DRTAB: DIAGNOSE, REVISE, AND RE-READ THE TABLE

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) learn transferable prediction algorithms through large-scale pretraining and apply them to unseen tables through in-context learning (ICL). Recent studies further narrow the gap between pretrained models and specific downstream tasks through task-time fine-tuning. However, how much of this gap is correctable through adaptation and how to diagnose this potential before parameter updates to guide adaptation effort, remain insufficiently characterized. To this end, under controlled gradient estimation error and local smoothness, we derive a lower bound on local improvement in the full-context objective from pre-update task signals, providing a theoretical basis for diagnosing adaptation potential. To put this theoretical insight into practice, we propose DrTab (Diagnose, Revise, and Re-read the Table). Grounded in our theoretical analysis, Diagnose uses information from the current training table to estimate the model's remaining potential for improvement on the task. Revise uses the diagnosis to determine whether to continue adaptation and to guide task-specific parameter updates, after which Re-read combines the revised model with the training table for in-context prediction. Experiments show that the pre-update gradient energy characterized by our theoretical analysis is strongly positively correlated with subsequent loss reduction, supporting its use as a diagnostic signal for adaptation potential. Furthermore, as an add-on adaptation framework for TFMs, DrTab demonstrates broader applicability across TFM backbones and achieves better tabular prediction performance than prior adaptation methods. Its performance gains on non-tabular out-of-distribution (OOD) tasks further demonstrate its effectiveness in enhancing TFM adaptation to new task types.

1 INTRODUCTION

Tabular foundation models (TFMs) learn transferable prediction algorithms through large-scale cross-task pretraining and apply them to unseen tables by using labeled training examples as context to predict new query targets via in-context learning (ICL) (Müller et al., 2021; Hollmann et al., 2022; 2025; Qu et al., 2025).

Recent work further narrows the gap between pretrained models and specific downstream tasks through task-time fine-tuning, using supervision from the current training table to update model parameters (Thomas et al., 2024; Rubachev et al., 2025). The benefits of fine-tuning vary across tasks and data conditions (Rubachev et al., 2025), making it important to determine how much adaptation a particular task requires. Yet this task dependence creates a practical decision problem: before any parameter update is made, we still need to determine from the current training table whether the model retains exploitable local correction opportunities and how much adaptation those opportunities can support. This raises a central question: *Can an observable quantity diagnose a task's remaining correction potential before parameter updates and guide adaptation accordingly?*

A central challenge is to relate gradients estimated from reduced contexts to local improvement on unseen queries conditioned on the full table. Under controlled gradient estimation error and local smoothness, we establish a positive lower bound on the objective reduction of a feasible update when the observed gradient norm exceeds its error bound. This provides a conditional basis for assessing local correction potential before fine-tuning. Experiments on binary classification, multiclass classi-

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.

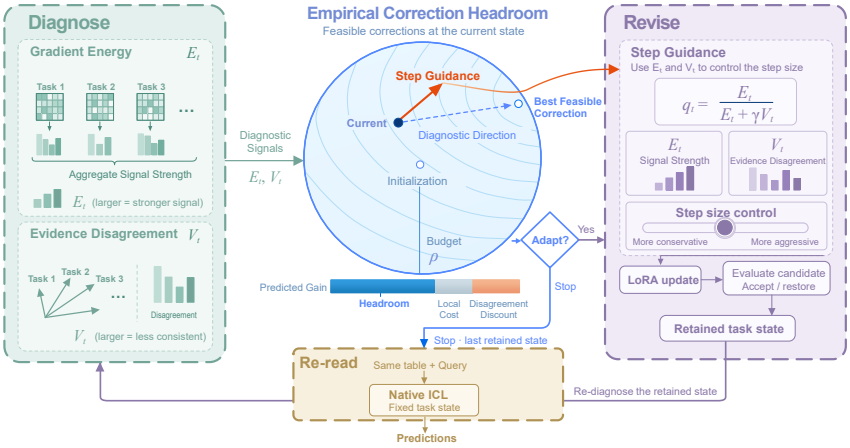


Figure 1: Overview of DrTab. Diagnose assesses empirical correction headroom; Revise guides and evaluates lightweight task-specific updates; Re-read combines the retained state with the training table for prediction. Diagnosis is repeated at each retained state.

fication, and regression further show positive associations between pre-update gradient energy and one-step loss reduction.

Building on these theoretical and empirical findings, we propose **DrTab** (Diagnose, Revise, and Re-read the Table), a diagnosis-guided task-time adaptation framework for TFMs. **Diagnose** aggregates table-wide task evidence and balances predicted gains against local quadratic costs and evidence disagreement to estimate the remaining empirical correction headroom. **Revise** uses this diagnosis to determine whether to continue adaptation and adaptively adjust the magnitude of lightweight task-specific updates; the framework then reruns diagnosis at the retained model state so that further adaptation responds to the remaining correction opportunity. **Re-read** combines the selected task state with the training table through the backbone's native ICL interface to predict new queries.

Experiments show that DrTab, as an add-on adaptation framework, consistently improves task-time adaptation across multiple mainstream TFM backbones on binary classification, multiclass classification, and regression, while outperforming prior adaptation methods on matched backbones. Further controlled analysis shows that task-specific evidence from the current training table translates into more effective local correction than evidence averaged at the task-family level. Moreover, DrTab yields performance gains on non-tabular out-of-distribution (OOD) tasks, including handwritten digit recognition and word-level language identification. DrTab-adapted TFMs even outperform multiple vision-language model (VLM) baselines on handwritten digit recognition and achieve performance comparable to multiple large language model (LLM) baselines on word-level language identification, further demonstrating DrTab's effectiveness in enhancing TFM adaptation to new task types. These results highlight the importance of linking remaining improvement potential, task feedback, and parameter updates: assessing opportunities for improvement based on the strength and reliability of task feedback, and using this assessment to guide update magnitudes, thereby aligning adaptation effort with the current task's correction needs.

2 Preliminaries

Tabular foundation models. Let $\tau \sim \Pi_{\text{pre}}$ denote a tabular prediction task with dataset $\mathcal{D}_\tau = \{(x_i, y_i)\}_{i=1}^{N_\tau}$. A context–query split $(C, Q) \sim \mathcal{S}_\tau$ exposes the labels of samples in C while withholding those in Q. Given a labeled context C and a query input x, a tabular foundation model with parameters θ predicts $p_\theta(y \mid x, C)$. Unlike conventional models trained independently for each dataset, TFMs share a single parameter vector across tasks and are pretrained to implement a reusable prediction algorithm. At deployment, the labeled rows of an unseen task are provided as context, and query labels are predicted through in-context learning. For PFN-style models, Π_{pre} is induced by a synthetic task prior, and minimizing the held-out prediction loss can be interpreted as approximating the corresponding Bayesian posterior predictive distribution (Müller et al., 2021).

LoRA-based post-training. We adapt a pretrained TFM using Low-Rank Adaptation (LoRA) (Hu et al., 2021), keeping its pretrained parameters θ_0 frozen. For each selected weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$,

$$W_{\text{eff}} = W_0 + \frac{\alpha}{r} BA, \qquad A \in \mathbb{R}^{r \times d_{\text{in}}}, \quad B \in \mathbb{R}^{d_{\text{out}} \times r}, \tag{1}$$

where r is the adaptation rank and α is a scaling coefficient. Let $\phi \in \mathbb{R}^p$ collect all trainable LoRA factors, initialized at ϕ_0 with random A and zero B. Each task reinitializes the adapters and optimizes the context–query loss using only its labeled training table:

$$J_\tau(\phi) = \mathbb{E}_{(C,Q)\sim\mathcal{S}_\tau}\left[\frac{1}{|Q|}\sum_{(x,y)\in Q} \ell(p_{\theta_0,\phi}(\cdot \mid x, C), y)\right], \tag{2}$$

where $p_{\theta_0,\phi}$ denotes the adapted predictor. At iteration t, the cumulative displacement $s_t = \phi_t - \phi_0$ is constrained by $\|s_t\|_2 \le \rho$, with ρ specifying the adaptation budget.

For analysis, write $\mathcal{D} = \mathcal{D}_\tau$ and $w = P^{-1/2}(\phi - \phi_0)$ for a fixed metric $P \succ 0$. Let $\mathcal{J}_\mathcal{D}(w)$ denote the expected loss on fresh task queries with $\mathcal{D}$ as context, and $\widehat{\mathcal{J}}_\mathcal{D}^{\text{CF}}(w)$ the row-averaged cross-fitted loss, with each held-out fold predicted using the remaining rows as context. Their gradients at the frozen-equivalent initialization $w = 0$ are

$$h_\mathcal{D} = \nabla_w \mathcal{J}_\mathcal{D}(0), \qquad \widehat{h}_\mathcal{D} = \nabla_w \widehat{\mathcal{J}}_\mathcal{D}^{\text{CF}}(0). \tag{3}$$

Further discussion of related work is provided in Appendix B.

3 DrTab: Diagnose, Revise, and Re-read the Table

DrTab organizes task-time adaptation as a closed loop. Diagnose assesses whether the current table still supports a coherent and locally actionable correction; Revise writes the supported correction into lightweight task-specific parameters; and Re-read deploys the selected state through the backbone's native in-context prediction interface. The current table thus supplies both the signal for revising the learner and the evidence for deciding how much adaptation it needs.

3.1 Diagnose: Assessing Remaining Adaptation Potential

Task-specific correction potential can coexist with a vanishing mean task gradient (Appendix A). We therefore examine how current-table evidence can inform adaptation.

Table-wide task evidence. To this end, we first partition the training table $\mathcal{D}_\tau$ into K approximately balanced evidence slices. In diagnostic view k, slice P_k serves as a label-hidden diagnostic probe set, while the remaining rows form its labeled context pool, $C_k = \mathcal{D}_\tau \setminus P_k$. After one complete rotation, every row is evaluated exactly once as a diagnostic probe at the same parameter state ϕ_t; the weighted mean of the resulting gradients defines the task direction $\bar{g}_t$, its squared norm $E_t = \|\bar{g}_t\|_2^2$ measures the task signal strength, and the cross-slice variation in the gradients is used to compute the evidence disagreement V_t.

Correction potential before the first update. To determine how much correction the table-wide evidence can support, let $\widehat{h}_\mathcal{D} = \nabla_w \widehat{\mathcal{J}}_\mathcal{D}^{\text{CF}}(0)$ denote the observed cross-fitted gradient at the pretrained state, and let $h_\mathcal{D} = \nabla_w \mathcal{J}_\mathcal{D}(0)$ denote the gradient of the full-context objective. Under the table-sampling and context-stability conditions in Appendix A, these gradients satisfy $\|\widehat{h}_\mathcal{D} - h_\mathcal{D}\|_2 \le \epsilon_{N,K}(\delta)$ with probability at least $1-\delta$. The following theorem further shows that a lower bound on objective improvement can be established from this relationship before any model parameters are updated.

