TAFFY: A TASK-ADAPTIVE TABULAR FOUNDATION MODEL WITH IN-CONTEXT DIVERSITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent progress in tabular foundation models suggests that training on synthetic tasks can substantially improve in-context learning capabilities, with overall performance largely depending on how well models can infer task-specific predictive relationships from the available context during inference. In this paper, we introduce **TAFFY**, a tabular foundation model with an In-Context Diversity Prior and a Task-Conditioned Looped Transformer that strengthen this ability. Specifically, to construct each synthetic pretraining context, the In-Context Diversity Prior samples from multiple related environments derived via controlled interventions and distribution shifts on a shared causal process. This in-context diversity encourages the model to learn a more comprehensive and task-specific representation. Moreover, the Task-Conditioned Looped Transformer iteratively and selectively applies a shared group of Transformer blocks to refine contextual representations, with a task-conditioned gate modulating the final hidden-state update. This enables task-adaptive iterative refinement. Together, these components encourage the model to identify predictive relationships from contextual contrasts during pretraining and dynamically modulate context integration for each task. Across six classification and five regression benchmark datasets, **TAFFY** attains the lowest average rank.

1 INTRODUCTION

Tabular foundation models leverage in-context learning (ICL) to predict query labels from labeled support examples without explicit parameter updates. Prior-data fitted networks formulate this paradigm as approximating Bayesian inference under a task prior: synthetic tasks provide support–query pairs, and pretraining optimizes query predictions conditioned on the support set (Müller et al., 2022). Recent architectures such as TabPFN, TabICL, LimiX, and Mitra have advanced tabular ICL through developments in synthetic data generation and prediction architectures (Hollmann et al., 2023; 2025; Qu et al., 2025; 2026; Zhang et al., 2025a; Wang et al., 2026; Zhang et al., 2025b). Because semantic meanings and feature dependencies shift across datasets, a model cannot rely on fixed feature semantics; instead, it must infer task-specific functional dependencies directly from the support context—an intuition formalized by recent kernel and nearest-neighbor views of tabular ICL (Koshil et al., 2025; Miftachov et al., 2026). Consequently, generalization to unseen tasks depends on how effectively the model learns task-specific predictive relationships from context.

To strengthen this ability, we introduce **TAFFY**, which combines an *In-Context Diversity Prior* for synthetic task construction with a *Task-Conditioned Looped Transformer* for task-dependent context processing. First, the *In-Context Diversity Prior* incorporates observations from multiple related environments into a single synthetic pretraining context. Starting from a randomly sampled causal prototype, we first generate a set of environments through controllable distribution shifts (Zhang et al., 2015) and interventions (Pearl, 2009), then combine their support data and query data to form one prediction task. This design is inspired by causal discovery under changing distributions (Huang et al., 2020a) and latent-variable learning with auxiliary information (Hyvärinen et al., 2019), which allow the model to explore a sufficient amount of variation across different data generation conditions. Although the model is not provided with environment identifiers or access to the generating process, the mixing prior enables observations from related conditions which provides contrasting evidence about predictive relationships. Training on these contrasts encourages TAFFY to infer task-specific predictive relationships from the available context.

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.

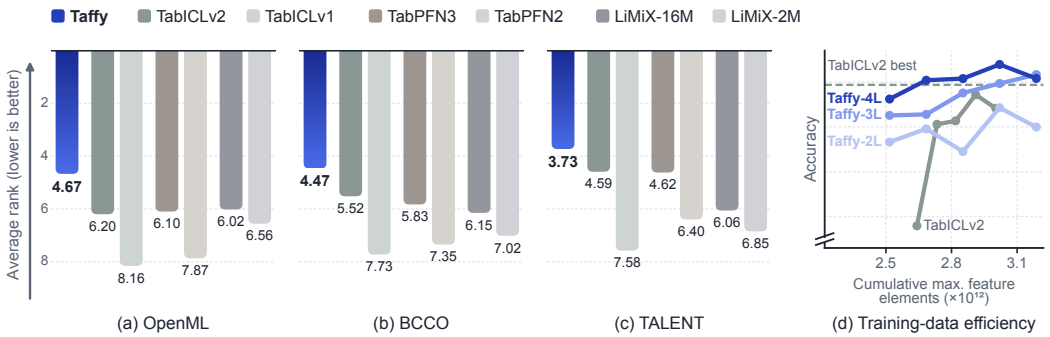

Figure 1: Predictive performance and pretraining-data efficiency. (a–c) Average accuracy ranks on OpenML-CC18, BCCO, and TALENT using the ranking pools of Table 1 (lower is better; rank axes start at 2). (d) Accuracies on TALENT classification datasets versus cumulative maximum feature elements. Markers denote measured checkpoints connected by lines; the grey line marks TabICLv2's best measured accuracy.

