TABCAM: ONLINE CALIBRATED REGULARIZATION FOR TABULAR DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern tabular deep learning models are mostly built on overparameterized architectures, which require carefully calibrated regularization to ensure out-of-sample predictive accuracy and reliability. Standard regularization approaches rely on cross-validation via grid search to tune a small number of global penalty coefficients, which inevitably over-regularizes some parameters and under-regularizes others. Ideally, each model parameter should be assigned its own regularization strength, enabling the training to selectively suppress dataset-specific noise while preserving signals that capture true population-level structure. However, since the cost of grid search grows exponentially in the number of hyperparameters, per-parameter regularization is computationally infeasible under standard approaches. To address this bottleneck, we propose TabCAM (**Tab**ular **C**alibrated **A**uto-**M**odeling), an add-on module for differentiable backbones that jointly estimates the model parameters and per-parameter regularization strengths inside a single training loop. Through a novel online construction of calibrated future targets, TabCAM adds minimal computational overhead beyond vanilla model training, making it particularly promising for future adoption with even more complex architectures where principled per-parameter regularization becomes increasingly valuable. Moreover, because TabCAM estimates the population conditional distribution during training, covariate-conditional uncertainty quantification is built in. On the 51-dataset TabArena benchmark, TabCAM ranks third among 16 non-foundation default methods and improves substantially under modest tuning while matching the runtime of vanilla model training. In simulations, TabCAM converges to the population minimizer and produces prediction intervals with near-nominal coverage that is more consistent and narrower than split-conformal baselines.

1 INTRODUCTION

Tabular data constitute the dominant data modality in a wide range of real-world domains, and supervised learning on tabular data remains fundamental to nearly every quantitative discipline (Hastie et al., 2009). Modern machine learning and deep learning models have demonstrated remarkable capability in capturing complex, high-order interactions in heterogeneous tabular features (Gorishniy et al., 2025; Holzmüller et al., 2024; Hollmann et al., 2025; Prokhorenkova et al., 2018). However, these models are mostly built on heavily overparameterized architectures, which often cause overfitting that degrades the models' out-of-sample predictive accuracy and reliability (Zhang et al., 2017; Belkin et al., 2019; Nakkiran et al., 2020). To mitigate overfitting, regularization is commonly applied during training to steer the model toward population-level structure rather than dataset-specific noise (Srivastava et al., 2014; Loshchilov & Hutter, 2019; Kadra et al., 2021). Ideally, each model parameter should be assigned its own regularization strength, as different parameters vary in their tendency to overfit. Even for deep learning models relying on architectural choices such as dropout, weight decay, and learning rate scheduling to control overfitting, since these design decisions essentially serve as proxies for regularization, effectively applying a per-parameter approach can potentially reduce the architectural tuning burden significantly.

That said, standard regularization approaches rely on cross-validation with grid search to identify optimal penalty strengths (Stone, 1974; Gorishniy et al., 2021; Kadra et al., 2021). Since the cost of grid search grows exponentially in the number of regularization hyperparameters, only a small number of global penalty coefficients can be tuned in practice. Under standard regularization

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

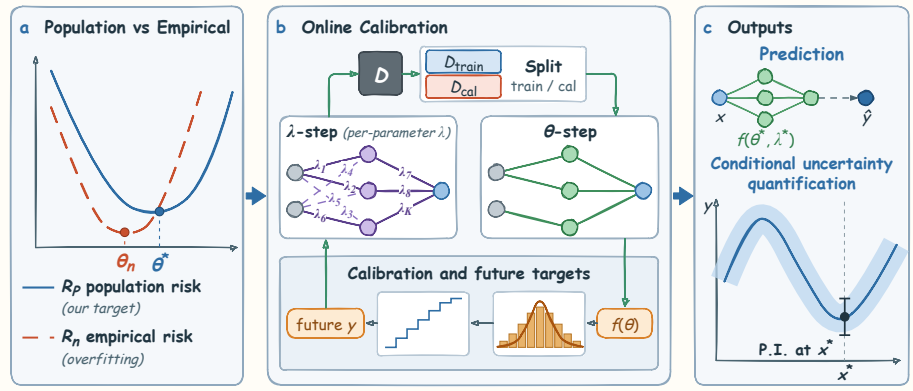


Figure 1: **Overview of TabCAM. (a)** Standard training minimizes the empirical risk R_n, which causes overfitting, while TabCAM targets the population risk R_P directly. **(b)** At each epoch, data are split into training and calibration sets, where the training set is used to update the model parameters through standard gradient descent, while the calibration set is used to construct future targets online, which are used to update the per-parameter regularization strengths. **(c)** The fitted model produces both point predictions and covariate-conditional uncertainty quantification.

approaches, assigning each parameter its own regularization strength is therefore infeasible. This leaves a fundamental gap between the ideal of per-parameter regularization and what standard cross-validation approaches can deliver. Recently, Jiang & Liu (2025) attempted to address this gap, but at a computational cost that renders it impractical; for instance, it fails to meet basic time budgets on many TabArena datasets (Erickson et al., 2025). Gradient-based hyperparameter optimization can learn many regularization coefficients during training through validation risk (MacKay et al., 2019; Lorraine et al., 2020) rather than explicitly modeling the empirical-to-population generalization gap with calibrated future targets.

In this paper, we aim to put forth a computationally efficient alternative that achieves per-parameter regularization with minimal extra computational cost beyond standard training. Specifically, we propose TabCAM (**Tab**ular **C**alibrated **A**uto-**M**odeling), illustrated in Figure 1, an add-on module for differentiable backbones, which reformulates the population risk $R_P(\theta)$ into a penalized empirical objective G and a discrepancy term V that captures the gap between the empirical and population risks. This reformulation allows us to simultaneously estimate the model parameters $\theta \in \mathbb{R}^k$ and a per-parameter regularization vector $\lambda \in \mathbb{R}^k_{\geq 0}$ within a single training loop. A central mechanism enabling TabCAM to estimate the generalization gap without observing the population risk is a novel online construction of calibrated pseudo-observations. At each training epoch, TabCAM splits the data, fits θ on one part, and calibrates on the other part to generate future observations used to approximate the generalization gap between the population risk and empirical risk. These calibrated targets are used to update the per-parameter regularization vector λ. The learned penalties can also provide interpretable signals, such as variable importance. In addition, since TabCAM estimates the population conditional distribution of the response given covariates during training, uncertainty quantification is built in, and we can produce not only marginal prediction intervals/sets but also prediction intervals/sets conditional on the covariates.

Contributions. Our paper contains three main contributions. First, to our knowledge, TabCAM provides the first computationally feasible framework that learns **per-parameter regularization** online while directly targeting population-level risk through calibration, which effectively helps mitigate overfitting and achieve desired prediction performance with reduced architectural tuning overhead (Section 2). Second, the **joint estimation** of the model parameters θ and the per-parameter regularization strengths λ runs inside a single training loop through an online future target construction mechanism, so TabCAM is **end-to-end** and adds minimal extra computational cost beyond standard model training (Section 2). Third, because TabCAM estimates the population conditional distribution during training, it automatically provides **covariate-conditional uncertainty quantification** as a byproduct (Section 2.3).

We illustrate the practical significance of these contributions in Section 3. On the TabArena benchmark (Erickson et al., 2025), TabCAM ranks third among 16 non-foundation default methods and improves substantially under modest architectural tuning on TabArena-Lite. Moreover, its runtime is comparable to that of standard methods. For instance, when using MLP as the base predictor, TabCAM has training runtime comparable to RealMLP in Figure 3. In simulations, TabCAM converges to the population minimizer, and its prediction intervals achieve near-nominal coverage more consistently than split-conformal baselines while being narrower.

It is worth emphasizing that TabCAM operates as an add-on to differentiable backbones. Beyond the MLP backbone used in the main text, Appendix I.1 further applies TabCAM to TabM and FT-Transformer, showing that the module can be readily integrated with different backbones and improve their performance across multiple tasks. In addition, the computational efficiency of the proposed TabCAM makes it particularly promising for future adoption with even more complex architectures, where the need for principled per-parameter regularization is likely to be even more pronounced.

2 METHOD

2.1 TABCAM

Consider a tabular dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $\mathcal{D}$ is an i.i.d. sample from an unknown distribution $\mathbb{P}$ over $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{Y} \subseteq \mathbb{R}^q$. Given a predictor f_θ and task loss $\ell(\theta; x, y)$ with $\theta \in \Theta^{(n)} \subseteq \mathbb{R}^k$, define the population risk by

$$R_P(\theta) = \mathbb{E}_{(X,Y)\sim\mathbb{P}}\big[\ell(\theta; X, Y)\big].$$

To estimate the model parameters, the ideal target is any minimizer of the population risk, namely

$$\theta^\star \in \operatorname*{arg\,min}_{\theta\in\Theta^{(n)}} R_P(\theta). \tag{1}$$

In practice, $\mathbb{P}$ is unobserved, but only the sample $\mathcal{D}$ is available. Let $\mathbb{P}_n$ denote the empirical distribution induced by $\mathcal{D}$, and define the empirical risk by

$$R_n(\theta) = \mathbb{E}_{(X,Y)\sim\mathbb{P}_n}\big[\ell(\theta; X, Y)\big] = \frac{1}{n}\sum_{i=1}^n \ell(\theta; x_i, y_i).$$

Empirical risk minimization replaces (1) with

$$\hat{\theta} \in \operatorname*{arg\,min}_{\theta\in\Theta^{(n)}} R_n(\theta). \tag{2}$$

However, minimizing the empirical risk may cause overfitting among overparameterized models. To address overfitting, motivated by Jiang & Liu (2025), TabCAM directly targets the population risk via the following decomposition:

$$R_P(\theta) = G(\theta, \lambda) + V(\theta, \lambda), \tag{3}$$

where

$$G(\theta, \lambda) = R_n(\theta) + \pi(\theta, \lambda), \qquad V(\theta, \lambda) = R_P(\theta) - R_n(\theta) - \pi(\theta, \lambda). \tag{4}$$

Here, $\pi(\theta, \lambda)$ represents a duality function, which serves as a tractable approximation of the generalization gap $(R_P - R_n)$. Two natural choices of π which allow for each parameter θ_j to receive its own regularization strength λ_j, include

$$\pi_{\mathrm{L1}}(\theta, \lambda) = \sum_{j=1}^k \lambda_j |\theta_j|, \qquad \pi_{\mathrm{L2}}(\theta, \lambda) = \sum_{j=1}^k \lambda_j \theta_j^2. \tag{5}$$

Accordingly, G represents a penalized empirical objective that is tractable to optimize, while V captures the residual discrepancy between the true generalization gap and its approximation by π. To estimate the model, the goal is then to choose λ to make V as small as possible, so that minimizing G becomes approximately equivalent to minimizing the population risk R_P directly. This procedure is equivalent to solving the problem

$$\min_{\theta,\lambda}\ G(\theta, \lambda) + V(\theta, \lambda), \qquad \text{subject to } \theta \in \operatorname*{arg\,min}_{\vartheta\in\Theta^{(n)}} G(\vartheta, \lambda). \tag{6}$$

Proposition 1 in Appendix A provides theoretical insight into how the constrained optimization problem (6) can be solved. Specifically, it shows that under some regularity conditions, a feasible

point $(\tilde{\theta}, \tilde{\lambda})$ satisfies $\nabla_\theta V(\tilde{\theta}, \tilde{\lambda}) = 0$. Because the constraint in problem (6) already specifies that $\nabla_\theta G(\tilde{\theta}, \tilde{\lambda}) = 0$, this result suggests minimizing $\|\nabla_\theta V\|_2^2$ over λ to solve the joint optimization problem.

Computing $V(\theta, \lambda)$ or its gradient requires access to $\mathbb{P}$ which is unobservable. TabCAM addresses this by iteratively estimating the model and constructing calibrated future observations to approximate V. Specifically, at e-th epoch, we randomly partition the dataset into two disjoint subsets $\mathcal{D}_{\text{tr}}^{(e)}$ and $\mathcal{D}_{\text{cal}}^{(e)}$. The training set $\mathcal{D}_{\text{tr}}^{(e)}$ is used to update the model estimate under the current regularization parameter λ, yielding $\theta^{(e)}$. Given $\theta^{(e)}$, the calibration set $\mathcal{D}_{\text{cal}}^{(e)}$ is used to construct a future target set $Q^{(e)} = \{q_i^{(e)}, i \in \mathcal{D}_{\text{cal}}^{(e)}\}$, where $q_i^{(e)} = q(x_i; \theta^{(e)})$. The form of constructor $q(\cdot)$ is task-dependent, designed to assess how well $\pi(\theta^{(e)}, \lambda)$ approximates the generalization gap $R_P(\theta^{(e)}) - R_n(\theta^{(e)})$. More details on the future targets construction are provided in Section 2.2.

Set $n_{\text{tr}} = |\mathcal{D}_{\text{tr}}^{(e)}|$ and $n_{\text{cal}} = |\mathcal{D}_{\text{cal}}^{(e)}|$. At the current epoch, the penalized empirical objective is

$$G^{(e)}(\theta, \lambda) = \frac{1}{n_{\text{tr}}} \sum_{i \in \mathcal{D}_{\text{tr}}^{(e)}} \ell(\theta; x_i, y_i) + \pi(\theta, \lambda),$$

and the generalization gap surrogate is

$$V^{(e)}(\theta, \lambda) = \frac{1}{n_{\text{cal}}} \sum_{i \in \mathcal{D}_{\text{cal}}^{(e)}} \ell\Big(\theta; x_i, q_i^{(e)}\Big) - \frac{1}{n_{\text{tr}}} \sum_{i \in \mathcal{D}_{\text{tr}}^{(e)}} \ell(\theta; x_i, y_i) - \pi(\theta, \lambda).$$

Replacing V with its calibrated approximation based on the pseudo future observations, the epoch-wise counterpart of the constrained problem (6) becomes

$$\min_{\theta, \lambda} \; G^{(e)}(\theta, \lambda) + V^{(e)}(\theta, \lambda), \qquad \text{subject to } \theta \in \arg\min_{\vartheta} G^{(e)}(\vartheta, \lambda). \tag{7}$$

In light of Proposition 1, problem (7) motivates two alternating updates. Given the current regularization strength $\lambda^{(e-1)}$, θ is updated toward $\nabla_\theta G^{(e)}(\theta, \lambda^{(e-1)}) = 0$ via SGD or Adam on $\mathcal{D}_{\text{tr}}^{(e)}$, yielding $\theta^{(e)}$. Then, $\lambda^{(e)}$ is updated by minimizing the gradient of the surrogate gap:

$$\lambda^{(e)} \in \arg\min_{\lambda \in \Lambda} \left\| \nabla_\theta V^{(e)}\Big(\theta^{(e)}, \lambda\Big) \right\|_2^2. \tag{8}$$

Define

$$g_Q^{(e)} = \frac{1}{n_{\text{cal}}} \sum_{i \in \mathcal{D}_{\text{cal}}^{(e)}} \nabla_\theta \ell\Big(\theta^{(e)}; x_i, q_i^{(e)}\Big), \qquad g_{\text{tr}}^{(e)} = \frac{1}{n_{\text{tr}}} \sum_{i \in \mathcal{D}_{\text{tr}}^{(e)}} \nabla_\theta \ell(\theta^{(e)}; x_i, y_i). \tag{9}$$

Then $\nabla_\theta V^{(e)}(\theta^{(e)}, \lambda) = g_Q^{(e)} - g_{\text{tr}}^{(e)} - \nabla_\theta \pi(\theta^{(e)}, \lambda)$, and the λ-update reduces to minimizing

$$\zeta^{(e)}(\lambda) = \left\| g_Q^{(e)} - g_{\text{tr}}^{(e)} - \nabla_\theta \pi(\theta^{(e)}, \lambda) \right\|_2^2. \tag{10}$$

Algorithm 1 summarizes the full procedure consisting of three phases. In the warmup, θ is trained on all data with a light fixed penalty. TabCAM then alternates between the θ-step and λ-step. Finally, a refit phase freezes λ and refines θ on all data. The procedure discussed thus far considers the case in which both θ and λ are updated at every TabCAM epoch. In practice, we can update λ only every ξ_λ epochs to reduce the additional computation required for constructing future targets and updating the regularization strengths, while using the full data for θ-updates in epochs without a λ-update.