Theorem 1 (Pre-adaptation correction from table-wide evidence)**.** *On the event $\|\widehat{h}_\mathcal{D} - h_\mathcal{D}\|_2 \le \epsilon$, let $\widehat{G}_\mathcal{D} = \|\widehat{h}_\mathcal{D}\|_2 > \epsilon$. If $\mathcal{J}_\mathcal{D}$ is L-smooth along the following feasible update segment, then*

$$w_1 = -\frac{1}{L}\left(1 - \frac{\epsilon}{\widehat{G}_\mathcal{D}}\right)\widehat{h}_\mathcal{D}, \qquad \mathcal{J}_\mathcal{D}(0) - \mathcal{J}_\mathcal{D}(w_1) \ge \frac{(\widehat{G}_\mathcal{D} - \epsilon)^2}{2L}. \tag{4}$$

Taking $\epsilon = \epsilon_{N,K}(\delta)$ gives the corresponding high-probability result before the first update.

Theorem 1 shows that the observed gradient norm $\widehat{G}_{\mathcal{D}}$, estimation error $\epsilon_{N,K}$, and local smoothness constant L jointly determine a lower bound on local objective reduction. Figure 2 further examines the relationship between pre-update gradient energy $\|\widehat{g}_\tau\|_2^2$ and one-step objective reduction on real-world tasks. The upper panels characterize this relationship for the source cross-fit objective used to construct the update; the lower panels evaluate the same update under an alternate partition of the same training table, testing whether the association persists when the context–query organization changes. Alternate-partition correlations have positive point estimates, with greater uncertainty for multiclass classification. Motivated by the gain–uncertainty–cost structure in Appendix A.3, Diagnose combines task gradients with empirical local geometry and cross-slice disagreement to assess feasible correction opportunities. We formalize this assessment as empirical correction headroom.

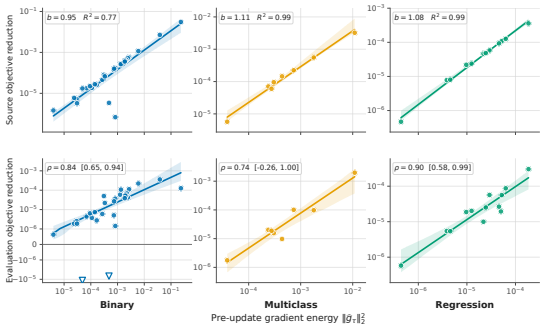


Figure 2: Pre-update gradient energy tracks one-step loss reduction. We evaluate tabular foundation models on binary classification, multiclass classification, and regression. The upper row relates $\|\widehat{g}_\tau\|_2^2$ to the loss reduction on the cross-fit folds used to construct the update, $\Delta J_\tau^{\mathrm{upd}} = J_\tau^{\mathrm{upd}}(\phi_0) - J_\tau^{\mathrm{upd}}(\phi_1)$. The lower row evaluates the same update on an alternate fold partition, $\Delta J_\tau^{\mathrm{alt}} = J_\tau^{\mathrm{alt}}(\phi_0) - J_\tau^{\mathrm{alt}}(\phi_1)$. Each point denotes one dataset; downward triangles indicate loss increases.

Empirical correction headroom. Let $s_t = \phi_t - \phi_0$ denote the cumulative task displacement. The vector g_t denotes the update gradient after incorporating the retention term, which aligns the diagnosis with the objective optimized during revision. The matrix H_t denotes a diagonal local geometry constructed from the evidence-slice gradients and captures the direction-dependent cost of parameter movement. Within a cumulative adaptation radius ρ, DrTab solves for the diagnostic displacement

$$d_t^\star = \arg\min_{\|s_t+d\|_2 \le \rho} \left(g_t^\top d + \frac{1}{2} d^\top H_t d \right). \tag{5}$$

The constraint keeps the cumulative displacement $s_t + d$ within a radius ρ of initialization. The diagnostic displacement $d_t^\star$ quantifies the maximum decrease predicted by this quadratic surrogate within the remaining displacement budget. Combining this prediction with an evidence-disagreement discount gives the empirical relative correction headroom:

$$\widehat{\mathcal{H}}_{\tau,t}^{\mathrm{emp}} = \frac{1}{J_{\tau,0}} \left[\underbrace{-g_t^\top d_t^\star}_{\text{predicted gain}} \underbrace{- \frac{1}{2}(d_t^\star)^\top H_t d_t^\star}_{\text{local cost}} \underbrace{- \beta \|d_t^\star\|_2 \sqrt{V_t}}_{\text{disagreement discount}} \right]_+ . \tag{6}$$

Here $J_{\tau,0}$ is the initial task objective with a positive numerical floor. The first term measures the first-order gain predicted along the diagnostic displacement. The second term subtracts the quadratic cost induced by the local geometry. The third term discounts gains that lack stable support across evidence slices, with β controlling the strength of this discount. The positive-part operator retains corrections with positive predicted value, and normalization by $J_{\tau,0}$ expresses the remaining opportunity relative to the initial task objective.

Diagnose permits progression to Revise when the empirical correction headroom exceeds a prescribed threshold and the evidence-slice gradients satisfy the consistency requirement. These conditions are reassessed at the current parameter state in each adaptation iteration.

3.2 Revise: Adaptive Task-Specific Updates

Revise translates the assessment from Diagnose into targeted parameter revisions. The empirical correction headroom $\widehat{\mathcal{H}}_{\tau,t}^{\mathrm{emp}}$ and evidence consistency jointly determine whether to propose an update. When further adaptation is supported, Revise reuses the update gradient g_t to generate candidates and uses the signal strength, evidence disagreement, and local geometry from the same diagnostic pass to guide and screen them. Within this process, Step Guidance uses E_t and V_t to adaptively modulate the candidate step magnitude.

Step Guidance. Motivated by the signal–variance analysis in Appendix A.4, we define a guidance coefficient based on the strength of the task signal relative to evidence disagreement:

$$q_t = E_t/(E_t + \gamma V_t). \tag{7}$$

Here, $E_t = \|\bar{g}_t\|_2^2$, γ controls the weight assigned to disagreement, and we set $q_t = 0$ when $E_t = 0$.

Step Guidance adaptively adjusts the update magnitude through q_t: when the task signal captured by E_t dominates, a larger q_t favors retaining more of the candidate step; when the evidence disagreement captured by V_t dominates, a smaller q_t favors contracting the candidate step for a more cautious update.

We therefore incorporate Step Guidance into the training of task-specific LoRA adapters. When the diagnosis supports further adaptation, the optimizer uses the update gradient g_t to generate a candidate, which is projected into the cumulative adaptation region to obtain ϕ_t^{prop}. Step Guidance then produces the guided candidate state:

$$\phi_t^{\text{guided}} = \phi_t + \underbrace{\big(1 - \lambda_g + \lambda_g q_t\big)}_{\text{retained step fraction}}\big(\phi_t^{\text{prop}} - \phi_t\big). \tag{8}$$

Here, $\lambda_g \in [0, 1]$ controls the guidance strength. This update preserves the direction of the projected candidate displacement and adaptively adjusts its magnitude through q_t; the resulting state ϕ_t^{guided} subsequently undergoes candidate screening and evaluation. Using E_t and V_t already computed by Diagnose, Step Guidance requires only scalar arithmetic and linear scaling of the candidate displacement, with no additional model forward or backward passes.

For the guided candidate displacement $d_t = \phi_t^{\text{guided}} - \phi_t$, we reuse the update gradient, local geometry, and evidence-slice gradients from Diagnose to score its predicted gain, local quadratic cost, and evidence disagreement along that displacement (Appendix A.5). A candidate that passes this screening is then evaluated using a fixed alternate partition of the same training table to assess aggregate objective reduction and the consistency of improvements across evidence slices. Accepted revisions are retained; otherwise, the previous parameter state is restored. DrTab then reruns Diagnose at the retained state and uses the updated diagnosis to continue or terminate adaptation.

3.3 Re-read: Contextual Prediction with the Revised Learner

After adaptation, DrTab fixes the last accepted task state $\phi_\tau^\star$ and combines it with the native deployment pipeline of the backbone, using the training table $\mathcal{D}_\tau$ as context to predict outer test queries x. Accordingly, throughout the complete loop, the same table serves three complementary roles: it provides diagnostic evidence for assessing the current state, yields numerical guidance for controlling the revision magnitude, and remains explicit contextual input for final prediction.

Overall, with Diagnose centered on the empirical correction headroom, DrTab performs a diagnosis-guided, targeted Revise and ultimately conducts a multi-factor Re-read, enabling the TFM to adapt on demand according to the remaining correctable space of the current table.

4 Experiments

Figure 2 shows that pre-update gradient energy is positively correlated with subsequent one-step loss reduction, providing empirical support for using observable task signals to diagnose local correction potential and guide adaptation. Building on this evidence, we evaluate DrTab's overall predictive performance, the role of diagnosis-guided revision, and its effectiveness in adapting TFMs to new task types. Specifically, we investigate three research questions: **(RQ1)** Does DrTab improve tabular prediction across backbones? **(RQ2)** How do task evidence and diagnostic control contribute to effective revision? **(RQ3)** Can DrTab enhance TFM adaptation to non-tabular OOD tasks? Additional implementation and statistical details are deferred to the appendix.

4.1 RQ1: Does DrTab Improve End-to-End Prediction?

We evaluate whether the complete Diagnose–Revise–Re-read pipeline translates the task-specific correction potential characterized in Section 3.1 into end-to-end predictive gains across TFM backbones.

Table 1: **Prediction performance across five TFM backbones.** Dr. denotes our framework with the indicated backbone. Green subscripts report rank gains over the corresponding frozen backbone on TabArena-v0.1, and boldface indicates the best value within each comparison block.