Second, the *Task-Conditioned Looped Transformer* combines a looped transformer stack with a task-specific gate to strengthen the model's ability to learn task-specific predictive relationships from context. Building on representations produced by the column and row encoders, our model repeatedly applies a shared ICL Transformer stack (Dehghani et al., 2019; Yang et al., 2024) to further integrate information from the labeled context. Later passes recompute attention using representations already informed by support features and labels, allowing them to build on information integrated earlier. Since the usefulness of further refinement may vary across tasks, we introduce a task-conditioning module that maps support-dataset statistics to a task-dependent scalar gate modulating the final hidden-state update. The shared loop enables iterative context processing, while the gate adjusts how strongly the final refinement contributes to prediction with the number of passes held fixed. Since these feature and label statistics are computed solely from the labeled support and unlabeled query dataset, this task-dependent refinement is also applicable when inferring predictive relationships for unseen real-world tasks. Together, the in-context diversity prior provides relational evidence, while the task-conditioned looped Transformer refines the tabular representations.

We evaluate TAFFY across six classification and five regression benchmark suites. Using Taffy-4L throughout the classification evaluation, TAFFY obtains the lowest average rank on all six classification suites; the regression model ranks first on all five regression suites. Figure 1(a–c) shows selected foundation models on OpenML-CC18, BCCO, and TALENT, with the full comparison in Table 1. Figure 1(d) compares TAFFY with reproduced TabICLv2 (Qu et al., 2026) across measured pretraining budgets, providing an exploratory view of pretraining data efficiency. Section 5.4 defines the cumulative maximum-feature-element budget and reports the measured checkpoints.

Our contributions are summarized as follows:

- We introduce a synthetic prior that places causally related environments within the same task.
- We develop a looped Transformer whose final refinement is modulated by task-specific statistics.
- TAFFY obtains the lowest average rank on all classification and regression benchmarks.

2 Related Work

2.1 Tabular Foundation Models

Tabular foundation models learn from synthetic tasks to make predictions on real-world datasets through in-context learning (Qu et al., 2025; Zhang et al., 2025b). Building on prior-data fitted networks (Müller et al., 2022), the TabPFN family extends this approach across classification, regression, and larger tables (Hollmann et al., 2023; 2025; Grinsztajn et al., 2025). Generator design includes tree-based tasks, mixtures of priors, and distributional perturbations (den Breejen et al., 2024; Zhang et al., 2025b; Bouadi et al., 2026), while the TabICL and LimiX families develop table representations and contextual interactions (Qu et al., 2025; 2026; Zhang et al., 2025a; Wang et al., 2026). Other approaches use real-data pretraining (Ma et al., 2025; Garg et al., 2025) or

self-supervised representation learning (Wu et al., 2024). Closely related to our task construction, Drift-Resilient TabPFN generates evolving SCM mechanisms and uses ordered domain indices for temporal extrapolation (Helli et al., 2024). TAFFY instead constructs unordered mixtures of related environments through distribution shifts and interventions, without supplying environment identifiers. Its focus is on placing observations from these conditions within the same support–query task, making their joint use part of the prediction problem practiced during pretraining.

