RAD-TFM: Robust and Domain-Adapted Tabular Foundation Models

Matthew Peroni
MIT, IBM
mperoni@ibm.com

Franck Le
IBM
fle@us.ibm.com

Vadim Sheinin
IBM
vadims@us.ibm.com

Abstract

The development of tabular foundation models (TFMs) has accelerated in recent years, outperforming traditional ML methods on many tabular benchmarks. A key finding is that TFMs can be pretrained entirely on synthetic datasets, opening opportunities to design data generators that encourage desirable model properties. Prior work has mainly focused on crafting high-quality priors over generators to improve overall pretraining performance. Our key insight is that parameterizing the generator distribution enables an adversarial, distributional robustness perspective: during training, we can adapt the generator to emphasize datasets that are particularly challenging for the model, which we formalize by introducing an optimality gap measure. Further, we develop a method to align the synthetic data generation with real-world datasets from a given domain, constraining the adversary to generate realistic data. Together, these algorithms comprise the ROBUST AND DOMAIN-ADAPTED TABULAR FOUNDATION MODELS (RAD-TFM) pipeline, a model-agnostic adversarial training framework. Applied to the TabPFN V2 classifier, RAD-TFM improves performance across 6 diverse tabular benchmarks, with up to a 10.8 percentage-point improvement in mean normalized AUC over the strongest baseline, using fewer than 100k additional synthetic training datasets, less than 0.1% of the original pretraining data. Applying the same pipeline to TabICL V2 also improves performance across three major benchmarks, demonstrating the generality of the framework. These results highlight a promising new direction for targeted adversarial continued training of TFMs using synthetic data alone.

1 Introduction

Recent survey studies have challenged the deep learning (DL) community to improve upon traditional methods for learning from tabular data, such as boosted trees methods, having found that DL approaches still lagged behind in terms of performance [19]. In response, substantial progress has been made to close this gap, resulting in a variety of strong deep learning approaches being proposed over the past few years [35, 18, 20, 22, 7, 39, 42]. Among these new approaches, tabular foundation models (TFMs) which rely on in-context learning (ICL) [22, 31, 42] have emerged as a promising direction for classification and regression tasks with structured datasets. The benefits of this approach are two-fold. In many cases, the performance of the TFMs improves upon strong traditional baselines such as boosted trees [16, 14]. Additionally, this performance is achieved in a zero-shot framework, enabling high-quality predictions on new datasets in milliseconds when GPU-accelerated. Further, as we will explore in this work, since these TFMs are pretrained using synthetic data, there is significant opportunity for improvement as the data generation process itself is expanded and improved.

Pretraining TFMs relies on generating a large amount of diverse synthetic datasets [22]. As we will formally define in the following section, this generation process relies on constructing structural causal models (SCMs) from which datasets can be sampled [28, 34]. The structure of these SCMs is implicitly parameterized, giving significant control over the data generation process. All current

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

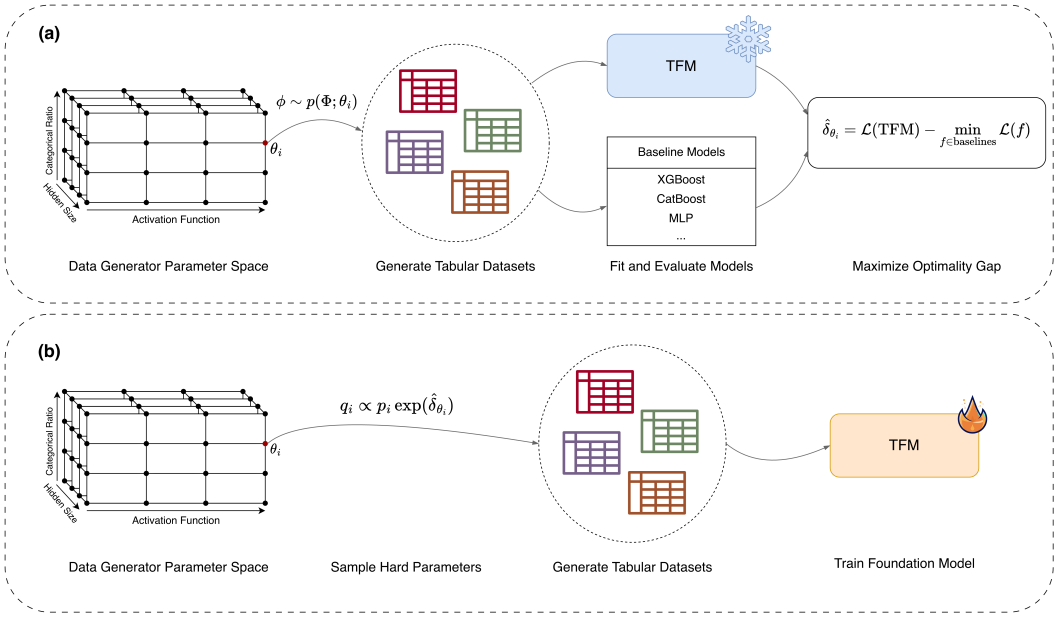

Figure 1: Overview of RAD-TFM. We solve the core distributionally robust optimization problem by alternating between two stages: **(a)** searching over the SCM parameter space, we find synthetic data distributions with large estimated optimality gaps, and **(b)** sampling parameters proportional to their optimality gap, we generate synthetic datasets and continue training the TFM.

publicly available, competitive TFMs [22, 31, 42] have been pretrained on datasets generated from a fixed prior distribution over these SCM parameters. However, fixed priors underrepresent certain regions of the parameter space, potentially degrading performance on real-world datasets with similar structure. This contributes to state-of-the-art TFMs still lagging behind tree-based methods on some benchmarks [25, 41]. In this work, we leverage the significant control provided by the data generation process to frame TFM training from an adversarial robustness perspective [24, 32]. Recent work has explored a narrower version of this idea, focusing on adjusting the weights of a specific class of SCMs in GAN-style training of TFMs [37]. In this work, we focus on a broad and highly flexible framework for adversarially robust training of TFMs. Going one step further, we ground this adversarial training process, constraining it to generating realistic data, yielding a robust and domain-adapted TFM.

Recent developments in TFMs, such as Real-TabPFN, have demonstrated the benefit of incorporating real-world data into the training process of TFMs [17]. However, as we will show in this work, finetuning on real datasets alone is often insufficient due to the lack of large-scale, high-quality tabular datasets, especially for more specific domains such as genomics or physical sciences. In this work, we will train TFMs on synthetic data, but we will constrain the data generation process to produce datasets that are similar to real-world datasets from a given domain. Previous work that aims to align a synthetic data generator with a tabular dataset typically take a GAN-based approach, focusing on constructing a generator to closely mimic a single real dataset [9, 27, 38]. In this work, we will expand beyond this one-to-one relationship, developing an algorithm that efficiently constructs a *distribution over SCM-based tabular dataset generators* that, in aggregate, align with real-world data across a large variety of meta-features capturing data schema, geometry, and learnability.

Our contributions can be summarized as follows: **1)** We formalize distributionally-robust adversarial training over the SCM parameter space, allowing the model to adapt to challenging regions (e.g., varying numbers of features, categorical feature ratios, nonlinearities). We introduce an *optimality gap* objective and use it to target regions where the TFM underperforms relative to the best achievable performance. **2)** We formalize the notion of domain-adapted TFMs in our DRO framework and introduce an efficient algorithm to construct a reference distribution over synthetic data generators that closely matches the real-data distribution for a given domain. **3)** We propose an efficient, model-agnostic two-stage adversarial training algorithm for TFMs, which we call ROBUST AND DOMAIN-ADAPTED TABULAR FOUNDATION MODELS (RAD-TFM). **4)** We apply RAD-TFM to

TabPFN V2 [23], showing that with fewer than 100k additional training datasets, less than 0.1% of the original pretraining data, we can significantly improve its ranking on several real-world tabular benchmarks. We further demonstrate that the same training pipeline improves TabICL V2 [30], an architecturally distinct TFM. To organize this work, in Section 2, we first define the DRO formulation for robust and domain-adapted training of TFMS. Then, in Section 3, we introduce our method for finding an approximate reference distribution over data generators that aligns with real-world datasets. With these elements developed, in Section 4, we define our alternating two-stage algorithm for solving the core DRO problem. Finally, in Section 5, we apply our RAD-TFM pipeline and evaluate it on a variety of tabular benchmarks, both general and application-specific.

2 Problem Definition

We begin by defining some notation, and then introduce our theoretical formulation for RAD-TFM. Let Φ represent the space of hypothesis functions which define the relationship between features x and outcome y. The outcome y can be discrete or continuous, but for this work, we focus on classification tasks. Then a hypothesis $\phi \in \Phi$ is a specific function which we can treat as a data generating mechanism. In the case of SCMs, Φ is the entire space of SCMs, and $\phi \in \Phi$ is a specific instance of an SCM, from which a dataset $D \sim p(\phi)$ can be generated. In this work, we implement SCMs as randomly initialized multi-layer perceptrons (MLPs), as introduced in [22]. Full details of this process are provided in Appendix D. For example, we can sample the number of layers $l \sim p(l)$, the hidden size $h \sim p(h)$, the activation function $a \sim p(a)$, and the ratio of categorical features $r_{cat} \sim p(r_{cat})$, among other dimensions. An SCM is generated by first sampling from the joint distribution over these hyperparameters, each of which is parameterized. For example, we might sample the ratio of categorical features $r_{cat} \sim TruncNorm(\mu_{r_{cat}}, a = 0, b = 1)$ from the truncated normal distribution with some mean $\mu_{r_{cat}}$ which parametrizes the distribution. Then, assuming we have similar parameterizations for the distribution over each hyperparameter, we define $\theta = (\mu_l, \mu_h, \mu_{r_{cat}}, \dots)$ as the parameterization of the joint distribution over all the hyperparameters. We define Θ as the space of the parameters, such that $\theta \in \Theta$ represents a specific parameterization. The generators are then sampled from $\phi \sim p(\Phi; \theta)$.

We now define the original objective for the Prior-Fitted Network (PFN) learning task and provide our adversarially robust formulation. Let $g_{\mathbf{W}}$ be our predictive model parameterized by weight matrix $\mathbf{W}$. As a PFN, $g_{\mathbf{W}}$ takes as input a sequence of labeled training samples and unlabeled test samples, and predicts labels for the test samples. For a given data generator $\phi \in \Phi$, the PFN loss is defined as the cross entropy loss over the test samples,

$$\mathcal{L}_{PFN}(\mathbf{W}; \phi) = \mathbb{E}_{(\{(x_t, y_t)\} \cup D) \sim p(\phi)}[-\log(g_{\mathbf{W}}(y_t | x_t, D)]. \tag{1}$$

Since the model will be trained by sampling many data generators from the distribution $\phi \sim p(\Phi; \theta)$, we can write the original optimization problem from [22] as follows,

$$\min_{\mathbf{W}} \mathbb{E}_{\phi \sim p(\Phi;\theta)}[\mathcal{L}_{PFN}(\mathbf{W}; \phi)]. \tag{2}$$

Since our goal is to learn a foundation model which performs well across all data generation schemes, we want to introduce an adversary who can shift the distribution over data generators to identify the regions over which $g_{\mathbf{W}}$ yields the worst performance. However, there is a key subtlety in this formulation. Maximizing the loss alone could lead the maximizer to select data generators which produce datasets that are hard to learn for any model. Instead, we want to maximize the *optimality gap* between the model's current performance and the best achievable performance. Since we are using the cross-entropy loss, we can frame this optimality gap in terms of the conditional entropy of the data. For any data generator ϕ, let us denote the features as a random variable Z_x and the target as Z_y such that $(Z_x, Z_y) \sim p(\phi)$. Then the Bayes' optimal predictor is exactly $f^* = p(Z_y | Z_x)$ which achieves the cross-entropy loss $L_{CE}(f^*) = H_\phi(Z_y | Z_x)$, the conditional entropy of the target Z_y given the features Z_x. It follows that for any fixed data generator, $\mathcal{L}_{PFN} \geq H(Z_y | Z_x)$, and $\mathcal{L}_{PFN} - H(Z_y | Z_x) \geq 0$ is the optimality gap. We can then state the adversarially robust optimization problem in terms of maximizing the optimality gap,

$$\min_{\mathbf{W}} \max_{\theta \in \Theta} \mathbb{E}_{\phi \sim p(\Phi;\theta)}[\mathcal{L}_{PFN}(\mathbf{W}; \phi) - H_\phi(Z_y | Z_x)]. \tag{3}$$

For brevity, we denote the objective function as $\delta_\theta(\mathbf{W}) = \mathbb{E}_{\phi \sim p(\Phi;\theta)}[\mathcal{L}_{PFN}(\mathbf{W}; \phi) - H_\phi(Z_y | Z_x)]$. While Equation 3 is the precise theoretical optimization problem we wish to solve, in reality, estimating the conditional entropy $H_\phi(Z_y | Z_x)$ or evaluating it analytically is often infeasible. Therefore,

rather than estimate $H_\phi(Z_y|Z_x)$ directly, in this work, we replace this term with the minimum cross-entropy loss over a collection of baseline estimators $f_1, \ldots, f_e$, such as XGBoost, Catboost, and Random Forests [8, 13, 6], which are fit to each generated dataset. Since $\min_{i\in[e]} \mathcal{L}_{PFN}(f_i;\phi) \geq H(Z_y|Z_x)$, it follows that $\mathbb{E}_{\phi\sim p(\Phi;\theta)}[\mathcal{L}_{PFN}(\mathbf{W};\phi) - \min_{i\in[e]} \mathcal{L}_{PFN}(f_i;\phi)] \leq \delta_\theta(\mathbf{W})$, providing us with a lower bound on the optimality gap. We then modify our original objective function, defining

$$\widehat{\delta}_\theta(\mathbf{W}) = \mathbb{E}_{\phi\sim p(\Phi;\theta)}[\mathcal{L}_{PFN}(\mathbf{W};\phi) - \min_{i\in[e]} \mathcal{L}_{PFN}(f_i;\phi)]. \tag{4}$$

Since $\widehat{\delta}_\theta(\mathbf{W}) \leq \delta_\theta(\mathbf{W})$, it follows that $\widehat{\delta}_\theta(\mathbf{W}) > 0$ implies $\delta_\theta(\mathbf{W}) > 0$, such that for any parameter estimated to have a positive optimality gap, the model $g_\mathbf{W}$ does indeed have room for improvement. Further, this lower bound can always be improved by adding more baseline estimators with varying strengths and inductive biases. In practice, maximizing over a single parameterization could lead to the model overfitting to a particular region of the parameter space. Further, for any given domain or application, we want to ensure that the generated synthetic datasets do not stray too far from the types of datasets observed in that domain. We therefore frame this problem from a distributionally robust optimization (DRO) perspective by constraining the actions of the maximizer [32]. Rather than a specific parameterization θ, we allow the maximizer to select a distribution $Q \in \Delta_\Theta$ over parameters, where Δ_Θ is the simplex over Θ. We assume Θ is discrete, which is reasonable in practice since the parameters are either categorical or can be easily discretized (e.g. $\mu_{r_{cat}} \in \{0, 0.1, \ldots, 1\}$). To constrain the maximizer, we will assume access to a reference distribution $P \in \Delta_\Theta$ that aligns with the data from a given domain. In Section 3, we will introduce our methodology for finding such a reference distribution given a corpus of real-world datasets from a specific domain. Then, with the reference distribution P, we can constrain the KL-divergence between Q and P, leading to the following optimization problem,

$$\min_{\mathbf{W}} \max_{\substack{Q\in\Delta_\Theta \\ D_{KL}(Q||P)\leq\rho}} \mathbb{E}_{\theta\sim Q}[\hat{\delta}_\theta(\mathbf{W})]. \tag{5}$$

As we prove in Appendix C, the optimal distribution for the maximizer in Equation 5 is a softmax distribution. Specifically, denoting $Q^* = (q_1^*, \ldots, q_{|\Theta|}^*)$ as the optimal distribution and $P = (p_1, \ldots p_{|\Theta|})$ as the reference distribution, for any $i \in |\Theta|$, $q_i^* \propto p_i \exp(\eta \cdot \delta_{\theta_i}(\mathbf{W}))$. The temperature $\eta = \eta(\rho, \delta_\theta(\mathbf{W}), P)$ is uniquely determined by the maximum KL-divergence, the optimality gaps, and the reference distribution. The DRO problem can then be rewritten as

$$\min_{\mathbf{W}} \sum_{i=1}^{|\Theta|} \frac{p_i \exp(\eta \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}))}{\sum_{i\in|\Theta|} p_i \exp(\eta \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}))} \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}). \tag{6}$$

In the following sections, we introduce our RAD-TFM approach to solving Equation 6. In Section 3, we develop our key contribution to domain adaptation for TFMs, learning the reference distribution P over data generator parameters to align the generated data with real-world data. Then in Section 4, we incorporate the learned reference distribution into our DRO formulation and develop our method for solving this optimization problem using an iterative, two-stage approach.