	(a) TabArena-v0.1				(b) AutoML Subset + OpenML-CTR23		
Model	**Overall** Elo ↑	**Binary** Elo ↑	**Multi.** Elo ↑	**Reg.** Elo ↑	**Binary** AUC (%) ↑	**Multi.** LogLoss ↓	**Reg.** RMSE ↓
LightGBM	1386.7	1394.2	1440.1	1426.7	86.271	0.3258	8313.2
XGBoost	1337.6	1347.9	1384.7	1365.3	86.472	0.3349	8175.9
CatBoost	1377.3	1377.9	1431.8	1434.9	87.285	0.3848	7803.5
Random Forest	1166.8	1152.0	1319.2	1151.7	86.843	0.4464	9473.0
MLP (NN-Torch)	1257.4	1290.8	1299.3	1205.9	80.796	0.4107	8724.1
AutoGluon	**1453.8**	**1458.3**	**1501.2**	**1521.2**	**87.919**	**0.3138**	**7637.4**
TabICLv2	1543.1	1565.0	1543.2	1613.8	89.560	0.2013	7272.0
Dr. TabICLv2	**1572.3** $_{\uparrow 4}$	**1571.3** $_{\uparrow 1}$	**1615.5** $_{\uparrow 4}$	**1679.3** $_{\uparrow 4}$	**89.570**	**0.1947**	**7188.5**
LimiX-2M	1198.3	1169.0	1404.7	1193.2	88.821	0.3958	8190.0
Dr. LimiX-2M	**1206.4** $_{\uparrow 2}$	**1180.0** $_{\uparrow 3}$	**1410.1** $_{\uparrow 1}$	**1198.2** $_{\uparrow 2}$	88.821	**0.3951**	**8128.4**
TabPFN-v2	1186.8	1181.4	1457.8	1068.4	88.560	0.3719	8514.6
Dr. TabPFN-v2	**1189.7** $_{\uparrow 1}$	**1183.0** $_{\uparrow 3}$	**1462.8** $_{\uparrow 1}$	**1074.9** $_{\uparrow 1}$	**88.563**	**0.3709**	**8503.7**
TabSwift	1119.2	1123.2	1221.0	1077.4	85.939	0.4001	8556.3
Dr. TabSwift	**1135.7** $_{\uparrow 3}$	**1128.8** $_{\uparrow 1}$	**1243.7** $_{\uparrow 3}$	**1122.9** $_{\uparrow 5}$	**86.003**	**0.3862**	**8497.0**
TabFM	1460.8	1471.9	1651.8	1421.9	89.988	0.2402	7225.0
Dr. TabFM	**1461.0** $_{\uparrow 1}$	1471.9	1651.8	**1422.5** $_{\uparrow 1}$	89.988	**0.2401**	**7222.4**

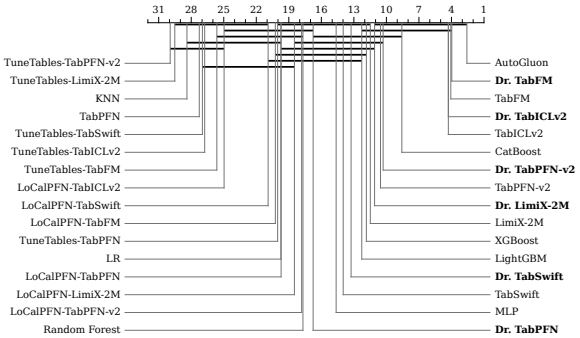


Figure 3: Critical difference diagram of average ranks on 33 classification datasets. Because the compared adaptation baselines do not cover the full benchmark, comparisons among TFM adaptation methods are restricted to this common subset. Thick bars indicate groups of methods whose average ranks are not significantly different according to the Nemenyi test (Demšar, 2006) at $\alpha = 0.05$.

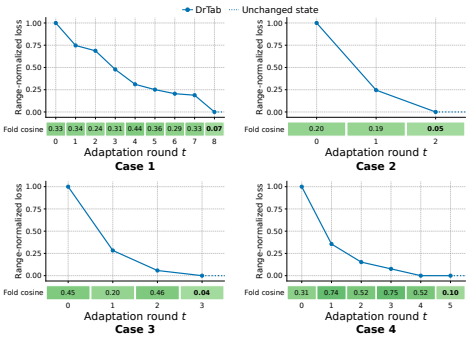
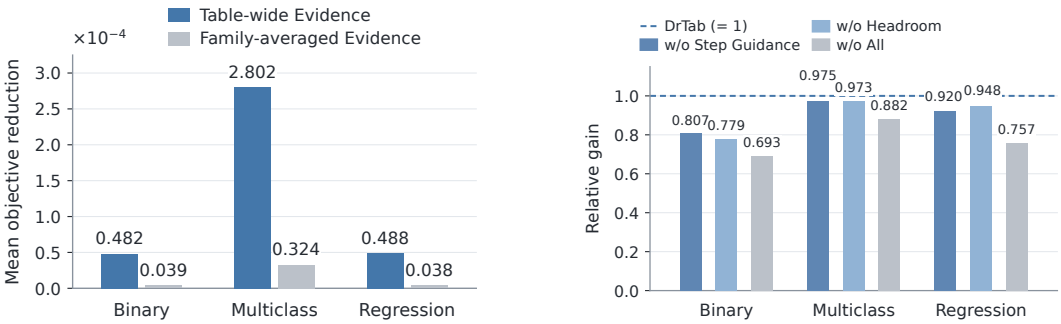


Figure 4: **DrTab adaptation trajectories on four representative binary classification tasks.** Blue curves show range-normalized loss over adaptation rounds. Green cells report the pre-update cosine similarity between fold gradients at state t, used to assess the subsequent adaptation attempt from state t to $t + 1$.

Baselines and datasets. We evaluate DrTab on five TFM backbones, including TabICLv2 (Qu et al., 2026), LimiX-2M (Zhang et al., 2025), TabPFN-v2 (Hollmann et al., 2025), TabSwift (Liu & Ye, 2026), and TabFM (Kong & Das, 2026), and compare each with its corresponding Frozen, TuneTables (Feuer et al., 2024), and LoCalPFN (Thomas et al., 2024) variants. For the overall classification ranking, we additionally include the original TabPFN (Hollmann et al., 2022) with its Frozen, DrTab, TuneTables, and LoCalPFN variants, together with conventional baselines including LightGBM, XGBoost, CatBoost, Random Forest, MLP, AutoGluon, LR, and KNN. We evaluate on TabArena-v0.1 (Erickson et al., 2026) and AutoML Subset (Gijsbers et al., 2024) + OpenML-CTR23 (Fischer et al., 2023), covering binary classification, multiclass classification, and regression.



(a) Evidence-source ablation.

(b) Diagnostic-control ablation.

Figure 5: Task evidence and diagnostic-control ablations. (a) Dataset-equal mean reduction in the alternate cross-fit objective after a one-step update. (b) Relative gain under different diagnostic-control ablations, with complete DrTab normalized to one within each task family.

Evaluation protocol and metrics. All task-local adaptation signals are computed exclusively from the outer training data, and matched methods on each backbone share the same outer split, training context, and native inference pipeline. On TabArena-v0.1, we report Elo scores and average ranks for classification tasks. On AutoML Subset + OpenML-CTR23, we use ROC AUC, log loss, and RMSE for binary classification, multiclass classification, and regression, respectively, with equal weight assigned to each dataset. Full dataset, baseline, and experimental details are provided in the appendix.

DrTab improves tabular prediction across mainstream TFM backbones. As shown in Table 1, DrTab increases Overall Elo for all five TFM backbones on TabArena-v0.1. On AutoML Subset + OpenML-CTR23, it reduces regression RMSE for all five backbones while improving or preserving binary AUC and multiclass log loss. Figure 3 further places these results in the broader classification benchmark: DrTab improves the average rank of five of six TFM backbones, ties with Frozen TabICLv2, and ranks ahead of TuneTables and LoCalPFN on every matched backbone. Together, these results show that DrTab provides consistent adaptation benefits across different TFM architectures.

Repeated diagnosis guides task-dependent iterative revision. Figure 4 shows task-dependent adaptation trajectories: Cases 2 and 3 stop after a few rounds, whereas Case 1 continues longer. The green cells report the pre-update cosine similarity between fold gradients, which DrTab combines with empirical correction headroom to determine whether to continue revision. At the terminal states, cosine similarity falls to 0.04–0.10 and the model remains unchanged, consistent with the evidence-consistency criterion in Section 3.1 and illustrating how repeated diagnosis makes adaptation depth responsive to evolving task evidence.

4.2 RQ2: How do task evidence and diagnostic control contribute to effective revision?

We examine two sources of effective revision: task-specific evidence and diagnostic control. The first ablation compares table-wide and family-averaged gradients; the second removes Headroom, Step Guidance, or all diagnostic control while retaining DrTab's LoRA parameterization. Figure 5b reports relative gain normalized to complete DrTab ($= 1$) within each task family.

Table-wide task evidence translates into more effective local correction. Theorem 1 shows that the table-wide evidence $\widehat{h}_{\mathcal{D}}$ extracted from the current training table can reveal local correction potential before parameter updates and construct an update with a positive lower bound on objective reduction. Table-wide evidence therefore provides the task-conditioned basis for Diagnose.

To further examine the role of this task-conditioned evidence in actual revision, we perform a controlled ablation that changes only the source of evidence. *Table-wide Evidence* constructs the one-step update from the cross-fitted gradient of the current training table, corresponding to $\widehat{h}_{\mathcal{D}}$ in Theorem 1. As a control, *Family-averaged Evidence* retains information at the task-family level while

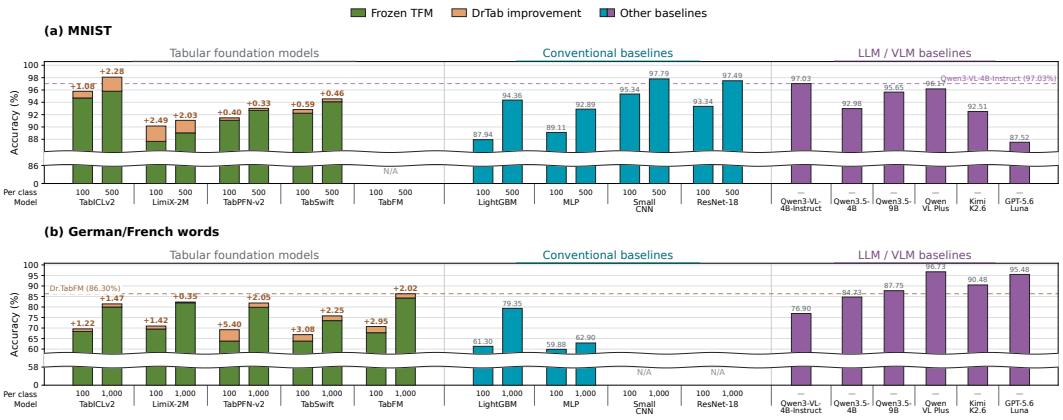


Figure 6: **DrTab improves TFM performance on non-tabular OOD tasks.** For each TFM, the base bar shows the accuracy of the frozen model, while the overlaid segment shows the absolute improvement achieved by DrTab; the annotated $+x.xx$ values denote percentage-point gains.

removing current-task specificity, replacing $\widehat{h}_{\mathcal{D}}$ with the gradient averaged across tasks from the same family. All other settings are held fixed, and both variants use the same step size within each task family. We then evaluate the one-step objective reduction on an alternate cross-fit partition that is not used to construct either update.