2.2 LOOPED TRANSFORMERS

Universal Transformers and ALBERT share parameters across processing depth (Dehghani et al., 2019; Lan et al., 2020), while Relaxed Recursive Transformers add depth-specific low-rank adapters (Bae et al., 2025a). In ICL, looped Transformers have been studied as models of iterative algorithms (Yang et al., 2024), including connections to multi-step preconditioned gradient descent for linear regression (Gatmiry et al., 2024). Work on latent reasoning explores repeated computation in hidden states (Saunshi et al., 2025; Geiping et al., 2025), and CoTFormer permits attention to intermediate representations from earlier iterations (Mohtashami et al., 2025). For tabular prediction, Balef et al. (2026) investigate layerwise inference dynamics and a parameter-efficient single-layer looped model. A separate line of work learns halting or token-level computation allocation (Graves, 2016; Banino et al., 2021; Raposo et al., 2024; Chen et al., 2025; Bae et al., 2025b). In this paper, TAFFY uses shared multi-block recurrence to refine representations already conditioned on each task. A gate based on task statistics modulates only the final update, making its contribution task-dependent without token routing or changing the number of passes.

2.3 CAUSAL LEARNING

Our task prior is related to causal learning through its use of structured changes across environments (Schölkopf et al., 2021). Causal discovery studies changing mechanisms (Huang et al., 2020a) and cross-environment invariance (Peters et al., 2016), while multi-source domain adaptation relates distributional variation to transferable mechanisms (Zhang et al., 2015). Nonlinear ICA and iVAE establish latent identifiability using auxiliary information under specific assumptions (Hyvärinen et al., 2019; Khemakhem et al., 2020). Related work considers latent causal identification from interventions (Varici et al., 2024) and content identification from paired augmentations (von Kügelgen et al., 2021). These results motivate studying structured variation, but do not establish identifiability for our mixtures without environment identifiers. In TAFFY, causal prototypes and interventions serve as tools for constructing related environments within the same pretraining task. The goal is to train prediction from their pooled observations, rather than to recover the generating process.

3 PROBLEM FORMULATION

Tabular Foundation Model with in-context learning. We consider classification and regression tasks on tabular data. For a given task, each sample has a feature vector $x \in \mathbb{R}^d$ and the corresponding label $y \in \mathcal{Y}$, where $\mathcal{Y} = \{1, \dots, K\}$ for classification and $\mathcal{Y} = \mathbb{R}$ for regression. The available data consist of a labeled support dataset $S = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and unlabeled query dataset $X_Q = (x_j^q)_{j=1}^{n_q}$ whose corresponding labels $y_Q = (y_j^q)_{j=1}^{n_q}$ are unknown to the model. A tabular foundation model with parameters θ predicts the query labels by conditioning on the support dataset:

$$(\hat{y}_1, \dots, \hat{y}_{n_q}) = f_\theta(X_Q; S), \tag{1}$$

where $\hat{y}_j$ denotes the predictive output for query x_j^q. The output is a class-probability vector for classification or a vector of conditional quantile predictions for regression.

Pretraining objective. Following the prior-data fitted framework (Müller et al., 2022; Qu et al., 2026), we learn θ from synthetic prediction tasks. Let $\mathcal{E} = (S, X_Q, y_Q)$ denote a synthetic pretraining task, and let Π denote the distribution over such tasks induced by the synthetic prior. Pretraining minimizes the expected query prediction loss as follows:

$$\mathcal{L}_{\text{pre}}(\theta) = \mathbb{E}_{\mathcal{E}\sim\Pi}\left[\frac{1}{n_q}\sum_{j=1}^{n_q} \ell_{\mathcal{E}}\left([f_\theta(X_Q; S)]_j, y_j^q\right)\right], \tag{2}$$