2.2 Future-target construction

The future-target constructor q in Algorithm 1 generates calibrated future observations. In this subsection, we develop the construction for regression. The specific construction for classification is provided in Appendix C.

Let $F_\theta(y \mid x)$ denote the conditional cumulative distribution function (CDF) of Y given $X = x$ induced by a model with parameter θ. If $\theta = \theta^\star$, then for any observation $(x_i, y_i) \sim \mathbb{P}$,

$$u_i^\star \;=\; F_{\theta^\star}(y_i \mid x_i) \;\sim\; \text{Unif}(0, 1).$$

Conversely, generating a future response from $\theta^\star$ is equivalent to drawing $U^\star \sim \text{Unif}(0, 1)$ and applying the CDF inversion; namely, $Y_{\text{fut}} = F_{\theta^\star}^{-1}(U^\star \mid x)$. In practice we only have the current

Algorithm 1 TabCAM: Tabular Calibrated Auto-Modeling

Require: Dataset $\mathcal{D}$; predictor f_θ; duality function $\pi(\theta, \lambda)$; warmup epochs E_{warm}; TabCAM epochs E_{am}; refit epochs E_{refit}; calibration ratio ρ; λ-update frequency ξ_λ; future-target constructor q.

```
Initialize (θ, λ).
Warmup phase: train θ on all of D for E_warm epochs with a light fixed penalty.
for e = 1, ..., E_am do
    if e mod ξ_λ = 0 then
        Split D into D_tr^(e) and D_cal^(e) with |D_cal^(e)| = ⌊ρ n⌋.
        θ-step: Update θ on D_tr^(e) via SGD/Adam by minimizing G^(e)(·, λ) with current λ, yielding θ^(e).
        Future-target step: construct future targets {q_i^(e)}_{i∈D_cal^(e)} using q(x_i; θ^(e)) for i ∈ D_cal^(e).
        λ-step: compute g_Q^(e) and g_tr^(e) via (9); update λ by minimizing (10).
    else
        Update θ on all of D using the penalized loss G^(e)(θ, λ) with current λ.
    end if
end for
Refit phase: freeze λ* ← λ; retrain θ on all of D for E_refit epochs with penalty π(θ, λ*).
return (θ*, λ*).
```

estimate $\hat{\theta}^{(e)}$, under which $\tilde{u}_i^{(e)} = F_{\hat{\theta}^{(e)}}(y_i \mid x_i)$, $i \in \mathcal{D}_{\text{cal}}^{(e)}$, are no longer exactly uniform. Let $H_{\hat{\theta}^{(e)}}$ denote the empirical CDF of $\{\tilde{u}_i^{(e)}\}_{i \in \mathcal{D}_{\text{cal}}^{(e)}}$. We use $H_{\hat{\theta}^{(e)}}$ as a nonparametric correction for the mismatch between the estimated conditional CDF $F_{\hat{\theta}^{(e)}}(\cdot \mid x)$ and the true one $F_{\theta^\star}(\cdot \mid x)$. Specifically, rather than inverting directly through $F^{-1}_{\hat{\theta}^{(e)}}$, we first pass $U^\star$ through $H^{-1}_{\hat{\theta}^{(e)}}$ and then invert, yielding a surrogate future response:

$$q_{\text{reg},i}^{(e)} = F^{-1}_{\hat{\theta}^{(e)}}\big(H^{-1}_{\hat{\theta}^{(e)}}(u_i^\star) \,\big|\, x_i\big), \qquad u_i^\star \sim \text{Unif}(0,1). \tag{11}$$

We evaluate the standard squared-error regression loss at $(x_i, q_{\text{reg},i}^{(e)})$ and aggregate over the calibration split to form $g_Q^{(e)}$ in (9). The procedure of Jiang & Liu (2025) serves a similar purpose through an expensive offline m-out-of-n bootstrap, while TabCAM replaces this with the efficient online construction in (11).

One working approach to specify the conditional CDF is via the Gaussian model such that $F_{\hat{\theta}^{(e)}}(y \mid x) = \Phi\big(\big(y - f_{\hat{\theta}^{(e)}}(x)\big)/\hat{\sigma}^{(e)}(x)\big)$, where $f_{\hat{\theta}^{(e)}}(x)$ is the current conditional mean predictor, $\hat{\sigma}^{(e)}(x)$ denotes the corresponding conditional scale, and Φ is the standard normal CDF. Appendix B discusses estimation of the scale parameter and a heteroscedastic extension. Algorithm 2 in Appendix B.1 summarizes the resulting regression future-target construction at the e-th epoch. The Gaussian model is only a working choice rather than a structural assumption of TabCAM. Equation (11) applies to any continuous conditional CDF with an analytical or numerical inverse, including heavy-tailed, skewed, and mixture distributions, and the working family can be selected from the observed data. Moreover, our method reduces sensitivity to working-family misspecification by recalibrating the fitted CDF before generating future targets. Appendix H.1 describes a data-driven selection procedure, Appendix H.2 provides further insight into this robustness, and Appendix I.3 gives supporting examples.

2.3 Uncertainty quantification

Because TabCAM models the conditional law of the response during training, conditional uncertainty quantification follows directly. For regression, any fitted continuous conditional CDF $\widehat{F}(\cdot \mid x)$ can be used to construct conditional quantiles and prediction intervals through CDF inversion. For example, under the Gaussian working model, $\widehat{F}(y \mid x) = \Phi\big((y - f_{\hat{\theta}}(x))/\hat{\sigma}(x)\big)$, the conditional τ-quantile is $\hat{q}_\tau(x) = f_{\hat{\theta}}(x) + \hat{\sigma}(x)\Phi^{-1}(\tau)$ where $\tau \in (0,1)$, and the $(1-\alpha)$ prediction interval is

$$\widehat{C}_{1-\alpha}(x) = \big[f_{\hat{\theta}}(x) + \hat{\sigma}(x)\Phi^{-1}(\alpha/2),\ f_{\hat{\theta}}(x) + \hat{\sigma}(x)\Phi^{-1}(1-\alpha/2)\big], \tag{12}$$

where $(\hat{\theta}, \hat{\sigma}(x))$ denote the fitted parameters. When the assumed Gaussian family matches the true conditional distribution, $\widehat{C}_{1-\alpha}(x)$ achieves exact conditional coverage at every x. For classification, the point prediction is $\arg\max_k p_{\hat{\theta},k}(x)$, and the conditional uncertainty is described by the full probability vector $p_{\hat{\theta}}(\cdot \mid x)$. A natural $(1-\alpha)$ prediction set is the smallest subset $A \subseteq \{1, \ldots, K\}$ such that $\sum_{k \in A} p_{\hat{\theta},k}(x) \geq 1 - \alpha$.

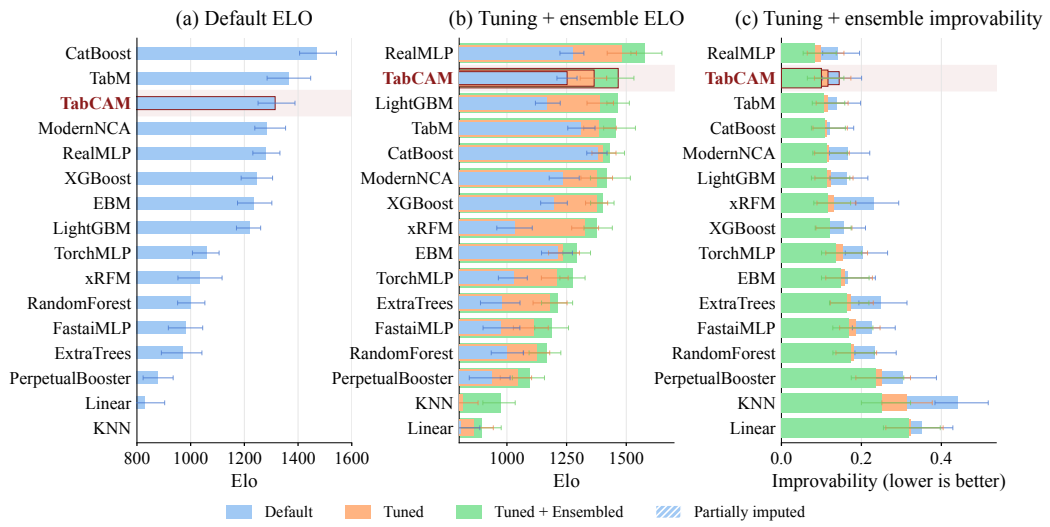


Figure 2: Non-foundation comparison: (a) default Elo, (b) tuned-and-ensembled Elo, and (c) tuned-and-ensembled improvability (lower is better).

3 Experiments

3.1 TabArena

TabArena consists of 51 datasets (13 regression, 30 binary, 8 multiclass). We evaluate TabCAM in two regimes, default hyperparameters on TabArena and 100-config tuning with ensembling on TabArena-Lite that uses a single outer fold per dataset. For these experiments, we use a three-layer MLP backbone with width 256 and PBLD numerical embeddings (Holzmüller et al., 2024). Full implementation details are provided in Appendix D. Note that TabCAM is backbone-agnostic and can be paired with other architectures. Elo values are computed via the standard TabArena toolkit, and baseline results are taken from the official TabArena leaderboard as of May 10, 2026.

Figure 2 reports Elo scores and improvability under the two regimes for the 16 non-foundation TabArena methods. The comparison over more methods is provided in Appendix Figure 10. Under default hyperparameters across all folds, TabCAM attains an Elo of 1314.7, ranking third among the sixteen non-foundation default models, behind only CatBoost default and TabM default. Under 100-config tuning with ensembling on TabArena-Lite, TabCAM reaches an Elo of 1466.3 in the non-foundation pool, comparable to tuned-and-ensembled LightGBM (1463.1) and TabM (1456.9), and clearly ahead of tuned CatBoost (1429.1) and ModernNCA (1419.4). Among the tuned-and-ensembled configurations in the non-foundation pool, only RealMLP ensemble (1577.0) lies above TabCAM, which may be partially due to the fact that our implementation of the proposed TabCAM uses only 100 configurations while RealMLP uses 200. This observation supports our earlier claim that adopting per-parameter regularization via TabCAM can potentially reduce the architectural tuning burden of deep learning models.

Figure 3 reports end-to-end GPU training time on a single NVIDIA A100 (40 GB) and CPU full-batch inference throughput for TabCAM, RealMLP, and AM on ten TabArena tasks. Each dot is one dataset, and diamonds show aggregate means. TabCAM's training time is comparable to RealMLP, while AM, re-implemented on the same backbone with the same 256-epoch schedule, is more than 10 times slower and fails to complete within a four-hour budget on several datasets (*credit_card_clients_default*, *internet_firewall*). This shows that TabCAM has competitive end-to-end training cost in this comparison. More comparisons with other models can be found in Figure 11 of Appendix G. In addition, to evaluate TabCAM as an add-on, we compare TabM and FT-Transformer with and without TabCAM on ten TabArena tasks (Appendix I.1).

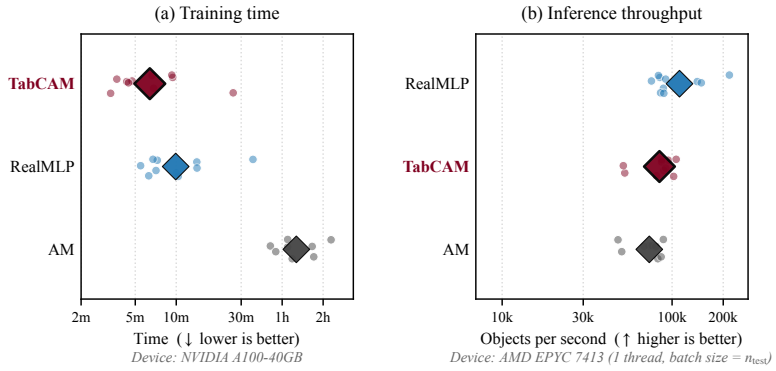


Figure 3: Computational speed comparison among RealMLP, MLP paired with TabCAM and AM (Jiang & Liu, 2025) over ten TabArena tasks. Panel (a) reports GPU training runtime and Panel (b) reports CPU full-batch inference throughput. Each dot represents one dataset and the diamond indicates the aggregate mean.

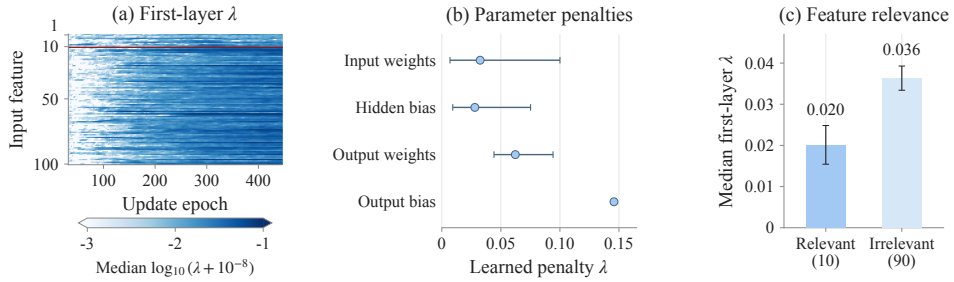


Figure 4: Learned per-parameter regularization in the sparse-teacher MLP simulation ($n = 3200$, 10 relevant and 90 irrelevant covariates). (a) Evolution of first-layer penalties during training. (b) Final penalties across parameter groups. (c) Relevant covariates receive smaller first-layer penalties than irrelevant covariates across 20 seeds, motivating a variable-importance interpretation.

3.2 Simulation studies

We first examine the learned per-parameter regularization in a sparse-teacher MLP simulation with 3200 training observations and 100 covariates, of which only 10 are relevant. Figure 4 shows that the learned λ values are different across parameters. In particular, first-layer weights connected to relevant covariates receive systematically smaller penalties than those connected to irrelevant covariates. Since smaller λ implies weaker shrinkage, aggregating the first-layer penalties associated with each input provides a natural measure of variable importance. Thus, the learned λ not only controls regularization but also provides an interpretable feature-level signal.

To study whether TabCAM recovers the population-optimal regularization, we use a ridge-regression simulation in which the minimizer of population risk is available in closed form. This gives an oracle target to compare with. Specifically, we simulate a sparse linear model with $n = 1000$ and $p = 500$, and define $\lambda^\star = \arg\min_{\lambda \geq 0} R(\lambda)$ as the population minimizer. TabCAM uses L2 duality $\pi(\beta, \lambda) = \lambda \|\beta\|_2^2$ with update frequencies $\xi_\lambda \in \{1, 4, 8\}$. The AM baseline follows Jiang & Liu (2025) with $K = 5$-fold cross-validation and $N = 10$ bootstrap imputations. Results are averaged over 50 replications, and the full data-generating process is given in Appendix F.1.

Figure 5 summarizes the convergence results. With $\xi_\lambda = 1$, TabCAM achieves a risk ratio of $R(\hat{\lambda})/R(\lambda^\star) \approx 1.015$, nearly recovering the population minimizer. AM reaches a higher risk ratio of 1.11, which is close to the performance of TabCAM with $\xi_\lambda = 4$. At the same time, TabCAM is 46–128 times faster than AM, as shown in panel (c). The speedup is larger here than in Figure 3 because the backbone model is simpler.

Finally, we evaluate the uncertainty quantification provided by TabCAM. We consider a nonlinear high-dimensional setting where the responses are generated by a sparse one-hidden-layer ReLU

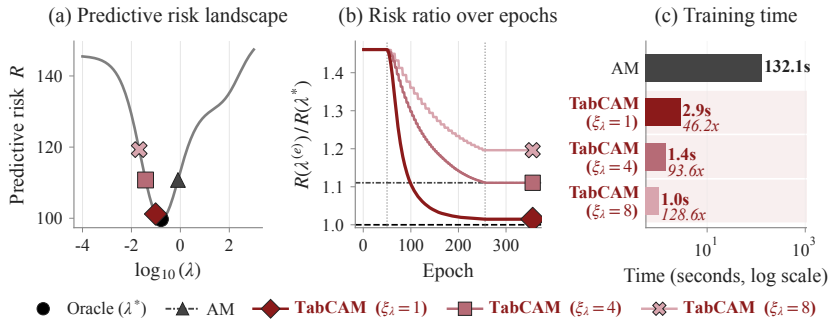


Figure 5: Ridge regression simulation. (a) Predictive risk R vs. $\log_{10}\lambda$; markers indicate each method's converged $\hat{\lambda}$. (b) Risk ratio across epochs; the dashed line marks the AM ratio. (c) Running time (log scale).