As shown in Figure 5a, *Table-wide Evidence* achieves a mean objective reduction approximately an order of magnitude larger than that of Family-averaged Evidence across binary classification, multiclass classification, and regression. These results show that task-conditioned evidence extracted from the current training table translates more effectively into task-local correction than evidence averaged at the task-family level. This directly supports DrTab's design of preserving table-wide task evidence in Diagnose and reusing it to construct parameter updates in Revise.

Diagnostic control helps translate task evidence into effective revision. Figure 5b compares the complete framework with different diagnostic-control ablations. Removing either Step Guidance or Headroom reduces the relative gain across binary classification, multiclass classification, and regression, while the *w/o All* variant yields a further reduction relative to both single-component ablations in all three task families. These results support the role of both forms of diagnostic control in local revision: Headroom determines whether the current state still warrants further revision, whereas Step Guidance adjusts the candidate update magnitude according to task-signal strength and evidence disagreement. Together, they allow the information produced by Diagnose to directly control whether an update is performed and how large that update should be.

Held-out validation of diagnostic decisions. We further examine whether updates accepted by the diagnostic controller remain beneficial on rows excluded from adaptation. We reserve 20% of the outer training rows exclusively for auditing; these rows are excluded from context construction, gradient computation, and all adaptation decisions. Each accepted update is evaluated from its saved pre-update state. The accepted positive rate, defined as the fraction of audited accepted updates that decrease the regularized audit objective, reaches 80.95%. This state-level audit complements the component ablations by showing that most accepted revisions improve the audit objective on rows not used for adaptation. Detailed experimental results are provided in Appendix D.4.

4.3 RQ3: Can DrTab Improve TFM Performance on OOD Tasks?

Recent work shows that frozen TFMs can perform image and word classification when inputs are represented as tables (Nakerst et al., 2026). These tasks probe transfer across task types, complementing evaluation on conventional tables. We therefore examine whether DrTab can further improve this transfer through task-specific adaptation.

Experimental setup. We consider two non-tabular OOD tasks: handwritten digit recognition on MNIST (LeCun et al., 1998) and word-level German/French language identification. Following the

class-balanced experimental construction of Nakerst et al. (2026), we vary the training/context size by the number of labeled examples per class; for example, `100/class` denotes 100 labeled examples from each class (1,000 in total for MNIST and 200 for German/French). In addition to the TFM and conventional machine-learning baselines used above, we include a set of general-purpose LLM/VLM baselines for broader comparison, including Qwen3-VL-4B-Instruct (Bai et al., 2025), Qwen3.5-4B, Qwen3.5-9B (Qwen Team, 2026), Qwen VL Plus (Bai et al., 2023), Kimi K2.6 (Moonshot AI, 2026), and GPT-5.6 Luna (OpenAI, 2026). For MNIST, we additionally include Small CNN and ResNet-18 (He et al., 2016) as task-specific vision baselines. We report classification accuracy for all settings, with full details of data construction, input representations, and model configurations provided in the appendix.

DrTab consistently strengthens TFM adaptation on non-tabular OOD tasks. As shown in Figure 6, DrTab improves the accuracy of the corresponding frozen model across both MNIST and German/French, across different training-set sizes, and for every evaluated TFM backbone. Notably, on MNIST, the frozen TabICLv2 initially performs below the best LLM/VLM baseline considered in our comparison, whereas DrTab adaptation raises its accuracy to 98.07%, surpassing all LLM/VLM baselines shown in Figure 6(a). Meanwhile, on German/French, all five TFM backbones likewise benefit from DrTab. In particular, Dr. TabFM reaches 86.30% accuracy with 1,000 examples per class, placing it within the performance range of the LLM/VLM baselines and making it competitive with several general-purpose language models. These consistent gains across backbones and data regimes indicate that, even when the task type departs substantially from standard tabular prediction, DrTab can still identify and exploit correctable task-specific signals, thereby strengthening TFM adaptation to new task types. On the two tasks considered here, this adaptation enables TFMs to match or surpass the evaluated VLM baselines on handwritten digit recognition and to achieve performance comparable to several LLM/VLM baselines on word-level language identification.

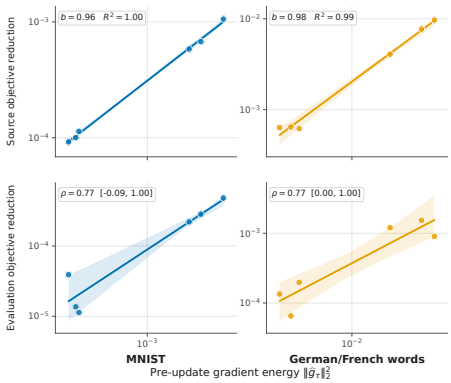


Figure 7: **Pre-update gradient energy remains diagnostic on OOD tasks.** Its positive association with subsequent objective reduction persists on MNIST and German/French word classification.

Pre-update gradient energy exhibits consistent diagnostic value on new task types. We further validate the gradient-energy analysis in Figure 2 on two non-tabular OOD tasks. As shown in Figure 7, pre-update gradient energy remains strongly positively associated with subsequent objective reduction on both MNIST and German/French word-level language identification. This consistency indicates that the diagnostic signal extends from standard tabular prediction to the additional task types considered here, further supporting the use of gradient energy to assess task-specific correction potential.

5 Conclusion

We investigated how pre-update task evidence can diagnose local correction potential and guide task-time adaptation of tabular foundation models. Under controlled gradient estimation error and local smoothness, we established a lower bound on one-step improvement in the full-context objective. Building on this insight, we introduced DrTab, an add-on framework whose Diagnose–Revise–Re-read loop assesses empirical correction headroom, guides lightweight task-specific updates, and retains the training table for in-context prediction. Experiments across six backbones and two non-tabular OOD tasks demonstrate predictive gains, while mechanistic analyses support the use of gradient energy as a diagnostic signal and highlight the contributions of task-specific evidence and diagnostic control. These findings support using the training table as both predictive context and adaptation evidence to align parameter revision with the task's remaining correction opportunities. Future work will investigate diagnostic reliability across adaptation rounds and explore more efficient task-evidence construction for large-scale tables.

AI USE STATEMENT

In accordance with the ICLR 2027 AI Policy for Authors, the authors declare that ChatGPT was used solely for language polishing. All research design, theoretical development, experiments, analyses, and conclusions were conducted and verified by the authors.

REFERENCES

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

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.

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

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 Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *Advances in Neural Information Processing Systems*, 38, 2026.

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

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. Openml-ctr23–a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023.

Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. Amlb: an automl benchmark. *Journal of Machine Learning Research*, 25(101):1–65, 2024.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors. In *International Conference on Learning Representations*, volume 2024, pp. 18209–18249, 2024.

Halil Alperen Gozeten, Muhammed Emrullah Ildiz, Xuechen Zhang, Mahdi Soltanolkotabi, Marco Mondelli, and Samet Oymak. Test-time training provably improves transformers as in-context learners. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20266–20295, 2025.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 2022.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 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. *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, June 2026. URL `https://research.google/blog/introducing-tabfm-a-zero-shot-foundation-model-for-tabular-data/`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 3045–3059, 2021.

Si-Yang Liu and Han-Jia Ye. Tabpfn unleashed: A scalable and effective solution to tabular classification problems. *arXiv preprint arXiv:2502.02527*, 2025.

Si-Yang Liu and Han-Jia Ye. Tabswift: An efficient tabular foundation model with row-wise attention. *arXiv preprint arXiv:2606.07345*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L Caterini, and Maks Volkovs. Tabdpt: Scaling tabular foundation models on real data. *Advances in Neural Information Processing Systems*, 38:172692–172722, 2025.

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

Moonshot AI. Kimi K2.6: Advancing Open-Source Coding, 2026. URL `https://www.kimi.ai/blog/kimi-k2-6`.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

Goran Nakerst, John Brennan, Wouter Beugeling, and Masudul Haque. Tabular foundation models for non-tabular tasks. *arXiv preprint arXiv:2608.22594*, 2026.

OpenAI. GPT-5.6 Luna Model. OpenAI API Documentation, 2026. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed September 26, 2026.

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

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 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.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Ivan Rubachev, Akim Kotelnikov, Nikolay Kartashev, and Artem Babenko. On finetuning tabular foundation models. *arXiv preprint arXiv:2506.08982*, 2025.

Robyn Speer. rspeer/wordfreq: v3.0. Zenodo, September 2022. URL https://doi.org/10.5281/zenodo.7199437. Software version v3.0.2.

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

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

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

A Mathematical Support for the DrTab Method

A.1 Coordinates and task-gradient heterogeneity

Let $P \succ 0$ be a fixed metric, shared across tasks for Proposition 1, and set $w = P^{-1/2}(\phi - \phi_0)$. The full-context objective is $\mathcal{J}_{\mathcal{D}}(w) = J^{\text{full}}_{\tau,\mathcal{D}}(\phi_0 + P^{1/2}w)$, where the training table is retained as context and the expectation is over fresh queries. Thus $h_{\mathcal{D}} = P^{1/2}\nabla_\phi J^{\text{full}}_{\tau,\mathcal{D}}(\phi_0)$. A deterministic retention term may be included in both the full-context and cross-fitted objectives; its common gradient cancels in their discrepancy.

Task-specific correction after pretraining. For the analysis, let $\mathcal{J}_\tau(w)$ denote the smooth full-context adaptation objective in fixed normalized adapter coordinates, with $w = 0$ representing the pretrained state, and let $h_\tau = \nabla_w \mathcal{J}_\tau(0)$. The objective averages over fresh queries while retaining the task's training table as context; the expectation below includes the task and its training table.

Proposition 1 (Task averaging conceals task-specific corrections)**.** *Suppose the task objectives are locally L-smooth, the task gradients have finite second moments, and the exact-gradient updates remain in their local feasible regions. For* $0 < \eta < 2/L$*,*

$$\mathbb{E}_\tau[\mathcal{J}_\tau(0) - \mathcal{J}_\tau(-\eta h_\tau)] \geq \eta\left(1 - \frac{L\eta}{2}\right) \times \left[\operatorname{tr}(\operatorname{Cov}(h_\tau)) + \|\mathbb{E}_\tau h_\tau\|_2^2\right]. \tag{9}$$

Proof of Proposition 1. Local smoothness along the feasible update segment gives

$$\mathcal{J}_\tau(0) - \mathcal{J}_\tau(-\eta h_\tau) \geq \eta\left(1 - \frac{L\eta}{2}\right)\|h_\tau\|_2^2.$$

Taking expectations and using

$$\mathbb{E}_\tau\|h_\tau\|_2^2 = \operatorname{tr}(\operatorname{Cov}(h_\tau)) + \|\mathbb{E}_\tau h_\tau\|_2^2$$

proves the claim. The coefficient is positive for $0 < \eta < 2/L$. □

The statement concerns the task distribution of the displayed full-context objectives. Relating its mean gradient to pretraining stationarity additionally requires the corresponding task-distribution and context-view alignment. The covariance decomposition itself holds for any task distribution satisfying the stated assumptions.