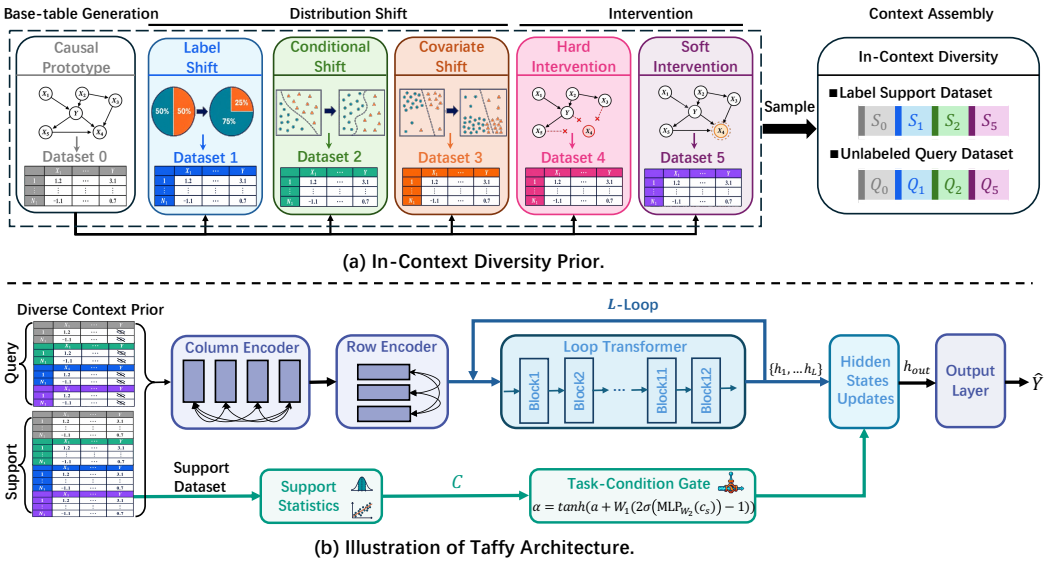

Figure 2: The pretraining pipeline of TAFFY combines the in-context diversity prior with the task-conditioned looped transformer. (a) In-Context Diversity Prior proceeds through three steps: base-table generation, multiple-environment construction, and context assembly. (b) Column and row encoders produce representations for a looped Transformer, whose final update is modulated by a support-derived task gate before query prediction.

where $\ell_{\mathcal{E}}$ is defined separately for classification and regression, following TabICLv2 (Qu et al., 2026). For classification, we use cross-entropy loss. For regression, we use pinball loss averaged over 999 predicted quantiles. At inference, point predictions are obtained by averaging these quantiles. Query labels supervise the pretraining objective but are excluded from the model inputs. The prior Π specifies the joint construction of support and query data.

4 TAFFY

4.1 OVERALL FRAMEWORK

TAFFY is designed to learn task-specific predictive relationships from a new support context. Figure 2 shows how the training-task construction and prediction network serve this goal. Specifically, the *In-Context Diversity Prior* defines the contextual prediction problems used during pretraining, including tasks that combine observations from multiple causally related environments. The *Task-Conditioned Looped Transformer* refines support-conditioned representations through a reusable update of Transformers, with later passes building on information integrated by earlier ones. The query-label objective in Eq. (2) trains the network to predict from each sampled task's support data.

Given a labeled support dataset and query features, column and row encoders produce row representations that incorporate support-label information. The looped Transformer applies a shared ICL stack repeatedly to these representations. A separate branch summarizes support features and labels statistically and computes a scalar gate, which modulates the final recurrent update before query prediction. At downstream inference, only the prediction network and its task-conditioning branch are used; the synthetic generator and environment identifiers of synthetic data are not required.

4.2 IN-CONTEXT DIVERSITY PRIOR

Design principle. The synthetic prior guides how the model learns feature–label predictive relationships from context. Our *In-Context Diversity Prior* places data from multiple causally related environments within the same task. Starting from a shared causal prototype, we construct these environments through different types of distribution shifts and interventions. Because the environments share the same causal prototype but slightly different mechanisms, their joint context provides con-

trasting evidence about how feature–label relationships behave under changing conditions. In causal discovery, such mechanism diversity can help identify causal structures among variables that are difficult to identify from a single observational distribution (Zhang et al., 2015; Pearl, 2009; Huang et al., 2020a). We adapt this insight to synthetic pretraining by using multi-environment diversity to help the model learn predictive relationships from context, rather than to recover the underlying causal graph. Figure 2(a) summarizes the construction in three steps: base-table generation, multiple-environment construction, and context assembly, which are illustrated as follows.

Step 1: Base-table generation. We first sample a causal process (Hollmann et al., 2023) as the causal prototype that defines a joint distribution over features and labels, then draw a base table from this distribution, serving as the starting point for constructing the causally related environments.

Step 2: Multiple-environment construction. Starting from the shared causal prototype, we construct multiple environments through distribution shifts in data sampling and interventions on structural mechanisms. Let D denote the number of environments, with (S_e, Q_e) denoting the support and query datasets for environment e. We construct datasets for different environments as follows.