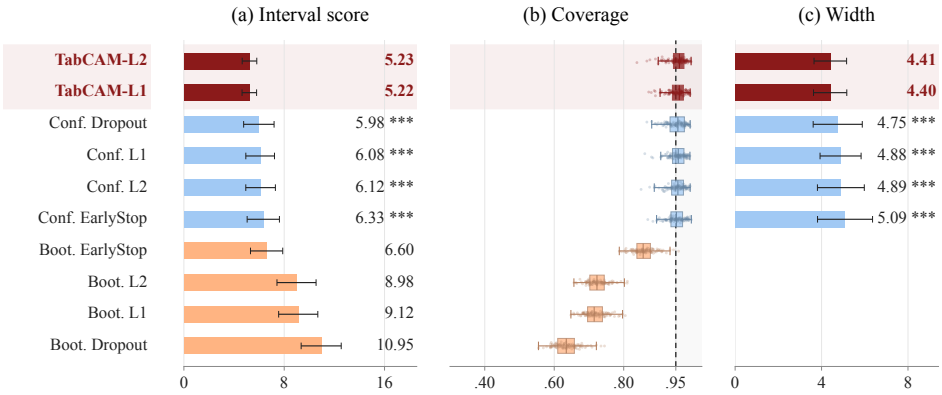


Figure 6: Uncertainty quantification simulation. (a) Mean interval score (lower is better). (b) Empirical coverage. (c) Mean interval width (lower is better). Bars show means with standard-deviation error bars; coverage boxes show medians, interquartile ranges, and 1.5-IQR whiskers, with dots indicating individual runs. "Conf." and "Boot." denote split conformal and residual bootstrap, respectively; L1, L2, Dropout, and EarlyStop indicate the MLP regularization choice. Stars report paired Wilcoxon p-values comparing TabCAM-L2 with each split-conformal baseline ($^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$).

teacher network ($n_{\text{train}} = 200$, $p = 200$) with input-dependent $\sigma(x)$. The noise standard deviation is 12 times larger when $x_1 > \Phi^{-1}(0.8)$, so that 20% of the inputs fall in a high-noise region and the rest form a low-noise region (Appendix F.2). We sweep the signal-to-noise ratio $\kappa \in \{1, \ldots, 8\}$ with 12–13 replications per κ, giving 100 paired runs per method. All methods use the same one-hidden-layer ReLU student MLP with 48 hidden units. TabCAM and split conformal share a $140/60$ fitting–calibration split. We report the results for TabCAM-L1 and TabCAM-L2, corresponding to the implementations under L1 and L2 duality functions defined in (5). The baselines include two uncertainty-quantification methods, split conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2023) and residual bootstrap (Efron & Tibshirani, 1994) ($B = 200$), each paired with four regularization or training choices (L1, L2, dropout (Srivastava et al., 2014) with $p = 0.1$, or early stopping). Full details are in Appendix F.2.

Figure 6 summarizes the results. Both TabCAM variants and all four split-conformal baselines attain near-nominal mean coverage (panel b). The bootstrap baselines under-cover substantially (0.64–0.86). Among the methods with near-nominal coverage, TabCAM produces the shortest intervals (panel c), with average widths of 4.40 (L1) and 4.41 (L2), versus 4.75–5.09 for split-conformal baselines, and achieves the lowest interval scores (Gneiting & Raftery, 2007) in panel (a). Within the low- and high-noise regions, TabCAM-L2 attains coverage of 0.956 and 0.945, compared with 0.975–0.979 and 0.850–0.856 for split conformal (Table 4 in Appendix F.2). Paired Wilcoxon tests across 100 runs confirm that TabCAM has significantly smaller interval score than split-conformal baselines

($p < 0.001$) and significantly narrower width ($p < 0.05$ or better). The lower interval scores and more balanced coverage across noise regions persist with Laplace or standardized Student-$t(3)$ noise (Figures 7 and 8 in Appendix).

4 Related work

Tabular deep learning. Methods trained from scratch, including gradient-boosted trees (Chen & Guestrin, 2016; Ke et al., 2017; Prokhorenkova et al., 2018), engineered neural networks (Holzmüller et al., 2024; Gorishniy et al., 2025; Ye et al., 2025), and kernel machines (Beaglehole et al., 2026), rely on manual tuning to mitigate overfitting (Gorishniy et al., 2021; Kadra et al., 2021; Borisov et al., 2024) rather than directly targeting population risk. In contrast, tabular foundation models (Hollmann et al., 2025; Qu et al., 2025; Zhang et al., 2025b; Ma et al., 2025; Zhang et al., 2025a; Grinsztajn et al., 2025; Bonet et al., 2025; Qu et al., 2026; Grinsztajn et al., 2026) pretrain on large table corpora and predict on new tables with little or no per-dataset training. We therefore benchmark TabCAM against from-scratch methods and view foundation model fine-tuning as a natural extension for TabCAM.

Regularization and complexity control. Dropout (Srivastava et al., 2014), weight decay (Loshchilov & Hutter, 2019), and early stopping provide coarse mechanisms for controlling model complexity. Constrained parameter regularization (Franke et al., 2024) assigns penalties at the per-matrix level, while AlphaDecay (He et al., 2025) assigns per-module weight decay. Neither reaches per-parameter resolution. Gradient-based hyperparameter optimization can learn many regularization coefficients, including online or per-weight updates (MacKay et al., 2019; Lorraine et al., 2020), but still optimizes them against a validation objective. TabCAM instead learns per-parameter regularization online through calibrated future-data gradients that target population-level risk.

Cross-validation, risk estimation, and auto-modeling. Cross-validation (Stone, 1974; Geisser, 1975; Stone, 1977; Bates et al., 2024) and Stein's unbiased risk estimate (Stein, 1981) can tune a scalar regularization parameter but scale poorly to high-dimensional λ. The Auto-Modeling framework of Jiang & Liu (2025) introduces the risk decomposition, but its offline K-fold cross-validation and m-out-of-n bootstrap stage is prohibitively expensive for modern machine learning models. We propose TabCAM with an online calibration procedure, reducing cost by more than 10 times.

Conformal prediction and uncertainty. Split conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2023) provides distribution-free marginal coverage, and Romano et al. (2019) extend this to quantile regression. Temperature scaling (Guo et al., 2017) estimates a probability-calibration map for classification. Foygel Barber et al. (2021) show that exact distribution-free conditional coverage is impossible in finite samples without further assumptions. By modeling the population conditional distribution, TabCAM delivers covariate-conditional uncertainty quantification under mild model assumptions.

5 Discussion

TabCAM brings the statistical idea of population-risk targeting into modern tabular deep learning. By decomposing the population risk and constructing calibrated future targets online, it learns per-parameter regularization within a single training loop and yields covariate-conditional uncertainty quantification as a byproduct. On TabArena, TabCAM ranks third among 16 non-foundation methods by default and improves substantially under modest tuning. Ablations (Appendix G.2) isolate each component's contribution, and additional experiments confirm near-nominal coverage at competitive width on synthetic high-dimensional and real TabArena data (Appendices G.1, G.3). The working-model interface allows the conditional law to be selected from the data, while the add-on structure pairs learned regularization with different predictive architectures. A natural next step is to apply TabCAM to larger differentiable architectures, such as deeper transformers or the fine-tuning of pretrained tabular models, where the need for per-parameter regularization is likely to be more pronounced. Another direction is to replace the parametric working model with neural conditional estimators, which would further relax the distributional assumptions behind the future targets.

AI USE STATEMENT

Large language models (Claude, Anthropic; ChatGPT, OpenAI) were used solely to assist with the writing and presentation of this paper. Based on the authors' draft, this assistance included refining wording for clarity and concision, improving paragraph organization, and adjusting the manuscript's formatting. The authors developed the research ideas and theoretical arguments, designed the experiments, and analyzed and interpreted the results. The authors reviewed each edit suggested by the models and verified that it did not alter any technical claim or reported numerical value. The authors take full responsibility for the final text, technical claims, and reported results.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Appendices B and C specify the full training procedure, and Appendix D lists the architecture, preprocessing, default hyperparameters, and tuning search space. The TabArena protocol, hardware, and baseline configurations are described in Appendix E, and the data-generating processes and settings of all simulations are given in Appendices F and I. Baseline results on TabArena are taken from the official leaderboard as of May 10, 2026. The supplementary material contains anonymized code for TabCAM and for the experiments.

ETHICS STATEMENT

Our experiments use synthetic data and the publicly available datasets of the TabArena benchmark (Erickson et al., 2025). The work involves no human subjects. We adhere to the ICLR Code of Ethics.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Stephen Bates, Trevor Hastie, and Robert Tibshirani. Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association*, 119(546):1434–1445, 2024.

Daniel Beaglehole, David Holzmüller, Adityanarayanan Radhakrishnan, and Mikhail Belkin. xRFM: Accurate, scalable, and interpretable feature learning models for tabular data. In *International Conference on Learning Representations*, 2026.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

David Bonet, Marçal Comajoan Cara, Alvaro Calafell, Daniel Mas Montserrat, and Alexander G. Ioannidis. iLTM: Integrated large tabular model. *arXiv preprint arXiv:2511.15941*, 2025.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024.

Morton B. Brown and Alan B. Forsythe. Robust tests for the equality of variances. *Journal of the American Statistical Association*, 69(346):364–367, 1974.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Peter K. Dunn and Gordon K. Smyth. Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3):236–244, 1996.

Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994.

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

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, 10(2):455–482, 2021.

Jörg K. H. Franke, Michael Hefenbrock, Gregor Koehler, and Frank Hutter. Improving deep learning optimization through constrained parameter regularization. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

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

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

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

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1321–1330. PMLR, 2017.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

Di He, Songjun Tu, Ajay Jaiswal, Li Shen, Ganzhao Yuan, Shiwei Liu, and Lu Yin. AlphaDecay: Module-wise weight decay for heavy-tailed balancing in LLMs. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

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

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

Bingqing Hu and Bin Nan. Conditional distribution function estimation using neural networks for censored and uncensored data. *Journal of Machine Learning Research*, 24(223):1–26, 2023.

Yiran Jiang and Chuanhai Liu. Estimation of over-parameterized models from an auto-modeling perspective. *Journal of the American Statistical Association*, 120(552):2422–2434, 2025.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. In *Advances in Neural Information Processing Systems*, volume 34, pp. 23928–23941, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with Dirichlet calibration. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108, pp. 1540–1552. PMLR, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

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

Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations*, 2019.

Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *International Conference on Learning Representations*, 2020.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

D. Peel and G. J. McLachlan. Robust mixture modelling using the t distribution. *Statistics and Computing*, 10(4):339–348, 2000.

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

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 50817–50847. PMLR, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Samuel S. Shapiro and Martin B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3–4):591–611, 1965.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):1135–1151, 1981.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B*, 36(2):111–133, 1974.

M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society: Series B*, 39(1):44–47, 1977.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Han-Jia Ye, Huai-Hong Yin, De-Chuan Zhan, and Wei-Lun Chao. Revisiting nearest neighbor for tabular data: A deep tabular baseline two decades later. In *International Conference on Learning Representations*, 2025.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

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

Xiyuan Zhang, Danielle Maddix Robinson, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael Mahoney, Tony Hu, Huzefa Rangwala, George Karypis, and Yuyang (Bernie) Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

A Proof of Proposition 1 and matching analysis

A.1 Proposition 1 and proof

Proposition 1. Let $(\tilde{\theta}, \tilde{\lambda})$ be a feasible point of (6), with $G(\cdot, \tilde{\lambda})$ twice continuously differentiable and $\nabla^2_\theta G(\tilde{\theta}, \tilde{\lambda}) \succ 0$. Suppose $\pi(\theta, \lambda) = \sum_{j=1}^k \lambda_j \rho_j(\theta)$ with each ρ_j differentiable at $\tilde{\theta}$. If $\tilde{\lambda}$ satisfies

$$\sum_{j=1}^{k} \tilde{\lambda}_j \, \nabla_\theta \rho_j(\tilde{\theta}) = \nabla R_P(\tilde{\theta}) - \nabla R_n(\tilde{\theta}), \tag{13}$$

then $\nabla_\theta V(\tilde{\theta}, \tilde{\lambda}) = 0$. Moreover, if $V(\cdot, \tilde{\lambda})$ is locally convex around $\tilde{\theta}$, then $\tilde{\theta}$ is also a local minimizer of $V(\cdot, \tilde{\lambda})$.

Since $(\tilde{\theta}, \tilde{\lambda})$ is a feasible point of (6), we have

$$\tilde{\theta} \in \arg\min_{\vartheta} G(\vartheta, \tilde{\lambda}).$$

Moreover, since

$$\nabla^2_\theta G(\tilde{\theta}, \tilde{\lambda}) \succ 0,$$

this lower-level solution is locally non-degenerate, so $\tilde{\theta}$ is a strict local minimizer of $G(\cdot, \tilde{\lambda})$ rather than a saddle point or flat region.

By the definition of V in (4),

$$\nabla_\theta V(\theta, \lambda) = \nabla R_P(\theta) - \nabla R_n(\theta) - \nabla_\theta \pi(\theta, \lambda). \tag{14}$$

Under the condition

$$\pi(\theta, \lambda) = \sum_{j=1}^{k} \lambda_j \rho_j(\theta)$$

with each ρ_j differentiable at $\tilde{\theta}$, we have

$$\nabla_\theta \pi(\tilde{\theta}, \tilde{\lambda}) = \sum_{j=1}^{k} \tilde{\lambda}_j \, \nabla_\theta \rho_j(\tilde{\theta}).$$

The matching condition (13) then states exactly that

$$\nabla_\theta \pi(\tilde{\theta}, \tilde{\lambda}) = \nabla R_P(\tilde{\theta}) - \nabla R_n(\tilde{\theta}).$$

Substituting this identity into (14) at $(\tilde{\theta}, \tilde{\lambda})$ yields

$$\nabla_\theta V(\tilde{\theta}, \tilde{\lambda}) = 0.$$

If, in addition, $V(\cdot, \tilde{\lambda})$ is locally convex around $\tilde{\theta}$, then this first-order condition implies that $\tilde{\theta}$ is a local minimizer of $V(\cdot, \tilde{\lambda})$. □

A.2 Solvability of the gradient-matching system for L1 and L2 duality functions

In this section, we characterize when the duality functions defined in (5) satisfy condition (13).

The matching condition (13) is a linear system in the k unknowns $\tilde{\lambda} = (\tilde{\lambda}_1, \ldots, \tilde{\lambda}_k) \in \mathbb{R}^k$,

$$M(\tilde{\theta})\,\tilde{\lambda} = d, \qquad M(\tilde{\theta}) = \big[\nabla_\theta \rho_1(\tilde{\theta}), \ldots, \nabla_\theta \rho_k(\tilde{\theta})\big] \in \mathbb{R}^{k \times k}, \qquad d = \nabla R_P(\tilde{\theta}) - \nabla R_n(\tilde{\theta}). \tag{15}$$

Thus exact solvability is determined by whether d lies in the column space of $M(\tilde{\theta})$. When $M(\tilde{\theta})$ is invertible, the solution is unique.