A.2 Complementary contexts and the finite-table gradient bridge

Fix a task and let $\mathcal{D} = (Z_1, \ldots, Z_N) \sim P_\tau^N$. Use $K \geq 2$ equally sized folds, with $K \mid N$, chosen independently of the row values, and let $m = N(K-1)/K$. All gradients are evaluated at the same fixed parameter state. A row encoder ψ and query-gradient maps U_s, fixed before sampling the table, describe the complete context dependence through

$$\mu_{\mathcal{C}} = \frac{1}{|\mathcal{C}|}\sum_{z\in\mathcal{C}}\psi(z), \qquad U_s(\mu_{\mathcal{C}}, z), \qquad s \in \{m, N\}.$$

Assume $\|\psi(z)\|_2 \leq R_\psi$, $\|U_s(a,z)\|_2 \leq B$, and

$$\|U_s(a,z) - U_s(b,z)\|_2 \leq \Lambda\|a-b\|_2.$$

Define the fresh-query mean gradient $\mathcal{H}_s(a) = \mathbb{E}_Z[U_s(a,Z)]$. Assume that the Jacobian of $\mathcal{H}_N$ is M-Lipschitz over the relevant convex summary region and

$$\sup_a \|\mathcal{H}_m(a) - \mathcal{H}_N(a)\|_2 \leq b_{m,N}.$$

Let $v_\psi = \mathbb{E}\|\psi(Z) - \mathbb{E}\psi(Z)\|_2^2$. The two gradients are

$$h_{\mathcal{D}} = \mathcal{H}_N(\mu_{\mathcal{D}}), \qquad \widehat{h}_{\mathcal{D}} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i\in\mathcal{P}_k} U_m(\mu_{\mathcal{C}_k}, Z_i).$$

These assumptions give an explicit model of the gradient pipeline. Approximate balance, label-stratified partitions, and more general query losses require the corresponding adapted bridge conditions.

Complementary-context cancellation. Each row occurs in exactly $K-1$ contexts of size m, so

$$\frac{1}{K}\sum_k \mu_{\mathcal{C}_k} = \mu_{\mathcal{D}}.$$

Writing $d_k = \mu_{\mathcal{C}_k} - \mu_{\mathcal{D}}$, Taylor expansion gives

$$\mathcal{H}_N(\mu_{\mathcal{C}_k}) = \mathcal{H}_N(\mu_{\mathcal{D}}) + D\mathcal{H}_N(\mu_{\mathcal{D}})d_k + r_k, \quad \|r_k\|_2 \le \frac{M}{2}\|d_k\|_2^2.$$

Because $\sum_k d_k = 0$, aggregation cancels the linear term. Adding the size discrepancy yields

$$\left\| \frac{1}{K}\sum_k \mathcal{H}_m(\mu_{\mathcal{C}_k}) - h_{\mathcal{D}} \right\|_2 \le b_{m,N} + \frac{M}{2K}\sum_k \|d_k\|_2^2.$$

Independent rows further give

$$\mathbb{E}\|d_k\|_2^2 = \left(\frac{1}{m} - \frac{1}{N}\right) v_\psi = \frac{v_\psi}{N(K-1)}.$$

Thus the expected context discrepancy is at most $b_{m,N} + Mv_\psi/[2N(K-1)]$.

Finite-table error bound. For $0 < \delta < 1$, with probability at least $1-\delta$ over the training table and the independent partition,

$$\begin{aligned}\|\widehat{h}_{\mathcal{D}} - h_{\mathcal{D}}\|_2 \le \epsilon_{N,K}(\delta) = 2(B + 2\Lambda R_\psi)\sqrt{\frac{1+\log(1/\delta)}{N}} \\ + \frac{Mv_\psi}{2N(K-1)} + b_{m,N}.\end{aligned} \tag{10}$$

To prove this, fix the partition and set $T(\mathcal{D}) = \widehat{h}_{\mathcal{D}} - h_{\mathcal{D}}$ and $A = B + 2\Lambda R_\psi$. Each held-out row is independent of its context, so the cancellation result gives

$$\|\mathbb{E}T\|_2 \le b_{m,N} + \frac{Mv_\psi}{2N(K-1)}.$$

Replacing row i by an independent copy changes its own query term by at most $2B/N$, the m query terms using it as context by at most $2\Lambda R_\psi/N$ in total, and the full-context gradient by at most $2\Lambda R_\psi/N$. Therefore

$$\|T(\mathcal{D}) - T(\mathcal{D}^{(i)})\|_2 \le \frac{2A}{N}.$$

The vector Efron–Stein inequality and Jensen's inequality imply

$$\mathbb{E}\|T - \mathbb{E}T\|_2^2 \le \frac{2A^2}{N}, \qquad \mathbb{E}\|T\|_2 \le \|\mathbb{E}T\|_2 + \frac{\sqrt{2}A}{\sqrt{N}}.$$

Applying McDiarmid's inequality to the scalar function $\|T\|_2$ then gives

$$\|T\|_2 \le \|\mathbb{E}T\|_2 + \frac{\sqrt{2}A}{\sqrt{N}}\left(1 + \sqrt{\log(1/\delta)}\right)$$

with probability at least $1-\delta$. The inequality $(1+\sqrt{u})^2 \le 2(1+u)$ proves Equation 10. This argument controls the entire estimator as a function of independent rows and allows overlap between fold contexts.

A.3 Pre-update correction and general candidate displacements

Proof of Theorem 1. Write $h = h_{\mathcal{D}}$, $\widehat{h} = \widehat{h}_{\mathcal{D}}$, and $\widehat{G} = \|\widehat{h}\|_2$. On the gradient-error event,

$$\langle h, \widehat{h}\rangle = \|\widehat{h}\|_2^2 + \langle h - \widehat{h}, \widehat{h}\rangle \ge \widehat{G}(\widehat{G} - \epsilon).$$

Smoothness along the update segment yields

$$\mathcal{J}(0) - \mathcal{J}(-\eta\widehat{h}) \ge \eta\widehat{G}(\widehat{G} - \epsilon) - \frac{L\eta^2}{2}\widehat{G}^2.$$

For $\widehat{G} > \epsilon$, this quadratic lower bound is maximized at $\eta = L^{-1}(1 - \epsilon/\widehat{G})$; substitution gives $(\widehat{G} - \epsilon)^2/(2L)$. □

The same argument at a state w gives, for any feasible candidate displacement d with a smooth update segment and a valid error bound $\|\widehat{h} - h\|_2 \le \epsilon$,

$$\mathcal{J}(w) - \mathcal{J}(w+d) \ge -\widehat{h}^\top d - \epsilon\|d\|_2 - \frac{L}{2}\|d\|_2^2.$$

This expresses the gain–uncertainty–local-cost structure used to design the empirical headroom and proposal score. Their practical quantities H_t and V_t are empirical geometry and disagreement statistics. The finite-table result above applies at the fixed initial state; extending its probability guarantee simultaneously to adaptive states requires uniform error control over the visited states.

A.4 Signal–variance motivation and the executed Guidance rule

At a fixed state and in fixed coordinates, suppose $\widehat{h}$ is an unbiased gradient estimate with $\mathbb{E}\widehat{h} = h$, $G^2 = \|h\|_2^2$, and $\mathbb{E}\|\widehat{h} - h\|_2^2 \le v$. Assume the objective is L-smooth on all update segments under consideration. Then

$$\mathbb{E}[\mathcal{J}(w) - \mathcal{J}(w - \eta\widehat{h})] \ge \eta G^2 - \frac{L\eta^2}{2}(G^2 + v).$$

Differentiating the right-hand side gives

$$\eta^\star = \frac{1}{L}\frac{G^2}{G^2 + v}$$

when $G^2 + v > 0$ and this step is feasible. For a prescribed step, a sufficient condition for strictly positive expected gain is

$$0 < \eta < \frac{2}{L}, \qquad G^2 > \frac{L\eta}{2 - L\eta}\, v.$$

The step-size range is part of the sufficient condition.

For the actual controller, let $\omega_k = |\mathcal{P}_k|/N$ and let $g_{k,t}$ be the unregularized slice gradients. Define

$$\bar{g}_t = \sum_k \omega_k g_{k,t}, \quad E_t = \|\bar{g}_t\|_2^2, \quad D_t = \sum_k \omega_k \|g_{k,t} - \bar{g}_t\|_2^2,$$

$$K_{\text{eff}} = \frac{1}{\sum_k \omega_k^2}, \qquad V_t = \frac{D_t}{\max(K_{\text{eff}} - 1, 1)}, \qquad q_t = \begin{cases} E_t/(E_t + \gamma V_t), & E_t > 0, \\ 0, & E_t = 0. \end{cases}$$

The retained fraction and guided candidate are

$$a_t = (1 - \lambda_g) + \lambda_g q_t, \qquad \phi_t^{\text{guided}} = \phi_t + a_t(\phi_t^{\text{prop}} - \phi_t),$$

where $0 \le \lambda_g \le 1$, $\gamma \ge 0$, and ϕ_t^{prop} is the Adam (Kingma & Ba, 2014) candidate after projection into the cumulative adaptation ball. The order is Adam proposal, cumulative projection, Guidance, and proposal scoring. The interpolation preserves the projected candidate direction and stays inside the ball by convexity. The reported engineering configuration uses $\lambda_g = 0.25$ and $\gamma = 1$, giving $a_t \in [0.75, 1]$.

The ideal signal–variance calculation motivates this empirical rule. The implementation uses correlated cross-fit evidence, Adam, retention, and acceptance checks; its executed step is defined by the displayed candidate interpolation. Alternate-partition acceptance evaluates the same training table under another context–query organization, relative to the current retained state.

A.5 Empirical local geometry and context construction

The main full-context construction is instantiated by TabICLv2. Other supported backbones construct their actual context from the complementary pool subject to their native capacity; the supplied engineering configuration caps these contexts at 2,048 rows. Each completed diagnostic rotation still covers all training rows as probes. The analytic bridge incorporates a prediction-view discrepancy when a corresponding bound is available.