Distribution shifts. Keeping the sampled causal graph and its structural mechanisms fixed, we first generate a fresh candidate table for each environment. We then construct environment-specific datasets through three sampling schemes (Zhang et al., 2015) corresponding to three types of distribution shifts. For *label shift*, we change $p(y)$ relative to the base table while keeping $p(x \mid y)$ unchanged. We do so by assigning different sampling probabilities to the labels and sampling data points based on their labels. For *covariate shift*, we change $p(x)$ relative to the base table while keeping $p(y \mid x)$ unchanged. We achieve it by assigning sampling probabilities based on feature values and sampling data points accordingly. For *conditional shift*, we change $p(x \mid y)$ relative to the base table while keeping $p(y)$ unchanged. Specifically, we implement it by assigning feature-dependent sampling probabilities separately within each label and sampling data points accordingly. Definitions of these shifts and implementation details are provided in Appendix A.2.

Interventions. Starting from the same sampled causal graph in step 1, we construct interventional environments by modifying the mechanisms of selected non-label variables (Pearl, 2009), without directly intervening on the label node. For a *hard intervention*, we replace each selected variable's structural equation with an externally specified value that does not depend on its parents. This removes its incoming edges, while its outgoing effects remain. For a *soft intervention*, we keep the causal graph fixed but replace the selected variable's structural mechanism with an environment-specific, potentially nonlinear mechanism over the same parents, thereby changing how the parents determine the variable without removing their influence (Brouillard et al., 2020). We then generate a new table from the intervened causal process, propagating intervention effects to descendant variables. Definitions and implementation details are provided in Appendix A.3.

Step 3: Context assembly. We combine environment-specific support and query data as follows:

$$S = \biguplus_{e=1}^{D} S_e, \qquad Q = \biguplus_{e=1}^{D} Q_e, \tag{3}$$

where $\biguplus$ denotes row-wise concatenation with repeated examples retained. The resulting S and $Q = (X_Q, y_Q)$ form the assembled support and query datasets. For multi-environment tasks, we first sample D (up to 4 distinct environments, including the unchanged base-table environment), then sample the changes used to construct the remaining environments from the candidate distribution shifts and interventions in Step 2. The model receives the labeled support dataset and query features, while query labels provide supervision through the pretraining objective. To retain exposure to single-environment tasks that may arise during downstream inference, we leave 50% of pretraining tasks as single-environment tasks and apply the multi-environment construction to the remainder.

It is noted that the environment indices are used only to assemble synthetic tasks and are not supplied to the model. The predictor learns from the pooled support and query data through query-label supervision, with no environment-prediction or causal-identification objective.

4.3 Column and Row Encoders

We follow the column-then-row encoding design of TabICLv2 (Qu et al., 2026) to obtain fixed-dimensional representations from tables with varying feature counts. Circular shifts form overlap-

ping groups of three feature values, each projected to a 128-dimensional token by a shared linear layer. Support-label embeddings are added to the corresponding support tokens to incorporate feature–label associations. Three induced-attention blocks then use 128 inducing vectors to summarize support information for each feature group and propagate it to support and query tokens.

Next, the row encoder models interactions among the feature-group tokens within each example using three Transformer blocks and four learned `[CLS]` tokens. Concatenating the four `[CLS]` outputs yields a 512-dimensional row representation. A separate label embedding is then added to each support-row representation before the looped ICL learner. These augmented support representations and the query-row representations form the initial hidden states $h^{(0)}$. Query labels are never used as inputs. Both encoders run once per forward pass; only the ICL learner is repeated.

4.4 Task-Conditioned Looped Transformer

Design principle. The In-Context Diversity Prior exposes the model to varied predictive relationships by placing observations from multiple environments within the same context. These relationships differ in complexity: some can be identified from the initial context representations, whereas others benefit from further integration across examples and features. Looped computation with task-conditioned gating provides a way to accommodate this variation. Reapplying a shared Transformer allows each pass to build on contextual information integrated by earlier passes (Dehghani et al., 2019; Yang et al., 2024), while a task-specific gate regulates how much the final refinement contributes to prediction. The module therefore adapts the influence of iterative computation to the current task, while keeping the number of passes fixed and sharing parameters across passes.

