0.776	0.793	0.566	0.902	0.735	0.598	0.426	0.680	0.449
L13	0.970	0.971	0.942	0.971	0.975	0.970	0.976	0.971	0.976	0.970	0.903	0.936	0.975	0.971
L14	0.967	0.971	0.952	0.971	0.976	0.971	0.977	0.971	0.976	0.971	0.923	0.943	0.975	0.971
L15	0.620	0.702	0.579	0.694	0.647	0.708	0.519	0.371	0.592	0.696	0.526	0.692	0.586	0.691
L16	0.801	0.718	0.805	0.721	0.842	0.751	0.843	0.754	0.829	0.741	0.702	0.636	0.827	0.738
L17	0.447	0.291	0.477	0.316	0.490	0.328	0.500	0.334	0.508	0.338	0.499	0.333	0.498	0.333
L18	0.729	0.920	0.703	0.920	0.723	0.920	0.727	0.920	0.733	0.920	0.554	0.919	0.711	0.920
L19	0.623	0.760	0.556	0.760	0.648	0.760	0.646	0.758	0.614	0.760	0.500	0.760	0.578	0.760
L20	0.870	0.883	0.832	0.843	0.907	0.891	0.824	0.886	0.882	0.884	0.670	0.859	0.869	0.874
L21	0.801	0.612	0.784	0.606	0.839	0.662	0.616	0.404	0.800	0.628	0.741	0.574	0.796	0.605
L22	0.892	0.878	0.863	0.874	1.000	0.998	0.855	0.826	1.000	0.998	0.820	0.869	0.869	0.875
L23	0.501	0.108	0.502	0.109	0.501	0.108	0.498	0.111	0.500	0.099	0.501	0.099	0.499	0.111
L24	0.700	0.717	0.619	0.710	0.774	0.742	0.705	0.717	0.795	0.741	0.530	0.710	0.703	0.717
L25	0.673	0.963	0.663	0.963	0.666	0.963	0.667	0.963	0.668	0.963	0.580	0.963	0.666	0.963
L26	0.994	0.934	0.900	0.924	0.998	0.978	0.996	0.936	0.988	0.974	0.914	0.920	0.954	0.889
L27	0.761	0.696	0.711	0.660	0.869	0.766	0.799	0.711	0.845	0.759	0.624	0.584	0.740	0.670
L28	0.843	0.985	0.807	0.985	0.846	0.985	0.815	0.985	0.840	0.985	0.637	0.981	0.833	0.985
L29	0.952	0.873	0.941	0.871	0.962	0.897	0.961	0.892	0.964	0.899	0.926	0.853	0.954	0.880

Method abbreviations: XGB: XGBoost; CatB: CatBoost; AG: AutoGluon; TPv2: TabPFN v2; LoCal: LoCalPFN; LUCoS: LUCoS; TACO: TACO.

Table 24: Detailed AUC and ACC results on TabMacro-Cls for TabPFN v2.6 sampling. Dataset IDs follow Appendix C.3.