For p adapter parameters, let $s_t = \phi_t - \phi_0$ and $J_{\tau,0} = \max(J_{\text{initial}}, 10^{-6})$. The retention-adjusted gradient and positive diagonal geometry are

$$g_t = \bar{g}_t + \frac{2\lambda_{\text{L2}}}{p} s_t,$$

$$(H_t)_{jj} = \max\left\{10^{-12}, \kappa \frac{\sum_k \omega_k g_{k,t,j}^2}{J_{\tau,0}} + \text{damping} \cdot J_{\tau,0} + \frac{2\lambda_{\text{L2}}}{p}\right\}.$$

The signal energy E_t uses $\bar{g}_t$, while the diagnostic quadratic and proposal score use g_t. For a guided displacement d, set $b_k = -g_{k,t}^\top d$ and $\bar{b} = \sum_k \omega_k b_k$. The proposal score is

$$P_t(d) = -g_t^\top d - \frac{1}{2} d^\top H_t d - \beta \sqrt{\frac{\sum_k \omega_k (b_k - \bar{b})^2}{\max(K_{\text{eff}} - 1, 1)}}.$$

The controller tests the prescribed learning-rate multipliers in order and selects the first candidate whose normalized score exceeds the threshold. These empirical definitions implement the local diagnostic principle and specify the relationship between headroom, Guidance, and candidate scoring.

B Related Work

Tabular foundation models. Gradient-boosted decision trees remain strong baselines for tabular prediction (Grinsztajn et al., 2022; McElfresh et al., 2023), while neural approaches such as FT-Transformer and TabR advance dataset-specific learning through attention and retrieval (Gorishniy et al., 2021; 2024). Tabular foundation models instead learn predictive algorithms that transfer across datasets (Ma et al., 2025; Grinsztajn et al., 2025). Prior-data fitted networks approximate Bayesian posterior prediction under a specified task prior (Müller et al., 2021). TabPFN applies this paradigm to tabular classification, and TabPFNv2 extends its predictive capabilities and supported data regimes (Hollmann et al., 2022; 2025). TabICL and TabICLv2 further improve the scalability and effectiveness of in-context prediction through architectural and pretraining advances (Qu et al., 2025; 2026). These developments provide reusable predictive priors, motivating adaptation methods that better align pretrained learners with individual downstream tables.

Adapting tabular foundation models. Existing approaches adapt pretrained tabular learners through context optimization, retrieval, and parameter updates. TuneTables learns compact task-specific contexts through soft prompt tuning while keeping the backbone frozen (Feuer et al., 2024; Lester et al., 2021). LoCalPFN combines neighborhood-based context retrieval with dataset-level fine-tuning (Thomas et al., 2024), whereas MixturePFN couples a sparse mixture of in-context prompters with bootstrap-based adapter tuning (Xu et al., 2025). BETA introduces lightweight input encoders to align downstream features with TabPFN and combines multiple encoders with bootstrapping to reduce predictive variance (Liu & Ye, 2025). Beyond specialized adaptation pipelines, Rubachev et al. (2025) systematically compare fine-tuning strategies for TabPFN-v2 and identify full fine-tuning as a strong practical baseline. Parameter-efficient methods such as LoRA provide an alternative parameterization for task-specific updates (Houlsby et al., 2019; Hu et al., 2021). DrTab builds on this adaptation paradigm, using LoRA to retain task-specific corrections while focusing on the evidence that determines whether and how strongly the learner should be revised.

Understanding task-time adaptation. Recent studies also investigate why adapting an in-context learner can help. Rubachev et al. (2025) relate improvements in TabPFN-v2 to query–key similarities that better reflect target similarity. Gozeten et al. (2025) theoretically characterize one-step test-time training for linear transformers, identifying the roles of pretraining–task alignment, context length, and adaptation sample size, with empirical validation on TabPFN. Our work asks a complementary question: what local correction opportunity does the current training table support before a parameter update? Under controlled gradient estimation error and local smoothness, we connect reduced-context task gradients to a lower bound on improvement in the full-context objective. DrTab uses this perspective to assess empirical correction headroom from task-signal strength, local movement costs, and evidence disagreement, then guides iterative revision while retaining the backbone's native in-context prediction interface.

Table 2: Reference adapter configurations for backbone integration. Q, V, and O denote query, value, and attention-output projections. Parameter counts refer to trainable LoRA factors in the listed configurations; the legacy configuration is recorded separately for each run.

Backbone	Adapted component	Selected blocks	Parameters
TabICLv2	Sample-level `TFicl` attention: Q/V/O	All 12	245,760
TabSwift	Row-wise ICL attention: packed Q/V and O	Last 12 of 24	92,160
TabPFN-v2	Sample/cell attention: Q/V/O	Last 6 of 12	55,296
LimiX-2M	Sequence attention: head-aware Q/V and O	Last 6	46,080
TabFM	Task-level `TFicl` attention: Q/V/O	Last 1 of 24	98,304
TabPFN (Original)	Selected projections in `decoder.0` and `decoder.2`	–	Run-specific

C Integration with Tabular Foundation Models

We implement DrTab on five main TFM backbones: TabICLv2 (Qu et al., 2026), TabSwift (Liu & Ye, 2026), TabPFN-v2 (Hollmann et al., 2025), LimiX-2M (Zhang et al., 2025), and TabFM. We additionally provide a separate compatibility implementation for the original TabPFN (Hollmann et al., 2022) checkpoint used in the classification comparison.

C.1 Shared Integration Interface

The implementation separates backbone-specific model operations from adaptation control. Each backbone adapter exposes its trainable parameters, differentiable context–query forward pass, state save/restore operations, and native prediction interface. The adaptation controller uses these operations to construct table-wide evidence and execute the Diagnose–Revise–Re-read loop. This interface allows DrTab to reuse its adaptation logic while preserving each backbone's architecture, preprocessing procedure, and prediction pipeline.

For the five main backbones, we introduce task-specific LoRA (Hu et al., 2021) parameters into selected query, value, and attention-output projections, while keeping the pretrained parameters frozen. The output factor B is initialized to zero, yielding zero initial weight perturbation. Input encoders, task-output heads, and the backbone's prescribed inference procedures are retained. Table 2 summarizes the reference adapter configurations.

C.2 Backbone-specific Instantiations

TabICLv2. TabICLv2 processes a table through column-level encoding, row-level encoding, and sample-level in-context learning. We inject independent Q/V/O LoRA modules into all 12 `TFicl` blocks, leaving `TFcol`, `TFrow`, and the output heads frozen. Adaptation therefore modifies the sample-interaction stage through which the model uses labeled training rows to predict queries. The accepted parameters are installed in the native Re-read estimator, which retains the prescribed preprocessing and inference views.

TabSwift. TabSwift uses a row-wise ICL predictor with register tokens and packed QKV attention. We adapt the final 12 of its 24 ICL blocks. Within each packed projection, low-rank increments are applied to Q and V, while K remains unchanged; the attention-output projection receives a separate increment. The integration preserves the native numerical and categorical transformations, normalization, feature permutations, class shifts, register tokens, and view ensemble. Final predictions are produced through the native estimator.

TabPFN-v2. We attach independent Q/V/O LoRA modules to the sample/cell-attention components in the final six Transformer blocks. These components correspond to `per_column_attention_between_cells` and exchange information across samples within feature columns. The integration retains the official preprocessing, classification logits, regression bar-distribution output, and inference ensemble. Regression adaptation uses the native bar-

distribution loss. For Re-read, the adapted state is transferred to an independent evaluation estimator.

LimiX-2M. LimiX-2M separates feature attention from sequence attention. We freeze feature attention and adapt the sequence-attention modules in the final six encoder layers. Head-aware low-rank increments accommodate the packed QKV tensor layout, modifying Q and V together with a separate attention-output increment. We use the native no-retrieval view family and retain class permutations for classification, support-fitted target standardization for regression, and native multi-view prediction.

TabFM. TabFM combines cell/column embeddings and row interactions with a task-level ICL predictor. We place Q/V/O LoRA modules only in the final `TFicl` block, freezing the preceding 23 blocks, input encoders, and output heads. For a fixed context–query view, the frozen prefix representations can be cached and reused across adaptation rounds. Cross-fitted query supervision updates the final block, and Re-read uses the native classifier or regressor with the corresponding cache and chunked-inference policy.

Original TabPFN. The original TabPFN checkpoint is handled by a separate compatibility implementation because its PFN interface differs from that of TabPFN-v2. The DrTab branch inserts LoRA into selected projections in `decoder.0` and `decoder.2`, adapts them using cross-fitted query loss, and performs final prediction through the original PFN forward interface. This decoder-based integration is distinct from the sample/cell-attention adapter used for TabPFN-v2. The LoCalPFN local-retrieval baseline is maintained as a separate method rather than being treated as part of the DrTab adapter.

C.3 Task Evidence, State Isolation, and Native Re-read

For the main backbones, classification folds are stratified, while regression folds use a fixed random seed. Every completed diagnostic rotation covers all outer-training rows as queries, with each query excluded from its corresponding support set. All fold gradients are evaluated at the same parameter state and aggregated in proportion to the number of query rows; no parameter update occurs between folds. TabICLv2 uses all remaining training rows as context. For capacity-limited adapters, support selection is restricted to the complementary pool and uses a fixed seed to select at most 2,048 rows. This context restriction does not reduce the coverage of diagnostic queries.

Preprocessing, task feedback, and adaptation-state selection use only the outer training data. After adaptation, the last accepted state is deployed through the backbone's native prediction pipeline, using the same context-selection and inference-view policies as the matched Frozen model. When no update is accepted, the method retains the Frozen state. Outer-test labels are used only for evaluation after the adaptation state has been fixed.

Each task starts from the same prescribed pretrained checkpoint with newly initialized LoRA parameters and cleared optimizer, selection, and cache states. We check that zero-initialized adapters preserve Frozen predictions within the native numerical tolerance. Adapter snapshots support candidate rollback, and each run records the applied parameter scope, cross-fit coverage, selected state, and prediction configuration.

D Tabular Benchmarks and Evaluation Protocol

This appendix details the datasets, adaptation protocols, baseline configurations, and statistical procedures used in Table 1 and Figure 3. Evaluation metrics are recomputed from saved outer-test predictions and the corresponding ground-truth labels.

D.1 Datasets, Splits, and Preprocessing

TabArena-v0.1. TabArena-v0.1 (Erickson et al., 2026) contains 51 datasets: 30 binary classification, eight multiclass classification, and 13 regression datasets, totaling 816 official outer splits. We use the official OpenML task identifiers and split artifacts. Each dataset uses three repetitions

Table 3: Coverage of the tabular evaluations. The common classification subset belongs to TabArena-v0.1. Counts are reported separately for each evaluation collection.

Evaluation collection	Binary	Multi.	Reg.	Datasets	Splits
TabArena-v0.1	30	8	13	51	816
Common classification subset	27	6	0	33	528
AutoML Subset	5	3	0	8	80
OpenML-CTR23	0	0	35	35	350

with either three or ten folds per repetition, yielding nine or 30 splits, respectively. Results retain the official repeat–fold identifiers, `r{repeat}f{fold}`. The task seed is $42 + s$, where s is the numerical split index recorded for the run. Table 3 summarizes the evaluation coverage.