3 Domain-Adapted Tabular Foundation Models

In our RAD-TFM methodology, the key to adapting a TFM to a specific domain is successfully constructing a reference distribution $P \in \Delta_\Theta$ that connects a corpus of real-world datasets to a corresponding distribution over dataset generators. We will then use this reference distribution in our DRO formulation given in Equation 5 to adversarially generate datasets that challenge the TFM while remaining similar to the real-world datasets observed in the given domain. However, tabular datasets are structurally heterogeneous, with varying numbers and types of features, patterns, and targets. This motivates the need for a shared latent space for all tabular datasets that encodes semantic meaning through which we can project and compare datasets. To accomplish this, we rely on meta-feature extraction, a well-studied topic in meta-learning [5], to efficiently summarize any given tabular dataset into a fixed-size vector of statistical and information-theoretic properties. An illustration of our domain-adaptation scheme is given in Figure 2. Full details are given in Appendix A.

Following the definition of meta-features from [29] and [33], for a given dataset $(\mathbf{X}, \mathbf{y}) \in \mathcal{S}$, where $S = \cup_{n\leq N, d\leq D} \mathbb{R}^{n\times d}$ represents the space of tabular datasets up to some maximum size $N \times D$,

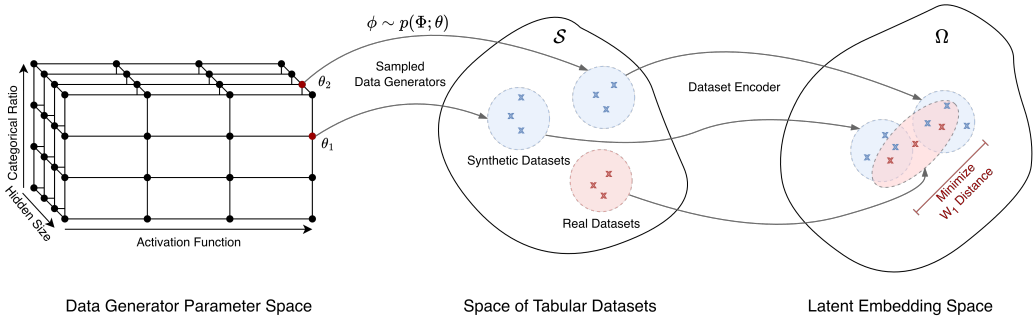

Figure 2: We perform domain adaptation for TFMs (e.g. healthcare, finance) by learning a mixture distribution over SCM generators that covers the distribution of real-world datasets from the given domain. We use a collection of meta-features capturing *dataset schema, geometry, and learnability* to encode and measure distance between datasets.

we can construct a *characterization measure* $m : \mathcal{S} \to \cup_d \mathbb{R}^d$, which produces a variable number of values depending on the shape of the dataset. For example, m might extract the skewness of each numeric feature in the dataset separately. We can then introduce a *summarization function* $\sigma : \cup_d \mathbb{R}^d \to \mathbb{R}^k$ which condenses the characterization measure (e.g. by taking averages) into a fixed-size summarization vector of dimension k. Combining these functions, we define the tabular dataset encoder as $f((\mathbf{X}, \mathbf{y})) = \sigma(m((\mathbf{X}, \mathbf{y})))$, projecting any tabular dataset into a semantically meaningful latent space. We provide details of the exact meta-features used in this work in Appendix A. Using this encoder, we can reason about the relationship between real datasets and synthetic datasets generated from the class of SCMs.

Suppose we have a corpus $D_r = \{(\mathbf{X}_i, \mathbf{y}_i)\}_{i=1}^{n_r} \in \mathcal{S}^{n_r}$ of real datasets from a given domain. At a high level, our algorithm works as follows: (1) *encode* each real dataset in the corpus as $f((\mathbf{X}_i, \mathbf{y}_i)) = \mathbf{e}_i$ using our meta-feature extractor; (2) use Bayesian Optimization (BO) to find distributions over synthetic data generators such that the embeddings of the synthetic datasets *provide partial coverage* of the real dataset embeddings; (3) Aggregate the results to form a *global distribution over data generators* that minimizes the W_1-distance from the distribution of real datasets in the latent space. By creating a mixture over synthetic data distributions that each provide partial coverage of the real datasets, we can construct a reference distribution P that aligns with an entire corpus of real datasets.

We now formalize the key steps in our DA algorithm where we align the distributions of real and synthetic datasets. We provide additional details in Appendix A. For each $\theta \in \Theta$ we have a distribution $\phi \sim p(\Phi;\theta)$ over data generators and, using the dataset encoder, a distribution $\mathbf{e}(\theta) \sim p(f(\phi);\theta)$ over dataset embeddings, where $f(\phi)$ is short for the embedding of $(\mathbf{X}, y) \sim p(\phi)$, the dataset sampled from the specific generator. If we assume $|\Theta| = M$ and denote $\Theta = \{\theta_1, \ldots, \theta_M\}$, we can define $Q(\theta_i) = p(f(\phi);\theta_i)$ as the distribution over datasets in the latent space induced by θ_i. Then, given that the embeddings of the corpus of real datasets is sampled from some distribution P_r, our goal is to solve the following optimization problem,

$$P^* \in \arg\min_{P \in \Delta_\Theta} W_1\left(P_r, \sum_{i=1}^{M} p_i \cdot Q(\theta_i)\right), \tag{7}$$

finding the mixture over these parameterized generator distributions that minimizes the 1-Wasserstein distance from the real data distribution. Solving this optimization problem recovers the reference distribution P^* with the best possible coverage of the underlying domain-specific data distribution. In practice, we instead optimize the entropy-regularized optimal transport distance, which can be efficiently solved via the Sinkhorn Algorithm [11]. Since M is large for any reasonable parameter search space, we approximate this search using Bayesian Optimization [15] with a surrogate objective designed to reward partial coverage of P_r. The resulting candidate distributions $Q(\theta_i)$ are then combined by solving the mixture optimization problem in Equation 7, which has the side effect of pruning redundant and unhelpful generator distributions. The details of our method are given in Algorithm 1.

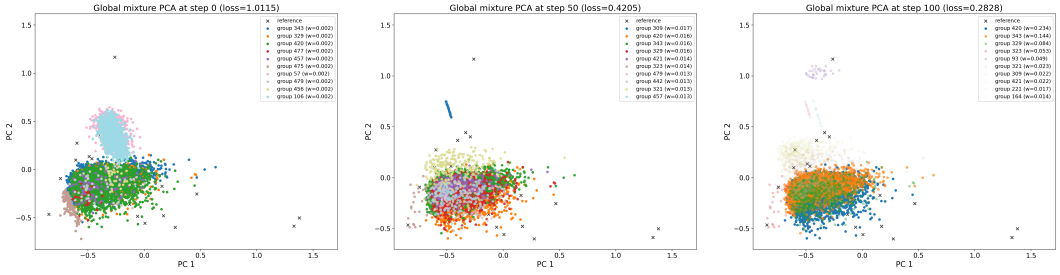

Figure 3: An example of our domain-adaptation methodology applied to the engineering datasets from OpenML [4]. Each color corresponds to a different parameterization of the synthetic data generator distribution. The black marks represent the embeddings of the real datasets. We plot the first two principal components of the meta-features extracted for the real datasets.

4 Robust Tabular Foundation Models

In this section, we give an overview of our two-stage optimization algorithm, ROBUST AND DOMAIN-ADAPTED TABULAR FOUNDATION MODELS (RAD-TFM), which is visualized in Figure 1. The pseudocode and additional details are provided in Appendix B. We first decompose the problem naturally into a maximization stage (parameter search) and a minimization stage (model training). We assume access to a pretrained TFM, $g_{\mathbf{W}}$, with weights $\mathbf{W}$ which will be used as the input model for our RAD-TFM algorithm. While not absolutely necessary, starting with a pretrained model gives us a warm-start for the model weights. In addition, we specify a fixed set of baseline models $f_1, \ldots, f_e$ which will be used to estimate the optimality gap. We begin with the maximization stage, during which we freeze $g_{\mathbf{W}}$ to maximize the optimality gap in its current state. Since the cardinality of the parameter space is large in practice, we use a black-box optimization algorithm [1, 36, 3] to efficiently search the space for parameters with large optimality gaps.

Using the black-box optimization algorithm, parameters θ_i are proposed in sequence. For each θ_i proposed, we estimate the optimality gap by sampling a fixed number, n_{ds}, of generators $\phi_1, \ldots, \phi_{n_{ds}} \sim p(\Phi; \theta_i)$ and datasets $d_j \sim p(\phi_j)$ and compute $\widehat{\delta}_{\theta_i} = \frac{1}{n_{ds}} \sum_{j=1}^{n_{ds}} \mathcal{L}_{PFN}(\mathbf{W}; d_j) - \min_{k \in [e]} \mathcal{L}_{PFN}(f_k, d_j)$. Then for each proposed θ_i, we must fit $n_{ds} \cdot e$ baseline estimators. However, each pair of dataset and baseline estimator can be fit independently, such that this step can be trivially parallelized, where each baseline estimator is fit to each dataset in parallel. In practice, we found that for $n_{ds} = 20$ and $e = 7$, the estimated optimality gap $\widehat{\delta}_{\theta_i}$ could be computed in a matter of seconds when parallelized, given sufficient CPU cores (we used $n_{cores} = n_{ds} \cdot e$). We repeat this process for a fixed number of trials, n_{trials}, with the underlying optimization algorithm proposing new parameters θ_i based on the observed estimated optimality gaps. At the end of this stage, we have a collection $\{(\theta_i, \delta_{\theta_i})\}_{i=1}^{n_{trials}}$ of parameters and associated estimated optimality gaps.

In the minimization stage, our goal is to train $g_{\mathbf{W}}$ on the regions of the parameter space with large optimality gaps. Following our derivation in Equation 6, we define a distribution Q over Θ such that $q_i \propto p_i \exp(\eta \cdot \delta_i(\mathbf{W}))$. The function $\eta \mapsto D_{KL}(Q(\eta)||P)$ is strictly increasing on $(0, \infty)$, so we can quickly find η given a fixed ρ via one-dimensional root finding (e.g., bisection). Using the observations from the maximization stage, our optimization problem becomes

$$\min_{\mathbf{W}} \sum_{i=1}^{n_{trials}} \frac{p_i \exp(\eta \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}))}{\sum_{i \in [n_{trials}]} p_i \exp(\eta \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}))} \cdot \widehat{\delta}_{\theta_i}(\mathbf{W}). \tag{8}$$

Then, to train the TFM, we generate each dataset within a batch by sampling $\theta_i \sim Q$ and generator $\phi \sim p(\Phi; \theta_i)$. This sampling and dataset generation process is repeated for each batch during model training such that, in expectation, the loss is weighted according to Equation 8. After training $g_{\mathbf{W}}$ for n_{iter} iterations, we freeze the model return to the maximization stage, using the updated model weights to generate the next set of estimated optimality gaps. This max-min iteration is repeated until the objective in Equation 8 converges. In addition to the baseline estimators $f_1, \ldots, f_e$, after a fixed number of max-min iterations, we incorporate the TFM with its original weights as an additional baseline model. This can help mitigate any unlearning that has occurred, encouraging the model to both improve upon strong baselines as well as its own performance at the start of training.

Table 1: Experiment results across three tabular dataset benchmarks. For consistency, we report normalized AUC, with values scaled to [0,1] for each dataset [25, 23]. For rank-1 wins, we do not count ties, only outright wins.

Benchmark	Metric	LR	MLP	RF	CatBoost	XGB	TabPFN$_{\text{n.e.}}$	TabPFN$_{\text{n.e.}}$ (FT)	TabPFN$_{\text{n.e.}}$ (RAD-TFM)
TabPertNet[40]	Mean rank AUC	5.4	6.0	4.6	4.2	5.2	3.7	3.9	**3.1**
	Mean Norm. AUC	0.4684	0.3627	0.6646	0.7404	0.5743	0.7306	0.7299	**0.8486***
	Rank-1 Wins	6	5	7	6	7	9	2	**13**
TabArena[14]	Mean rank AUC	5.7	7.2	5.5	4.0	5.2	3.6	2.7	**2.1**
	Mean Norm. AUC	0.5070	0.2513	0.6870	0.8214	0.7293	0.8829	0.9215	**0.9421***
	Rank-1 Wins	2	0	0	1	3	1	4	**9**
PMLB[26]	Mean rank AUC	5.8	7.0	4.5	4.3	4.7	3.3	3.3	**3.1**
	Mean Norm. AUC	0.4497	0.2782	0.7580	0.7798	0.7036	0.8889	0.8894	**0.9070**
	Rank-1 Wins	8	2	10	8	7	9	4	**16**

5 Experiments

Experiment Setup. We evaluate our RAD-TFM algorithm on broad benchmark classification datasets from TabArena [14], TabPertNet [40], and PMLB [26], as well as three domain-specific benchmarks derived from datasets available on OpenML [4]. We use the open-source model weights from TabPFN V2 [23] to initialize our TFM, and further apply the same pipeline without architectural modifications to the TabICL V2 classifier [30]. We provide a full description of the experiment setup and results in Appendix G. For the RAD-TFM algorithm, we use $n_{trials} = 100$ and $n_{ds} = 20$ datasets to estimate the optimality gap for each trial. During the minimization stage, we use a learning rate of $lr = 1e\text{-}5$ and a batch size of $b = 64$. We set $\rho = 0.1 \cdot \rho_{\text{max}}$, where ρ_{max} is the maximum KL-divergence given the learned reference distribution P. Depending on the task, we train on 1,000–3,000 generated datasets per epoch and run the full max-min loop for $e \in [5, 30]$ epochs, using at most 90k generated training datasets in any run. For baseline models, we use Random Forest, CatBoost, XGBoost, Logistic Regression, and MLPs. For the primary TabPFN benchmark tables, we also finetune TabPFN-V2 on each evaluation dataset for 10 epochs and report held-out performance as TabPFN (FT). In a separate compute-controlled comparison, we match fine-tuning to RAD-TFM by training-dataset exposure, batch size, and number of optimizer updates, evaluating checkpoints every 500 training datasets (Appendix H). We evaluate the conventional baseline algorithms using their respective default settings, and report TFM results without ensembling. All experiments were run on a single node with one A100 GPU; further details of the compute environment and SCM parameter space are provided in Appendices G and D. Following previous work, we report normalized ROC-AUC as our primary metric, which scales raw ROC-AUC scores to [0,1], with the worst-performing model receiving a score of 0, and the best a score of 1 [21, 17]. This provides consistency when comparing performance across datasets, and measures how close each model comes to the best possible model on average. For multi-class tasks, we use the One-versus-One ROC-AUC score. We also report the mean rank and number of rank-1 wins for each model, based on the normalized AUC scores.

In Table 1, we report the main results from our experiments on the TabArena ($N = 21$), TabPertNet ($N = 70$), and PMLB ($N = 91$) benchmarks. Across our three primary metrics, we observe that RAD-TFM improves performance over all baselines, including the original TabPFN-V2 and the

Table 2: Experiment results across three domain-specific tabular dataset benchmarks from OpenML [4]. For consistency, we report normalized AUC, with values scaled to [0,1] for each dataset [25, 23]. For rank-1 wins, we do not count ties, only outright wins.