Looped Transformer. After column and row encoding in Section 4.3, we obtain the initial representations $h^{(0)}$. We then use a looped ICL Transformer to progressively integrate information across the labeled context. Let F_θ denote the twelve-block ICL Transformer with hidden width 512. Following shared recurrent computation of Transformers (Dehghani et al., 2019; Yang et al., 2024; Balef et al., 2026), we apply the same stack to successive hidden states:

$$h^{(\ell)} = F_\theta(h^{(\ell-1)}; M_S), \qquad \ell = 1, \dots, L, \tag{4}$$

where L is the number of passes and M_S denotes a support-only attention pattern. In each block, all $N = n_s + n_q$ positions serve as queries, whereas only the first n_s support positions serve as keys and values. We implement this pattern by placing support representations before query representations and slicing the first n_s hidden states for the key/value sequence. Thus, both support and query representations are updated, while query examples do not attend to one another.

Task-specific gate. The task-specific gate makes the final recurrent update adaptively depend on a statistical description of the current task. Specifically, we summarize the support dataset using statistics such as feature means, standard deviations, and class proportions, and combine them with input-shape information into a parameter-free descriptor $c_S \in \mathbb{R}^{51}$ to guide contextual refinement. The complete list of statistics is provided in Appendix B.1. Since query labels are unavailable in real-world tasks, we compute the label summaries solely from the labeled support dataset. For regression, the four class-specific entries are set to zero. To translate this task descriptor into a gate that regulates the final refinement, we use a scalar-output multilayer perceptron (MLP) parameterized by W_2 to compute a bounded task-dependent correction to a global scalar a:

$$b_S = W_1\big(2\sigma(\mathrm{MLP}_{W_2}(c_S)) - 1\big), \qquad \alpha_S = \tanh(a + b_S), \tag{5}$$

where σ is the logistic sigmoid, W_1 is a scalar scaling coefficient, and W_2 denotes the parameters of the MLP. The gate combines a global update scale with a task-specific correction.

Conditioned final update. After L passes, the looped Transformer produces hidden states $h^{(1)}, \dots, h^{(L)}$, each containing the support and query representations. We use the task-specific gate to scale the refinement from the final pass and add it to the preceding state:

$$h^{\mathrm{out}} = h^{(L-1)} + \alpha_S\big(h^{(L)} - h^{(L-1)}\big). \tag{6}$$

Here, $h^{(L)} - h^{(L-1)}$ is the update contributed by the final pass. Setting $\alpha_S = 0$ retains the preceding state; a positive gate preserves a fraction of this update, while a negative gate reverses its direction. All L passes are executed, so task conditioning changes the final representation rather than the number of computation steps. Finally, the prediction module produces $\widehat{Y}_Q = g_\theta(h_Q^{\mathrm{out}})$, where g_θ maps query representations to class probabilities or conditional quantile predictions for regression.

Table 1: Average ranks across all evaluated classification and regression datasets (lower is better). The classification row uses Taffy-4L on every suite; regression uses the separately trained model.