For L2, $\pi_{\mathrm{L2}}(\theta, \lambda) = \sum_j \lambda_j \theta_j^2$ gives

$$\nabla_\theta \rho_j(\tilde{\theta}) = 2\tilde{\theta}_j \, e_j,$$

where e_j is the j-th standard basis vector. Hence

$$M(\tilde{\theta}) = 2\,\mathrm{diag}(\tilde{\theta}_1, \ldots, \tilde{\theta}_k).$$

If every $\tilde{\theta}_j \neq 0$, then $M(\tilde{\theta})$ is invertible, with closed-form solution

$$\tilde{\lambda}_j = \frac{d_j}{2\tilde{\theta}_j}.$$

For L1, $\pi_{\mathrm{L1}}(\theta, \lambda) = \sum_j \lambda_j |\theta_j|$ gives, on coordinates with $\tilde{\theta}_j \neq 0$,

$$\nabla_\theta \rho_j(\tilde{\theta}) = \mathrm{sign}(\tilde{\theta}_j)\, e_j.$$

Thus, if every $\tilde{\theta}_j \neq 0$, then

$$M(\tilde{\theta}) = \mathrm{diag}\big(\mathrm{sign}(\tilde{\theta}_1), \ldots, \mathrm{sign}(\tilde{\theta}_k)\big)$$

is invertible, with closed-form solution

$$\tilde{\lambda}_j = d_j\, \mathrm{sign}(\tilde{\theta}_j).$$

When some coordinate satisfies $\tilde{\theta}_j = 0$, the preceding coordinatewise matching argument may fail. For L2, the corresponding column of $M(\tilde{\theta})$ is zero, so exact matching requires $d_j = 0$ on that coordinate and, even then, $\tilde{\lambda}_j$ is not identified by the matching equation. For L1, the function $|\theta_j|$ is not differentiable at zero, so the gradient-based argument applies only away from zero unless a subgradient formulation or a smooth approximation is used. Combined with the additional nonnegativity constraint $\lambda \geq 0$ adopted in practice, the closed-form solutions above may also fall outside the feasible set even when $M(\tilde{\theta})$ is invertible. In these situations, TabCAM replaces the equality system by minimizing $\|\nabla_\theta V(\theta, \lambda)\|_2^2$ over $\lambda \geq 0$.

A.3 KKT INTERPRETATION OF THE GRADIENT-MATCHING CONDITION

This subsection explains the role of $\|\nabla_\theta V(\theta, \lambda)\|_2^2$ from the viewpoint of constrained optimization. The KKT argument below clarifies what condition (13) means for the constrained problem (6).

The constraint in (6) requires

$$\theta \in \arg\min_{\vartheta} G(\vartheta, \lambda).$$

Locally, when $G(\cdot, \lambda)$ is smooth and the solution is non-degenerate, this can be represented by the first-order condition

$$\nabla_\theta G(\theta, \lambda) = 0.$$

Let

$$F(\theta, \lambda) = G(\theta, \lambda) + V(\theta, \lambda).$$

Then (6) can be viewed locally as

$$\min_{\theta, \lambda} F(\theta, \lambda) \qquad \text{subject to} \qquad \nabla_\theta G(\theta, \lambda) = 0.$$

Introduce a Lagrange multiplier u for this equality constraint. The Lagrangian is

$$\mathcal{L}(\theta, \lambda, u) = G(\theta, \lambda) + V(\theta, \lambda) + u^\top \nabla_\theta G(\theta, \lambda).$$

The KKT stationarity condition with respect to θ is

$$\nabla_\theta F(\theta, \lambda) + \nabla^2_{\theta\theta} G(\theta, \lambda) u = 0.$$

At a feasible point, $\nabla_\theta G(\theta, \lambda) = 0$. Since $F = G + V$, we have

$$\nabla_\theta F(\theta, \lambda) = \nabla_\theta V(\theta, \lambda).$$

Therefore the KKT condition becomes

$$\nabla_\theta V(\theta, \lambda) = -\nabla^2_{\theta\theta} G(\theta, \lambda) u. \tag{16}$$

Equation (16) shows that a general constrained optimum does not necessarily satisfy $\nabla_\theta V(\theta, \lambda) = 0$. The gradient $\nabla_\theta V$ may instead be balanced by the multiplier u. If $\nabla^2_{\theta\theta} G(\theta, \lambda)$ is non-singular, the multiplier u is determined by

$$u = -\big[\nabla^2_{\theta\theta} G(\theta, \lambda)\big]^{-1} \nabla_\theta V(\theta, \lambda).$$

Thus $\nabla_\theta V(\theta, \lambda) = 0$ corresponds to the special case $u = 0$.

A.4 Closed-form expressions for the λ-update

The following example records the gradients of π and the resulting per-coordinate updates of $\zeta^{(e)}$ used inside the λ-step in Algorithm 1.

Example 1. For the duality functions in (5),

$$\nabla_\theta \pi_{\mathrm{L1}}(\theta, \lambda) = (\lambda_j \operatorname{sign}(\theta_j))_{j=1}^k, \qquad \nabla_\theta \pi_{\mathrm{L2}}(\theta, \lambda) = (2\lambda_j \theta_j)_{j=1}^k.$$

Writing $r^{(e)}(\lambda) = g_Q^{(e)} - g_{\mathrm{tr}}^{(e)} - \nabla_\theta \pi(\theta^{(e)}, \lambda)$, the coordinatewise gradient of $\zeta^{(e)}$ is

$$\frac{\partial \zeta^{(e)}(\lambda)}{\partial \lambda_j} = \begin{cases} -2\, r_j^{(e)}(\lambda)\ \operatorname{sign}(\theta_j^{(e)}), & \text{vector L1}, \\ -4\, r_j^{(e)}(\lambda)\, \theta_j^{(e)}, & \text{vector L2}. \end{cases} \tag{17}$$

B Regression future-target construction and scale estimation

B.1 Future target construction algorithm for regression

The future target construction for regression task is summarized in Algorithm 2.

Algorithm 2 Regression future-target construction

Require: Calibration set $\mathcal{D}_{\mathrm{cal}}^{(e)}$; current model $F_{\hat\theta^{(e)}}(\cdot \mid x)$.

1: **for all** $i \in \mathcal{D}_{\mathrm{cal}}^{(e)}$ **do**
2: Compute PIT score $\tilde u_i^{(e)} \leftarrow F_{\hat\theta^{(e)}}(y_i \mid x_i)$.
3: **end for**
4: Form the empirical CDF $H_{\hat\theta^{(e)}}$ of $\{\tilde u_i^{(e)}\}$ and its generalized inverse $H^{-1}_{\hat\theta^{(e)}}$.
5: **for all** $i \in \mathcal{D}_{\mathrm{cal}}^{(e)}$ **do**
6: Draw $u_i^\star \sim \mathrm{Unif}(0,1)$; set $\tilde u_i \leftarrow H^{-1}_{\hat\theta^{(e)}}(u_i^\star)$.
7: $q_{\mathrm{reg},i}^{(e)} \leftarrow F^{-1}_{\hat\theta^{(e)}}(\tilde u_i \mid x_i)$.
8: **end for**
9: **return** $\{q_{\mathrm{reg},i}^{(e)}\}_{i \in \mathcal{D}_{\mathrm{cal}}^{(e)}}$ for use in (9).

B.2 Estimation of scale parameter

Write

$$Y \mid X = x \sim \mathcal{N}(f_\theta(x), \sigma^2),$$

and let

$$\vartheta = (\theta, \sigma), \qquad \sigma > 0.$$

The corresponding per-sample negative log-likelihood is

$$\ell_{\mathrm{reg}}(\vartheta; x, y) = \tfrac{1}{2}\log(\sigma^2) + \tfrac{(y - f_\theta(x))^2}{2\sigma^2}.$$

In this formulation, the mean parameter θ and the scale parameter σ are estimated jointly. A natural duality function is

$$\pi_{\mathrm{reg}}(\vartheta, \lambda) = \sum_{j=1}^k \lambda_j \rho_j(\theta) + \lambda_\sigma \log(1/\sigma^2), \qquad 0 \le \lambda_\sigma < 1/2.$$

The resulting conditional quantile and prediction interval take the form

$$\hat q_\tau(x) = f_{\hat\theta}(x) + \hat\sigma\, \Phi^{-1}(\tau), \qquad \tau \in (0,1),$$

and

$$\widehat C_{1-\alpha}(x) = \big[f_{\hat\theta}(x) + \hat\sigma\, \Phi^{-1}(\alpha/2),\ f_{\hat\theta}(x) + \hat\sigma\, \Phi^{-1}(1-\alpha/2)\big].$$

Jointly optimizing σ within the full training loop is sometimes numerically unstable. The implementation therefore supports joint optimization as an option and also supports the plug-in estimator $\hat\sigma^{(e)} = \sqrt{\frac{1}{n_{\mathrm{cal}}} \sum_{i \in \mathcal{D}_{\mathrm{cal}}^{(e)}} (r_i^{(e)})^2}$ as another option.

B.3 Additional implementation details for regression future-target construction

In the main implementation, we adopt a Gaussian working model for the conditional CDF. At e-th epoch,

$$\hat{\mu}_i^{(e)} = f_{\theta^{(e)}}(x_i), \qquad r_i^{(e)} = y_i - \hat{\mu}_i^{(e)},$$

with $\hat{\sigma}^{(e)}$ clipped below at 10^{-6} for numerical stability. The working conditional CDF is then

$$F_{\hat{\theta}^{(e)}}(y \mid x) = \Phi\left(\frac{y - \hat{\mu}^{(e)}(x)}{\hat{\sigma}^{(e)}}\right),$$

where Φ is the standard normal CDF. The corresponding PIT scores are

$$u_i^{(e)} = \Phi(r_i^{(e)}/\hat{\sigma}^{(e)}),$$

clipped to $[10^{-8}, 1-10^{-8}]$. Let $u_{(1)} \le \cdots \le u_{(n_{\mathrm{cal}})}$ denote the sorted PIT scores. We define the generalized inverse of the empirical CDF $H_{\hat{\theta}^{(e)}}$ by

$$H^{-1}_{\hat{\theta}^{(e)}}(v) = u_{(\lceil n_{\mathrm{cal}} \cdot v \rceil)}.$$

In the main text, the regression future target is a single draw

$$q^{(e)}_{\mathrm{reg},i} = \hat{\mu}_i^{(e)} + \hat{\sigma}^{(e)}\, \Phi^{-1}\big(H^{-1}_{\hat{\theta}^{(e)}}(u_i^\star)\big), \qquad u_i^\star \sim \mathrm{Unif}(0,1).$$

The corresponding future-target loss is

$$\ell(\theta; x_i, q^{(e)}_{\mathrm{reg},i}) = (f_\theta(x_i) - q^{(e)}_{\mathrm{reg},i})^2,$$

which is the standard squared loss with $q^{(e)}_{\mathrm{reg},i}$ in place of y_i.

The implementation can optionally average multiple future-target draws. Specifically, we generate

$$q^{(b)}_{\mathrm{reg},i} = \hat{\mu}_i^{(e)} + \hat{\sigma}^{(e)}\, \Phi^{-1}\big(H^{-1}_{\hat{\theta}^{(e)}}(u_b^\star)\big), \qquad b = 1, \dots, B,$$

with independent $u_b^\star \sim \mathrm{Unif}(0,1)$, and replace the single draw by

$$\bar{q}^{(e)}_{\mathrm{reg},i} = \frac{1}{B}\sum_{b=1}^{B} q^{(b)}_{\mathrm{reg},i}.$$

In that case, the future-target loss becomes

$$\ell(\theta; x_i, \bar{q}^{(e)}_{\mathrm{reg},i}) = (f_\theta(x_i) - \bar{q}^{(e)}_{\mathrm{reg},i})^2.$$

B.4 Heteroscedastic implementation

We replace the constant scale σ with an input-dependent scale $\sigma_\gamma(x) > 0$ parameterized by γ. A convenient choice is

$$\log \sigma_\gamma^2(x) = g_\gamma(x).$$

In the simulation, $g_\gamma(x) = \gamma_0 + \gamma_1 \mathbf{1}\{x_1 > c\}$, with $c = \Phi^{-1}(0.8)$. The two scale coefficients are fitted from residuals. The working model then becomes

$$Y \mid X = x \sim \mathcal{N}(f_\theta(x), \sigma_\gamma^2(x)).$$

The corresponding Gaussian deviance, up to an additive constant, is

$$\ell_{\mathrm{reg}}((\theta,\gamma); x, y) = \log \sigma_\gamma^2(x) + \frac{(y - f_\theta(x))^2}{\sigma_\gamma^2(x)}.$$

Under this extension, the PIT score at epoch e becomes

$$u_i^{(e)} = \Phi\left(\frac{y_i - f_{\theta^{(e)}}(x_i)}{\sigma_{\gamma^{(e)}}(x_i)}\right).$$

With the fitted scale held fixed, the mean update uses the scale-weighted squared loss. Writing $z_j^{(e)} = (y_j - f_{\theta^{(e)}}(x_j))/\sigma_{\gamma^{(e)}}(x_j)$, calibrated future targets take the form $q_i^{(e)} = f_{\theta^{(e)}}(x_i) + \sigma_{\gamma^{(e)}}(x_i) z_J^{(e)}$, where J is sampled uniformly from the current calibration split. Let $\hat{G}$ be the empirical CDF of standardized residuals. The calibrated conditional quantile and prediction interval are

$$\hat{q}_\tau(x) = f_{\hat{\theta}}(x) + \hat{\sigma}_{\hat{\gamma}}(x)\, \hat{G}^{-1}(\tau)$$

and

$$\widehat{C}_{1-\alpha}(x) = \big[f_{\hat{\theta}}(x) + \hat{\sigma}_{\hat{\gamma}}(x)\, \hat{G}^{-1}(\alpha/2),\ f_{\hat{\theta}}(x) + \hat{\sigma}_{\hat{\gamma}}(x)\, \hat{G}^{-1}(1-\alpha/2)\big].$$

For the symmetric-noise experiments in Appendix F.2, we use absolute standardized residuals to calibrate symmetric intervals; the final scale fit and interval calibration use disjoint data.

C Future target construction for classification

For classification with K classes, the natural analogue of $F_\theta(y \mid x)$ is the conditional class distribution $p_\theta(\cdot \mid x) \in \Delta^{K-1}$, where $\Delta^{K-1} = \{q \in [0,1]^K : \sum_{k=1}^K q_k = 1\}$ denotes the probability simplex, and $p_{\theta,k}(x) = [p_\theta(\cdot \mid x)]_k$ denotes the model probability assigned to class k. Under the true $\theta^\star$, $p_{\theta^\star}(\cdot \mid x)$ is the same as $\mathbb{P}(Y{=}\cdot \mid X{=}x)$. The current estimate $\hat{\theta}^{(e)}$ yields $\hat{p}_i^{(e)} = p_{\hat{\theta}^{(e)}}(\cdot \mid x_i)$, which deviates from the true conditional. Given calibration pairs $\{(\hat{p}_i^{(e)}, y_i)\}_{i \in \mathcal{D}_{\mathrm{cal}}^{(e)}}$, we fit a calibration map $\mathcal{C}^{(e)} : \Delta^{K-1} \to \Delta^{K-1}$ that corrects the raw probabilities toward the true conditional, playing a role analogous to the calibration correction in regression. The default calibration map is temperature scaling (Guo et al., 2017).

Applying $\mathcal{C}^{(e)}$ yields a calibrated soft future-label distribution $q_{\mathrm{class},i}^{(e)} = \mathcal{C}^{(e)}(\hat{p}_i^{(e)}) \in \Delta^{K-1}$. Equivalently, one may view a future class label $\widetilde{Y}_i^{(e)}$ as being drawn from the calibrated class distribution, in the sense that $\mathbb{P}(\widetilde{Y}_i^{(e)} = k \mid x_i) = \left[q_{\mathrm{class},i}^{(e)}\right]_k$ for $k = 1, \ldots, K$. Instead of sampling such a future label, we use the corresponding expected cross-entropy loss,

$$\ell(\theta; x_i, q_{\mathrm{class},i}^{(e)}) = \mathbb{E}_{\widetilde{Y}_i^{(e)} \mid x_i}\left[-\log p_{\theta, \widetilde{Y}_i^{(e)}}(x_i)\right] = -\sum_{k=1}^K \left[q_{\mathrm{class},i}^{(e)}\right]_k \log p_{\theta,k}(x_i). \tag{18}$$

We aggregate this loss over the calibration split to form $g_Q^{(e)}$ in (9). Algorithm 3 summarizes the corresponding classification future-target construction.