Benchmark	Metric	LR	MLP	RF	CatBoost	XGB	TabPFN$_{\text{n.e.}}$	TabPFN$_{\text{n.e.}}$ (FT)	TabPFN$_{\text{n.e.}}$ (RAD-TFM)
Engineering	Mean rank AUC	6.9	7.3	4.3	3.4	4.9	3.3	3.4	**2.6**
	Mean Norm. AUC	0.2944	0.3951	0.8405	0.8952	0.7805	0.9004	0.8998	**0.9313***
	Rank-1 Wins	0	0	1	4	2	0	0	**7**
Genomics	Mean rank AUC	7.2	5.9	4.8	3.4	4.9	3.5	3.3	**2.7**
	Mean Norm. AUC	0.1481	0.4309	0.7439	0.8611	0.6979	0.8801	0.8805	**0.9088**
	Rank-1 Wins	0	0	1	0	0	0	0	**2**
Healthcare	Mean rank AUC	5.3	6.9	4.5	3.5	5.0	3.8	3.8	**3.2**
	Mean Norm. AUC	0.5831	0.2050	0.7245	0.8446	0.6719	0.8696	0.8699	**0.8974**
	Rank-1 Wins	2	0	2	2	4	1	0	**7**

Table 3: Comparison of fine-tuning (FT) and RAD-TFM on the TabPFN V2 classifier across three benchmarks. Wins count outright wins among the three TabPFN variants; AUC change reports the relative percentage change in mean raw AUC from the frozen TabPFN. A leap occurs when an adapted variant outperforms at least one conventional baseline that the frozen TabPFN did not outperform.

Benchmark	Metric	$\textbf{TabPFN}_{\textbf{n.e.}}$	$\textbf{TabPFN}_{\textbf{n.e.}}$ **(FT)**	$\textbf{TabPFN}_{\textbf{n.e.}}$ **(RAD-TFM)**
TabPertNet[40]	#Wins	11	5	**37**
	AUC Change (%)	-	-0.001	$\mathbf{2.061}$
	#Leaps	-	1	**18**
	Norm. AUC (Std ↓)	0.345	0.343	**0.215**
TabArena[14]	#Wins	1	6	**13**
	AUC Change (%)	-	0.166	$\mathbf{0.462}$
	#Leaps	-	3	**7**
	Norm. AUC (Std ↓)	0.223	0.195	**0.178**
PMLB[26]	#Wins	15	8	**37**
	AUC Change (%)	-	0.016	$\mathbf{0.332}$
	#Leaps	-	1	**8**
	Norm. AUC (Std ↓)	0.210	0.209	**0.188**

finetuned versions. To measure the significance of our results, we first perform the Friedman test for repeated samples on the median normalized AUC scores [12, 2]. For the TabPertNet datasets, the Friedman test yielded $p = 3.5\times10^{-15}$, for TabArena, the test yielded $p = 4.3\times10^{-13}$, and for PMLB, the test yielded $p = 5.2 \times 10^{-43}$. Having established significance for the benchmark performance varying by model class, we performed the Wilcoxon signed-rank test pair-wise, comparing the RTFM version of TabPFN to the other models [10]. In Tables 1 and 2, we add an asterisk to indicate results for which the all Wilcoxon signed-rank tests between RAD-TFM and the alternative models achieves $p < 0.05$. For the main benchmarks reported in Table 1, we see that all results are significant for RAD-TFM for TabPertNet and TabArena. Further, for PMLB, the largest p-value obtained was $p = 0.06$, between RAD-TFM and TabPFN-V2 (FT), with $p < 0.05$ for all other pair-wise tests. Our results are therefore almost strictly statistically significant, with RAD-TFM achieving the best performance for each metric across a variety of challenging tabular benchmarks.

In Table 2, we report results from experiments applying RAD-TFM to domain-specific benchmarks in Engineering ($N = 20$), Genomics ($N = 11$), and Healthcare ($N = 31$), derived from OpenML. To construct these benchmarks, we aggregate all binary and multi-task classification datasets on OpenML matching a given tag (e.g. Engineering), then filter to datasets with at least $n = 200$ samples, no more than $n = 10k$ samples, and at most $d = 500$ features. These benchmarks can be highly domain-specific, and differ significantly from common benchmarks such as TabArena. Similar to the other benchmarks, applying RAD-TFM improves the overall performance of TabPFN-V2 across domains, improving mean rank, normalized AUC, and outright wins. Additionally, for these more specific domains, finetuning TabPFN does not exhibit significant benefits. Across the three domain-specific benchmarks, RAD-TFM achieves 16 rank-1 wins, compared with only one for the original TabPFN and none for the fine-tuned model. The Friedman test returned $p = 1.3 \times 10^{-11}$, $p = 1.3 \times 10^{-5}$, and $p = 4.1 \times 10^{-14}$ for Healthcare, Genomics, and Engineering respectively. Although our approach moves TabPFN's performance in the right direction for these domains, the Wilcoxon signed-rank tests indicate that RAD-TFM only has statistically significant ($p < 0.05$) improvement in normalized AUC for the Engineering benchmark. This is due in part to the small Genomics benchmark ($N = 11$). The Healthcare benchmark is particularly challenging, indicating that it may be necessary to expand either the SCM parameter space or the meta-features used to represent the datasets to enable RAD-TFM to properly target this domain. Overall, these results indicate the potential for RAD-TFM to go beyond finetuning TFMs for domain-specific applications, noticeably improving model performance across three diverse domains.

RAD-TFM outperforms Fine-Tuning. To further investigate the impact of finetuning versus our RAD-TFM approach, in Table 3, we compare these two methods against the original TabPFN V2 model in isolation. Across the three benchmarks, we report the number of outright wins among the three TabPFN variants and the relative percentage change in mean raw AUC from the frozen model. Additionally, we report a metric we refer to as the *number of leaps*, defined as the number of

Table 4: Generalization of RAD-TFM to the TabICL V2 backbone across TabPertNet, TabArena, and PMLB. Normalized AUC and ranks are computed over the expanded comparison of 11 models (including five classical baselines); selected models are shown. Scores are pooled across 177 datasets with defined normalized AUC and use a common normalization across all evaluated models.

Model	Mean Norm. AUC	Rank-1 Wins	Mean Rank
TabICL V2 (RAD-TFM)	**0.868**	21	4.23
TabPFN V2 (RAD-TFM)	0.860	11	4.73
TabICL V2 (FT)	0.850	15	4.37
TabICL V2	0.837	18	4.67
TabPFN V2 (FT)	0.810	2	5.37
TabPFN V2	0.806	1	5.45
CatBoost	0.743	8	6.21

datasets for which the adapted TabPFN surpasses at least one conventional baseline that the original TabPFN-V2 did not outperform. This metric captures how often the method not only improves upon TabPFN-V2 incrementally, but improves it enough to surpass alternative model classes. This metric is key to our analysis, given that a major motivation for this work is the observation that, although TFMs are competitive with traditional methods, they do not consistently outperform them. In Table 3, we can see that applying RAD-TFM not only significantly improves the average change in AUC over finetuning alone, but also dominates in the number of wins and, importantly, the number of leaps. In the most extreme case, TabPertNet, finetuning has almost no effect on performance, while applying RAD-TFM leads to 18 leaps, such that TabPFN would now outperform at least one alternative method on 25.7% of datasets in the benchmark. Additionally, we report the standard deviation of normalized AUC scores for each model, which measures the dispersion of performance across datasets. We observe that the robust training process of RAD-TFM reduces this cross-dataset variability.

Generalization Across TFM Backbones. To evaluate the generality of the proposed training framework, we apply RAD-TFM to the open-source TabICL V2 classifier [30], a TFM with a different architecture from TabPFN V2. We use the same domain-adaptation and robust training pipeline, without modifying the TabICL architecture, and compare against the frozen backbone and compute-matched fine-tuning on TabPertNet, TabArena, and PMLB. Table 4 summarizes normalized AUC, rank-1 wins, and mean rank across the three benchmarks. Both RAD-TFM variants achieve the highest mean normalized AUC among the evaluated methods, and TabICL with RAD-TFM improves over the original TabICL on all three benchmarks (Appendix H). The improvement over the frozen TabICL is significant under a one-sided Wilcoxon signed-rank test on per-dataset normalized AUC ($p = 0.032$), whereas the corresponding improvement from fine-tuning is not ($p = 0.148$). The direct comparison of TabICL with RAD-TFM against fine-tuning favors RAD-TFM in mean normalized AUC, but is not statistically significant (mean gap 0.018; 95% bootstrap confidence interval $[-0.005, 0.043]$). These results demonstrate that the benefits of RAD-TFM are not specific to TabPFN.

Isolating Domain Adaptation and Robustness. We further ablate the two components of RAD-TFM by training TabPFN V2 using only the domain-adapted reference distribution (DA-only), or only robust parameter selection (Robust-only). In Table 5, each component independently improves upon the frozen model, while the full approach achieves the highest mean AUC on five of the six benchmarks. Across 236 datasets, both DA-only and Robust-only improve normalized AUC significantly relative to the frozen TabPFN ($p = 10^{-4}$ and $p = 0.028$, respectively, using one-sided paired Wilcoxon tests). The effects are approximately additive on average, although the relative contribution varies by domain: domain adaptation accounts for most of the improvement on PMLB, while the two components complement one another on Engineering. Additional analysis is provided in Appendix H.

Compute-Matched Fine-Tuning. To separate the benefit of the training distribution from training duration, we additionally train TabPFN V2 on real reference datasets with the same batch size and number of training datasets and optimizer updates as the corresponding RAD-TFM runs. Both approaches exhibit small gains at early checkpoints, but their trajectories diverge with additional training. Across TabArena, PMLB, and TabPertNet, fine-tuning eventually falls below the frozen backbone, while RAD-TFM continues to improve. On TabArena, for example, the selected RAD-

Table 5: Ablation of domain adaptation (DA) and robust SCM parameter selection, reporting mean test AUC for TabPFN V2 across six benchmarks. The full method combines both components.

Benchmark	Base	DA-only	Robust-only	RAD-TFM
PMLB	0.9088	**0.9121**	0.9109	0.9119
TabPertNet	0.8019	0.8093	0.8113	**0.8185**
TabArena	0.8725	0.8738	0.8743	**0.8765**
Genomics	0.9675	0.9699	0.9679	**0.9713**
Healthcare	0.8915	0.8937	0.8938	**0.8951**
Engineering	0.9402	0.9403	0.9386	**0.9422**
Pooled	0.8780	0.8819	0.8818	**0.8851**

TFM checkpoint from a training run with 30k generated-dataset exposures yields a mean raw AUC gain of $+0.0040$ relative to the frozen model, compared with -0.0844 for fine-tuning evaluated at the 30k checkpoint. The full checkpoint comparisons are reported in Appendix H.

Learned SCM Parameters and Generator Coverage. The learned adversarial distributions vary substantially across benchmarks (Appendix E). For example, Genomics favors generators with many numerical features, whereas Healthcare favors small, predominantly categorical datasets with higher noise. The Engineering benchmark instead selects larger, dense generators with numerical features. These differences illustrate the importance of adapting both dataset-level properties and the underlying data-generating mechanisms. However, domain adaptation is necessarily limited by the expressiveness of the available SCM family. On a foreign-exchange benchmark with pronounced temporal structure, the original non-temporal generators provide poor coverage of the real-data distribution, and RAD-TFM does not improve upon the frozen TabPFN. Extending the generator family with a temporal SCM improves alignment and yields a modest gain in AUC. This case illustrates both a limitation of the current generator space and the potential of domain-specific synthetic generation (Appendix F).

6 Conclusion

We introduce RAD-TFM, a model-agnostic framework for robust, domain-adapted continued training of TFMs. We demonstrate its ability to improve TabPFN-V2 using less than 0.1% of the original pretraining data, across six different tabular benchmarks. We further improve the architecturally distinct TabICL V2 classifier across three major benchmarks, demonstrating the generality of our approach. We show that constraining the robust training process to generate synthetic data that is similar to real-world data can be done efficiently, and that both domain adaptation and robust parameter selection independently contribute to performance. Compute-matched comparisons further distinguish the gains from training on robust, domain-adapted synthetic data from simply training for longer on real datasets. The foreign-exchange experiments further highlight the role of the synthetic generator family: extending its support to temporal data improves domain coverage and enables further adaptation. Our framework can also be extended to other TFMs and regression tasks. The RAD-TFM algorithm is highly-parallelizable, enabling large-scale training and efficient parameter search. Together, these results demonstrate how robust optimization and flexible SCM-based synthetic data generation can improve TFM performance across diverse domains.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by a research grant from the MIT–IBM Computing Research Lab.

Competing interests. The authors declare no competing interests.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework, July 2019. arXiv:1907.10902 [cs].

[2] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks? *The Journal of Machine Learning Research*, 17(1):152–161, 2016.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[4] Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G. Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML Benchmarking Suites, November 2021. arXiv:1708.03731 [stat].

[5] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning: Applications to data mining*. Springer, 2009.

[6] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[7] Kimberly Villalobos Carballo, Liangyuan Na, Yu Ma, Léonard Boussioux, Cynthia Zeng, Luis R. Soenksen, and Dimitris Bertsimas. TabText: A Flexible and Contextual Approach to Tabular Data Representation, July 2023. arXiv:2206.10381 [cs].

[8] 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*, KDD '16, pages 785–794, New York, NY, USA, August 2016. Association for Computing Machinery.

[9] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks, 2018.

[10] William Jay Conover. *Practical nonparametric statistics*. john wiley & sons, 1999.

[11] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

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

[13] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. CatBoost: gradient boosting with categorical features support, October 2018. arXiv:1810.11363 [cs].

[14] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A Living Benchmark for Machine Learning on Tabular Data, June 2025. arXiv:2506.16791 [cs].

[15] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[16] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, October 2001. Publisher: Institute of Mathematical Statistics.

[17] Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data, 2025.

[18] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting Deep Learning Models for Tabular Data, October 2023. arXiv:2106.11959 [cs].

[19] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, July 2022. arXiv:2207.08815 [cs].

[20] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. TabLLM: Few-shot Classification of Tabular Data with Large Language Models. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR, April 2023. ISSN: 2640-3498.

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

[22] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second, September 2023. arXiv:2207.01848 [cs].

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

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks, September 2019. arXiv:1706.06083 [stat].

[25] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Benjamin Feuer, Chinmay Hegde, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When Do Neural Nets Outperform Boosted Trees on Tabular Data?, July 2024. arXiv:2305.02997 [cs].

[26] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):36, 2017.

[27] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

[28] Judea Pearl. *Causality*. Cambridge University Press, Cambridge, 2 edition, 2009.

[29] Fábio Pinto, Carlos Soares, and Joao Mendes-Moreira. Towards automatic generation of metafeatures. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 215–226. Springer, 2016.

[30] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 102477–102555. PMLR, 2026.

[31] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A Tabular Foundation Model for In-Context Learning on Large Data, May 2025. arXiv:2502.05564 [cs].

[32] Hamed Rahimian and Sanjay Mehrotra. Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*, 2019.

[33] Adriano Rivolli, Luís P. F. Garcia, Carlos Soares, Joaquin Vanschoren, and André C. P. L. F. de Carvalho. Characterizing classification datasets: a study of meta-features for meta-learning, 2019.

[34] Ramalingam Shanmugam. Elements of causal inference: foundations and learning algorithms. *Journal of Statistical Computation and Simulation*, 88(16):3248–3248, November 2018.

[35] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training, June 2021. arXiv:2106.01342 [cs].

[36] Shuhei Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance, May 2023. arXiv:2304.11127 [cs].

[37] Yulun Wu and Doron L. Bergman. Zero-shot Meta-learning for Tabular Prediction Tasks with Adversarially Pre-trained Transformer, June 2025. arXiv:2502.04573 [cs].

[38] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan, 2019.

[39] Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Z. Chen, Jimeng Sun, Jian Wu, and Jintai Chen. Making Pre-trained Language Models Great on Tabular Prediction, March 2024. arXiv:2403.01841 [cs].

[40] Chao Ye, Guoshan Lu, Haobo Wang, Liyao Li, Sai Wu, Gang Chen, and Junbo Zhao. Towards Cross-Table Masked Pretraining for Web Data Mining, February 2024. arXiv:2307.04308 [cs].

[41] Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A Closer Look at TabPFN v2: Understanding Its Strengths and Extending Its Capabilities, June 2025. arXiv:2502.17361 [cs].

[42] Xiyuan Zhang and Danielle Maddix Robinson. Mitra: Mixed synthetic priors for enhancing tabular foundation models, July 2025.