TabPFN v2.6 sampling

Dataset	TP-R5		TP-R10		TP-R20		TP-R50		TP-kNN		TP-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
L01	0.978	0.972	0.978	0.972	0.979	0.972	0.980	0.972	0.899	0.951	0.969	0.911
L02	0.979	0.980	0.986	0.985	0.983	0.989	0.994	0.992	0.773	0.882	0.752	0.436
L03	0.507	0.243	0.520	0.256	0.530	0.260	0.558	0.260	0.515	0.225	0.499	0.180
L04	0.678	0.634	0.682	0.641	0.686	0.643	0.699	0.652	0.656	0.622	0.631	0.603
L05	0.671	0.633	0.685	0.643	0.690	0.647	0.698	0.651	0.659	0.624	0.613	0.573
L06	0.505	0.504	0.497	0.498	0.501	0.499	0.501	0.501	0.497	0.499	0.501	0.502
L07	0.794	0.729	0.795	0.732	0.798	0.732	0.800	0.733	0.779	0.720	0.778	0.721
L08	0.950	0.809	0.951	0.817	0.955	0.820	0.957	0.828	0.853	0.761	0.918	0.671
L09	0.961	0.672	0.961	0.674	0.962	0.676	0.962	0.676	0.966	0.705	0.955	0.654
L10	0.805	0.535	0.814	0.545	0.821	0.547	0.827	0.552	0.836	0.558	0.770	0.463
L11	0.733	0.913	0.734	0.913	0.733	0.913	0.734	0.913	0.733	0.913	0.732	0.511
L12	0.773	0.559	0.791	0.575	0.802	0.583	0.822	0.597	0.736	0.510	0.605	0.236
L13	0.976	0.971	0.976	0.971	0.979	0.971	0.979	0.971	0.863	0.943	0.969	0.893
L14	0.976	0.971	0.978	0.971	0.978	0.971	0.978	0.971	0.937	0.954	0.972	0.916
L15	0.632	0.696	0.622	0.700	0.654	0.702	0.662	0.706	0.558	0.693	0.528	0.203
L16	0.839	0.748	0.845	0.754	0.847	0.758	0.849	0.761	0.763	0.692	0.773	0.697
L17	0.503	0.337	0.500	0.334	0.498	0.331	0.506	0.337	0.465	0.306	0.504	0.337
L18	0.723	0.920	0.721	0.920	0.728	0.920	0.745	0.920	0.588	0.916	0.706	0.334
L19	0.603	0.760	0.635	0.760	0.643	0.760	0.671	0.759	0.549	0.757	0.540	0.556
L20	0.871	0.882	0.874	0.886	0.885	0.887	0.892	0.889	0.785	0.839	0.848	0.557
L21	0.819	0.631	0.827	0.631	0.834	0.642	0.839	0.651	0.758	0.563	0.753	0.547
L22	0.868	0.876	0.886	0.876	0.931	0.911	0.973	0.942	0.889	0.875	0.853	0.739
L23	0.502	0.110	0.498	0.109	0.499	0.111	0.504	0.111	0.501	0.101	0.499	0.097
L24	0.692	0.719	0.709	0.719	0.725	0.721	0.733	0.723	0.676	0.688	0.621	0.380
L25	0.669	0.963	0.664	0.963	0.666	0.963	0.670	0.963	0.653	0.963	0.659	0.712
L26	0.928	0.931	0.930	0.936	0.995	0.944	0.999	0.969	0.914	0.921	0.994	0.923
L27	0.752	0.685	0.784	0.698	0.821	0.719	0.863	0.755	0.744	0.655	0.674	0.500
L28	0.828	0.985	0.827	0.985	0.838	0.985	0.848	0.985	0.621	0.968	0.839	0.726
L29	0.955	0.878	0.959	0.890	0.960	0.890	0.963	0.896	0.934	0.872	0.944	0.867

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 25: Detailed AUC and ACC results on TabMacro-Cls for LimiX v1 sampling. Dataset IDs follow Appendix C.3.