Algorithm 3 Classification future-target construction

Require: Calibration set $\mathcal{D}_{\mathrm{cal}}^{(e)}$; current classifier $p_{\hat{\theta}^{(e)}}(\cdot \mid x)$; calibration-map family $\mathfrak{C}$.

1: **for all** $i \in \mathcal{D}_{\mathrm{cal}}^{(e)}$ **do**
2: Compute raw probabilities $\hat{p}_i^{(e)} \leftarrow p_{\hat{\theta}^{(e)}}(\cdot \mid x_i)$.
3: **end for**
4: Fit calibration map $\mathcal{C}^{(e)} \in \mathfrak{C}$ using $\{(\hat{p}_i^{(e)}, y_i)\}_{i \in \mathcal{D}_{\mathrm{cal}}^{(e)}}$.
5: **for all** $i \in \mathcal{D}_{\mathrm{cal}}^{(e)}$ **do**
6: $q_{\mathrm{class},i}^{(e)} \leftarrow \mathcal{C}^{(e)}(\hat{p}_i^{(e)}) \in \Delta^{K-1}$.
7: **end for**
8: **return** $\{q_{\mathrm{class},i}^{(e)}\}_{i \in \mathcal{D}_{\mathrm{cal}}^{(e)}}$ for use in (9).

For classification, the default calibration map is temperature scaling (Guo et al., 2017). Let $z_i^{(e)} \in \mathbb{R}^K$ denote the current logits, so that $\hat{p}_i^{(e)} = \mathrm{softmax}(z_i^{(e)})$. For a temperature $T^{(e)} > 0$, we have $\mathcal{C}_{\mathrm{temp}}^{(e)}(\hat{p}_i^{(e)}) = \mathrm{softmax}(z_i^{(e)}/T^{(e)})$, where the temperature $T^{(e)}$ is selected from $\mathcal{T}_{\mathrm{grid}} = \{1.0, 1.25, 1.5, 2.0, 3.0, 5.0\}$ by minimizing the calibration cross-entropy on $\mathcal{D}_{\mathrm{cal}}^{(e)}$. The calibrated probabilities are $q_{\mathrm{class},i}^{(e)} = \mathrm{softmax}(z_i^{(e)}/T^{(e)})$.

In practice, the calibration split for classification uses stratified sampling, ensuring each class retains at least two training examples whenever possible.

D TabCAM Implementation details

D.1 Model architecture

Note that TabCAM is backbone-agnostic and can be paired with other architectures. In the TabArena experiments, TabCAM uses an MLP backbone: a ScalingLayer, PBLD embeddings for numerical features, NTP linear layers, CatEmbedding for high-cardinality categoricals (> 9 unique values), and Mish activations (Misra, 2019). The default architecture has 3 hidden layers of width 256 (Holzmüller et al., 2024).

D.2 Preprocessing pipeline

The pipeline follows RealMLP-TD. Categoricals with ≤ 9 unique values use a custom one-hot encoding (binary $\rightarrow \{-1, 0, 1\}$, NaN $\rightarrow$ all zeros); those with > 9 values use ordinal encoding for `nn.Embedding` with a reserved unseen/NaN index. Numeric features pass through directly. All continuous features (numeric plus one-hot) are RobustScaled then SmoothClipped via $x/\sqrt{1 + (x/3)^2}$. The output layout is [scaled continuous | categorical ordinal indices].

D.3 Training hyperparameters

Table 1 lists default hyperparameters for TabCAM.

Table 1: Default hyperparameters for TabCAM.

Parameter	Default	Description
Architecture		
`hidden_width`	256	Hidden layer width
`n_hidden_layers`	3	Number of hidden layers
`plr_h1` / `plr_h2`	16/4	PBLD embedding hidden dimensions
`plr_sigma`	0.1	PBLD initialization scale
`embedding_size`	8	Dimension for high-cardinality categorical embeddings
`act`	mish	Activation function
Optimization		
`n_epochs`	256	Total training epochs
`base_lr_cls`	0.04	Base learning rate (classification)
`base_lr_reg`	0.2	Base learning rate (regression)
`scale_lr_factor`	6.0	Learning rate multiplier for ScalingLayer
TabCAM-specific		
`warmup_epochs`	20	Epochs before online calibration phase begins
`refit_epochs`	30	Refit epochs with frozen λ
`calib_ratio`	$1/3$	Fraction of data used for calibration
`b_imp`	10	Number of Monte Carlo imputations (regression)
`lambda_lr`	3×10^{-6}	Adam learning rate for λ updates
`s_lambda`	1	Number of Adam steps per λ-update
`lambda_step_freq`	2	λ-update frequency (every ξ_λ TabCAM epochs)
`lambda_warmup`	20	Linear warmup epochs for λ learning rate
`lambda_type`	vector L2	Vector L2 duality function in (5)

The λ-update in (10) uses an Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. By default, a single Adam step ($S_\lambda = 1$) is taken per λ-update epoch, with linear learning-rate warmup over the first 20 TabCAM epochs. After each step, λ is projected onto $\mathbb{R}^k_{\geq 0}$. Table 2 defines the hyperparameter search space used for tuned configurations.

Table 2: Hyperparameter search space for TabCAM. The tuning experiments use a random search of 100 configurations on TabArena-Lite.

Parameter	Type	Range / Options
`hidden_width`	choice	$\{256, 384, 512\}$
`n_hidden_layers`	choice	$\{2, 3, 4\}$
`base_lr_cls/reg`	log-uniform	$[0.02, 0.3]$
`plr_sigma`	log-uniform	$[0.01, 50.0]$
`plr_h1` / `plr_h2`	choice	$\{8, 16, 32, 64\}$
`n_epochs`	choice	$\{256, 384, 512\}$
`scale_lr_factor`	log-uniform	$[2.0, 10.0]$
`embedding_size`	choice	$\{4, 8, 16\}$
`lambda_lr`	log-uniform	$[10^{-7}, 10^{-4}]$
`warmup_epochs`	choice	$\{20, 30, 50, 80\}$
`lambda_warmup`	choice	$\{0, 10, 20, 40\}$
`refit_epochs`	choice	$\{30, 50, 100, 150\}$
`calib_ratio`	choice	$\{0.2, 0.33, 0.5\}$
`b_imp`	choice	$\{1, 5, 10, 20\}$
`lambda_type`	choice	{`single L1`, `vector L1`, `single L2`, `vector L2`}
`lambda_step_freq`	choice	$\{1, 2, 4, 8, 16\}$
`act`	choice	{`mish`, `selu`}
`label_smooth`	choice	$\{0.0, 0.05, 0.1\}$

E TabArena benchmark details

E.1 Dataset summary and evaluation protocol

TabArena (Erickson et al., 2025) consists of 51 datasets (13 regression, 30 binary, 8 multiclass) with sizes from 748 to 150,000 samples. Each dataset has a fixed number of outer CV folds. The evaluation metrics are RMSE (regression), ROC-AUC (binary), and log-loss (multiclass). We follow the official TabArena evaluation protocol.

E.2 Datasets used in the runtime figure

Table 3 lists the ten TabArena tasks used in Figure 3. For every method we use the canonical fold $= 0$, repeat $= 0$ TabArena split.

Table 3: The ten TabArena tasks used in Figure 3.

TID	Dataset	Task	n
363619	Bank_Customer_Churn	binary	10,000
363632	E-CommerceShippingData	binary	10,999
363691	online_shoppers_intention	binary	12,330
363681	in_vehicle_coupon_recommendation	binary	12,684
363686	miami_housing	regression	13,776
363709	fifa	regression	19,178
363678	houses	regression	20,640
363705	superconductivity	regression	21,263
363627	credit_card_clients_default	binary	30,000
363710	internet_firewall	multiclass-4	65,532

E.3 Computational setup

The original full-benchmark TabArena evaluations are run on an NVIDIA A100–40 GB GPU. The supplementary matched experiments in Appendix I.1 are run locally on CPU. The runtime experiment is executed with identical hardware and thread budgets for every method. Training is run on one NVIDIA A100–40 GB GPU with 8 OpenMP/MKL threads. Inference throughput is measured on the host CPU of the same A100 node, so all nine methods are timed on the same CPU hardware. Before timing we move every learned `torch.nn.Module` to CPU, pin the process to a

single thread (`torch.set_num_threads(1)`, `torch.set_num_interop_threads(1)`, `OMP/MKL/OPENBLAS/NUMEXPR_NUM_THREADS` = 1), run one `predict_proba(X_test)` warm-up, and report the throughput $n_{\text{test}}/t_{\text{infer}}$ from one timed full-batch call.

E.4 Baseline descriptions

All methods are wrapped as AutoGluon (Erickson et al., 2020) `AbstractModel` subclasses inside TabArena.

TabCAM. 256 epochs, `lambda_step_freq`= 2, vector-L2 duality, calibration ratio $1/3$.

AM. Re-implemented on top of the same backbone as TabCAM, with 256 epochs and 5 imputation/estimation rounds following Jiang & Liu (2025).

RealMLP. AutoGluon's `RealMLPModel` (Holzmüller et al., 2024), default hyperparameters.

TabM. AutoGluon's `TabMModel` (Gorishniy et al., 2025), default hyperparameters.

ModernNCA. TabArena's `ModernNCAModel` (Ye et al., 2025), default hyperparameters.

TorchMLP. AutoGluon's `TabularNeuralNetTorchModel` (Erickson et al., 2020), default hyperparameters.

LightGBM. AutoGluon's `LGBModel` (Ke et al., 2017), default hyperparameters.

XGBoost. AutoGluon's `XGBoostModel` (Chen & Guestrin, 2016), default hyperparameters.

EBM. AutoGluon's `EBMModel` (Nori et al., 2019), default hyperparameters.

F Simulation study details

F.1 Ridge regression simulation details

Following Jiang & Liu (2025), covariates $x_i \in \mathbb{R}^p$ are drawn i.i.d. from $\mathcal{N}(0, \Sigma)$ with compound-symmetric covariance

$$\Sigma = (1-\rho)I_p + \rho \mathbf{1}\mathbf{1}^\top, \qquad \rho = 0.5,$$

so all pairwise correlations equal ρ. The true coefficient vector $\beta \in \mathbb{R}^p$ is sparse with $s = \lceil n^\alpha \rceil$ nonzero entries chosen uniformly at random, with values drawn from $\mathrm{Laplace}(0, 1)$. Responses are generated by

$$y_i = x_i^\top \beta + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2), \qquad \sigma^2 = \beta^\top \Sigma\, \beta/\kappa.$$

Both covariates and responses are standardized before fitting, and all reported metrics are computed on the original response scale. The ridge oracle experiment uses $n = 1000$, $p = 500$, $\alpha = 0.6$, and $\kappa = 1$.

We run TabCAM with linear predictor $f_\beta(x) = x^\top \beta$, half squared-error loss, and L2 duality function $\pi(\beta, \lambda) = \lambda \|\beta\|_2^2$. The penalized empirical objective admits the closed-form minimizer

$$\hat{\beta}(\lambda) = (X^\top X + 2\lambda n\, I_p)^{-1} X^\top y.$$

The corresponding population prediction risk is

$$R(\lambda) = \sigma^2 + \big(\hat{\beta}(\lambda) - \beta\big)^\top \Sigma \big(\hat{\beta}(\lambda) - \beta\big),$$

and the oracle $\lambda^\star = \arg\min_{\lambda \geq 0} R(\lambda)$ is computed by numerical optimization using full knowledge of $(\beta, \Sigma, \sigma^2)$.

When specialized to ridge with the L2 duality function, the gradient discrepancy (10) becomes

$$\zeta^{(e)}(\lambda) = \big\| g_Q^{(e)} - g_{\text{tr}}^{(e)} - 2\lambda\, \beta^{(e)} \big\|_2^2,$$

with coordinatewise gradient

$$\frac{\partial \zeta^{(e)}}{\partial \lambda} = -4 \big\langle \beta^{(e)},\, r^{(e)}(\lambda) \big\rangle, \qquad r^{(e)}(\lambda) = g_Q^{(e)} - g_{\text{tr}}^{(e)} - 2\lambda \beta^{(e)},$$

and $g_{\text{tr}}^{(e)}$, $g_Q^{(e)}$ given by the standard ridge gradients evaluated on $\mathcal{D}_{\text{tr}}^{(e)}$ and $\mathcal{D}_{\text{cal}}^{(e)}$, respectively. We use $E_{\text{warm}} = 50$, $E_{\text{am}} = 206$, $E_{\text{refit}} = 100$, $\rho = 1/3$, $B = 10$, and Adam with learning rate 10^{-3} for the λ-update. AM uses $K = 5$-fold cross-validation with m-out-of-n bootstrap resampling (ratio selected from $\{0.1, 0.2, 0.3, 0.5\}$ by KS p-value), faithfully reproducing Jiang & Liu (2025).

F.2 UNCERTAINTY QUANTIFICATION EXPERIMENT DETAILS

Data generating process. The teacher is $f_\star(x) = W_2^\top \mathrm{ReLU}(W_1^\top x)$ with $p = 200$ input features, $h_{\mathrm{true}} = 32$ hidden units, and only the first $s_{\mathrm{active}} = 10$ rows of W_1 nonzero. Nonzero entries are drawn as $[W_1]_{jk} \sim \mathcal{N}(0, 2/s_{\mathrm{active}})$ and $[W_2]_k \sim \mathcal{N}(0, 2/h_{\mathrm{true}})$. Covariates are $x_i \sim \mathcal{N}(0, I_p)$ and responses are $y_i = f_\star(x_i) + \varepsilon_i$, where $\varepsilon_i = \sigma(x_i) Z_i$, $Z_i \sim \mathcal{N}(0,1)$, and

$$\sigma(x) = \sqrt{\frac{\mathrm{Var}(f_\star(X))}{\kappa}} \, \frac{1 + 11\mathbf{1}\{x_1 > c\}}{\sqrt{29.6}}, \qquad c = \Phi^{-1}(0.8).$$

Thus $\mathbb{E}[\sigma^2(X)] = \mathrm{Var}(f_\star(X))/\kappa$; 80% of inputs lie in the low-noise region and 20% in the high-noise region. The teacher is fixed at seed 42, and its signal variance is estimated once from $100{,}000$ independent covariates. Covariate and response standardization uses $\mathcal{D}_{\mathrm{fit}}$ only; all metrics use the original response scale. We use $n_{\mathrm{train}} = 200$, $n_{\mathrm{test}} = 1500$, and $\kappa \in \{1, \ldots, 8\}$ with 13 replications for $\kappa = 1, \ldots, 4$ and 12 for $\kappa = 5, \ldots, 8$, giving 100 paired runs per method. The main-text figure reports this Gaussian-noise setting; below we additionally consider non-Gaussian noise variants while keeping the same signal-to-noise calibration.

Student architecture. All methods use the same one-hidden-layer ReLU student MLP with 48 hidden units. Dropout baselines insert a dropout layer after the hidden activation with dropout probability 0.1; all other methods use the plain MLP.

TabCAM configuration. We report TabCAM-L1 and TabCAM-L2, corresponding to the L1 and L2 duality functions. We use $E_{\mathrm{warm}} = 5$, $E_{\mathrm{am}} = 35$, and $E_{\mathrm{refit}} = 12$, calibration ratio 0.2, $B_{\mathrm{imp}} = 4$, model Adam learning rate 0.015, and λ-update learning rate $3{\times}10^{-4}$. The mean loss and calibrated future targets use the input-dependent scale in Appendix B.4. At each online calibration step, the variance in region g is updated to $(\sum_{i \in g} r_i^2 + 2\bar{r}^2)/(n_g + 2)$, where $\bar{r}^2$ is the pooled mean squared residual; scales are clipped below at 0.05 on the standardized response scale. The initial penalty is 10^{-8}, with 10^{-5} during warmup.