A Details for Domain-Adapted TFMs

In this Section, we provide a detailed overview of the domain-adaptation process introduced in Section 3. We begin by defining the set of meta-features used to encode tabular datasets, then define our surrogate reward for the Bayesian Optimization process to maximize partial coverage of the real data distribution, and finally introduce our algorithm for constructing the global mixture over synthetic data generators. Although the overall pipeline involves multiple steps, the meta-features we define are inexpensive to compute, and we are able to solve the global mixture optimization efficiently via the Sinkhorn algorithm. As a result, we found that for all of our experiments, we can recover the approximate reference distribution over synthetic data generators in under 10 minutes, using a single A100 GPU on standard hardware. Further, this process only needs to be run once for each domain, since the same reference distribution can be re-used across many training runs of the RAD-TFM algorithm. We summarize the pseudocode for our domain-adaptation pipline in Algorithms 1 and 2.

Meta-Feature Extraction. For a tabular dataset (X, y) with n samples and p input features, we construct a meta-feature embedding

$$\phi(X, y) = \begin{bmatrix} \phi_{\text{schema}}(X) \\ \phi_{\text{geom}}(X) \\ \phi_{\text{learn}}(X, y) \end{bmatrix} \in \mathbb{R}^d.$$

The representation is designed to separate three sources of variation: coarse dataset schema and marginals, affine-invariant geometric structure of the numeric covariates, and inexpensive proxies for supervised learnability. In the classification setting used in our experiments, the corresponding block dimensions are $(d_{\text{schema}}, d_{\text{geom}}, d_{\text{learn}}) = (24, 24, 16)$.

Throughout, for any finite vector $v = (v_1, \ldots, v_m)$, we write

$$S(v) = \big(\text{mean}(v),\ \text{std}(v),\ Q_{0.25}(v),\ \text{median}(v),\ Q_{0.75}(v)\big)$$

for the five-dimensional summary operator applied repeatedly below. The schema block contains the global descriptors: $\log(1 + p)$, categorical feature ratio, total missing fraction, together with the fraction of categorical features that are binary, and the summaries S(per-feature missing rate), S(numeric unique ratio), S(categorical unique ratio), S(categorical normalized entropy). Note that for this work, we do not encode schema information regarding the number of samples in the dataset, since we currently fix the number of samples generated for synthetic datasets to be a constant during model training. In future work, we plan to include these features of the dataset schema, but we exclude them for now for training stability purposes. These quantities preserve coarse structural properties of the dataset that should not be removed by downstream normalization.

Let $X_{\text{num}} \in \mathbb{R}^{n \times p_{\text{num}}}$ denote the numeric submatrix. For each numeric feature j, we compute the mean μ_j and standard deviation σ_j using observed entries only, impute missing values by μ_j, and form the standardized column

$$\widetilde{X}_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j}.$$

All geometric features are computed from $\widetilde{X}_{\text{num}}$, making this block invariant to per-feature affine transformations $X_{ij} \mapsto a_j X_{ij} + b_j$ with $a_j > 0$. The geometric block contains $S(|\text{feature skewness}|)$, $S(\text{feature kurtosis})$, and $S(\text{feature outlier rate})$, where the outlier rate is the fraction of observed standardized entries satisfying $|\widetilde{X}_{ij}| > 3$. To capture dependence structure, we form the correlation matrix R of the standardized numeric features and include

$$S\big(|R_{jk}| : j < k\big), \qquad \max_{j<k} |R_{jk}|,$$

together with three spectral summaries,

$$\frac{\lambda_1}{\sum_\ell \lambda_\ell}, \qquad \frac{\sum_{\ell=1}^{\min(5,r)} \lambda_\ell}{\sum_\ell \lambda_\ell}, \qquad \exp\Big(-\sum_\ell \pi_\ell \log \pi_\ell\Big),$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ are the eigenvalues of R, r is the rank, and $\pi_\ell = \lambda_\ell / \sum_k \lambda_k$. The final term is the entropy-based effective rank.

The learnability block augments these unsupervised statistics with inexpensive surrogates for task difficulty. Specifically, it includes #classes, $H(Y)$, $\max_c \Pr(Y = c)$, $\min_c \Pr(Y = c)$, where $H(Y) = -\sum_c \Pr(Y = c) \log \Pr(Y = c)$. It further includes the summaries $S(\eta_j^2)$ over numeric features, and $S(V_j)$ over categorical features, where η_j^2 is the ANOVA effect size between numeric feature j and the class label, and V_j is Cramér's V between categorical feature j and the class label. Finally, we include two global proxies for separability: leave-one-out nearest-centroid accuracy, and the ratio

$$\frac{\text{mean pairwise distance between class centroids}}{\text{mean within-class distance}}.$$

Given a collection of reference datasets $\{(X^{(i)}, y^{(i)})\}_{i=1}^N$, we first compute raw embeddings

$$u^{(i)} = \phi(X^{(i)}, y^{(i)}) \in \mathbb{R}^d.$$

For each coordinate k, we then estimate the reference-set mean and variance,

$$\mu_k = \frac{1}{N} \sum_{i=1}^N u_k^{(i)}, \qquad s_k^2 = \frac{1}{N} \sum_{i=1}^N (u_k^{(i)} - \mu_k)^2,$$

and standardize each embedding coordinate according to

$$z_k^{(i)} = \frac{u_k^{(i)} - \mu_k}{s_k}.$$

Any generated dataset is mapped into the same latent space using the *same* (μ_k, s_k) estimated from the reference collection. After this per-coordinate standardization, we apply an explicit block reweighting. Writing

$$z = \begin{bmatrix} z_{\text{schema}} \\ z_{\text{geom}} \\ z_{\text{learn}} \end{bmatrix}, \qquad z_b \in \mathbb{R}^{d_b},$$

we define the final transformed representation by

$$\widehat{z}_b = \sqrt{\frac{w_b}{d_b}}\, z_b,$$

where $w_b > 0$ is a user-specified block weight. Distances are then evaluated in the transformed space:

$$\|\widehat{z} - \widehat{z}'\|_2^2 = \sum_b \frac{w_b}{d_b} \|z_b - z_b'\|_2^2.$$

This normalization prevents large blocks from dominating the geometry solely by virtue of dimensionality. If each standardized coordinate has variance approximately one and within-block correlations are not too strong, then

$$\mathbb{E}\big[\|z_b\|_2^2\big] \approx d_b, \qquad \mathbb{E}\big[\|\widehat{z}_b\|_2^2\big] = \frac{w_b}{d_b}\, \mathbb{E}\big[\|z_b\|_2^2\big] \approx w_b.$$

Thus the expected squared contribution of block b is controlled by w_b rather than by its raw dimension. In practice, we set $w_b = \frac{1}{3}$ for each of the three categories, balancing the contribution of each to the distance metric.

Parameter Search for Partial Coverage. With our dataset encoder and distance metric defined, we now introduce our search method for finding parameterizations of synthetic data generators that align with real-world data from some reference corpus. Since this is an unconstrained black-box optimization problem over a discrete search space, we use Bayesian Optimization to find parameters that cover parts of the reference corpus well [15]. To accomplish this, we now define our surrogate objective function that rewards partial coverage of the embedding space spanned by the reference corpus. For a given reference corpus of datasets, let $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ denote its standardized reference embeddings after the same preprocessing and block reweighting described in the previous section. A trial parameter vector θ induces a *set* of synthetic datasets; in the implementation, we generate m datasets from $\phi \sim p(\Phi; \theta)$, embed them, and obtain a cloud $\{\mathbf{x}_j(\theta)\}_{j=1}^m \subset \mathbb{R}^d$. The search objective is then defined in terms of how well this cloud covers the reference embeddings.

For each reference point $\mathbf{x}_i$, the we compute a soft nearest-neighbor distance to the generated cloud,

$$d_i(\theta) = -\tau \log\left(\frac{1}{m}\sum_{j=1}^{m} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j(\theta)\|_2}{\tau}\right)\right), \tag{9}$$

with temperature $\tau > 0$. Distances are Euclidean and $\tau = 1$ by default. As $\tau \to 0$, $d_i(\theta)$ approaches $\min_j \|\mathbf{x}_i - \mathbf{x}_j(\theta)\|_2$, while for larger τ it becomes a smoother average over nearby generated embeddings. This gives a differentiable surrogate for pointwise coverage without requiring an exact minimum.

The scalar objective returned to the optimizer at iteration t is a residual-weighted average of these pointwise distances,

$$f_t(\theta) = -\sum_{i=1}^{n} r_i^{(t)} d_i(\theta), \qquad r_i^{(t)} \geq 0, \qquad \sum_{i=1}^{n} r_i^{(t)} = 1. \tag{10}$$

Maximizing f_t is therefore equivalent to minimizing a weighted coverage loss. The weights $r_i^{(t)}$, which we refer to as the *residuals*, capture which parts of the reference data remain poorly covered by the synthetic data generators found so far. To define these residual weights, we track the coverage distance $c_i^{(t)}$ for each reference point. After evaluating a new parameter vector θ_t, the candidate distances $d_i(\theta_t)$ are merged with the existing coverage state by a hard minimum,

$$c_i^{(t+1)} = \begin{cases} d_i(\theta_t), & t = 0, \\ \min\big(c_i^{(t)},\, d_i(\theta_t)\big), & t \geq 1. \end{cases} \tag{11}$$

Thus $c_i^{(t)}$ records the best coverage attained so far by any previously evaluated generator in that cluster. The residual weights for the next trial are then recomputed from the updated coverage distances. The update is the linear normalization

$$r_i^{(t+1)} = \frac{c_i^{(t+1)}}{\sum_{\ell=1}^{n} c_\ell^{(t+1)}}. \tag{12}$$

Reference points that are already well covered have small $c_i^{(t+1)}$ and therefore receive little residual mass, whereas persistently under-covered points retain large weights. The initial residual state is uniform, $r_i^{(0)} = 1/n$. Consequently, early trials try to improve average coverage of the whole space, while later trials are pushed toward the portions of embedding space that remain inadequately represented. This is why we refer to the score as a *partial coverage* surrogate, since a single generator is not required to match the entire reference corpus by itself. Instead, the sequential parameter search encourages successive generators to cover complementary regions of the target embedding distribution. In practice, as we can see in Figure 3, there are no generators the adequately cover the entire space spanned by the reference corpus. Instead, by finding generators that partially cover this space and combining them, we are able to sufficiently cover the reference corpus.

Solving the Global Mixture Problem. After the parameter search from the previous section is performed, we obtain T candidate generators. Generator t is represented by a cloud of embedded

synthetic datasets $\{\boldsymbol{x}_j(\theta_t)\}_{j=1}^{m_t} \subset \mathbb{R}^d$, while the target collection of real datasets is represented by reference embeddings $\{\boldsymbol{x}_i\}_{i=1}^{n} \subset \mathbb{R}^d$. In practice, we set $m_t = m$ as a constant for consistency across trials. We want to fit a global distribution over generators by choosing mixture weights $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_T)$ on the simplex that minimizes the transport distance between this global distribution over synthetic data generators and the distribution over the reference datasets.

The reference embeddings define the empirical measure

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \delta(\boldsymbol{x}_i), \tag{13}$$

and a candidate generator mixture induces

$$\nu_{\boldsymbol{\alpha}} = \sum_{k=1}^{T} \alpha_t \left(\frac{1}{m_t} \sum_{j=1}^{m_t} \delta(\boldsymbol{x}_j(\theta_t)) \right). \tag{14}$$

Thus each synthetic table embedding produced by generator k receives mass α_t/m_t. We compare μ and $\nu_{\boldsymbol{\alpha}}$ using squared Euclidean cost $c(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_2^2$ and the entropically regularized optimal transport objective

$$\mathrm{OT}_\varepsilon(\mu, \nu_{\boldsymbol{\alpha}}) = \min_{\boldsymbol{\Pi} \in U(\boldsymbol{a}, \boldsymbol{b}(\boldsymbol{\alpha}))} \langle \boldsymbol{\Pi}, \boldsymbol{C} \rangle + \varepsilon \sum_{i,j} \Pi_{ij}(\log \Pi_{ij} - 1), \tag{15}$$

where $\varepsilon > 0$ is the regularization parameter, $\boldsymbol{a}_i = 1/n$, $\boldsymbol{b}(\boldsymbol{\alpha})$ collects the per-sample masses α_t/m_t, and $U(\boldsymbol{a}, \boldsymbol{b})$ is the set of couplings with marginals $\boldsymbol{a}$ and $\boldsymbol{b}$. The entropic term makes the problem strictly convex in the transport plan and enables efficient solution by Sinkhorn matrix scaling [11].

Rather than solving the transport problem from scratch by general-purpose linear programming, we use the Sinkhorn algorithm in the log domain. Writing $C_{ij} = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2^2$, the dual potentials are updated as

$$f_i \leftarrow \varepsilon \left(\log a_i - \log \sum_j \exp\left(\frac{-C_{ij} + g_j}{\varepsilon} \right) \right), \tag{16}$$

$$g_j \leftarrow \varepsilon \left(\log b_j - \log \sum_i \exp\left(\frac{-C_{ij} + f_i}{\varepsilon} \right) \right), \tag{17}$$

which yields the transport plan $\Pi_{ij} = \exp((-C_{ij} + f_i + g_j)/\varepsilon)$. This log-domain form is numerically stable and gives an efficient inner solver for repeated OT evaluations.

We then solve the outer mixture problem by minimizing the regularized transport cost with respect to $\boldsymbol{\alpha}$:

$$\min_{\boldsymbol{\alpha} \in \Delta^{T-1}} \mathrm{OT}_\varepsilon(\mu, \nu_{\boldsymbol{\alpha}}). \tag{18}$$

In practice, we reparameterize $\boldsymbol{\alpha}$ by unconstrained logits $\boldsymbol{\eta} \in \mathbb{R}^T$ with $\alpha_t = \exp(\eta_t)/\sum_{\ell=1}^{T} \exp(\eta_\ell)$, and optimize $\boldsymbol{\eta}$ by gradient descent. Each outer step constructs the sample-weight vector $\boldsymbol{b}(\boldsymbol{\alpha})$, calls the Sinkhorn solver to evaluate $\mathrm{OT}_\varepsilon(\mu, \nu_{\boldsymbol{\alpha}})$, and backpropagates through that solution to update the logits. In this way, Sinkhorn serves as the inner OT subroutine inside the global mixture optimization. The result is a set of mixture weights over generators whose aggregated embedding distribution best matches the reference embeddings under the entropically regularized transport geometry.