Dataset	*LimiX v1 sampling*											
	LX-R5		LX-R10		LX-R20		LX-R50		LX-kNN		LX-CB	
	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑	AUC ↑	ACC ↑
L01	0.977	0.972	0.977	0.972	0.979	0.972	0.980	0.972	0.910	0.949	0.968	0.901
L02	0.973	0.980	0.978	0.985	0.982	0.989	0.994	0.991	0.774	0.888	0.813	0.530
L03	0.502	0.261	0.509	0.261	0.505	0.261	0.505	0.261	0.516	0.219	0.511	0.189
L04	0.679	0.636	0.684	0.641	0.687	0.644	0.688	0.644	0.649	0.616	0.635	0.603
L05	0.673	0.637	0.684	0.641	0.688	0.645	0.689	0.645	0.651	0.619	0.601	0.570
L06	0.500	0.505	0.506	0.509	0.503	0.502	0.498	0.499	0.496	0.498	0.500	0.503
L07	0.796	0.730	0.796	0.731	0.799	0.733	0.799	0.733	0.772	0.714	0.782	0.720
L08	0.943	0.804	0.945	0.808	0.949	0.817	0.953	0.821	0.846	0.723	0.922	0.673
L09	0.960	0.667	0.961	0.671	0.961	0.672	0.961	0.673	0.967	0.711	0.953	0.636
L10	0.812	0.539	0.825	0.550	0.828	0.550	0.829	0.547	0.845	0.572	0.768	0.464
L11	0.733	0.913	0.734	0.913	0.734	0.913	0.734	0.913	0.733	0.913	0.732	0.545
L12	0.783	0.578	0.800	0.586	0.800	0.588	0.807	0.582	0.762	0.531	0.662	0.338
L13	0.973	0.971	0.974	0.971	0.977	0.971	0.979	0.971	0.895	0.944	0.957	0.873
L14	0.973	0.970	0.977	0.971	0.977	0.971	0.978	0.971	0.947	0.959	0.967	0.912
L15	0.633	0.703	0.625	0.707	0.646	0.704	0.654	0.708	0.557	0.696	0.541	0.294
L16	0.840	0.748	0.844	0.752	0.846	0.756	0.848	0.761	0.757	0.685	0.749	0.680
L17	0.502	0.335	0.497	0.328	0.498	0.333	0.499	0.332	0.478	0.316	0.502	0.336
L18	0.718	0.920	0.730	0.920	0.722	0.920	0.700	0.920	0.633	0.920	0.684	0.347
L19	0.614	0.760	0.634	0.760	0.642	0.760	0.650	0.760	0.551	0.757	0.525	0.566
L20	0.877	0.883	0.867	0.886	0.889	0.886	0.884	0.886	0.787	0.849	0.848	0.546
L21	0.829	0.639	0.836	0.643	0.844	0.657	0.848	0.659	0.760	0.569	0.765	0.560
L22	0.875	0.880	0.889	0.878	0.898	0.883	0.893	0.881	0.890	0.880	0.856	0.737
L23	0.500	0.103	0.498	0.110	0.499	0.110	0.499	0.108	0.501	0.101	0.499	0.109
L24	0.702	0.719	0.722	0.725	0.730	0.720	0.736	0.720	0.700	0.704	0.631	0.374
L25	0.668	0.963	0.666	0.963	0.669	0.963	0.669	0.963	0.655	0.963	0.659	0.738
L26	0.930	0.933	0.950	0.937	0.996	0.941	0.997	0.945	0.914	0.923	0.994	0.924
L27	0.785	0.701	0.807	0.715	0.826	0.724	0.845	0.734	0.745	0.654	0.671	0.471
L28	0.814	0.985	0.817	0.985	0.831	0.985	0.844	0.985	0.747	0.985	0.835	0.729
L29	0.956	0.885	0.958	0.890	0.960	0.889	0.963	0.896	0.939	0.874	0.944	0.873

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 26: Detailed R^2 and RMSE results on TabArena-Reg for TFM backbones and adaptation methods. Dataset IDs follow Appendix C.3.

Dataset	LX		LX+TT		LX+SKV		TP2.6		TP2.6+TT		TP2.6+SKV	
	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓	R^2 ↑	RMSE ↓
R01	0.865	0.358	0.816	0.418	0.870	0.352	0.869	0.353	0.802	0.433	0.863	0.361
R02	0.334	0.821	0.131	0.937	0.369	0.799	0.382	0.791	0.131	0.937	0.386	0.791
R03	0.633	0.634	0.514	0.731	0.662	0.609	0.652	0.618	0.514	0.731	0.642	0.627
R04	0.960	0.201	0.180	0.916	0.978	0.150	0.983	0.133	0.180	0.916	0.984	0.133
R05	0.945	0.240	0.636	0.619	0.952	0.224	0.950	0.230	0.636	0.619	0.948	0.232
R06	0.981	0.141	0.242	0.890	0.984	0.131	0.981	0.131	0.242	0.890	0.979	0.149
R07	0.873	0.332	0.740	0.475	0.878	0.326	0.877	0.326	0.740	0.475	0.875	0.329
R08	0.877	0.357	0.246	0.882	0.873	0.362	0.881	0.348	0.246	0.882	0.886	0.348
R09	0.940	0.250	0.243	0.890	0.941	0.248	0.942	0.246	0.243	0.890	0.929	0.273
R10	0.709	0.538	0.106	0.944	0.683	0.562	0.747	0.502	0.091	0.952	0.747	0.502
R11	0.921	0.279	0.455	0.731	0.924	0.273	0.927	0.267	0.456	0.731	0.929	0.263
R12	0.518	0.671	0.064	0.934	0.487	0.692	0.530	0.662	0.064	0.935	0.508	0.679
R13	0.600	0.670	0.013	0.952	0.600	0.682	0.705	0.568	0.134	0.974	0.755	0.478