Conditional scale calibration. For both TabCAM and split conformal, 30% of the 200 observations form $\mathcal{D}_{\mathrm{scale}}$ ($m = 60$), and the remaining 140 form $\mathcal{D}_{\mathrm{fit}}$. With $g(x) = \mathbf{1}\{x_1 > c\}$, we estimate the two-region scale from five-fold out-of-fold residuals $\tilde{r}_i$ on $\mathcal{D}_{\mathrm{fit}}$:

$$\hat{s}^2(x) = \frac{1}{|\mathcal{I}_{g(x)}|} \sum_{i \in \mathcal{I}_{g(x)}} \tilde{r}_i^2, \qquad \mathcal{I}_g = \{i \in \mathcal{D}_{\mathrm{fit}} : g(x_i) = g\}.$$

The mean model is then evaluated on the independent $\mathcal{D}_{\mathrm{scale}}$. Let $a_i = |y_i - f_{\hat{\theta}}(x_i)|/\hat{s}(x_i)$ and let $\hat{q}$ be the $\lceil (m+1)(1-\alpha) \rceil$-th smallest a_i (the 58th of 60 at $\alpha = 0.05$). The final interval is

$$\widehat{C}_{1-\alpha}(x) = \big[f_{\hat{\theta}}(x) - \hat{q}\hat{s}(x), \; f_{\hat{\theta}}(x) + \hat{q}\hat{s}(x) \big].$$

With $\hat{s}$ fitted on $\mathcal{D}_{\mathrm{fit}}$, this is split-conformal calibration of standardized residuals, retaining the learned input-dependent interval width.

Under Gaussian noise, TabCAM-L2 has mean interval widths 3.87 and 6.53 in the low- and high-noise regions, respectively.

Baselines. We compare against two standard UQ wrappers, split conformal prediction and residual bootstrap, each paired with four training choices: L2 penalty ($\mathrm{wd} = 10^{-3}$), L1 penalty (coefficient 10^{-4}), dropout ($p = 0.1$), and early stopping (validation fraction 0.15, patience 6 checks). This gives eight MLP baselines in total. Split-conformal students use the same 140 fitting observations as TabCAM, while residual bootstrap uses all 200 observations. Baseline training uses up to 80 optimizer steps, learning rate 0.015, and minibatches of size $\min(n, 256)$.

For split conformal, the regularized student is trained on $\mathcal{D}_{\mathrm{fit}}$ and calibrated on the same 60 observations in $\mathcal{D}_{\mathrm{scale}}$ used by TabCAM. The implementation takes the linearly interpolated empirical quantile of absolute residuals at level $(1-\alpha)(m+1)/m$, yielding the constant-width interval $[\hat{\mu}(x) - \hat{q}, \hat{\mu}(x) + \hat{q}]$.

For residual bootstrap, we first fit the student on all 200 observations and store the training residuals. For each of $B_{\mathrm{boot}} = 200$ bootstrap replicates, we resample these 200 observations with replacement,

train a fresh student on the resampled data, and evaluate that student on the test points. The bootstrap predictive sample at each test point is obtained by adding a randomly sampled training residual to each bootstrap prediction. The empirical $\alpha/2$ and $1-\alpha/2$ quantiles of these bootstrap predictive samples form the interval.

Metrics. We report empirical coverage, average interval width, and interval score (Gneiting & Raftery, 2007) which is defined as

$$\mathrm{IS}(\ell, u, y) = (u-\ell) + \tfrac{2}{\alpha}(\ell - y)_+ + \tfrac{2}{\alpha}(y-u)_+.$$

We also report coverage separately for $x_1 \le c$ and $x_1 > c$, averaging each regional coverage over the 100 runs. All paired significance tests use the one-sided Wilcoxon signed-rank test across the 100 paired runs, with the alternative that TabCAM-L2 is smaller on the corresponding metric.

Table 4: MLP UQ summary ($n = 200$, $p = 200$, $\kappa \in \{1, \dots, 8\}$, 100 paired runs per method). Nominal coverage is 95%. Bootstrap rows produce relatively narrow intervals but still under-cover substantially, so their widths are not directly comparable with the near-nominal methods.

Method	Coverage	Low noise	High noise	Avg. Width	Interval Score
TabCAM-L2	0.954	0.956	0.945	4.405	5.231
TabCAM-L1	0.954	0.956	0.945	**4.398**	**5.218**
Split conformal | Dropout	0.952	0.977	0.850	4.748	5.979
Split conformal | L1	0.954	0.979	0.856	4.880	6.082
Split conformal | L2	0.952	0.977	0.854	4.893	6.116
Split conformal | EarlyStop	0.950	0.975	0.855	5.087	6.332
Bootstrap | EarlyStop	0.858	0.907	0.662	3.027	6.597
Bootstrap | L2	0.726	0.782	0.503	2.093	8.982
Bootstrap | L1	0.718	0.774	0.494	2.047	9.119
Bootstrap | Dropout	0.637	0.690	0.424	1.713	10.949

Table 5: One-sided paired Wilcoxon p-values for TabCAM-L2 against each baseline, testing whether TabCAM-L2 achieves strictly smaller interval score and width. Bootstrap width comparisons are omitted because bootstrap intervals remain far below nominal coverage.

Baseline	Interval score p	Width p
Split conformal | Dropout	5.35×10^{-17}	6.85×10^{-4}
Split conformal | L1	3.35×10^{-18}	1.78×10^{-8}
Split conformal | L2	3.45×10^{-18}	1.71×10^{-7}
Split conformal | EarlyStop	1.95×10^{-18}	2.54×10^{-10}
Bootstrap | EarlyStop	2.41×10^{-18}	—
Bootstrap | L2	1.95×10^{-18}	—
Bootstrap | L1	1.95×10^{-18}	—
Bootstrap | Dropout	1.95×10^{-18}	—

Non-Gaussian noise robustness. To test robustness to misspecification of the true conditional distribution, we repeat the same experiment with the same teacher, student architecture, sample sizes, κ grid, optimization, and interval construction, and change the distribution of the standardized noise. For the Laplace variant, we set $\varepsilon_i = \sigma(x_i)\tilde{\varepsilon}_i$ with $\tilde{\varepsilon}_i \sim \mathrm{Laplace}(0, 1/\sqrt{2})$. For the Student-$t(3)$ variant, we draw $u_i \sim t_3$, set $\tilde{\varepsilon}_i = u_i/\sqrt{3}$, and again take $\varepsilon_i = \sigma(x_i)\tilde{\varepsilon}_i$. Thus, in both cases, $\mathbb{E}[\sigma^2(X)] = \mathrm{Var}(f_\star(X))/\kappa$ continues to define the same signal-to-noise ratio as in the Gaussian experiment. TabCAM retains lower interval scores and more balanced coverage across the two noise regions, while bootstrap continues to under-cover. Corresponding results are shown in Figure 7 and Figure 8.

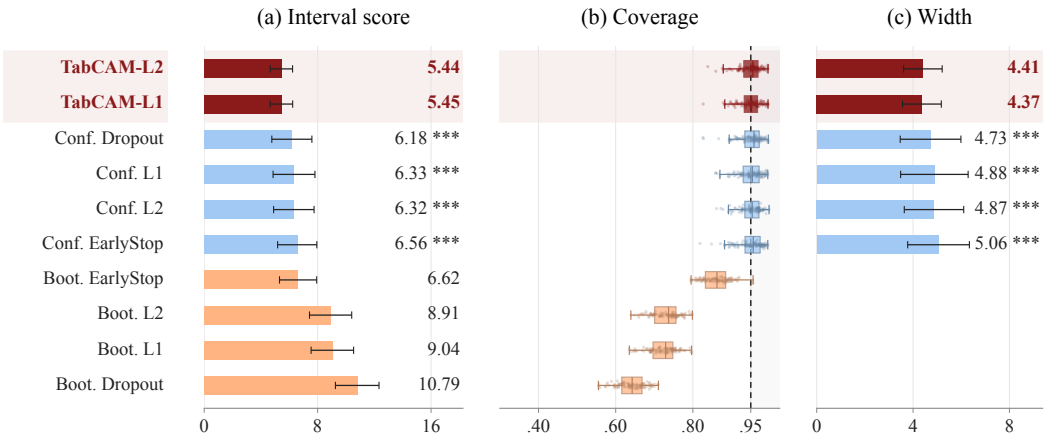


Figure 7: Nonlinear MLP UQ under Laplace noise. This experiment is identical to Figure 6 except that the response noise is unit-variance Laplace scaled by $\sigma(x)$. The plotting convention is the same as in the Gaussian case: bootstrap is shown in coverage and interval score, but omitted from the width panel. Stars report one-sided paired Wilcoxon p-values comparing TabCAM-L2 with each split-conformal baseline ($^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$).

Table 6: MLP UQ summary under Laplace noise ($n = 200$, $p = 200$, $\kappa \in \{1, \ldots, 8\}$, 100 paired runs per method).

Method	Coverage	Low noise	High noise	Avg. Width	Interval Score
TabCAM-L2	0.949	0.952	0.940	4.409	**5.444**
TabCAM-L1	0.948	0.950	0.940	**4.367**	5.447
Split conformal | Dropout	0.950	0.974	0.854	4.727	6.182
Split conformal | L1	0.949	0.972	0.857	4.885	6.333
Split conformal | L2	0.951	0.974	0.860	4.869	6.321
Split conformal | EarlyStop	0.949	0.971	0.860	5.058	6.561
Bootstrap | EarlyStop	0.861	0.903	0.694	3.022	6.622
Bootstrap | L2	0.730	0.778	0.543	2.095	8.915
Bootstrap | L1	0.723	0.770	0.534	2.051	9.038
Bootstrap | Dropout	0.641	0.687	0.462	1.721	10.792

Table 7: MLP UQ summary under standardized Student-$t(3)$ noise ($n = 200$, $p = 200$, $\kappa \in \{1, \ldots, 8\}$, 100 paired runs per method).

Method	Coverage	Low noise	High noise	Avg. Width	Interval Score
TabCAM-L2	0.948	0.951	0.937	**4.250**	**5.459**
TabCAM-L1	0.949	0.951	0.940	4.280	5.463
Split conformal | Dropout	0.952	0.973	0.866	4.492	5.941
Split conformal | L1	0.951	0.972	0.866	4.572	6.053
Split conformal | L2	0.949	0.969	0.866	4.631	6.142
Split conformal | EarlyStop	0.949	0.969	0.870	4.772	6.267
Bootstrap | EarlyStop	0.864	0.901	0.719	2.995	6.421
Bootstrap | L2	0.728	0.769	0.565	2.064	8.672
Bootstrap | L1	0.722	0.763	0.557	2.020	8.772
Bootstrap | Dropout	0.640	0.679	0.486	1.700	10.526

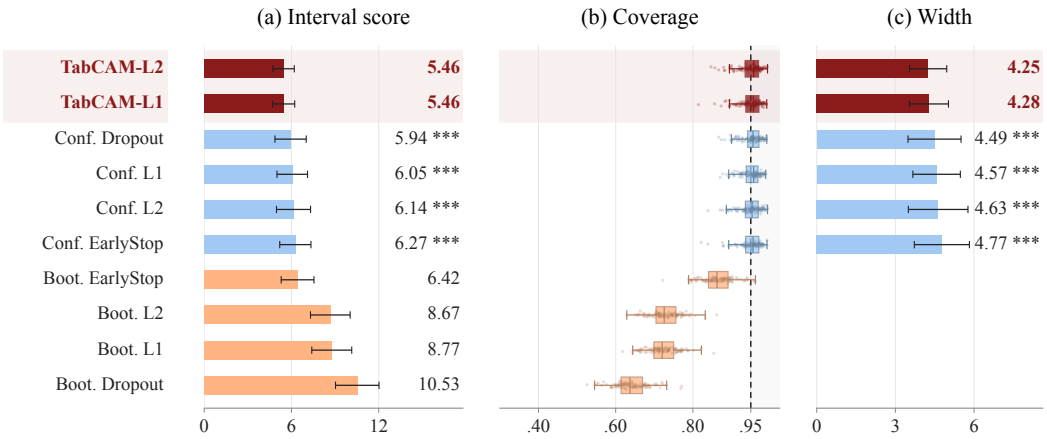


Figure 8: Nonlinear MLP UQ under standardized Student-$t(3)$ noise. This experiment is identical to Figure 6 except that the response noise is standardized t_3 scaled by $\sigma(x)$. The plotting convention is the same as in the Gaussian case: bootstrap is shown in coverage and interval score, but omitted from the width panel. Stars report one-sided paired Wilcoxon p-values comparing TabCAM-L2 with each split-conformal baseline ($^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$).

Table 8: One-sided paired Wilcoxon p-values for TabCAM-L2 under non-Gaussian noise, testing whether TabCAM-L2 achieves strictly smaller interval score and width. Bootstrap width comparisons are omitted because bootstrap intervals continue to under-cover substantially.

	Laplace noise		**Student-$t(3)$ noise**	
Baseline	**Score p**	**Width p**	**Score p**	**Width p**
Split conformal | Dropout	8.45×10^{-18}	9.30×10^{-4}	7.30×10^{-16}	5.45×10^{-4}
Split conformal | L1	3.35×10^{-18}	3.33×10^{-5}	3.08×10^{-17}	8.21×10^{-7}
Split conformal | L2	2.44×10^{-17}	2.83×10^{-7}	4.01×10^{-18}	9.17×10^{-6}
Split conformal | EarlyStop	3.06×10^{-18}	5.55×10^{-10}	4.52×10^{-18}	1.17×10^{-9}
Bootstrap | EarlyStop	1.10×10^{-17}	—	8.02×10^{-17}	—
Bootstrap | L2	1.95×10^{-18}	—	2.01×10^{-18}	—
Bootstrap | L1	1.95×10^{-18}	—	2.01×10^{-18}	—
Bootstrap | Dropout	1.95×10^{-18}	—	1.95×10^{-18}	—

G ADDITIONAL EXPERIMENT RESULTS

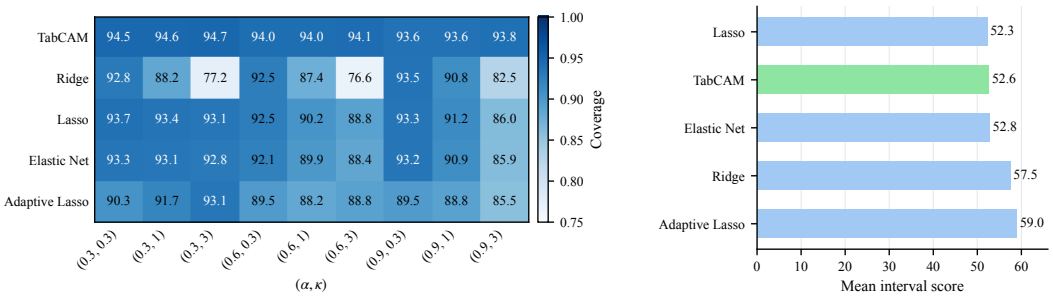
G.1 UNCERTAINTY QUANTIFICATION IN A LINEAR HIGH-DIMENSIONAL SETTING

The data generating process follows Appendix F.1. We sweep over nine combinations of sparsity $\alpha \in \{0.3, 0.6, 0.9\}$ and signal-to-noise ratio $\kappa \in \{0.3, 1, 3\}$, with 50 replications each. We use the vector L2 duality $\pi(\beta, \boldsymbol{\lambda}) = \sum_{j=1}^{p} \lambda_j \beta_j^2$ for our proposed TabCAM. We compare against Ridge, Lasso, Elastic Net, and Adaptive Lasso fit via `glmnet` with cross-validated tuning. The four baselines share a noise-scale estimate within each run; TabCAM estimates its scale from calibration residuals.

TabCAM attains mean coverage 0.941, with per-setting means between 0.936 and 0.947 (Figure 9a and Table 9). The corresponding means are 0.868 for Ridge and 0.914 for Lasso. Figure 9b reports interval scores from the same 450 runs per method.

Table 9: Prediction-interval coverage on the linear high-dimensional setting (n=200, p=500). For every method we run K=50 replications on each of the 9 (α, κ) cells (N=450 runs in total). Empirical coverage is measured on a fresh test set of 1,000 points; the nominal level is 95%. "Worst cell" / "best cell" report the minimum / maximum of the per-cell mean coverage (averaged over K replications of that cell). "MAD to 95%" is the mean of $|\text{coverage} - 0.95|$ across the N runs, and "in band" is the fraction of runs whose coverage falls inside $[0.93,\ 0.97]$. Lower MAD and higher "in band" indicate better calibration. Best value in each column is in **bold**.