B RAD-TFM Algorithm Details

We summarize ROBUST AND DOMAIN-ADAPTED TABULAR FOUNDATION MODELS in Algorithms 3 and 4. A description of the algorithm is given in Section 4. To begin, we perform the domain-adaption search from given by Algorithm 1 to construct a reference distribution P over SCM parameters that aligns the synthetic data with a given reference corpus of real data. We then alternate between a maximization stage, in which we search over SCM parameters to find those with large optimality gaps, as in Equation 5, and a minimization stage, where we sample generators from

Algorithm 1 Pseudocode for Domain-Adapted TFMs

Require: Reference datasets $\mathcal{D}_{\text{ref}} = \{(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$, parameter space Θ, trials T, datasets per trial m, soft-coverage temperature τ, transport regularization ε, Sinkhorn iterations L, outer iterations O

Compute reference embeddings $\{\boldsymbol{x}_i\}_{i=1}^{N}$ from $\mathcal{D}_{\text{ref}}$
Standardize and reweight the embedding coordinates using statistics estimated from $\mathcal{D}_{\text{ref}}$
Initialize an empty list of candidate generators and generated embedding clouds
Initialize residual weights uniformly as $\boldsymbol{r}^{(0)} = \frac{1}{N}$
$\{(\theta_t, \{\mathbf{x}_j(\theta_t)\}_{j=1}^{m})\}_{t=1}^{T} \leftarrow \text{DomainParamSearch}(\{\boldsymbol{x}_i\}_{i=1}^{N}, \mathbf{r}, \Theta, m, \tau, T)$
Form the empirical reference measure μ from $\{\boldsymbol{x}_i\}_{i=1}^{N}$
For each candidate generator θ_t, form the empirical measure supported on its generated cloud $\mathcal{Y}_t = \frac{1}{m}\sum_{j=1}^{m} \delta(\boldsymbol{x}_j(\theta_t))$
Define $\nu_{\boldsymbol{\alpha}} = \sum_{t=1}^{T} \alpha_t \mathcal{Y}_t$ as the global mixture over synthetic data generators
Solve $\boldsymbol{\alpha}^{\star} \in \arg\min_{\boldsymbol{\alpha} \in \Delta^{M-1}} \text{OT}_{\varepsilon}\big(\mu, \nu_{\boldsymbol{\alpha}}\big)$ using gradient-based optimization over mixture logits, with Sinkhorn$(\mu, \nu_{\boldsymbol{\alpha}}, \varepsilon, L)$ as the inner OT solver, for O outer iterations
return Global mixture over generator parameters $\boldsymbol{\alpha}^{\star}$

Algorithm 2 DOMAINPARAMSEARCH

Require: Reference embeddings $\{\boldsymbol{x}_i\}_{i=1}^{N}$, initial residuals $\boldsymbol{r}^{(0)}$, parameter space Θ, datasets per trial m, soft-coverage temperature τ, trials T

Initialize cover distances $\boldsymbol{c}^{(0)} = (+\infty, \ldots, +\infty)$
Initialize history $\mathcal{H}^{(0)} = \emptyset$
for $t = 1, \ldots, T$ **do**
    $\theta_t \leftarrow \text{BayesOPT}(\mathcal{H}^{(t-1)}, \Theta)$
    Generate m synthetic datasets from θ_t, compute their embeddings, and collect them as $\{\mathbf{x}_j(\theta_t)\}_{j=1}^{m}$
    Compute the pointwise soft coverage distances $\{d_i(\theta_t)\}_{i=1}^{N}$ as in Equation 9
    Compute the partial coverage reward $f_t(\theta_t) = -\sum_{i=1}^{N} r_i^{(t-1)} d_i(\theta_t)$
    for $i = 1, \ldots, N$ **do**
        $c_i^{(t)} \leftarrow \min\{c_i^{(t-1)}, d_i(\theta_t)\}$
    end for
    for $i = 1, \ldots, N$ **do**
        $r_i^{(t)} \leftarrow c_i^{(t)} \big/ \sum_{\ell=1}^{N} c_\ell^{(t)}$
    end for
    Update the BO history as $\mathcal{H}^{(t)} = \mathcal{H}^{(t-1)} \cup \{(\theta_t, f_t(\theta_t))\}$
end for
return Parameters and generated dataset point clouds $\{(\theta_t, \{\mathbf{x}_j(\theta_t)\}_{j=1}^{m})\}_{t=1}^{T}$

an adversarial distribution to continue training the TFM. In Algorithm 4, we reference an abstract function SuggestParameters which represents any black-box optimization algorithm. In this case, we use the Optuna python package [1] and the Tree-structed Parzen Estimator [36] algorithm, which belongs to the class of Bayesian Optimization algorithms [15]. At each iteration, this algorithm uses the information gained from the previous trials (the estimated optimality gaps) to propose a new parameterization, balancing exploration and exploitation to efficiently find parameters corresponding to large optimality gaps. To begin the minimization stage, we first construct the distribution $Q \in \Delta_{\mathcal{P}}$. Rather than constructing ρ as a parameter itself, we introduce a scalar parameter $c \in (0, 1)$ and define $\rho = c \cdot \rho_{\max}$, such that ρ is a given as a fraction of the maximum possible KL-divergence of Q given the reference distribution P. With $\rho_{\max}$ specified, we can find η such that $D_{KL}(Q(\eta)||P) = \rho_{\max}$ using one-dimensional root finding (e.g. bisection). Then we generate each dataset in a training batch by sampling a parameter $\theta_i \sim Q$ and generator $\phi \sim p(\Phi; \theta_i)$. We train the model for a fixed number of iterations and then return to the maximization stage. In our implementation, after five epochs, we incorporate the original TabPFN model as another baseline model. This can potentially

Algorithm 3 ROBUST AND DOMAIN-ADAPTED TABULAR FOUNDATION MODELS (RAD-TFM)

Require: Initial TFM weights $\mathbf{W}$, baseline estimators $\{f_1, \ldots, f_e\}$, n_e, n_{iter}, n_t, n_d, $c \in (0, 1)$, reference distribution P from Algorithm 1, parameter space Θ

$\{(\theta_i, \delta_i)\}_{i=1}^{n_{trials}} \leftarrow$ ParameterSearch$(\Theta, \mathbf{W}, \{f_1, \ldots, f_e\}, n_t, n_d)$
$\rho \leftarrow c \cdot \rho_{\max}$
$\eta \leftarrow$ RootFinding(ρ, P, δ) ▷ Find correct temperature for distribution
Construct distribution $Q_i \propto p_i \exp(\eta \cdot \delta_i)$
for $e \in [n_e]$ **do**
  for $t \in [n_{iter}]$ **do**
    $\theta_1, \ldots, \theta_{n_b} \sim Q$ ▷ Sample parameters
    $\phi_1, \ldots, \phi_{n_b} \sim p(\Phi; \theta_1), \ldots, p(\Phi; \theta_{n_b})$
    $\{(x^i, y^i)\}_{i=1}^{n_b} \sim p(\phi_1), \ldots, p(\phi_{n_b})$ ▷ Sample synthetic datasets
    $\{x^i_{train}, x^i_{test}, y^i_{train}, y^i_{test}\}_{i=1}^{n_b} \leftarrow$ TrainTestSplit$(\{(x^i, y^i)\}_{i=1}^{n_b})$
    $l(g_{\mathbf{W}}) \leftarrow \mathcal{L}_{PFN}(g_{\mathbf{W}}, \{x^i_{train}, x^i_{test}, y^i_{train}, y^i_{test}\}_{i=1}^{n_b})$
    $\mathbf{W} \leftarrow$ update$(\mathbf{W}, l(g_{\mathbf{W}}))$ ▷ Gradient step
  end for
  $\{(\theta_i, \delta_i)\}_{i=1}^{n_{trials}} \leftarrow$ ParameterSearch$(\mathbf{W}, \{f_1, \ldots, f_e\}, n_t, n_d)$
  $\eta \leftarrow$ RootFinding(H_{min}, δ)
  Construct distribution $Q_i \propto p_i \exp(\eta \cdot \delta_i)$
end for
Return Robust TFM model $g_{\mathbf{W}}$

address regions of the parameter space where the model has reduced its performance by focusing on improving in other regions.

The RAD-TFM algorithm is highly parallelizable, both in the dataset generation steps and the parameter search. For a fixed parameterization θ, each dataset generator $\phi \sim p(\Phi; \theta)$ is sampled independently from the same distribution, such that all datasets generated for a given batch can be created concurrently. The ParameterSearch algorithm can be completely parallelized within each trial. In our implementation, for each trial, we generate a batch of datasets and then fit and evaluate each model and dataset pair concurrently. For example, in our experiments, we set $n_d = 20$ and used $e = 7$ baseline models, resulting in $n_{ds} * e = 140$ separate processes distributed across all available cores. In practice, the domain-adaptation stage is lightweight, while the overall training cost scales approximately linearly with the number of max-min iterations; we provide benchmark-specific compute details in Appendix G.

Algorithm 4 ParameterSearch

Require: TFM weights $\mathbf{W}$, baseline estimators $\{f_1, \ldots, f_e\}$, num. trials n_t, num. datasets n_d

$D_\theta = \{\}$
for $i \in [n_t]$ **do**
  $\theta_i \leftarrow$ SuggestParameters(D_θ) ▷ Black-box optimizer proposes new parameter
  for $j \in [n_d]$ **do**
    $\phi_j \sim p(\Phi; \theta_i)$
    $x^j_{train}, x^j_{test}, y^j_{train}, y^j_{test} \sim p(\phi_j)$
    $l_j(g_{\mathbf{W}}) \leftarrow \mathcal{L}_{PFN}(g_{\mathbf{W}}, x^j_{train}, y^j_{train}, x^j_{test}, y^j_{test})$
    $l_b \leftarrow \infty$
    for $f_k \in \{f_1, \ldots, f_e\}$ **do**
      $l_j(f_k) \leftarrow \mathcal{L}_{PFN}(f_k, x^j_{train}, y^j_{train}, x^j_{test}, y^j_{test})$
      $l_b \leftarrow \min(l_b, l_j(f_k))$
    end for
    $\delta^j_i \leftarrow l_j(g_{\mathbf{W}}) - l_b$ ▷ Estimate optimality gap
  end for
  $\delta_i \leftarrow \frac{1}{n_d} \sum_{j=1}^{n_d} \delta^j_i$
  $D_\theta \leftarrow D_\theta \cup \{(\theta_i, \delta_i)\}$
end for
Return Parameters and estimated optimality gaps D_θ

C Theoretical Properties

In this section, we provide the proof that the optimal distribution for the maximizer in equation 5 is a softmax distribution such that $q_i^* \propto p_i \exp(\eta \cdot \delta_{\theta_i}(\mathbf{W}))$. Further, we show that in the absence of a reference distribution, if we take P to be uniform, our formulation can be written as an entropy-constrained DRO, where Q is constrained to have a certain minimum entropy.

Proposition 1. *Let $\delta = (\delta_{\theta_1}(\mathbf{W}), \ldots, \delta_{\theta_n}(\mathbf{W})) \in \mathbb{R}^n$ and let $P = (p_1, \ldots, p_n) \in \Delta_n$ satisfy $p_i > 0$ for all i. Consider the distributionally robust inner problem*

$$\max_{Q \in \Delta_n} \sum_{i=1}^{n} q_i \delta_{\theta_i}(\mathbf{W}) \quad \textit{s.t.} \quad D_{\mathrm{KL}}(Q \| P) \leq \rho,$$

Let $\mathcal{A} = \arg\max_{i \in [n]} \delta_{\theta_i}(\mathbf{W})$, and $P(\mathcal{A}) = \sum_{i \in \mathcal{A}} p_i$. Assume that $\delta_{\theta_i}(\mathbf{W})$ is not constant across i and $0 < \rho < -\log P(\mathcal{A})$. Then the following properties hold.

(i) The optimizer Q^* is strictly positive, and there exists a unique temperature parameter $\eta > 0$ such that

$$q_i^* = \frac{p_i \exp\left(\delta_{\theta_i}(\mathbf{W})/\eta\right)}{\sum_{j=1}^{n} p_j \exp\left(\delta_{\theta_j}(\mathbf{W})/\eta\right)}, \qquad i = 1, \ldots, n,$$

where η is determined implicitly by $D_{\mathrm{KL}}(Q^* \| P) = \rho$.

(ii) If P is the uniform distribution on $[n]$, then the KL constraint $D_{\mathrm{KL}}(Q \| P) \leq \rho$ is equivalent to the entropy constraint $H(Q) \geq \log n - \rho$. In this case, the optimizer in part (i) reduces to the softmax distribution

$$q_i^* = \frac{\exp\left(\delta_{\theta_i}(\mathbf{W})/\eta\right)}{\sum_{j=1}^{n} \exp\left(\delta_{\theta_j}(\mathbf{W})/\eta\right)}, \qquad i = 1, \ldots, n,$$

with $\eta > 0$ chosen so that $H(Q^*) = \log n - \rho$.

Proof. **Part (i)**. For notational simplicity, write $\delta_i = \delta_{\theta_i}(\mathbf{W})$. The problem can be rewritten as

$$\min_{Q \in \Delta_n} -\sum_{i=1}^{n} q_i \delta_i \quad \text{s.t.} \quad D_{\mathrm{KL}}(Q \| P) \leq \rho.$$

Since $P \in \Delta_n$ satisfies $p_i > 0$ for all i and

$$D_{\mathrm{KL}}(P \| P) = 0 < \rho,$$

Slater's condition holds. Therefore, the KKT conditions are necessary and sufficient.

The Lagrangian is

$$\mathcal{L}(Q, \eta, \mu, \gamma) = -\sum_{i=1}^{n} q_i \delta_i + \eta \left(\sum_{i=1}^{n} q_i \log \frac{q_i}{p_i} - \rho \right) + \mu \left(\sum_{i=1}^{n} q_i - 1 \right) - \sum_{i=1}^{n} \gamma_i q_i,$$

where $\eta \geq 0$, $\mu \in \mathbb{R}$, and $\gamma_i \geq 0$.

We first show that the KL constraint must be active. If the KL constraint were inactive, then $\eta = 0$. In that case, the solution would have to maximize the linear objective over the simplex without an active KL constraint, and hence would place all mass on the set

$$\mathcal{A} = \arg\max_{i \in [n]} \delta_i.$$

Among all distributions supported on $\mathcal{A}$, the one with smallest KL divergence from P is the conditional distribution

$$q_i = \frac{p_i}{P(\mathcal{A})} \mathbf{1}\{i \in \mathcal{A}\},$$

whose KL divergence is

$$D_{\mathrm{KL}}(Q \| P) = \sum_{i \in \mathcal{A}} \frac{p_i}{P(\mathcal{A})} \log \frac{p_i / P(\mathcal{A})}{p_i} = -\log P(\mathcal{A}).$$

Thus, if $\rho < -\log P(\mathcal{A})$, no distribution supported entirely on $\mathcal{A}$ is feasible. Therefore the unconstrained maximizer is infeasible, so the KL constraint must be active and $\eta > 0$.

For an interior optimum, complementary slackness gives $\gamma_i = 0$. The stationarity condition is then

$$-\delta_i + \eta\left(1 + \log\frac{q_i}{p_i}\right) + \mu = 0.$$

Rearranging,

$$\log\frac{q_i}{p_i} = \frac{\delta_i - \mu}{\eta} - 1.$$

Exponentiating gives

$$q_i = p_i \exp\left(\frac{\delta_i}{\eta}\right)\exp\left(-\frac{\mu}{\eta} - 1\right).$$

Enforcing $\sum_i q_i = 1$ yields

$$q_i(\eta) = \frac{p_i \exp(\delta_i/\eta)}{Z(\eta)}, \qquad Z(\eta) = \sum_{j=1}^{n} p_j \exp(\delta_j/\eta).$$

Since $p_i > 0$ and $\eta > 0$, this distribution is strictly positive.

It remains to show that there is a unique $\eta > 0$ satisfying $D_{\mathrm{KL}}(Q(\eta)\|P) = \rho$. Let $E(\eta) = \sum_{i=1}^{n} q_i(\eta)\delta_i$. Then, For the distribution $Q(\eta)$,

$$\log\frac{q_i(\eta)}{p_i} = \frac{\delta_i}{\eta} - \log Z(\eta),$$

so

$$D_{\mathrm{KL}}(Q(\eta)\|P) = \sum_{i=1}^{n} q_i(\eta)\left(\frac{\delta_i}{\eta} - \log Z(\eta)\right) = \frac{E(\eta)}{\eta} - \log Z(\eta).$$

Next,

$$\frac{d}{d\eta}\log Z(\eta) = \frac{1}{Z(\eta)}\sum_{i=1}^{n} p_i \exp(\delta_i/\eta)\left(-\frac{\delta_i}{\eta^2}\right) = -\frac{E(\eta)}{\eta^2}.$$

And,

$$\frac{dE}{d\eta} = -\frac{\mathrm{Var}_{Q(\eta)}(\delta)}{\eta^2}.$$

Therefore,

$$\begin{aligned}\frac{d}{d\eta}D_{\mathrm{KL}}(Q(\eta)\|P) &= \frac{1}{\eta}\frac{dE}{d\eta} - \frac{E(\eta)}{\eta^2} - \frac{d}{d\eta}\log Z(\eta) \\ &= \frac{1}{\eta}\frac{dE}{d\eta} - \frac{E(\eta)}{\eta^2} + \frac{E(\eta)}{\eta^2} \\ &= -\frac{\mathrm{Var}_{Q(\eta)}(\delta)}{\eta^3}.\end{aligned}$$

Since δ_i is not constant across i and $Q(\eta)$ is strictly positive, we have

$$\mathrm{Var}_{Q(\eta)}(\delta) > 0$$

for every $\eta > 0$. We can then conclude that $\eta \mapsto D_{\mathrm{KL}}(Q(\eta)\|P)$ is strictly decreasing on $(0, \infty)$.

Further, we have $\lim_{\eta\to\infty} Q(\eta) = P$, so $\lim_{\eta\to\infty} D_{\mathrm{KL}}(Q(\eta)\|P) = 0$. As $\eta \downarrow 0$, $Q(\eta)$ converges to P conditioned on the set of maximizers $\mathcal{A}$, namely

$$q_i(0) = \frac{p_i}{P(\mathcal{A})}\mathbf{1}\{i \in \mathcal{A}\},$$

and therefore

$$\lim_{\eta\downarrow 0} D_{\mathrm{KL}}(Q(\eta)\|P) = -\log P(\mathcal{A}).$$

Thus, for every $0 < \rho < -\log P(\mathcal{A})$ there exists a unique $\eta > 0$ such that $D_{\mathrm{KL}}(Q(\eta)\|P) = \rho$. By the KKT conditions, this $Q(\eta)$ is the global optimizer.