Model	Classification						Regression				
	BCCO	OpenML	PFN	TALENT	TabArena	TabZilla	BCCO	CTR23	PFN	TALENT	TabArena
Ensemble methods											
AutoGluon	10.62	8.29	9.88	9.96	9.98	8.52	5.96	5.85	5.96	5.89	6.15
Tree-based methods											
CatBoost	10.29	9.88	9.30	10.70	10.78	11.02	7.20	7.39	7.25	7.02	7.77
XGBoost	11.66	10.52	9.95	11.81	11.03	11.07	11.30	10.55	10.93	10.39	10.92
ET	12.23	11.55	10.96	13.34	14.77	11.83	10.15	10.03	10.29	9.76	9.54
RF	10.66	13.45	11.38	13.06	10.48	13.15	10.61	11.48	10.32	10.06	10.54
Deep tabular methods											
SwitchTab	16.73	16.46	16.16	17.45	18.98	16.00	16.85	16.36	16.39	16.47	17.23
T2G-Former	16.43	17.70	19.00	16.85	14.95	17.06	13.24	13.88	14.43	13.42	14.23
TANGOS	14.25	16.16	16.93	15.78	14.39	14.93	12.04	11.18	12.43	12.74	13.00
TabCaps	14.73	14.02	14.45	13.90	14.86	13.48	—	—	—	—	—
TabM	12.20	11.68	12.30	11.56	13.00	11.37	16.11	15.52	15.18	15.55	17.08
TabNet	18.90	18.34	19.34	18.63	18.88	18.93	16.15	15.58	16.96	16.35	16.38
TabR	11.91	11.11	11.14	11.12	12.12	10.54	10.74	10.67	12.93	11.14	12.31
TabTransformer	16.60	13.67	13.34	15.66	16.14	14.09	18.52	17.55	16.57	17.47	17.92
Tabular foundation models											
TabICLv1	7.73	8.16	7.93	7.58	7.06	8.65	—	—	—	—	—
LimiX-2M	7.02	6.56	6.14	6.85	5.72	6.20	9.76	8.88	7.75	9.17	7.15
TabPFN2	7.35	7.87	6.68	6.40	6.44	7.33	5.09	6.30	5.50	6.16	6.15
Mitra	9.72	12.60	12.20	11.34	10.31	12.09	9.30	10.39	8.93	10.35	9.92
LimiX-16M	6.15	6.02	7.00	6.06	6.03	6.67	8.09	7.55	7.57	8.00	6.46
TabICLv2	5.52	6.20	5.57	4.59	5.88	6.41	3.72	3.73	3.39	4.15	3.23
TabPFN3	5.83	6.10	6.46	4.62	4.73	6.48	3.11	4.18	3.93	3.48	2.69
TAFFY	**4.47**	**4.67**	**4.89**	**3.73**	**4.45**	**5.19**	**2.07**	**2.94**	**3.29**	**2.45**	**1.31**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate classification and regression on TALENT (Liu et al., 2025), BCCO (Zhang et al., 2025a), the TabPFN benchmark (Hollmann et al., 2025), and TabArena (Erickson et al., 2025), plus OpenML-CC18 (Bischl et al., 2021) and TabZilla (McElfresh et al., 2023) for classification and CTR23 (Fischer et al., 2023) for regression. Preprocessing is detailed in Appendix C.

Baselines and inference. Our baseline roster covers four groups: (i) tree-based methods, including CatBoost (Prokhorenkova et al., 2018), XGBoost (Chen & Guestrin, 2016), random forests (RF) (Breiman, 2001), and extremely randomized trees (ET) (Geurts et al., 2006); (ii) deep tabular and related predictors, including TabM (Gorishniy et al., 2025), TabR (Gorishniy et al., 2024), and SwitchTab (Wu et al., 2024); (iii) tabular foundation models, including TabPFN2 (Hollmann et al., 2025), TabPFN3 (Grinsztajn et al., 2026), TabICLv1/v2 (Qu et al., 2025; 2026), LimiX-2M/16M (Wang et al., 2026; Zhang et al., 2025a), and Mitra (Zhang et al., 2025b); and (iv) AutoGluon (Erickson et al., 2020). Please refer to Appendix C.1 for more details of the baselines.

Implementation and evaluation. We conduct pretraining on a cluster with 192 AMD MI210 GPUs. We train for 25,000 steps with the Muon optimizer. Each pretraining context contains up to four environments, with equal numbers of rows per environment subject to a fixed total row budget. We denote by Taffy-XL the model whose shared ICL stack is applied X times. For the primary comparison, classification uses Taffy-4L on all suites; regression uses a separately trained model. Pretraining Taffy-2L, Taffy-3L, and Taffy-4L takes approximately 2.54, 3.07, and 3.65 days, respectively, on 64 GPUs (Appendix C.4). Our primary metric is average rank within each benchmark: methods are ranked on every dataset using classification accuracy or regression RMSE, and the resulting dataset-level ranks are averaged. We further consider the Elo metric (Elo, 1978) in Appendix C.10.