Method	Coverage (mean ± sd)	Worst cell	Best cell	MAD to 95%	In band
TabCAM	$\mathbf{0.941 \pm 0.028}$	**0.936**	0.947	**0.021**	**269/450**
Lasso	0.914 ± 0.049	0.860	0.937	0.039	193/450
Elastic Net	0.911 ± 0.049	0.859	0.933	0.041	178/450
Adaptive Lasso	0.895 ± 0.048	0.855	0.931	0.056	81/450
Ridge	0.868 ± 0.083	0.766	0.935	0.082	110/450



(a) Coverage (b) Interval score

Figure 9: Linear high-dimensional UQ (n=200, p=500, 50 replications per cell). **(a)** Coverage across nine (α, κ) settings. TabCAM has per-setting mean coverage between 0.936 and 0.947; the baselines have lower mean coverage. **(b)** Interval score averaged across all settings; lower is better.

G.2 ABLATION STUDY

Table 10 isolates the contribution of each design choice in TabCAM. We run all folds of 9 randomly selected TabArena datasets (4 binary, 2 multiclass, 3 regression) under four ablation variants, keeping the total epoch budget identical to the default configuration. Per-fold normalised error is $\text{error}_{\text{variant}}/\text{error}_{\text{Full}}$, so TabCAM $\equiv 1.000$ by construction; p-values are from two-sided paired Wilcoxon signed-rank tests on the 102 paired per-fold errors.

Per-parameter λ is the largest contributor. Switching to a single global scalar increases normalised error by 2.16% ($p < 10^{-4}$). The online calibration phase contributes a comparable average gain ($+2.9\%$). The refit phase adds a negligible $+0.05\%$ on average and is not statistically significant ($p = 0.107$), suggesting that the online calibration phase already captures most of the regularisation signal. The default calibration ratio $\rho = 1/3$ is significantly better than $\rho = 1/2$ ($p < 10^{-5}$).

Table 10: Ablation of core design choices. "Scalar λ" replaces the per-parameter L2 duality function with a single global L2 coefficient. "No refit" removes the final refit phase. "No online calibration" disables the online calibration and λ-update phase entirely, reducing TabCAM to its backbone trained for the same total epochs. "$\rho = 0.5$" increases the calibration ratio from the default $1/3$ to $1/2$.

Variant	Avg rank ↓	Avg norm err ↓	Δ vs Full	p-value
TabCAM	**1.94**	**1.0000**	—	—
− Scalar λ (global L2)	3.67	1.0216	$+0.0216$	9.79×10^{-5}
− No refit	2.17	1.0005	$+0.0005$	0.107
− No online calibration	3.56	1.0290	$+0.0290$	0.052
− $\rho = 0.5$	3.67	1.0041	$+0.0041$	5.6×10^{-6}

Table 11: Prediction-interval evaluation on four TabArena regression datasets ($\alpha = 0.05$, target coverage 0.95). Width and interval-score units are dataset-specific and comparable only within a dataset. **Bold** marks the better value per metric within each dataset (coverage closer to 0.95 is better).

Dataset	Method	Coverage	Width ↓	Interval Score ↓
`healthcare` (n=1338)	Split conformal	0.974 ± 0.007	$(2.92 \pm 0.24) \times 10^4$	$(3.26 \pm 0.15) \times 10^4$
	TabCAM	$\mathbf{0.939 \pm 0.003}$	$\mathbf{(2.01 \pm 0.11) \times 10^4}$	$\mathbf{(3.15 \pm 0.10) \times 10^4}$
`houses` (n=20640)	Split conformal	0.955 ± 0.003	0.920 ± 0.026	$\mathbf{1.36 \pm 0.01}$
	TabCAM	$\mathbf{0.948 \pm 0.005}$	$\mathbf{0.907 \pm 0.029}$	1.39 ± 0.02
`miami_housing` (n=13776)	Split conformal	$\mathbf{0.945 \pm 0.007}$	$\mathbf{(3.29 \pm 0.25) \times 10^5}$	$\mathbf{(6.63 \pm 0.15) \times 10^5}$
	TabCAM	0.959 ± 0.005	$(3.96 \pm 0.25) \times 10^5$	$(6.76 \pm 0.29) \times 10^5$
`wine_quality` (n=6497)	Split conformal	$\mathbf{0.953 \pm 0.009}$	2.83 ± 0.11	3.48 ± 0.15
	TabCAM	0.944 ± 0.004	$\mathbf{2.64 \pm 0.06}$	$\mathbf{3.47 \pm 0.08}$

G.3 Real-data prediction intervals on TabArena regression

Table 11 evaluates uncertainty quantification performance on four randomly selected TabArena datasets. For each dataset we use the canonical fold-0 train/test split, further splitting the training portion $80/20$ into TrainFit and Cal sets. The baseline uses the same backbone without online calibration and constructs split-conformal intervals ($\hat{f}(x) \pm \hat{q}_\alpha$, where $\hat{q}_\alpha$ is the $(1-\alpha)$-quantile of calibration residuals). TabCAM constructs Gaussian intervals ($\hat{f}(x) \pm z_{1-\alpha/2}\,\hat{\sigma}$). Both methods train on TrainFit only and use the same Cal split. Both methods achieve near-nominal coverage on all datasets. TabCAM produces meaningfully tighter intervals on `healthcare` ($\sim$31% narrower) and `wine_quality` ($\sim$7% narrower), is comparable on `houses`, and slightly wider on `miami_housing`. The interval score favors TabCAM on two of the four datasets.

G.4 Additional results on TabArena

This subsection provides the full TabArena results underlying the summary plots and discussion in Section 3.1. We report three complementary views: the non-foundation default leaderboard (Table 12), the per-task rank distribution under default configurations (Table 13), and the tuned-and-ensembled TabArena-Lite leaderboard (Table 14). We also provide an extended runtime comparison in Figure 11.

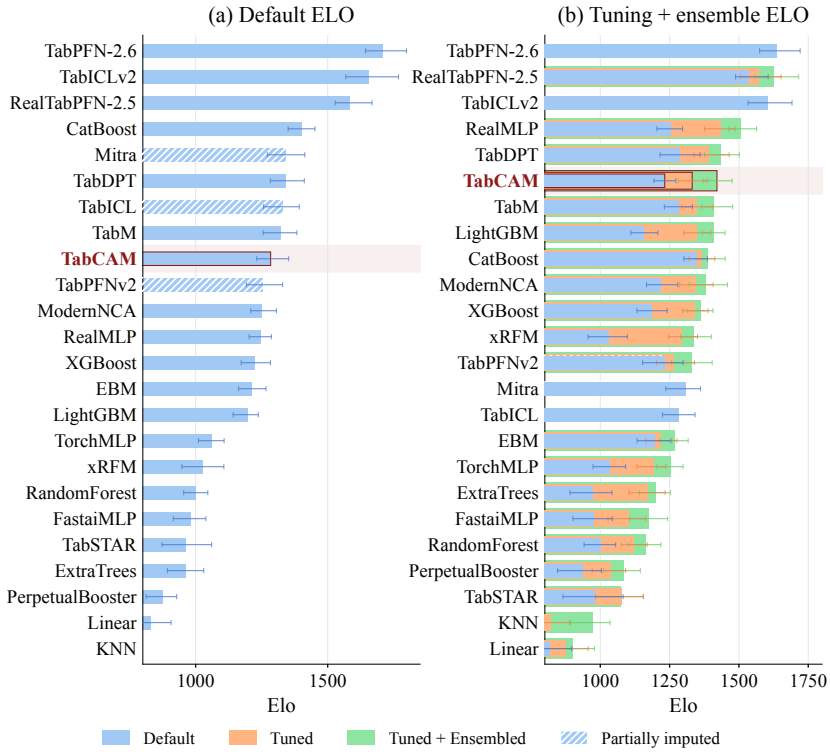


Figure 10: Default and 100-config tuned and ensembled Elo on the full TabArena pool.

Table 12: Full TabArena leaderboard for 16 non-foundation default methods. Columns: Elo with 95 % confidence interval, win-rate, improvability (regret to oracle), per-task rank counts (out of 51 tasks). Rows sorted by Elo.

Method	Elo	95 % CI	Win	Improv. (lower)	Top-3 count
CatBoost	1470.3	[1406, 1543]	0.86	0.039	36
TabM	1363.6	[1285, 1447]	0.77	0.059	27
TabCAM	1314.7	[1251, 1389]	0.72	0.070	16
ModernNCA	1284.7	[1239, 1354]	0.69	0.073	14
RealMLP	1281.0	[1232, 1332]	0.68	0.066	12
XGBoost	1247.4	[1188, 1305]	0.65	0.076	10
EBM	1234.4	[1174, 1302]	0.63	0.094	11
LightGBM	1219.2	[1171, 1261]	0.61	0.086	6
TorchMLP	1058.6	[1007, 1106]	0.43	0.129	2
xRFM	1035.0	[953, 1117]	0.41	0.155	8
RandomForest	1000.0	[952, 1053]	0.37	0.172	2
FastaiMLP	981.4	[917, 1045]	0.35	0.167	2
ExtraTrees	970.1	[891, 1042]	0.34	0.193	5
PerpetualBooster	878.2	[823, 935]	0.25	0.239	0
Linear	827.5	[727, 903]	0.20	0.299	2
KNN	590.0	[470, 681]	0.06	0.408	0

Table 13: Per-task rank distribution for the 16 non-foundation default methods across the 51 TabArena tasks. Each row gives the number of tasks on which the method was the best, second best, third best, or ranked outside the top three. The final column reports the fraction of tasks on which the method landed in the top three.

Method	Rank 1	Rank 2	Rank 3	Rank >3	Top-3	Top-3 (%)
CatBoost	13	13	10	15	36	70.6
TabM	14	9	4	24	27	52.9
TabCAM	4	6	6	35	16	31.4
ModernNCA	9	2	3	37	14	27.5
RealMLP	1	6	5	39	12	23.5
EBM	2	3	6	40	11	21.6
XGBoost	3	5	2	41	10	19.6
xRFM	4	2	2	43	8	15.7
LightGBM	0	1	5	45	6	11.8
ExtraTrees	0	3	2	46	5	9.8
TorchMLP	0	0	2	49	2	3.9
RandomForest	0	1	1	49	2	3.9
FastaiMLP	0	0	2	49	2	3.9
Linear	1	0	1	49	2	3.9
PerpetualBooster	0	0	0	51	0	0.0
KNN	0	0	0	51	0	0.0

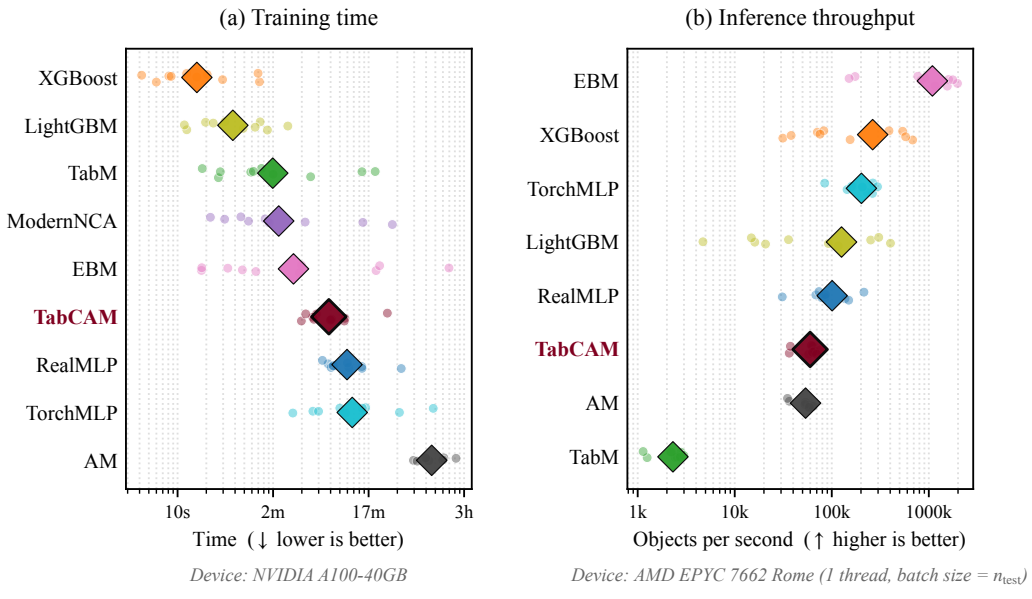


Figure 11: Training time (a) and CPU full-batch inference throughput (b) over ten TabArena tasks. One dot is one dataset; the diamond is the aggregate mean.

Table 14: Full TabArena leaderboard for tuning + ensembling on TabArena-Lite. TabCAM uses 100 random configurations. Methods are reported in three variants where available: default single configuration, tuned single model (selected from its search budget), and tuned + ensembled (weighted average of the top configurations). Rows sorted by Elo.

Method	Variant	Elo	95 % CI	Win	Improv. (lower)	Top-3 count
RealMLP	tuned + ens	1577.0	[1519, 1650]	0.83	0.084	6
RealMLP	tuned	1482.8	[1420, 1543]	0.75	0.098	5
TabCAM	tuned + ens	1466.3	[1417, 1532]	0.74	0.100	4
LightGBM	tuned + ens	1463.1	[1419, 1513]	0.73	0.114	1
TabM	tuned + ens	1456.9	[1405, 1538]	0.73	0.106	5
CatBoost	tuned + ens	1429.1	[1380, 1492]	0.70	0.108	1
ModernNCA	tuned + ens	1419.4	[1350, 1517]	0.69	0.114	10
CatBoost	tuned	1402.3	[1356, 1457]	0.68	0.113	3
XGBoost	tuned + ens	1401.3	[1350, 1449]	0.68	0.120	2
LightGBM	tuned	1388.7	[1336, 1446]	0.66	0.123	1
TabM	tuned	1385.0	[1322, 1458]	0.66	0.115	3
CatBoost	default	1379.5	[1334, 1420]	0.65	0.121	2
XGBoost	tuned	1374.9	[1330, 1422]	0.65	0.122	1
xRFM	tuned + ens	1374.5	[1323, 1442]	0.65	0.118	3
ModernNCA	tuned	1374.5	[1310, 1442]	0.65	0.120	6
TabCAM	tuned	1365.4	[1307, 1418]	0.64	0.116	3
xRFM	tuned	1323.9	[1271, 1384]	0.59	0.130	4
TabM	default	1309.9	[1255, 1369]	0.58	0.138	1
EBM	tuned + ens	1291.0	[1234, 1350]	0.56	0.149	3
TorchMLP	tuned + ens	1275.5	[1222, 1328]	0.54	0.136	1
RealMLP	default	1273.7	[1222, 1322]	0.54	0.140	0
TabCAM	default	1252.2	[1210, 1294]	0.52	0.144	2
ModernNCA	default	1234.2	[1177, 1303]	0.50	0.166	2
EBM	tuned	1233.1	[1174, 1304]	0.50	0.159	2
ExtraTrees	tuned + ens	1211.9	[1145, 1275]	0.47	0.164	0.5
EBM	default	1211.9	[1145, 1274]	0.47	0.167	1
TorchMLP	tuned	1209.0	[1144, 1257]	0.47	0.154	0
XGBoost	default	1196.2	[1141, 1253]	0.46	0.155	1
FastaiMLP	tuned + ens	1186.2	[1116, 1258]	0.45	0.170	0
ExtraTrees	tuned	1178.5	[1109, 1252]	0.44	0.173	1.5
LightGBM	default	1166.7	[1119, 1223]	0.42	0.164	0
RandomForest	tuned + ens	1166.5	[1093, 1226]	0.42	0.173	1
RandomForest	tuned	1123.2	[1067, 1179]	0.38	0.181	1
FastaiMLP	tuned	1110.8	[1027, 1173]	0.37	0.186	1
PerpetualBooster	tuned + ens	1095.0	[1023, 1157]	0.35	0.237	1
PerpetualBooster	tuned	1043.8	[972, 1103]	0.30	0.251	0
xRFM	default	1031.8	[958, 1106]	0.29	0.230	1
TorchMLP	default	1029.7	[964, 1085]	0.29	0.204	0
RandomForest	default	1000.0	[934, 1070]	0.26	0.234	0
ExtraTrees	default	979.6	[889, 1055]	0.25	0.250	0
FastaiMLP	default	975.8	[900, 1054]	0.24	0.227	0
KNN	tuned + ens	972.0	[899, 1035]	0.24	0.252	0
PerpetualBooster	default	935.9	[842, 1013]	0.21	0.304	1
Linear	tuned + ens	895.4	[792, 976]	0.18	0.318	1
Linear	tuned	862.5	[762, 943]	0.16	0.325	1
KNN	tuned	813.2	[732, 879]	0.13	0.313	0
Linear	default	806.7	[683, 887]	0.13	0.351	0
KNN	default	606.7	[482, 688]	0.05	0.440	0

H FLEXIBLE FUTURE-TARGET CONSTRUCTORS

The main benchmark implementation uses parametric working distributional models. For regression, we use a Gaussian working CDF. For classification, we use temperature scaling as the default calibration map. These choices are not essential to the TabCAM framework. The future-target construction step only requires an estimated conditional law that can generate calibrated future targets on the calibration split.