Table-wide adaptation. The outer training and test partitions follow the fixed benchmark splits. DrTab partitions the outer training table into evidence slices. In each adaptation round, the slices take turns serving as label-hidden diagnostic probe sets, with the remaining training rows forming the corresponding labeled context pools. Gradients are computed at a fixed parameter state throughout the rotation and aggregated with weights proportional to the number of probe rows in each slice. Each complete rotation covers every training row as a probe. Capacity-limited backbones select support examples from the complementary context pool while preserving table-wide probe coverage.

Candidates that pass proposal screening are evaluated on a fixed alternate partition of the same training table. The acceptance check assesses aggregate objective reduction and the consistency of improvements across evidence slices. Accepted candidates are retained; rejected candidates trigger restoration of the preceding state. After adaptation, the last retained task state predicts outer-test queries using the same context-selection policy and native inference pipeline as the matched Frozen backbone.

Preprocessing and data roles. Missing-value handling, categorical encoding, numerical transformations, feature scaling, and regression-target transformations are fitted exclusively on outer training data. Outer-test labels are reserved for scoring after the final task state is fixed; gradient computation, diagnosis, proposal screening, and acceptance decisions are confined to outer training data. The independent held-out audit in Section 4.2 reserves 20% of the outer training rows for auditing. These rows are excluded from context construction, gradient computation, and all adaptation decisions. This holdout is specific to the audit; the main performance experiments use the table-wide protocol above.

TabArena binary classification datasets. The 30 datasets and their OpenML task IDs are: `APSFailure` (363616), `Amazon_employee_access` (363613), `Bank_Customer_Churn` (363619), `Bioresponse` (363620), `Diabetes130US` (363630), `E-CommereShippingData` (363632), `Fitness_Club` (363671), `GiveMeSomeCredit` (363673), `HR_Analytics_Job_Change_of_Data_Scientists` (363679), `Is-this-a-good-customer` (363682), `Marketing_Campaign` (363684), `NATICUSdroid` (363689), `bank-marketing` (363618), `blood-transfusion-service-center` (363621), `churn` (363623), `coil2000_insurance_policies` (363624), `credit-g` (363626), `credit_card_clients_default` (363627), `customer_satisfaction_in_airline` (363628), `diabetes` (363629), `hazelnut-spread-contaminant-detection` (363674), `heloc` (363676), `in_vehicle_coupon_recommendation` (363681), `jm1` (363712), `kddcup09_appetency` (363683), `online_shoppers_intention` (363691), `polish_companies_bankruptcy` (363694), `qsar-biodeg` (363696), `seismic-bumps` (363700), and `taiwanese_bankruptcy_prediction` (363706).

TabArena multiclass classification datasets. The eight datasets and their OpenML task IDs are: `MIC` (363711), `SDSS17` (363699), `anneal` (363614), `hiva_agnostic` (363677), `maternal_health_risk` (363685), `splice` (363702), `students_dropout_and_academic_success` (363704), and `website_phishing` (363707).

TabArena regression datasets. The 13 datasets and their OpenML task IDs are: Another-Dataset-on-used-Fiat-500 (363615), Food_Delivery_Time (363672), QSAR-TID-11 (363697), QSAR_fish_toxicity (363698), airfoil_self_noise (363612), concrete_compressive_strength (363625), diamonds (363631), healthcare_insurance_expenses (363675), houses (363678), miami_housing (363686), physiochemical_protein (363693), superconductivity (363705), and wine_quality (363708).

Common classification subset for Figure 3. Figure 3 compares 32 methods: the Frozen, DrTab, TuneTables, and LoCalPFN variants of the five main backbones and the original TabPFN, together with eight conventional baselines—LightGBM, XGBoost, CatBoost, Random Forest, MLP, AutoGluon, LR, and KNN. The common-coverage results combine all six backbones with these baselines and retain the dataset–split units shared by all 32 methods. This yields 33 datasets and 528 units per method, comprising 27 binary and six multiclass classification datasets.

The subset includes all classification datasets listed above except APSFailure, Bioresponse, kddcup09_appetency, MIC, and hiva_agnostic. We first average errors over the official splits of each dataset, rank all methods within that dataset, and then average the ranks with equal weight per dataset. Section D.3 gives the statistical procedure.

AutoML Subset and OpenML-CTR23. The second panel of Table 1 contains 43 tasks and 430 official folds. The AutoML Subset (Gijsbers et al., 2024) contains five binary and three multiclass tasks, each evaluated over ten folds. The binary tasks are Australian, credit-g, jasmine, kc1, and madeline. The multiclass tasks are GesturePhaseSegmentationProcessed, car, and dna.

OpenML-CTR23 (Fischer et al., 2023) contains 35 regression tasks, each evaluated over ten folds: Moneyball, QSAR_fish_toxicity, abalone, airfoil_self_noise, auction_verification, brazilian_houses, california_housing, cars, concrete_compressive_strength, cps88wages, cpu_activity, diamonds, energy_efficiency, fifa, forest_fires, fps_benchmark, geographical_origin_of_music, grid_stability, health_insurance, kin8nm, kings_county, miami_housing, naval_propulsion_plant, physiochemical_protein, pumadyn32nh, red_wine, sarcos, socmob, solar_flare, space_ga, student_performance_por, superconductivity, video_transcoding, wave_energy, and white_wine.

These tasks follow the same protocol of adaptation on outer training data and evaluation on the outer test set. Before scoring, classification probabilities are clipped to finite positive values and normalized row-wise. Regression predictions are transformed back to the original target units before computing RMSE. Metrics are averaged over the ten folds of each task and then macro-averaged with equal weight per task.

D.2 Baselines and Matched-backbone Configurations

Frozen backbones. The five main backbones are TabICLv2, LimiX-2M, TabPFN-v2, TabSwift, and TabFM. Frozen denotes direct prediction from the corresponding pretrained checkpoint using fixed preprocessing and the native inference pipeline. Matched Frozen and DrTab evaluations share the outer split, context-selection policy, and inference settings. In Table 1 and Figure 3, frozen models are labeled by their backbone names, and the prefix Dr. denotes DrTab adaptation. Figure 3 additionally includes the original TabPFN.

Conventional baselines. On TabArena, conventional baselines use tuned entries from the official result pool. LightGBM (Ke et al., 2017), XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), Random Forest (Breiman, 2001), and MLP correspond to GBM, XGB, CAT, RF, and NN_TORCH, respectively, all with the tuned + ensemble configuration. AutoGluon (Erickson et al., 2020) uses AutoGluon 1.4 (best, 4h).

For the second panel of Table 1, conventional baselines use a fixed configuration with a 60-second time limit, a 20% validation holdout, and refitting of the selected model. Tree models use 500

boosting rounds, iterations, or estimators, with a learning rate of 0.05 where applicable. NN-Torch uses 100 epochs, two layers, a hidden size of 128, and a learning rate of 0.001. The random seed is 42, and configuration selection uses training data exclusively.

TuneTables and LoCalPFN. We use cross-backbone implementations of TuneTables (Feuer et al., 2024) and LoCalPFN (Thomas et al., 2024). Each method is instantiated independently for every backbone–task–split combination from the corresponding frozen checkpoint and task seed. Optimizer states, prompts, adapters, and fine-tuned weights are initialized separately for each run. In Figure 3, TuneTables and LoCalPFN labels append the corresponding backbone name.

TuneTables (Feuer et al., 2024) uses a learnable prompt of length 10 and an upper limit of 101 training epochs, with 10 warmup epochs and a cosine learning-rate schedule. LoCalPFN (Thomas et al., 2024) uses a learning rate of 0.01, with context and training-query sizes capped at $1{,}000$ each. Its training budget comprises at most 21 rounds, each containing up to 30 parameter updates, for a maximum of 630 updates. Both methods are subject to a 600-second adaptation time limit. The epoch and update counts specify configuration ceilings; training may terminate earlier when the time limit is reached.

TabPFN and TabPFN-v2. The original TabPFN uses `prior_diff_real_checkpoint_n_0_epoch_42.cpkt` and the original model interface for classification. Its four entries in Figure 3 are TabPFN, LoCalPFN-TabPFN, TuneTables-TabPFN, and Dr. TabPFN. TabPFN-v2 uses a separate v2 checkpoint and the unified evaluation runner. Checkpoint identities, interfaces, and prediction records are tracked separately for the two versions. A within-backbone comparison of the four TabPFN variants is reported in Section D.3.

D.3 Metrics, Aggregation, and Statistical Tests

Metrics and score direction. We report ROC-AUC for binary classification, multiclass log loss for multiclass classification, and RMSE for regression. AUC is displayed as a percentage, and RMSE is computed in the original target units. Log loss and RMSE retain their numerical scales. For ranking, binary classification uses $1 - \mathrm{AUC}$, while multiclass classification and regression use log loss and RMSE directly; lower error is better in all cases. In Table 1, boldface identifies the best displayed value within each comparison block. Figure 3 uses bold labels for DrTab variants. Statistical significance is assessed separately.

Split-level and dataset-level aggregation. Let $M_{d,s,m}$ denote the metric of method m on official split s of dataset d, which has S_d splits. We compute

$$\overline{M}_{d,m} = \frac{1}{S_d} \sum_{s=1}^{S_d} M_{d,s,m}, \qquad M_m^{\mathrm{macro}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \overline{M}_{d,m}, \tag{11}$$

where $\mathcal{D}$ contains the datasets for the relevant benchmark and task type. Each dataset receives equal weight, regardless of its sample size, feature count, test-set size, or number of splits. Elo is computed separately from matched task–split comparisons.

Elo evaluation. For Table 1(a), we compute Elo ratings using a single comparison pool comprising the official TabArena reference entries and the Frozen and DrTab variants of the five main backbones. The reported Elo ratings and rank gains are derived from this pool. Comparisons with TuneTables and LoCalPFN are conducted separately in Figure 3 using average ranks on the common classification subset.

Average ranks and Nemenyi comparisons. Within each dataset, methods are ranked by their mean split-level error, with average ranks assigned to ties. The average rank of method m is

$$\overline{r}_m = \frac{1}{N} \sum_{d=1}^{N} r_{d,m}, \tag{12}$$

where N is the number of common datasets. The Nemenyi critical difference (Demšar, 2006) at significance level α is

$$\mathrm{CD}_\alpha = \frac{Q_{1-\alpha}(k,\infty)}{\sqrt{2}}\sqrt{\frac{k(k+1)}{6N}}, \tag{13}$$

where $Q_{1-\alpha}(k,\infty)$ is the corresponding quantile of the studentized-range distribution and k is the number of methods included in the comparison. Figure 3 reports the Nemenyi comparison at $\alpha = 0.05$ for $k = 32$ methods and $N = 33$ datasets.