Part (ii). Suppose P is uniform, so $p_i = 1/n$ for all i. Then

$$\begin{aligned} D_{\mathrm{KL}}(Q\|P) &= \sum_{i=1}^{n} q_i \log \frac{q_i}{1/n} \\ &= \sum_{i=1}^{n} q_i \log q_i + \sum_{i=1}^{n} q_i \log n \\ &= \sum_{i=1}^{n} q_i \log q_i + \log n \\ &= \log n - H(Q). \end{aligned}$$

Therefore, $D_{\mathrm{KL}}(Q\|P) \leq \rho$ is equivalent to $\log n - H(Q) \leq \rho$, or $H(Q) \geq \log n - \rho$. Finally, substituting $p_i = 1/n$ into the solution from part (i) gives

$$q_i^* = \frac{(1/n)\exp(\delta_i/\eta)}{\sum_{j=1}^{n}(1/n)\exp(\delta_j/\eta)} = \frac{\exp(\delta_i/\eta)}{\sum_{j=1}^{n}\exp(\delta_j/\eta)},$$

with $\eta > 0$ chosen so that $D_{\mathrm{KL}}(Q^*\|P) = \rho$. Using $D_{\mathrm{KL}}(Q^*\|P) = \log n - H(Q^*)$, this is equivalently $H(Q^*) = \log n - \rho$. This proves the claim. □

D SCM Parameters

We summarize the MLP-based SCM parameter space used in our experiments in Table 6. In this work, we instantiate SCMs as randomized MLPs, so the search space includes both architectural parameters and dataset-level properties. The MLP-specific parameters are the hidden width μ_h, number of layers μ_l, input width $\mu_{x_{\text{in}}}$, activation function μ_a, shuffled categorical ratio $\mu_{p_{\text{shuffle}}}$, node noise scale μ_{noise}, and layer sparsity μ_{sparsity}. These are combined with shared dataset parameters controlling the number of observed features μ_{z_x}, number of classes μ_{n_c}, mean number of categories $\mu_{\text{mean_cat}}$, categorical feature ratio $\mu_{p_{\text{cat}}}$, missing-value ratio $\mu_{p_{\text{missing}}}$, missingness mechanism $\mu_{\text{missingness_type}}$, class-balance concentration $\mu_{\text{class_balance}}$, weight scaling $\mu_{\text{weight_scaling}}$, and input distribution μ_{dist}. Integer-valued parameters are sampled from truncated normal distributions over the ranges shown in Table 6 and then rounded to the nearest integer, while continuous parameters are sampled from truncated normals on their admissible intervals. Categorical parameters are sampled with an ϵ-greedy scheme: with probability $1 - \epsilon$ we select the reference value and with probability ϵ we sample uniformly from the remaining choices. We also implemented Tree-based SCMs, as in related literature, but found that they did not make a significant contribution to our RAD-TFM pipeline, so we exclude them here for brevity [31].

E Learned SCM Parameter Configurations

Different Domains Require Different SCM Parameters. In Table 7, we show a subset of the parameters used to configure the SCM generators, which determine the shape, geometry, and learnability of the generated datasets. For each benchmark, we show the parameters corresponding to the setting with the highest weight, as in Equation 5, the optimal response of the adversary when constrained to generate data that is similar to the real-world reference data. In the table, SCM size refers the dimensions of the MLP generating the data, which we split into five categorical values, balance refers to the class imbalance (higher corresponds to greater class balance), and noise refers to the scale of the noise added to the output of each neuron. We can clearly observe the variety in parameterizations across benchmarks, some of which align with what we know about the real-world datasets from that domain. For example, for the Genomics benchmark, most datasets have $d = 500$ numeric features, and the domain-adaptation algorithm has correctly constrained the parameter space to datasets with many features (we limit the number of generated features to $d = 200$ per dataset). While some parameters, such as the number of features, may be easier to interpret and adjust manually, for others, such as network sparsity or the activation function, it would be difficult to interpret or understand how changing these values would alter the generated datasets. We can see however, both in Table

Table 6: MLP-based SCM parameter space and sampling distributions. Numeric parameters are sampled from truncated normal distributions centered at the corresponding mean hyperparameter. Categorical parameters are sampled with an ϵ-greedy rule with $\epsilon = 0.3 \cdot \sigma$.

Parameter	Type	Range	Distribution
Mean hidden size μ_h	integer	[1, 256]	TruncNorm($\mu_h, 10 \cdot \sigma$)
Mean number of layers μ_l	integer	[1, 12]	TruncNorm(μ_l, σ)
Mean input width $\mu_{x_{\text{in}}}$	integer	[1, 15]	TruncNorm($\mu_{x_{\text{in}}}, 2 \cdot \sigma$)
Activation function μ_a	categorical	{`relu`, `elu`, `tanh`, `identity`}	ϵ-greedy
Mean number of features μ_{z_x}	integer	[2, 200]	TruncNorm($\mu_{z_x}, s_{z_x} \cdot \sigma$)
Mean number of classes μ_{n_c}	integer	[2, 10]	TruncNorm(μ_{n_c}, σ)
Mean number of categories $\mu_{\text{mean_cat}}$	integer	[2, 120]	TruncNorm($\mu_{\text{mean_cat}}, 2 \cdot \sigma$)
Categorical feature ratio $\mu_{p_{\text{cat}}}$	continuous	[0, 1]	TruncNorm($\mu_{p_{\text{cat}}}, 0.1 \cdot \sigma$)
Shuffled categorical ratio $\mu_{p_{\text{shuffle}}}$	continuous	[0, 1]	TruncNorm($\mu_{p_{\text{shuffle}}}, 0.1 \cdot \sigma$)
Missing-value ratio $\mu_{p_{\text{missing}}}$	continuous	[0, 1]	TruncNorm($\mu_{p_{\text{missing}}}, 0.1 \cdot \sigma$)
Missingness type $\mu_{\text{missingness_type}}$	categorical	{`mcar`, `featurewise`, `self_masking`}	ϵ-greedy
Class-balance concentration $\mu_{\text{class_balance}}$	continuous	[0.1, 10]	TruncNorm($\mu_{\text{class_balance}}, 1.5 \cdot \sigma$)
Node noise scale μ_{noise}	continuous	[0, 1]	TruncNorm($\mu_{\text{noise}}, 0.1 \cdot \sigma$)
Layer sparsity μ_{sparsity}	continuous	[0, 0.9]	TruncNorm($\mu_{\text{sparsity}}, 0.1 \cdot \sigma$)
Weight scaling $\mu_{\text{weight_scaling}}$	categorical	{`None`, `Small`, `Large`}	ϵ-greedy
Input distribution μ_{dist}	categorical	{`exponential`, `uniform`, `normal`}	ϵ-greedy

Table 7: SCM generator parameters with the highest weight in the first iteration of the RAD-TFM algorithm for various benchmarks. Different benchmarks correspond to significantly different data generation regimes. We report only a subset of the parameters in this table.

Benchmark	SCM Size	# Feat.	# Class	Act.	Sparsity	p_{cat}	Cat. Dim.	p_{miss}	Balance	Noise
TabPertNet[40]	Medium	13	2	tanh	0.9	0.0	-	0.4	3	0.0
TabArena[14]	Medium	44	2	tanh	0.9	0.5	50	0.2	3	0.0
PMLB[26]	Small	12	10	tanh	0.9	0.0	-	0.2	3	0.0
Engineering	Large	65	2	elu	0.0	0.0	-	0.0	10	0.01
Genomics	Large	200	2	relu	0.1	0.0	-	0.4	3	0.05
Healthcare	Small	5	2	identity	0.0	1.0	2	0.2	10	0.5

7 and Figure 3, that our domain-adaptation algorithm is not only able to adjust these parameters automatically to align with and cover the real-world datasets in the meta-feature space, but that it is necessary to do so, as their values vary significantly between domains.

F Generator Coverage on Foreign-Exchange Data

Effect of the Generator Family. To investigate the limitations imposed by the synthetic generator family, we consider a corpus of 64 foreign-exchange datasets from OpenML [4], derived from daily candlestick price series. These datasets exhibit strong temporal dependencies that are not naturally represented by the non-temporal SCMs used in our main experiments. With the original generators, the entropy-regularized optimal transport alignment objective between the real and synthetic dataset embeddings is approximately $S_d = 2$, compared with values below 0.2 on the other evaluated domains. We also observe no empirical coverage ($C = 0$), where coverage is the fraction of reference datasets whose nearest synthetic embedding is at least as close as a resampled real-data neighbor, averaged over resampling trials.

To expand the generator family, we introduce a temporal SCM with controllable latent dimension, autoregressive persistence, feature-noise ratio, maximum lag, prediction horizon, and target signal strength. This extended parameter space reduces the alignment objective to approximately $S_d = -0.6$ and increases coverage to $C = 0.05$. Although coverage remains limited, the additional temporal structure permits closer alignment with the reference domain. Table 8 reports the resulting mean test AUC. Neither fine-tuning nor RAD-TFM with the original generators improves upon the frozen backbone; in contrast, the temporal extension yields a modest improvement. The results illustrate that reweighting generators cannot compensate for missing data-generating mechanisms, while expanding

the SCM family can extend the range of domains amenable to robust, domain-adapted continued training.

Table 8: Mean test AUC across 64 foreign-exchange datasets, comparing the original and expanded synthetic generator families.

Model	Mean AUC	Change vs. base
TabPFN V2	0.6602	—
TabPFN V2 (FT)	0.6551	-0.0051
TabPFN V2 (RAD-TFM; original SCMs)	0.6597	-0.0005
TabPFN V2 (RAD-TFM; temporal SCM)	**0.6652**	$\mathbf{+0.0050}$

G Additional Experiment Details

In Tables 9-16 we provide the full experiment results from the TabPertNet [40], TabArena [14], PMLB [26], and domain-specific OpenML [4] tabular benchmarks. For all benchmarks, we evaluate on classification datasets only, and limit our evaluation to datasets with at most $n = 10k$ samples and $p = 500$ features. We use the training data from each dataset during the domain-adaptation process to approximate the reference distribution for the given domain. We report our test results on $N = 70$ datasets from TabPertNet, $N = 91$ datasets from PMLB, $N = 21$ datasets from TabArena, $N = 31$ dataset for Healthcare, $N = 11$ datasets for Genomics, and $N = 20$ datasets for Engineering. The original genomics datasets have $O(10k)$ features, so we first preprocess these datasets by randomly selecting a subset of $d = 500$ features and down-sampling. For TabArena, we use the same folds as defined in the original work and corresponding leaderboard. For the OpenML datasets, we use the same train/test splits as provided from the database. For TabPertNet, we use the same train/test split used in the original paper. We report ROC-AUC for binary classification tasks and AUC one-vs-one (OVO) for multi-class tasks.

Compute Resources. All experiments were run on a single 80GB A100 GPU together with up to 140 CPU cores during the robust parameter-search stage. Constructing the domain-adapted reference distribution for a benchmark required approximately 10 minutes of GPU time, while each max-min iteration of the RAD-TFM algorithm required approximately 1 GPU-hour; we used 5 epochs for Genomics, 10 epochs for Engineering and Healthcare, and 30 epochs for TabPertNet, TabArena, and PMLB. The CPU-intensive parameter-search stage used roughly 140 cores for about 6 hours per full training run.

Licensing for existing assets. Our experiments use existing open-source benchmarks, datasets, models, and software packages. We cite the original sources for each benchmark and model in the main paper. TabArena is used under its publicly released benchmark resources; its code repository is licensed under Apache License 2.0, and the associated dataset curation repository is licensed under the BSD 3-Clause License. The TabArena authors further state that datasets whose licenses do not allow sharing or use in an academic benchmark are excluded from the benchmark. PMLB is publicly available under the MIT License. OpenML data and metadata are made available under OpenML's terms of use; OpenML states that empirical data and metadata are available under a CC-BY license, while individual dataset metadata include dataset-specific license fields. For TabPertNet, we use the publicly released dataset split from the original benchmark; the TP-BERTa [39] repository distributing the TabPertNet/OpenTabs subset states that the authors obtained permission to distribute the used subset. We use TabPFN-V2 weights under the Apache 2.0 license and the open-source TabICL V2 classifier under the BSD 3-Clause License. Optuna is licensed under the MIT License. We use all assets only for academic research and benchmarking, and we do not redistribute modified versions of these assets.

Table 9: TabPertNet [40] Test Results, Part 1 (AUC ↑).

	TabPFN V2			TabICL V2			Conventional baselines				
Dataset	**RAD-TFM**	**Base**	**FT**	**RAD-TFM**	**Base**	**FT**	**CatBoost**	**RF**	**XGB**	**LR**	**MLP**
BankNoteAuthenticati	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.998
Bank_Personal_Loan_M	0.597	0.578	0.578	0.618	0.619	0.604	0.600	0.601	0.599	0.606	0.590
Breast_Cancer	0.882	0.876	0.876	0.892	0.888	0.886	0.864	0.891	0.839	0.885	0.869
Customer_Behaviour	0.958	0.958	0.958	0.769	0.782	0.773	0.944	0.968	0.972	0.944	0.940
Employee Satisfactio	0.655	0.433	0.441	0.474	0.431	0.474	0.543	0.473	0.436	0.439	0.598
NFL	0.725	0.731	0.731	0.723	0.729	0.724	0.720	0.679	0.671	0.670	0.685
TravelInsurancePredi	0.708	0.670	0.668	0.716	0.688	0.701	0.708	0.546	0.631	0.715	0.723
UniversalBank	0.597	0.578	0.578	0.618	0.619	0.604	0.600	0.593	0.599	0.606	0.590
audit_data	0.997	0.998	0.998	1.000	1.000	1.000	1.000	1.000	1.000	0.992	0.981
audit_risk	0.997	0.998	0.998	1.000	1.000	1.000	1.000	1.000	1.000	0.992	0.981
b_depressed	0.468	0.447	0.445	0.557	0.521	0.532	0.578	0.574	0.513	0.446	0.601
bt_dataset_t3	1.000	0.999	0.999	0.997	0.999	0.999	0.998	0.995	0.995	0.969	0.814
loan	0.673	0.659	0.659	0.697	0.702	0.685	0.669	0.691	0.697	0.677	0.654
mechanical_analysis	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.949	0.940
new_model	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
piracydataset	0.573	0.572	0.571	0.607	0.593	0.606	0.592	0.606	0.607	0.606	0.486
0284_bank8FM	0.994	0.994	0.994	0.994	0.994	0.994	0.994	0.991	0.992	0.993	0.994
0292_cpu_small	0.980	0.980	0.980	0.982	0.982	0.983	0.979	0.975	0.978	0.959	0.972
0312_cpu_act	0.987	0.988	0.988	0.989	0.988	0.988	0.984	0.983	0.986	0.970	0.982
0345_delta_ailerons	0.977	0.975	0.975	0.978	0.978	0.978	0.974	0.975	0.972	0.967	0.971
0356_delta_elevators	0.950	0.949	0.949	0.952	0.953	0.952	0.952	0.946	0.944	0.945	0.949
0400_analcatdata_sup	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.938	0.750
0419_pm10	0.643	0.438	0.433	0.762	0.640	0.659	0.567	0.677	0.570	0.404	0.562
0435_strikes	1.000	1.000	1.000	1.000	1.000	1.000	0.996	0.990	1.000	0.616	0.555
0437_quake	0.521	0.432	0.431	0.564	0.490	0.566	0.447	0.500	0.482	0.563	0.531
0445_arsenic-male-bl	0.744	0.600	0.600	0.756	0.667	0.700	0.533	0.500	0.778	0.411	0.711
0446_arsenic-female-	0.858	0.845	0.844	0.837	0.839	0.845	0.807	0.737	0.786	0.814	0.663
0472_analcatdata_mar	0.663	0.360	0.366	0.377	0.669	0.606	0.571	0.526	0.480	0.383	0.429
0509_pollen	0.512	0.500	0.502	0.530	0.512	0.523	0.503	0.498	0.530	0.456	0.528
0526_colleges_aaup	1.000	1.000	1.000	0.978	0.973	0.974	1.000	1.000	0.999	0.967	0.961
0541_plasma_retinol	0.585	0.591	0.591	0.585	0.602	0.591	0.636	0.699	0.352	0.625	0.511
0555_socmob	0.992	0.987	0.987	0.948	0.959	0.955	0.995	0.989	0.992	0.927	0.931
0885_compas-two-year	0.709	0.709	0.709	0.712	0.705	0.708	0.693	0.441	0.570	0.703	0.684
0948_Ishwar	0.996	0.998	0.998	0.950	0.946	0.954	0.990	0.983	0.988	0.975	0.979
1011_cleve	0.943	0.933	0.933	0.933	0.938	0.938	0.929	0.943	0.933	0.957	0.971
1142_Sick_numeric	0.995	0.991	0.991	0.998	0.997	0.997	0.995	0.983	0.978	0.917	0.578

Table 10: TabPertNet [40] Test Results, Part 2 (AUC ↑).