Method abbreviations: LX: LimiX v1; LX+TT: LimiX v1 + TuneTables; LX+SKV: LimiX v1 + SuffiKV; TP2.6: TabPFN v2.6; TP2.6+TT: TabPFN v2.6 + TuneTables; TP2.6+SKV: TabPFN v2.6 + SuffiKV.

Table 27: Detailed R^2 and RMSE results on TabArena-Reg for standard and context-selection baselines. Dataset IDs follow Appendix C.3.

Dataset	XGB		CatB		AG		TPv2		LoCal		LUCoS		TACO	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
R01	0.829	0.403	0.858	0.367	0.867	0.354	0.868	0.354	0.527	0.575	0.871	0.349	0.860	0.365
R02	0.208	0.895	0.377	0.794	0.377	0.794	0.374	0.795	0.192	0.864	0.293	0.845	0.285	0.850
R03	0.571	0.686	0.619	0.647	0.635	0.633	0.632	0.636	0.376	0.799	0.643	0.626	0.598	0.664
R04	0.945	0.237	0.953	0.218	0.973	0.167	0.977	0.154	0.941	0.232	0.981	0.140	0.856	0.384
R05	0.920	0.290	0.931	0.269	0.943	0.245	0.943	0.245	0.851	0.351	0.948	0.234	0.900	0.324
R06	0.993	0.084	0.995	0.070	0.994	0.074	0.982	0.136	0.337	0.909	0.975	0.161	0.943	0.244
R07	0.816	0.400	0.859	0.350	0.874	0.331	0.824	0.390	0.416	0.600	0.874	0.331	0.828	0.387
R08	0.836	0.411	0.849	0.395	0.872	0.363	0.873	0.362	0.740	0.472	0.823	0.427	0.793	0.461
R09	0.910	0.307	0.935	0.261	0.940	0.251	0.881	0.353	0.815	0.354	0.915	0.297	0.773	0.487
R10	0.596	0.635	0.599	0.633	0.729	0.520	0.673	0.571	0.633	0.574	0.474	0.724	0.416	0.763
R11	0.915	0.288	0.917	0.286	0.928	0.266	0.924	0.274	0.753	0.433	0.825	0.414	0.859	0.372
R12	0.416	0.738	0.444	0.720	0.523	0.667	0.378	0.762	0.318	0.782	0.348	0.780	0.398	0.750
R13	0.681	0.591	0.699	0.573	0.718	0.555	0.653	0.616	0.584	0.691	0.626	0.639	0.537	0.712

Method abbreviations: XGB: XGBoost; CatB: CatBoost; AG: AutoGluon; TPv2: TabPFN v2; LoCal: LoCalPFN; LUCoS: LUCoS; TACO: TACO.

Table 28: Detailed R^2 and RMSE results on TabArena-Reg for TabPFN v2.6 sampling. Dataset IDs follow Appendix C.3.

	TabPFN v2.6 sampling									
Dataset	TP-R5		TP-R10		TP-R20		TP-R50		TP-kNN	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
R01	0.840	0.390	0.850	0.377	0.853	0.374	0.862	0.361	0.862	0.362
R02	0.351	0.810	0.359	0.805	0.367	0.800	0.383	0.790	0.257	0.867
R03	0.419	0.799	0.522	0.724	0.558	0.697	0.603	0.660	0.603	0.660
R04	0.805	0.447	0.905	0.312	0.911	0.301	0.950	0.226	0.978	0.149
R05	0.751	0.512	0.827	0.426	0.887	0.345	0.908	0.311	0.929	0.273
R06	0.978	0.150	0.979	0.148	0.980	0.143	0.984	0.131	0.948	0.234
R07	0.836	0.378	0.837	0.376	0.857	0.352	0.872	0.333	0.863	0.345
R08	0.851	0.391	0.862	0.378	0.875	0.359	0.883	0.348	0.754	0.504
R09	0.921	0.287	0.937	0.258	0.941	0.247	0.941	0.248	0.863	0.378
R10	0.557	0.665	0.604	0.628	0.658	0.584	0.745	0.504	0.504	0.703
R11	0.877	0.347	0.891	0.327	0.915	0.289	0.927	0.268	0.757	0.489
R12	0.290	0.814	0.342	0.784	0.365	0.770	0.430	0.729	0.507	0.679
R13	0.384	0.821	0.457	0.770	0.571	0.685	0.664	0.606	0.097	0.994