Within-backbone comparison on TabPFN. A separate comparison evaluates TabPFN, LoCalPFN-TabPFN, TuneTables-TabPFN, and Dr. TabPFN on the same classification subset, with $k = 4$ and $N = 33$. The reported Friedman statistic is 67.5928 with $p = 1.3982 \times 10^{-14}$. The Nemenyi critical value after division by $\sqrt{2}$ is 2.5690, yielding $\mathrm{CD} = 0.81649$. TabPFN and LoCalPFN-TabPFN form the reported nonsignificant group. These statistics apply to the four-method comparison; Figure 3 presents the full 32-method comparison.

D.4 Held-out Validation of Diagnostic Decisions

Audit protocol. We examine whether updates accepted under diagnostic control reduce the audit objective on rows held out from adaptation. For this audit, we reserve 20% of the outer training rows exclusively for evaluation and use the remaining rows to construct adaptation trajectories. The held-out rows are excluded from context construction, gradient computation, and all adaptation decisions. We reconstruct each accepted update from the saved pre-update parameters, Adam state, and gradient, and measure the change in the audit objective. Audit results are used solely for post hoc evaluation.

Audit gain. For an accepted displacement d_t from state ϕ_t, we define the audit gain as

$$\begin{aligned} G_t &= \big[L_{\mathrm{audit}}(\phi_t) + \Omega(\phi_t)\big] - \big[L_{\mathrm{audit}}(\phi_t + d_t) + \Omega(\phi_t + d_t)\big], \\ \Omega(\phi) &= \frac{\lambda}{p}\|\phi - \phi_0\|_2^2, \end{aligned} \tag{14}$$

where L_{audit} is the loss on the held-out rows, λ is the retention coefficient, p is the number of trainable parameters, and ϕ_0 is the initial parameter vector for task adaptation. Positive, negative, and neutral outcomes correspond to $G_t > 0$, $G_t < 0$, and $G_t = 0$, respectively. The accepted positive rate is the fraction of audited accepted updates with $G_t > 0$.

Results. The audit comprises 63 accepted updates from 13 tasks and 27 task-seed units. Each record corresponds to one accepted update at a specific parameter state, and a task-seed unit may contribute multiple records. Of the audited updates, 51 have positive gains and 12 have negative gains. The overall accepted positive rate is 80.95%, computed by pooling the records with equal weight per update. Table 4 summarizes the state-level audit results by task family.

Table 4: State-level held-out audit of accepted updates by task family. Positive and negative outcomes correspond to decreases and increases in the regularized audit objective, respectively.

Task family	Accepted updates	Positive	Negative	Positive rate
Binary	4	4	0	100.00%
Multiclass	12	11	1	91.67%
Regression	47	36	11	76.60%
Overall	63	51	12	80.95%

E Non-tabular OOD Experiments

This appendix supplements Section 4.3 and Figure 6 with the data construction, input representations, model configurations, and evaluation protocol for MNIST digit recognition and German/French word-level language identification.

Table 5: Support-sampling seeds for the three OOD repeats. Context size is the number of labeled examples per class.

Task	Context/class	Repeat 30	Repeat 31	Repeat 32
MNIST	100	260868970	260869979	260870988
MNIST	500	260875770	260876779	260877788
Words	100	260879970	260880979	260881988
Words	1,000	260895270	260896279	260897288

E.1 Data Construction and Input Representations

MNIST. We use the official IDX gzip files, comprising 60,000 training images and 10,000 test images across ten digit classes. Each image has resolution 28×28. Class-balanced contexts contain 100 or 500 training images per class, giving 1,000 or 5,000 labeled examples, respectively. Evaluation uses the complete official test set in its original file order. Context examples are sampled from the official training split.

The shared data loader flattens the original unsigned 8-bit pixels row-wise into 784 numerical features. Model-specific preprocessing is handled by the corresponding backbone interface. LightGBM uses the flattened pixel columns, and the MLP uses a `StandardScaler` fitted on its inner fitting subset during model selection. Small CNN and ResNet-18 reshape the pixels into $1 \times 28 \times 28$ tensors and apply $x/127.5 - 1$ before the network.

German/French words. We start from the 100,000 highest-frequency entries for each language in `wordfreq` version 3.1.1 (Speer, 2022). Entries are lowercased and deduplicated. For German, ä, ö, ü, and ß are mapped to `ae`, `oe`, `ue`, and `ss`, respectively. We then apply Unicode NFKD normalization and retain ASCII letters `a`–`z`. Removing the 20,613 cleaned strings shared by the two languages yields 78,208 German and 71,984 French words. Class labels are 0 for German and 1 for French.

Each word is represented by its first 15 cleaned characters, encoded as $\texttt{a} = 1, \ldots, \texttt{z} = 26$ and padded with zeros when shorter than 15 characters. This produces 15 floating-point features per word. Class-balanced contexts contain 100 or 1,000 words per class, giving 200 or 2,000 labeled examples. The fixed final test set contains 2,000 words per class, sampled with seed `260826131`. It is disjoint from the development pool, which contains 2,000 words per class and uses seed `260822594`. Context examples are sampled from the remaining vocabulary.

Support sampling and repeats. We follow the class-balanced context-size convention used in Section 4.3. The three support repeats have identifiers $r \in \{30, 31, 32\}$. For task t and c context examples per class, the sampling seed is

$$s(t, c, r) = 260826000 + o_t + 17c + 1009r, \qquad o_{\mathrm{MNIST}} = 11000, \quad o_{\mathrm{words}} = 22000.$$

The resulting seeds are listed in Table 5. All repeats use the same fixed test set for each task.

E.2 Model Configurations

TFM backbones and DrTab. We evaluate TabICLv2, LimiX-2M, TabPFN-v2, TabSwift, and TabFM in the settings reported in Figure 6. Each Frozen model uses its pretrained checkpoint and native inference pipeline. DrTab starts from the same checkpoint and follows the Diagnose–Revise–Re-read procedure in Section 3, independently initializing task-specific adapters and optimizer state for each repeat. Frozen and DrTab share the labeled context, test split, preprocessing, and native inference configuration within each backbone and repeat.

The shared OOD configuration uses LoRA rank 8 and scaling coefficient 16, two training views, eight inference views, an inference chunk size of 4,096, and a 360-second adaptation budget. Capacity-limited backbones use a context cap of 2,048 rows, while each completed diagnostic rotation covers all training rows as probes. Adapter locations and context construction follow the corresponding backbone interfaces. Candidate evaluation uses a fixed alternate partition of the same labeled training context, and final prediction uses the last accepted task state.

Table 6: Configurations of the conventional and vision baselines.

Model	Configuration
LightGBM	At most 500 estimators; learning rate 0.05; 31 leaves; `min_child_samples`=10; L2 penalty 1; early-stopping patience 40.
MLP	Hidden widths 256/128; ReLU; dropout 0.1; AdamW (Loshchilov & Hutter, 2017) with learning rate 0.001 and weight decay 0.0001; batch size 128; at most 100 epochs; patience 12.
Small CNN	Three Conv–BN–ReLU–MaxPool blocks with widths 32/64/128; global average pooling; at most 100 epochs; patience 12.
ResNet-18	Random initialization; single-channel 3×3 stem with stride 1; initial max-pooling removed; at most 100 epochs; patience 12.

Conventional and vision baselines. LightGBM (Ke et al., 2017) and the MLP are evaluated on both tasks; Small CNN and ResNet-18 (He et al., 2016) are evaluated on MNIST. Their configurations are summarized in Table 6. Each repeat initializes a new model. Inner validation selects the number of trees or training epochs, after which the model is refitted on the complete labeled context.

LLM/VLM inputs and prompts. We evaluate Qwen3-VL-4B-Instruct (Bai et al., 2025), Qwen3.5-4B, Qwen3.5-9B, Qwen VL Plus, Kimi K2.6, and GPT-5.6 Luna. Each query consists of a task instruction and one sample: a PNG image for MNIST or the cleaned ASCII word for language identification. These models receive instruction-only queries, with zero labeled demonstrations. Their evaluations are independent of the context sizes used by the supervised baselines.

The system prompt is:

```
Follow the classification instruction.  Return only the
requested class ID, with no explanation or extra text.
```

The MNIST user prompt is:

```
Identify the handwritten digit in this image.  Output
exactly one digit from 0 to 9.
```

The word-classification user prompt is:

```
Classify the language of the following preprocessed word.  0
= German, 1 = French.  Output exactly 0 or 1.
Word:  {word}
```

Here, {`word`} is replaced by the query word, with a newline before `Word:`.

Model inference settings. Qwen3-VL-4B-Instruct uses bfloat16 and scaled dot-product attention (SDPA), with batch size 8, greedy decoding (`do_sample=false`), and the model processor's default image preprocessing. Batches are recursively split upon out-of-memory errors without changing the prompt or image resolution.

Qwen3.5-4B and Qwen3.5-9B use bfloat16 and SDPA, with batch sizes 8 and 4, respectively. Thinking is disabled, and decoding uses sampling with temperature 0.7, top-p 0.8, top-k 20, and repetition penalty 1.0. The three sampling seeds are `260915`, `260916`, and `260917`; each run covers all 10,000 MNIST test images and 4,000 German/French test words.

Qwen VL Plus uses thinking disabled, temperature 0.7, top-p 0.8, and image detail `auto`. Kimi K2.6 uses thinking disabled, temperature 0.6, top-p 0.95, and image detail `auto`. gpt-5.6-luna uses `reasoning_effort=none` and image detail `original`. All models are limited to at most 32 output tokens.

Output parsing and request handling. After stripping surrounding whitespace, each response must fully match `(0|[1-9][0-9]?)` and fall within the task's class range. Invalid outputs and terminal request failures count as incorrect predictions. Retries are limited to transport errors and HTTP 408, 429, or 5xx responses; HTTP 401, 402, and 403 responses terminate the queue.

E.3 Evaluation Protocol

We report classification accuracy in percent, averaged over three runs for each evaluated setting. Adaptation, candidate acceptance, and internal state selection use the labeled training context. Outer test labels are reserved for final scoring. For each run, accuracy is computed from the saved predictions on the complete task-specific test set.

Figure 6 reports the resulting mean accuracies. For each TFM, the base bar shows the mean Frozen accuracy, and the overlaid segment shows the percentage-point improvement from DrTab:

$$\Delta\mathrm{Acc} = \overline{\mathrm{Acc}}_{\mathrm{DrTab}} - \overline{\mathrm{Acc}}_{\mathrm{Frozen}}, \qquad \overline{\mathrm{Acc}}_m = \frac{1}{3}\sum_{j=1}^{3} \mathrm{Acc}_{m,j}.$$

The labeled context and test set are matched between Frozen and DrTab in each repeat. LLM/VLM accuracies are reported once per task because their instruction-only inputs are independent of the supervised context size.