	TabPFN V2			TabICL V2			Conventional baselines				
Dataset	**RAD-TFM**	**Base**	**FT**	**RAD-TFM**	**Base**	**FT**	**CatBoost**	**RF**	**XGB**	**LR**	**MLP**
1201_Gender-Recognit	1.000	1.000	1.000	0.999	0.999	0.999	0.998	0.999	0.999	0.996	0.993
1222_premier_league_	0.695	0.699	0.699	0.695	0.690	0.691	0.577	0.634	0.618	0.693	0.661
1408_national-longit	1.000	1.000	1.000	0.996	0.997	0.996	1.000	1.000	1.000	0.992	0.990
1413_shill-bidding	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.996	0.997
1458_kdd_ipums_la_97	0.938	0.937	0.937	0.938	0.939	0.936	0.939	0.932	0.923	0.890	0.909
1461_heart-failure	0.861	0.861	0.861	0.861	0.878	0.894	0.822	0.844	0.856	0.944	0.794
1512_eye_movements	0.714	0.746	0.745	0.818	0.747	0.817	0.690	0.698	0.707	0.594	0.627
1564_Mammographic-Ma	0.817	0.815	0.815	0.818	0.818	0.815	0.808	0.701	0.658	0.807	0.691
1578_kdd_ipums_la_97	0.954	0.953	0.953	0.956	0.954	0.955	0.959	0.954	0.946	0.907	0.927
1592_Diabetes-Data-S	0.873	0.871	0.871	0.877	0.873	0.871	0.876	0.896	0.871	0.839	0.772
1600_VulNoneVul	0.801	0.798	0.797	0.757	0.762	0.758	0.720	0.638	0.698	0.755	0.523
1635_Is-this-a-good-	0.810	0.805	0.804	0.779	0.787	0.781	0.820	0.779	0.752	0.747	0.750
1692_Gender-Classifi	0.995	0.995	0.995	0.993	0.993	0.994	0.993	0.992	0.989	0.993	0.993
1736_combined-wine-d	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.998
1742_Loan-Predicatio	0.785	0.743	0.743	0.712	0.733	0.728	0.766	0.769	0.737	0.742	0.752
1752_Wisconsin-breas	0.996	0.994	0.994	0.992	0.992	0.992	0.990	0.992	0.987	0.992	0.977
1759_Red_White-wine-	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.998
1898_Personal-Loan-M	0.618	0.624	0.623	0.615	0.619	0.616	0.631	0.645	0.603	0.627	0.587
2304_electricity_see	0.875	0.866	0.866	0.889	0.887	0.884	0.853	0.867	0.875	0.699	0.770
2305_electricity_see	0.934	0.937	0.937	0.915	0.913	0.919	0.905	0.917	0.922	0.788	0.858
2306_electricity_see	0.877	0.879	0.879	0.863	0.858	0.862	0.862	0.874	0.871	0.771	0.827
2308_electricity_see	0.912	0.911	0.911	0.903	0.902	0.901	0.891	0.886	0.886	0.717	0.837
2389_airlines_seed_0	0.679	0.683	0.683	0.650	0.644	0.656	0.640	0.618	0.637	0.683	0.666
2390_airlines_seed_1	0.646	0.640	0.640	0.643	0.638	0.650	0.637	0.607	0.585	0.662	0.646
2391_airlines_seed_2	0.650	0.641	0.641	0.636	0.629	0.632	0.619	0.618	0.650	0.647	0.635
2392_airlines_seed_3	0.607	0.623	0.623	0.583	0.585	0.580	0.607	0.600	0.564	0.612	0.617
2393_airlines_seed_4	0.677	0.680	0.679	0.686	0.686	0.686	0.647	0.639	0.607	0.690	0.682
2619_sf-police-incid	0.465	0.450	0.450	0.596	0.604	0.601	0.530	0.595	0.410	0.495	0.445
2620_sf-police-incid	0.509	0.487	0.485	0.464	0.456	0.455	0.518	0.510	0.457	0.512	0.456
2621_sf-police-incid	0.478	0.540	0.544	0.618	0.570	0.569	0.731	0.739	0.616	0.646	0.449
2622_sf-police-incid	0.522	0.458	0.458	0.623	0.386	0.361	0.482	0.442	0.430	0.449	0.467
2703_compas-two-year	0.729	0.730	0.730	0.733	0.729	0.733	0.711	0.525	0.585	0.732	0.733
trial	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
wines_SPA	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 11: TabArena [14] Test Results (OVO AUC ↑).

	TabPFN V2			TabICL V2			Conventional baselines				
Dataset	**RAD-TFM**	**Base**	**FT**	**RAD-TFM**	**Base**	**FT**	**CatBoost**	**RF**	**XGB**	**LR**	**MLP**
Bank_Customer_Churn	0.876	0.875	0.874	0.841	0.825	0.802	0.878	0.845	0.845	0.745	0.848
Fitness_Club	0.811	0.810	0.810	0.779	0.778	0.777	0.803	0.760	0.757	0.796	0.563
Is-this-a-good-custo	0.731	0.707	0.712	0.752	0.749	0.746	0.729	0.655	0.670	0.712	0.585
MIC	0.643	0.625	0.634	0.674	0.684	0.678	0.684	0.678	0.652	0.722	0.635
Marketing_Campaign	0.908	0.897	0.894	0.850	0.830	0.793	0.907	0.894	0.909	0.837	0.647
NATICUSdroid	0.986	0.984	0.985	0.987	0.987	0.987	0.985	0.978	0.984	0.982	0.981
anneal	0.992	1.000	1.000	0.999	1.000	1.000	0.958	0.999	0.994	0.953	0.961
blood-transfusion-se	0.741	0.729	0.732	0.741	0.740	0.744	0.697	0.663	0.649	0.744	0.723
churn	0.944	0.940	0.942	0.942	0.944	0.946	0.938	0.927	0.933	0.814	0.882
coil2000_insurance_p	0.752	0.746	0.749	0.765	0.758	0.762	0.721	0.689	0.702	0.735	0.696
credit-g	0.780	0.781	0.782	0.772	0.772	0.774	0.774	0.771	0.772	0.773	0.704
diabetes	0.841	0.842	0.844	0.845	0.847	0.845	0.830	0.828	0.808	0.839	0.781
hazelnut-spread-cont	0.982	0.982	0.983	0.992	0.992	0.992	0.959	0.950	0.968	0.584	0.931
maternal_health_risk	0.962	0.952	0.952	0.964	0.966	0.968	0.952	0.955	0.958	0.804	0.793
polish_companies_ban	0.967	0.963	0.966	0.980	0.980	0.982	0.970	0.945	0.963	0.532	0.814
qsar-biodeg	0.930	0.925	0.930	0.933	0.935	0.934	0.925	0.915	0.920	0.921	0.919
seismic-bumps	0.773	0.782	0.778	0.763	0.754	0.762	0.758	0.761	0.740	0.766	0.602
splice	0.998	0.997	0.998	0.998	0.997	0.998	0.996	0.997	0.995	0.992	0.988
students_dropout_and	0.869	0.864	0.866	0.862	0.851	0.860	0.857	0.856	0.869	0.864	0.832
taiwanese_bankruptcy	0.948	0.948	0.950	0.946	0.949	0.948	0.941	0.941	0.940	0.592	0.832
website_phishing	0.973	0.973	0.973	0.978	0.978	0.979	0.976	0.970	0.972	0.926	0.869

Table 12: PMLB [26] Test Results, Part 1 (OVO AUC ↑).

	TabPFN V2			TabICL V2			Conventional baselines				
Dataset	**RAD-TFM**	**Base**	**FT**	**RAD-TFM**	**Base**	**FT**	**CatBoost**	**RF**	**XGB**	**LR**	**MLP**
1028_SWD	0.841	0.843	0.843	0.861	0.868	0.864	0.861	0.861	0.861	0.860	0.827
1029_LEV	0.902	0.901	0.902	0.904	0.915	0.909	0.917	0.911	0.916	0.885	0.849
1030_ERA	0.788	0.787	0.787	0.785	0.782	0.782	0.742	0.737	0.739	0.788	0.724
294_satellite_image	0.995	0.995	0.995	0.996	0.996	0.996	0.989	0.988	0.991	0.957	0.985
GAMETES_Epistasis_2_	0.739	0.738	0.737	0.740	0.732	0.694	0.684	0.601	0.662	0.530	0.464
GAMETES_Epistasis_2_	0.876	0.871	0.872	0.883	0.876	0.897	0.851	0.711	0.807	0.498	0.666
GAMETES_Epistasis_3_	0.480	0.482	0.482	0.704	0.448	0.690	0.619	0.563	0.617	0.539	0.465
GAMETES_Heterogeneit	0.731	0.727	0.727	0.737	0.689	0.717	0.768	0.725	0.769	0.533	0.637
GAMETES_Heterogeneit	0.795	0.783	0.785	0.789	0.777	0.812	0.745	0.620	0.668	0.513	0.535
Hill_Valley_with_noi	0.532	0.511	0.510	0.490	0.465	0.532	0.486	0.558	0.467	0.496	0.502
Hill_Valley_without_	0.542	0.474	0.474	0.542	0.495	0.462	0.477	0.587	0.458	0.496	0.495
agaricus_lepiota	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.975	1.000
allbp	0.995	0.995	0.995	0.997	0.996	0.996	0.991	0.994	0.993	0.932	0.659
allhypo	0.986	0.984	0.984	0.991	0.994	0.993	0.982	0.983	0.981	0.840	0.707
allrep	0.904	0.899	0.899	0.926	0.910	0.915	0.898	0.929	0.890	0.821	0.456
analcatdata_authorsh	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999
analcatdata_dmft	0.621	0.605	0.614	0.602	0.620	0.603	0.533	0.517	0.548	0.597	0.449
analcatdata_germangs	0.839	0.835	0.837	0.941	0.853	0.957	0.660	0.595	0.660	0.698	0.573
analcatdata_lawsuit	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.987
ann_thyroid	0.998	0.999	0.999	1.000	1.000	1.000	1.000	1.000	1.000	0.789	0.922
balance_scale	0.985	0.986	0.986	1.000	1.000	1.000	0.915	0.754	0.871	0.941	0.852
banana	0.967	0.967	0.967	0.967	0.967	0.968	0.966	0.960	0.958	0.501	0.945
breast_cancer	0.800	0.774	0.774	0.852	0.783	0.791	0.800	0.783	0.539	0.800	0.626
breast_cancer_wiscon	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.997	1.000	0.929
breast_cancer_wiscon	0.985	0.986	0.986	0.986	0.986	0.988	0.992	0.987	0.984	0.986	0.997
bupa	0.671	0.667	0.667	0.746	0.658	0.679	0.650	0.625	0.496	0.688	0.667
calendarDOW	0.855	0.848	0.846	0.838	0.869	0.853	0.836	0.861	0.843	0.801	0.779
car_evaluation	0.999	0.999	0.999	1.000	1.000	1.000	1.000	0.999	1.000	0.921	0.862
cars	1.000	1.000	1.000	1.000	1.000	1.000	0.931	0.932	0.929	0.781	0.628
chess	0.999	0.999	0.999	1.000	1.000	1.000	1.000	1.000	0.999	0.996	0.998
churn	0.974	0.973	0.973	0.974	0.969	0.961	0.964	0.960	0.969	0.769	0.580
clean1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.918
clean2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
coil2000	0.769	0.763	0.763	0.759	0.751	0.761	0.742	0.656	0.741	0.723	0.601
congressional_voting	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.981	0.998	0.976
contraceptive_method	0.729	0.730	0.730	0.715	0.724	0.712	0.688	0.701	0.702	0.670	0.648
credit_approval_aust	0.959	0.958	0.958	0.965	0.962	0.964	0.960	0.966	0.944	0.951	0.918
credit_approval_germ	0.844	0.846	0.846	0.827	0.838	0.821	0.829	0.798	0.733	0.824	0.735

Table 13: PMLB [26] Test Results, Part 2 (OVO AUC ↑).

	TabPFN V2			TabICL V2			Conventional baselines				
Dataset	**RAD-TFM**	**Base**	**FT**	**RAD-TFM**	**Base**	**FT**	**CatBoost**	**RF**	**XGB**	**LR**	**MLP**
dis	0.999	1.000	1.000	0.999	0.997	0.998	0.996	0.995	0.998	0.732	0.771
dna	0.995	0.995	0.995	0.997	0.997	0.997	0.997	0.996	0.998	0.990	0.991
ecoli	0.960	0.960	0.960	0.960	0.959	0.960	0.955	0.956	0.947	0.941	0.872
haberman	0.725	0.683	0.683	0.692	0.658	0.675	0.608	0.529	0.375	0.717	0.875
heart_disease_clevel	0.904	0.894	0.894	0.904	0.909	0.918	0.904	0.916	0.904	0.904	0.880
heart_disease_hungar	0.911	0.900	0.900	0.932	0.963	0.974	0.947	0.945	0.895	0.905	0.574
horse_colic_outcome	0.764	0.749	0.753	0.865	0.855	0.861	0.812	0.828	0.792	0.731	0.776
horse_colic_surgery	0.821	0.810	0.810	0.786	0.806	0.821	0.790	0.804	0.722	0.841	0.587
hypothyroid	0.967	0.968	0.968	0.973	0.980	0.978	0.979	0.979	0.977	0.865	0.546
ionosphere	1.000	0.996	0.996	0.996	1.000	1.000	1.000	1.000	1.000	0.930	0.816
irish	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.869	0.902
kr_vs_kp	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.993	0.998
led24	0.948	0.948	0.948	0.951	0.952	0.951	0.941	0.942	0.929	0.948	0.946
led7	0.957	0.957	0.957	0.958	0.960	0.959	0.949	0.947	0.952	0.958	0.955
mfeat_factors	1.000	1.000	1.000	1.000	1.000	1.000	0.999	1.000	1.000	1.000	1.000
mfeat_fourier	0.992	0.993	0.993	0.996	0.996	0.997	0.986	0.986	0.988	0.981	0.977
mfeat_karhunen	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.999	0.999	1.000	0.999
mfeat_morphological	0.968	0.969	0.969	0.979	0.969	0.978	0.962	0.955	0.958	0.960	0.954
mfeat_pixel	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
mfeat_zernike	0.990	0.990	0.990	0.995	0.994	0.997	0.974	0.973	0.966	0.982	0.983
mofn_3_7_10	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
monk1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.374	0.760
monk2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.968	1.000	0.559	0.544
monk3	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.998	1.000	0.871	0.871
mushroom	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.977	1.000
optdigits	1.000	1.000	1.000	1.000	1.000	1.000	0.999	1.000	0.999	0.999	0.999
page_blocks	0.963	0.963	0.963	0.963	0.961	0.973	0.964	0.966	0.962	0.903	0.962
parity5+5	1.000	1.000	1.000	1.000	0.733	1.000	0.531	0.651	1.000	0.643	0.333
pendigits	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.996	1.000
penguins	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.901
phoneme	0.951	0.954	0.954	0.973	0.968	0.971	0.938	0.953	0.946	0.835	0.887
profb	0.698	0.670	0.671	0.700	0.710	0.580	0.652	0.671	0.680	0.656	0.579
ring	0.996	0.996	0.996	0.997	0.997	0.997	0.995	0.994	0.995	0.837	0.993
saheart	0.767	0.753	0.757	0.755	0.776	0.765	0.809	0.739	0.712	0.772	0.658
satimage	0.995	0.995	0.995	0.996	0.996	0.996	0.989	0.988	0.991	0.957	0.985
schizo	0.578	0.567	0.567	0.534	0.459	0.469	0.444	0.421	0.443	0.583	0.533
segmentation	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.995	0.994
spambase	0.988	0.989	0.989	0.992	0.993	0.993	0.988	0.980	0.990	0.976	0.971
spect	0.881	0.887	0.887	0.925	0.887	0.900	0.862	0.875	0.881	0.919	0.875
spectf	0.914	0.919	0.919	0.924	0.955	0.944	0.949	0.944	0.899	0.848	0.828
splice	0.997	0.997	0.997	0.999	0.999	0.999	0.998	0.995	0.998	0.944	0.956
threeOf9	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.961	0.969
tic_tac_toe	0.998	0.999	0.999	1.000	0.998	0.999	0.999	0.995	1.000	0.602	0.778
titanic	0.805	0.804	0.804	0.791	0.787	0.789	0.783	0.728	0.742	0.713	0.705
tokyo1	0.980	0.977	0.977	0.981	0.982	0.981	0.987	0.984	0.984	0.883	0.951
twonorm	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.997	0.997	0.998	0.998
vehicle	0.978	0.978	0.977	0.992	0.984	0.987	0.941	0.949	0.938	0.941	0.818
waveform_21	0.972	0.972	0.972	0.973	0.973	0.972	0.964	0.966	0.966	0.972	0.967
waveform_40	0.972	0.973	0.973	0.973	0.974	0.973	0.960	0.967	0.968	0.966	0.964
wine_quality_red	0.814	0.822	0.822	0.852	0.859	0.856	0.821	0.871	0.849	0.777	0.673
wine_quality_white	0.860	0.876	0.875	0.895	0.881	0.884	0.783	0.799	0.809	0.639	0.635
xd6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.886	0.996
yeast	0.873	0.876	0.875	0.874	0.888	0.882	0.853	0.870	0.870	0.827	0.814

Table 14: Engineering [4] Test Results (OVO AUC ↑).