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 29: Detailed R^2 and RMSE results on TabArena-Reg for LimiX v1 sampling. Dataset IDs follow Appendix C.3.

LimiX v1 sampling

Dataset	LX-R5		LX-R10		LX-R20		LX-R50		LX-kNN	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
R01	0.840	0.389	0.856	0.369	0.856	0.370	0.863	0.360	0.824	0.408
R02	0.321	0.828	0.337	0.819	0.344	0.814	0.352	0.809	0.226	0.884
R03	0.444	0.781	0.566	0.691	0.578	0.680	0.615	0.650	0.583	0.677
R04	0.626	0.618	0.817	0.432	0.887	0.340	0.931	0.266	0.949	0.228
R05	0.681	0.579	0.831	0.421	0.892	0.337	0.910	0.307	0.878	0.358
R06	0.978	0.152	0.979	0.150	0.980	0.145	0.982	0.138	0.940	0.249
R07	0.778	0.439	0.836	0.378	0.859	0.351	0.874	0.330	0.807	0.409
R08	0.855	0.387	0.868	0.369	0.880	0.351	0.884	0.345	0.726	0.531
R09	0.923	0.284	0.937	0.257	0.945	0.241	0.945	0.240	0.859	0.384
R10	0.566	0.658	0.605	0.627	0.660	0.583	0.736	0.513	0.457	0.736
R11	0.879	0.345	0.895	0.321	0.914	0.291	0.923	0.274	0.679	0.562
R12	0.290	0.814	0.334	0.788	0.376	0.763	0.456	0.712	0.474	0.701
R13	0.381	0.823	0.466	0.764	0.577	0.680	0.652	0.616	0.233	0.916

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 30: Detailed R^2 and RMSE results on TabMacro-Reg for TFM backbones and adaptation methods. Dataset IDs follow Appendix C.3.

Dataset	LX		LX+TT		LX+SKV		TP2.6		TP2.6+TT		TP2.6+SKV	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
G01	0.247	0.868	0.013	0.994	0.272	0.853	0.283	0.847	0.081	0.959	0.282	0.848
G02	0.996	0.062	0.288	0.843	0.999	0.033	1.000	0.008	0.233	0.875	1.000	0.008
G03	0.762	0.488	0.053	0.973	0.916	0.289	0.785	0.464	0.026	0.987	0.780	0.463
G04	1.000	0.015	0.060	0.969	1.000	0.014	0.997	0.058	0.403	0.773	1.000	0.013
G05	0.318	0.831	0.031	0.990	0.360	0.805	0.407	0.775	0.003	1.004	0.407	0.775
G06	0.269	0.855	0.758	0.492	0.277	0.850	0.268	0.855	0.061	0.969	0.266	0.856
G07	0.293	0.841	0.024	0.988	0.360	0.800	0.294	0.840	0.027	0.986	0.296	0.839
G08	0.286	0.845	0.021	0.989	0.356	0.803	0.295	0.840	0.028	0.986	0.294	0.838

Method abbreviations: LX: LimiX v1; LX+TT: LimiX v1 + TuneTables; LX+SKV: LimiX v1 + SuffiKV; TP2.6: TabPFN v2.6; TP2.6+TT: TabPFN v2.6 + TuneTables; TP2.6+SKV: TabPFN v2.6 + SuffiKV.