H.1 PRACTICAL WORKING-FAMILY SELECTION

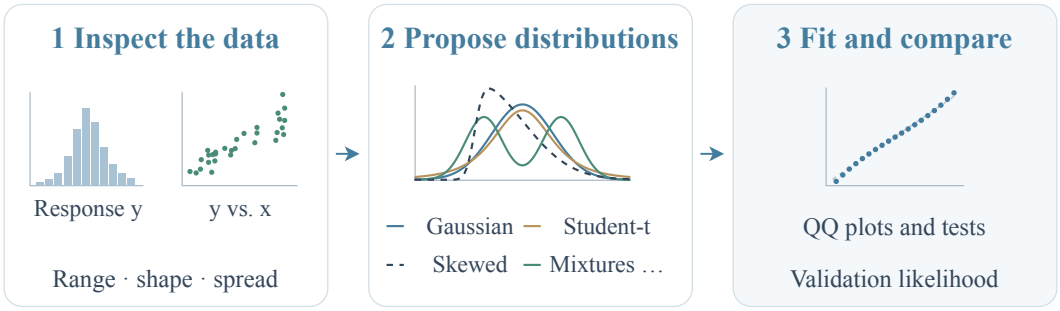


Figure 12: A general, data-driven procedure for choosing the working distribution. Inspect the observed response and its relationship to covariates to suggest candidate families. Fit candidate conditional distributions, check their fit using QQ plots and appropriate statistical tests, and compare validation negative log-likelihood (NLL). The schematic Gaussian, heavy-tailed Student-t, skewed, and mixture densities illustrate an extensible candidate set.

We adapt the location, scale, and shape approach to distributional regression (Rigby & Stasinopoulos, 2005). For independent observations with a continuous response, begin with the observed pairs (x_i, y_i). Reserve final-calibration and test observations, and divide the remaining data into $\mathcal{D}_{\text{fit}}$ for exploration and candidate fitting and $\mathcal{D}_{\text{val}}$ for predictive comparison.

Inspect the response support and histogram, then plot the response against numerical covariates or within categorical groups. The histogram suggests tail weight, asymmetry, and modal structure; the covariate plots show how location and spread vary with x. These descriptive features motivate candidates for the conditional law $Y \mid X$, whose fit is assessed in the next step. Gaussian, Student-t, Laplace, and finite mixture models provide candidates for light tails, heavy tails, peaked densities, and multiple modes; skewed families accommodate asymmetry. Mixtures can also use heavy-tailed components, such as Student-t (Peel & McLachlan, 2000), and their component count forms part of the candidate specification.

Fit candidate conditional densities on $\mathcal{D}_{\text{fit}}$. For example, a location-scale candidate takes the form $p_m(y \mid x) = s_m(x)^{-1} p_{0,m}((y - \mu_m(x))/s_m(x))$, where simple linear or additive terms can model location $\mu_m(x)$ and positive scale $s_m(x)$. A shared location estimate with residual-family fits provides another implementation, used in the toy example below. Compare the fitted candidates by the held-out negative logarithmic score (Gneiting & Raftery, 2007):

$$\widehat{m} = \arg\min_{m \in \mathcal{M}} \left\{ -\frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{i \in \mathcal{D}_{\text{val}}} \log \widehat{p}_m(y_i \mid x_i) \right\}. \tag{19}$$

The illustrated families are examples of $\mathcal{M}$; additional families and mixture specifications can enter the same predictive comparison. The factor $s_m(x)^{-1}$ keeps scores comparable across scale specifications. For candidate checking on $\mathcal{D}_{\text{val}}$, PIT scores $\widehat{F}_m(y_i \mid x_i)$ can be compared with uniform quantiles; equivalently, normal-score residuals $z_{im} = \Phi^{-1}(\widehat{F}_m(y_i \mid x_i))$ provide a common normal reference for continuous working families (Dunn & Smyth, 1996). This reference applies on the transformed scale for Gaussian, heavy-tailed, skewed, and mixture CDFs alike. QQ plots and plots against covariates assess their shape and remaining structure before empirical PIT correction.

Statistical tests can supplement these graphical checks. In the shared-location implementation, fit $\widehat{\mu}(x)$ on a subset of $\mathcal{D}_{\text{fit}}$ and compute residuals $r_i = y_i - \widehat{\mu}(x_i)$ on its remaining n_r observations.

The Shapiro–Wilk test (Shapiro & Wilk, 1965) checks the shared-scale Gaussian residual null $r_i \overset{\text{iid}}{\sim} \mathcal{N}(\mu_r, \sigma_r^2)$. To check scale variation, form four approximately equal-sized groups by $\widehat{\mu}(x_i)$ and apply the median-centered Brown–Forsythe test (Brown & Forsythe, 1974). It compares the group means of $|r_i - \widetilde{r}_{g(i)}|$, where $\widetilde{r}_g$ is the group median, using an approximate F_{3,n_r-4} reference. Under a common-shape location-scale model, its null corresponds to equal group scales. The toy example uses 0.05 as a diagnostic significance level; validation scores determine the final choice.

Supply the selected CDF to Equation (11) and run the θ- and λ-updates. The family and scale specification remain fixed while their parameters and the calibration map track the predictor. After refitting with fixed λ, estimate the final law and PIT correction on the reserved calibration observations, then evaluate predictions and intervals on the test set. Appendix I.3 illustrates this procedure. Classification uses calibrated class probabilities (Appendix C); randomized PIT extends the CDF construction to discrete responses.

H.2 Shared location-scale families

For a strictly increasing residual CDF F_0, write $F(y \mid x) = F_0((y - \mu(x))/s(x))$ and $r_j = (y_j - \mu(x_j))/s(x_j)$. With empirical PIT calibration, choosing an order statistic $r_{(j)}$ gives

$$F^{-1}\big(F_0(r_{(j)}) \mid x_i\big) = \mu(x_i) + s(x_i) r_{(j)}.$$

For fixed μ and s, empirical PIT calibration therefore produces the same targets across strictly increasing shared residual CDFs. We evaluate this identity by resampling standardized residuals. Family choice affects uncorrected parametric intervals and conditional models whose shape varies with x; $s(x)$ represents covariate-dependent spread.

H.3 More flexible conditional estimators

For regression, the Gaussian working CDF in Section 2.2 can be replaced by a more flexible conditional CDF estimator

$$\widehat{F}_e(y \mid x).$$

Given calibration observations, we compute PIT scores

$$\widehat{u}_i^{(e)} = \widehat{F}_e(y_i \mid x_i), \qquad i \in \mathcal{D}_{\text{cal}}^{(e)},$$

form their empirical CDF $\widehat{H}_e$, and generate future targets by

$$q_i^{(e)} = \widehat{F}_e^{-1}\Big(\widehat{H}_e^{-1}(u_i^\star) \mid x_i\Big), \qquad u_i^\star \sim \text{Unif}(0,1).$$

This is the same construction as in (11), except that the Gaussian CDF is replaced by a learned conditional CDF. For example, neural conditional CDF estimators can directly learn $\widehat{F}(y \mid x)$ (Hu & Nan, 2023).

For classification, the default temperature-scaling map can be replaced by a richer calibration map

$$\mathcal{C}_e : \Delta^{K-1} \to \Delta^{K-1}.$$

Given raw predicted probabilities $\widehat{p}_i^{(e)} = p_{\hat{\theta}^{(e)}}(\cdot \mid x_i)$, the calibrated future-label distribution becomes

$$q_{\text{class},i}^{(e)} = \mathcal{C}_e(\widehat{p}_i^{(e)}).$$

The λ-update then proceeds exactly as before using the expected cross-entropy loss. For example, Dirichlet calibration provides a more flexible multiclass probability calibration map than temperature scaling (Kull et al., 2019).

Both extensions retain the same future-target interface and coordinatewise λ-update, allowing the conditional estimator to be adapted to the data.

I Supplementary experiments and diagnostics

We complement the main experiments with backbone integration (Appendix I.1), learned regularization (Appendix I.2), and working-family selection (Appendix I.3).

I.1 MATCHED BACKBONE INTEGRATION

To evaluate TabCAM as an add-on to different predictive architectures, we pair it with three backbones: a two-hidden-layer ReLU MLP with width 64, TabM (Gorishniy et al., 2025) with two blocks of width 64 and 8 ensemble members, and FT-Transformer (Gorishniy et al., 2021) with one block, token dimension 32, and four attention heads. TabM and FT-Transformer use their public reference implementations.

We evaluate the three backbones on the five smallest regression tasks and five smallest binary-classification tasks in TabArena-v0.1 by sample size. For each task, we use canonical outer folds $0, 1, 2$ of repeat 0 and split each outer-training set $80/20$ into fitting and validation sets. Numerical features are median-imputed and standardized using the fitting set, while categorical features are one-hot encoded. The same processed data are supplied to each backbone.

For each backbone, the versions with and without TabCAM use the same architecture, initialization, preprocessing, training examples, minibatch order, optimizer, and number of model-parameter updates. Each run uses 256 epochs, consisting of 32 warmup, 192 online-update, and 32 refit epochs. The base L2 coefficient is selected on the validation split, and TabCAM starts from the same coefficient and learns per-parameter regularization strengths. The calibration ratio is $1/3$ and the λ-update frequency is 2.

Table 15 reports the paired test-error ratio of the model with TabCAM to its corresponding base backbone, using RMSE for regression and $1 - \mathrm{AUC}$ for binary classification. Ratios are first averaged across folds within each dataset and then aggregated equally across datasets. The aggregate ratios remain close to one for all three backbones, while TabCAM achieves lower error on subsets of tasks. These results show that the same TabCAM framework can be integrated with substantially different differentiable backbones without requiring a backbone-specific formulation.

Table 15: Matched backbone integration on ten TabArena tasks with three outer folds per task. Error ratios compare each backbone with TabCAM against its corresponding base model; values below one favor TabCAM.

Backbone	Error ratio	95% interval	Wins / 10
MLP	0.9892	[0.9762, 0.9925]	8
TabM	0.9902	[0.9861, 0.9933]	6
FT-Transformer	0.9940	[0.9874, 1.0001]	6

I.2 LEARNED REGULARIZATION STRENGTHS

We further examine the structure of the learned per-parameter regularization strengths in the sparse-teacher simulation used in Figure 4. We generate $p = 100$ independent standard-normal covariates, of which only the first 10 are relevant. A one-hidden-layer ReLU teacher with 32 hidden units generates the response, with Gaussian noise variance $\mathrm{Var}(f_\star(X))/3$. For each of 20 evaluation seeds (200–219), we generate $n = 3200$ training observations and 600 validation observations. The student is a one-hidden-layer ReLU MLP with 48 hidden units, trained for 512 epochs (32 warmup, 416 online-update, and 64 refit), with batch size 128, Adam learning rate 10^{-3} with cosine decay, initial penalty 10^{-5}, update frequency 2, and calibration ratio $1/3$. Five separate development seeds (100–104) are used to select the λ learning rate from $\{3 \times 10^{-4}, 3 \times 10^{-3}, 3 \times 10^{-2}, 10^{-1}\}$, yielding 3×10^{-2}.

Figure 4 shows the evolution and final distribution of the learned penalties. The learned λ values remain heterogeneous across both coordinates and parameter groups. More importantly, the first-layer weights connected to relevant covariates receive systematically weaker regularization. Across the 20 seeds, the median first-layer penalty averages 0.0201 for relevant features and 0.0363 for irrelevant features. Thus, in this simulation, the learned penalties reflect structure in the underlying signal rather than behaving as a uniform shrinkage mechanism.

I.3 EXAMPLE OF WORKING-FAMILY SELECTION

We illustrate the data-driven working-family selection procedure in Appendix H.1 with a simple nonlinear regression example. We generate 1000 observations from

$$Y = 2\sin(X_1) + X_2 + 0.5X_3X_4 + 1.5\varepsilon,$$

where the covariates are independent standard normal and ε follows a unit-variance Student-$t(3)$ distribution. The response is standardized before fitting.

We first fit a pilot MLP to estimate the conditional mean and use held-out residuals to compare Gaussian, Student-t, Laplace, and two-component Gaussian-mixture working families. The residual diagnostics in Figure 13 indicate substantial heavy tails (Shapiro–Wilk $p = 7.6 \times 10^{-11}$) and no significant scale variation at the 0.05 level (Brown–Forsythe $p = 0.196$). The Student-t family achieves the lowest held-out validation NLL (0.9945), compared with 1.0538 for the Gaussian family. The selection procedure therefore chooses Student-t as the working family.

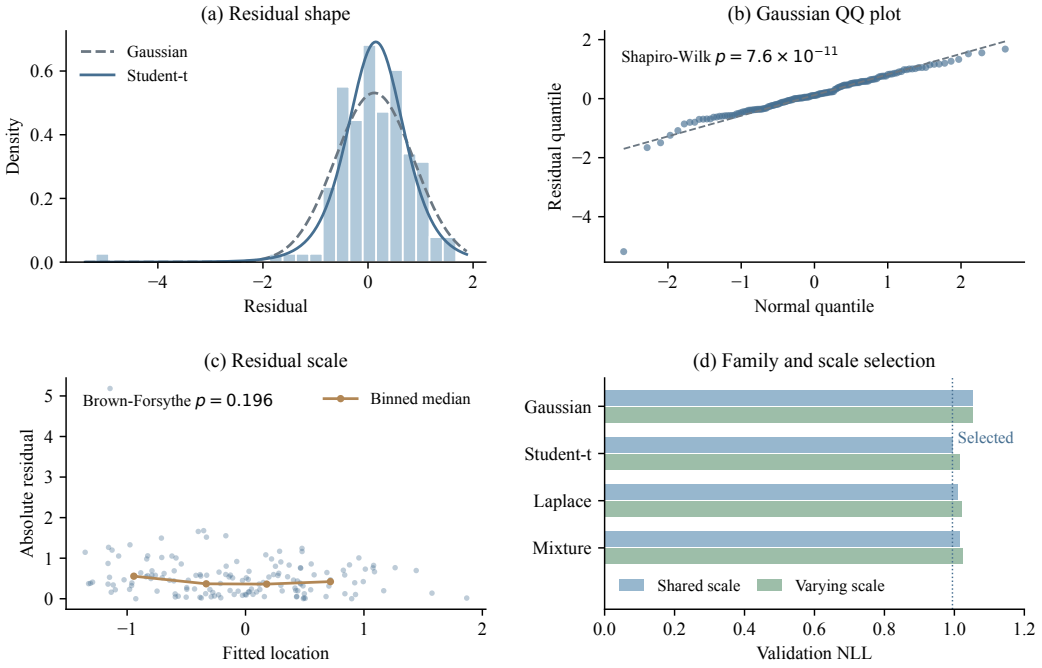


Figure 13: Illustration of the working-family selection procedure. (a,b) Residual diagnostics indicate heavy-tailed behavior. (c) Residual magnitude is examined for scale variation. (d) Held-out validation NLL is used to select among candidate working families and scale specifications.