Dataset	$\text{TabPFN}_{\text{n.e.}}$ (RAD-TFM)	$\text{TabPFN}_{\text{n.e.}}$ (Base)	$\text{TabPFN}_{\text{n.e.}}$ (Finetuned)	CatBoost	RF	XGB	LR	MLP
anneal_2	1.000	1.000	1.000	1.000	1.000	1.000	0.970	0.997
cnae-9_1468	0.999	0.998	0.998	0.998	0.997	0.997	0.998	0.993
glass_41	0.975	0.964	0.964	0.952	0.939	0.937	0.819	0.497
kc1_1067	0.868	0.861	0.861	0.840	0.849	0.802	0.789	0.820
kc2_1063	0.868	0.864	0.864	0.892	0.868	0.835	0.712	0.831
kc3_1065	0.780	0.774	0.774	0.750	0.848	0.685	0.554	0.702
mc1_1056	0.912	0.922	0.921	0.990	0.922	0.992	0.917	0.884
mw1_1071	0.842	0.868	0.868	0.719	0.711	0.588	0.833	0.640
pc1_1068	0.882	0.875	0.876	0.875	0.805	0.821	0.692	0.533
pc2_1069	0.990	0.990	0.990	0.993	0.982	0.985	0.035	0.250
pc3_1050	0.847	0.825	0.825	0.785	0.809	0.731	0.640	0.719
pc4_1049	0.946	0.930	0.930	0.938	0.934	0.949	0.918	0.880
seismic-bumps_1500	0.993	0.993	0.993	0.966	0.966	0.980	0.993	0.748
semeion_1501	0.998	0.996	0.996	0.997	0.996	0.995	0.995	0.991
sonar_40	0.963	0.963	0.963	0.981	0.954	0.972	0.926	0.843
steel-plates-fault_1	1.000	1.000	1.000	1.000	1.000	1.000	0.698	1.000
steel-plates-fault_4	0.980	0.980	0.980	0.974	0.973	0.971	0.737	0.950
wall-robot-navigatio	1.000	1.000	1.000	1.000	1.000	1.000	0.908	0.986
wall-robot-navigatio	1.000	1.000	1.000	1.000	1.000	1.000	0.986	1.000
wall-robot-navigatio	1.000	1.000	1.000	1.000	1.000	1.000	0.987	0.999

Table 15: Genomics [4] Test Results (AUC ↑).

Dataset	$\text{TabPFN}_{\text{n.e.}}$ (RAD-TFM)	$\text{TabPFN}_{\text{n.e.}}$ (Base)	$\text{TabPFN}_{\text{n.e.}}$ (Finetuned)	CatBoost	RF	XGB	LR	MLP
AP_Breast_Omentum_11	0.914	0.914	0.914	0.882	0.882	0.879	0.866	0.864
AP_Colon_Omentum_114	1.000	1.000	1.000	0.996	0.996	1.000	0.970	1.000
AP_Colon_Ovary_1153	0.984	0.986	0.986	0.984	0.974	0.972	0.948	0.950
AP_Colon_Uterus_1160	0.970	0.962	0.962	0.970	0.929	0.951	0.967	0.967
AP_Omentum_Kidney_11	0.990	0.990	0.990	0.995	0.990	0.995	0.971	0.971
AP_Omentum_Lung_1132	0.990	0.990	0.990	0.981	0.990	0.962	0.962	0.971
AP_Omentum_Ovary_115	0.975	0.956	0.956	0.975	0.944	0.944	0.862	0.644
AP_Omentum_Uterus_11	1.000	1.000	1.000	1.000	1.000	1.000	0.962	1.000
OVA_Colon_1161	0.921	0.918	0.918	0.941	0.948	0.932	0.912	0.932
OVA_Omentum_1139	0.991	0.986	0.986	0.982	0.960	0.975	0.711	0.944
madelon_1485	0.949	0.941	0.942	0.907	0.845	0.891	0.607	0.625

Table 16: Healthcare [4] Test Results (AUC ↑).

Dataset	$\text{TabPFN}_{\text{n.e.}}$ (RAD-TFM)	$\text{TabPFN}_{\text{n.e.}}$ (Base)	$\text{TabPFN}_{\text{n.e.}}$ (Finetuned)	CatBoost	RF	XGB	LR	MLP
SPECTF_1600	0.844	0.852	0.852	0.844	0.825	0.809	0.780	0.728
SPECTF_337	0.949	0.935	0.936	0.950	0.951	0.924	0.928	0.717
SPECT_336	0.865	0.862	0.862	0.854	0.828	0.831	0.847	0.767
abalone_720	0.905	0.905	0.905	0.892	0.889	0.880	0.888	0.838
audiology_999	0.990	1.000	1.000	1.000	1.000	0.971	1.000	0.902
blood-transfusion-se	0.775	0.771	0.771	0.762	0.773	0.729	0.778	0.733
bodyfat_778	1.000	1.000	1.000	1.000	1.000	1.000	0.947	0.858
breast-cancer-droppe	0.538	0.532	0.532	0.450	0.421	0.404	0.503	0.456
breast-cancer_13	0.696	0.655	0.655	0.619	0.595	0.655	0.613	0.589
cardiotocography_146	1.000	1.000	1.000	1.000	1.000	1.000	0.978	1.000
cardiotocography_156	1.000	1.000	1.000	1.000	1.000	1.000	0.975	1.000
chatfield_4_820	0.993	0.986	0.986	0.979	0.975	0.907	0.986	0.479
dermatology_1010	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
haberman_43	0.826	0.799	0.804	0.734	0.617	0.565	0.821	0.679
heart-c_982	0.924	0.933	0.933	0.916	0.891	0.887	0.920	0.882
heart-h_1565	0.608	0.606	0.607	0.705	0.694	0.659	0.541	0.641
heart-h_963	0.856	0.847	0.847	0.837	0.792	0.775	0.813	0.804
mfeat-factors_978	1.000	1.000	1.000	1.000	0.997	0.999	0.999	0.997
mfeat-fourier_971	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
mfeat-karhunen_1020	1.000	1.000	1.000	1.000	1.000	0.999	0.998	0.996
mfeat-morphological_	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
mfeat-pixel_1022	1.000	1.000	1.000	1.000	1.000	1.000	0.996	0.999
mfeat-zernike_995	1.000	0.999	0.999	1.000	0.999	1.000	0.972	0.968
mu284_880	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.738
pbc_810	0.753	0.757	0.757	0.778	0.765	0.831	0.705	0.703
pbcseq_802	0.833	0.857	0.855	0.902	0.901	0.933	0.728	0.666
pwLinear_721	0.950	0.960	0.960	0.980	0.980	1.000	0.970	0.700
thoracic-surgery_150	0.768	0.711	0.711	0.746	0.789	0.879	0.704	0.750
thoracic_surgery_432	0.768	0.775	0.771	0.796	0.730	0.768	0.768	0.571
vertebra-column_1523	0.944	0.933	0.933	0.947	0.949	0.900	0.902	0.783
vertebra-column_1524	0.962	0.962	0.962	0.952	0.931	0.924	0.967	0.824

H Additional Analyses

TabICL V2 Across Benchmarks. Table 17 reports the mean normalized AUC of the frozen, fine-tuned, and RAD-TFMtrained TabICL V2 classifier separately for each benchmark. In all three cases, the full method improves on both the original backbone and fine-tuning. These experiments use the same synthetic-generation and robust parameter-selection pipeline as TabPFN V2, without architecture-specific modifications. Normalization is performed over the expanded model comparison and therefore differs from the normalization used in Tables 1 and 2. Per-dataset AUC results for both TFM backbones and all five conventional baselines are reported in Tables 9–13.

Table 17: Mean normalized AUC for TabICL V2 variants on the three principal benchmarks.

Benchmark	TabICL Base	TabICL FT	TabICL (RAD-TFM)
PMLB	0.892	0.911	**0.924**
TabPertNet	0.748	0.754	**0.779**
TabArena	0.891	0.894	**0.914**

Training-Budget Sensitivity. To control for the number of training datasets, the fine-tuning runs use the same batch size and number of optimizer updates as the corresponding RAD-TFM runs. The models are evaluated at checkpoints spaced by 500 training datasets. We report the mean test-AUC gap between each trained model and its frozen backbone, evaluated on the respective reference benchmark. In particular, continued fine-tuning on the real-data distribution degrades sharply beyond the earliest checkpoints, whereas the synthetic distribution selected by RAD-TFM maintains positive gains throughout training (Table 18). These comparisons control for dataset exposures and parameter-update counts, rather than total wall-clock time. The 30k RAD-TFM entries correspond to selected checkpoints from training runs that processed 30k generated datasets in total, rather than evaluations at the terminal checkpoint. The fine-tuning entries at 30k report performance evaluated at that checkpoint.

Table 18: Mean raw test-AUC change relative to frozen TabPFN V2 at different training-dataset budgets. Positive values indicate improvement. At 30k, RAD-TFM entries marked with an asterisk correspond to selected checkpoints from runs that processed 30k generated datasets, whereas fine-tuning entries are evaluated at the 30k checkpoint.

Benchmark	Method	500	1k	1.5k	2k	6k	10k	30k
TabArena	RAD-TFM	+0.0005	+0.0010	+0.0016	+0.0015	+0.0026	+0.0028	$+0.0040^{*}$
	Fine-tuning	+0.0011	+0.0015	-0.0003	-0.0024	-0.0505	-0.0840	-0.0844
PMLB	RAD-TFM	+0.0019	+0.0025	+0.0026	+0.0033	+0.0029	+0.0030	$+0.0030^{*}$
	Fine-tuning	-0.0006	-0.0012	-0.0017	-0.0023	-0.0166	-0.0199	-0.0233
TabPertNet	RAD-TFM	+0.0031	+0.0049	+0.0052	+0.0058	+0.0083	+0.0105	$+0.0166^{*}$
	Fine-tuning	+0.0019	+0.0009	+0.0010	+0.0001	-0.0020	-0.0097	-0.0103

Individual and Combined Contributions. To further quantify the role of the two components, we measure the raw-AUC gain over the frozen model from domain adaptation alone (Δ_{DA}), robustness alone (Δ_{R}), and their combination (Δ_{RAD}). The residual $\Delta_{\mathrm{RAD}} - \Delta_{\mathrm{DA}} - \Delta_{\mathrm{R}}$ captures the deviation from additivity, as shown in Table 19. The average gain under the full model ($+0.0055$) is close to the sum of the individual average gains ($+0.0052$); the corresponding residual is $+0.0003$. While the interaction varies across benchmarks, the two components contribute complementary improvements on average. Paired one-sided Wilcoxon signed-rank tests of normalized AUC over 236 datasets give $p = 10^{-4}$ for domain adaptation alone, $p = 0.028$ for robustness alone, and $p < 10^{-4}$ for the combined method, each relative to the frozen model.

Table 19: Change in mean raw AUC from the frozen TabPFN V2 and the residual from additive contributions.

Benchmark	$\boldsymbol{\Delta_{\mathrm{DA}}}$	$\boldsymbol{\Delta_{\mathrm{R}}}$	$\boldsymbol{\Delta_{\mathrm{DA}} + \Delta_{\mathrm{R}}}$	$\boldsymbol{\Delta_{\mathrm{RAD}}}$	**Residual**
PMLB	+0.0033	+0.0021	+0.0053	+0.0030	-0.0023
TabPertNet	+0.0074	+0.0094	+0.0167	+0.0166	-0.0001
TabArena	+0.0013	+0.0018	+0.0031	+0.0040	+0.0010
Genomics	+0.0024	+0.0004	+0.0028	+0.0037	+0.0009
Healthcare	+0.0021	+0.0023	+0.0044	+0.0035	-0.0008
Engineering	+0.0001	-0.0016	-0.0015	+0.0020	+0.0035
Mean	+0.0028	+0.0024	+0.0052	+0.0055	+0.0003

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's theoretical, algorithmic, and empirical contributions, and these are matched by the formal development in Sections 2–4 and the experiments in Section 5.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses several concrete limitations, including limited statistical power for small benchmarks, challenging domains where the current SCM space and meta-features may be insufficient, and design restrictions such as fixed generated sample sizes; see Section 5 and Appendices A and D.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: The theoretical formulation and assumptions are introduced in Section 2, and formal proofs and supporting derivations are provided in Appendix C.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper specifies the algorithmic pipeline, benchmark construction, evaluation protocol, hyperparameters, and detailed pseudocode in Section 5 and Appendices A, B, D, and G.

Guidelines:

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

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [No]

   Justification: We do not release code or additional assets with the current submission. The experiments rely on publicly available benchmarks and pretrained model backbones, and the method, optimization objective, parameter space, training protocol, and evaluation procedure are described in the paper and its appendices.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 5 gives the main training setup and benchmark choices, and Appendices B, D, and G provide additional details on splits, hyperparameters, and implementation choices.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Rather than error bars, we report formal statistical significance tests for the main claims, including Friedman tests and pairwise Wilcoxon signed-rank tests, as described in Section 5 and detailed further in Appendix G. The tests span six benchmarks, with additional paired comparisons and bootstrap confidence intervals reported for the second TFM backbone and the component ablations.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper specifies the compute configuration and runtime breakdown used in practice, including GPU type and memory, CPU usage, domain-adaptation cost, and benchmark-dependent training duration; see Appendix G.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: To the best of our knowledge, this work conforms to the NeurIPS Code of Ethics. The paper studies generic tabular classification methods using cited benchmark resources and does not involve human subjects or the release of sensitive assets.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [No]

    Justification: The submission does not contain a dedicated broader-impact discussion. The work is methodological and application-agnostic, focusing on a generic tabular classifier rather than a specific downstream deployment setting.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release high-risk generative models, scraped datasets, or similar assets that would require a safeguards discussion for controlled release.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

Justification: We use existing benchmarks, datasets, software packages, and pretrained model weights. We cite the original creators in the main paper and provide an appendix paragraph listing the relevant licenses and terms of use for the principal assets, including TabArena, PMLB, OpenML, TabPertNet, Optuna, TabPFN-V2, and TabICL V2 in Appendix G. We use these assets only for academic research and benchmarking and do not redistribute modified versions of the assets.

Guidelines:

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

13. **New assets**

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

Answer: [N/A]

Justification: We do not release new code, models, datasets, or other assets as part of the current submission.

Guidelines:

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

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

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

Answer: [N/A]

Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve human subjects, so IRB approval and related participant-risk disclosures are not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs are not an important, original, or non-standard component of the core methodology studied in this paper.

    Guidelines:

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