Table 31: Detailed R^2 and RMSE results on TabMacro-Reg for standard and context-selection baselines. Dataset IDs follow Appendix C.3.

Dataset	XGB		CatB		AG		TPv2		LoCal		LUCoS		TACO	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
G01	0.265	0.857	0.279	0.849	0.293	0.841	0.260	0.860	0.105	0.914	0.226	0.880	0.234	0.875
G02	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.009	0.988	0.090	1.000	0.016	0.988	0.109
G03	0.776	0.473	0.794	0.454	0.841	0.399	0.668	0.576	0.516	0.596	0.659	0.584	0.542	0.676
G04	1.000	0.000	1.000	0.001	1.000	0.000	0.998	0.045	0.984	0.090	0.976	0.154	0.912	0.296
G05	0.306	0.838	0.339	0.818	0.424	0.764	0.352	0.809	0.311	0.840	0.279	0.854	0.191	0.905
G06	0.262	0.858	0.274	0.852	0.297	0.838	0.226	0.879	0.101	0.927	0.168	0.911	0.196	0.896
G07	0.250	0.866	0.259	0.861	0.414	0.766	0.178	0.906	0.158	0.900	0.069	0.965	0.055	0.972
G08	0.253	0.864	0.260	0.860	0.417	0.763	0.178	0.907	0.151	0.903	0.062	0.969	0.052	0.974

Method abbreviations: XGB: XGBoost; CatB: CatBoost; AG: AutoGluon; TPv2: TabPFN v2; LoCal: LoCalPFN; LUCoS: LUCoS; TACO: TACO.

Table 32: Detailed R^2 and RMSE results on TabMacro-Reg for TabPFN v2.6 sampling. Dataset IDs follow Appendix C.3.

TabPFN v2.6 sampling

Dataset	TP-R5		TP-R10		TP-R20		TP-R50		TP-kNN	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
G01	0.240	0.872	0.247	0.868	0.257	0.862	0.277	0.851	0.178	0.907
G02	1.000	0.008	1.000	0.007	1.000	0.006	1.000	0.007	0.867	0.365
G03	0.709	0.539	0.735	0.515	0.752	0.497	0.775	0.474	0.618	0.618
G04	0.992	0.089	1.000	0.018	0.997	0.050	0.998	0.042	0.878	0.350
G05	0.296	0.844	0.320	0.830	0.350	0.811	0.393	0.784	0.161	0.921
G06	0.198	0.895	0.228	0.878	0.236	0.874	0.265	0.857	0.121	0.937
G07	0.103	0.947	0.145	0.925	0.189	0.900	0.251	0.865	0.092	0.953
G08	0.096	0.951	0.155	0.919	0.182	0.904	0.251	0.866	0.085	0.956

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.

Table 33: Detailed R^2 and RMSE results on TabMacro-Reg for LimiX v1 sampling. Dataset IDs follow Appendix C.3.

LimiX v1 sampling

Dataset	LX-R5		LX-R10		LX-R20		LX-R50		LX-kNN	
	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$	$R^2 \uparrow$	RMSE $\downarrow$
G01	0.231	0.877	0.231	0.877	0.243	0.870	0.247	0.868	0.140	0.927
G02	0.999	0.030	0.999	0.027	0.999	0.032	0.998	0.048	0.867	0.365
G03	0.718	0.531	0.734	0.516	0.741	0.509	0.756	0.494	0.627	0.611
G04	1.000	0.015	1.000	0.012	1.000	0.013	1.000	0.013	0.843	0.396
G05	0.256	0.868	0.280	0.854	0.308	0.837	0.321	0.829	0.082	0.964
G06	0.188	0.901	0.232	0.876	0.244	0.869	0.269	0.854	0.104	0.946
G07	0.142	0.926	0.201	0.894	0.249	0.866	0.277	0.850	0.090	0.954
G08	0.121	0.937	0.182	0.904	0.219	0.884	0.280	0.849	0.093	0.952

R5/R10/R20/R50 denote random sampling at 5%/10%/20%/50% (2048/4096/8192/30000 samples); kNN denotes kNN-based sampling; CB denotes class-balanced sampling. TP and LX denote TabPFN v2.6 and LimiX v1, respectively.