5.2 OVERALL PERFORMANCE

Main Results. Following TabICLv2 (Qu et al., 2026), we exclude datasets overlapping with its development set from the primary evaluation and evaluate the 12 TALENT tasks with more than ten classes separately. For these many-class tasks, we use TabICL's hierarchical classification (Qu et al., 2025) with mixed-radix ensembling; their rank and Elo results are reported in Appendix C.7. We further report rank and Elo under the LimiX-2M dataset-selection criteria (Wang et al., 2026) in Appendix C.8. Table 1 summarizes average ranks across six classification and five regression benchmarks. TAFFY obtains the lowest average rank on all six classification and five regression

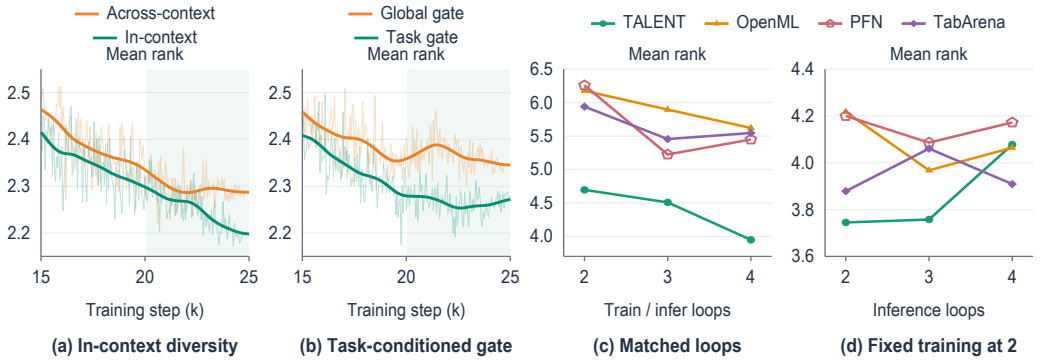

Figure 3: Component analyses: (a) in-context versus cross-context diversity; (b) task-specific versus global gating; (c) matched training/inference depth; (d) extra inference loops after two-loop training. In (a,b), each variant is ranked by accuracy against four fixed baselines (TabICLv1, TabICLv2, LimiX-2M, and LimiX-16M) on each dataset, with average ranks for ties before averaging across datasets. In (c), ranks are computed on all datasets of each benchmark. Lower ranks are better; thin lines show checkpoints, thick lines show smoothed trends.

benchmarks. On TALENT, it obtains ranks of 3.73 for classification and 2.45 for regression, compared with 4.62 and 3.48 for TabPFN3 and 4.59 and 4.15 for TabICLv2.

Ablation Study. To verify that each proposed component contributes to TAFFY, we compare three configurations in Figure 4: *Baseline*, which removes both the *In-Context Diversity Prior* and the *Task-Conditioned Looped Transformer*; + *Diversity*, which adds only the diversity prior; and *Full Taffy*, which includes both components and uses four loops during training and inference. Lower average rank indicates better performance. According to the ablation studies, adding in-context diversity improves the rank from 2.40 to 2.17, and incorporating the task-conditioned loop further reduces it to 2.07. This monotonic improvement shows that both additions contribute to the full model, while the final comparison evaluates the loop and task gate jointly rather than isolating them.

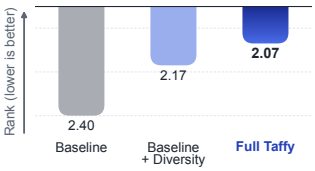

Figure 4: Average-rank ablation on TALENT classification datasets. Each variant is ranked separately against TabICLv1, TabICLv2, LimiX-2M, and LimiX-16M.

5.3 Analysis of Context Construction and Refinement

We further analyze context organization, task-specific gating, the loop depth during training and inference, and investigate which information the task-specific gate can capture.

In-Context Diversity vs. Cross-Context Diversity. To assess whether multiple environments are more useful when presented jointly within a prediction task, we compare in-context diversity with cross-context diversity. The cross-context variant retains environments generated through distribution shifts and interventions but shuffles the environment-specific data across training tasks, ensuring that each task contains data from only one environment. The in-context variant instead combines multiple related environments within half of its training tasks. Both variants use the same single-loop backbone and are evaluated every 50 steps on the TALENT classification datasets. Figure 3(a) shows the smoothed mean-rank trajectories. In-context diversity attains a lower mean rank at step 25,000, with its advantage becoming increasingly pronounced as training progresses. These results show that environmental diversity is more effective within prediction tasks than across training tasks.

Task-Specific vs. Global Gating. To evaluate the contribution of task-specific statistics, we compare a task-specific gate with a global gate. Specifically, both variants are trained and evaluated with two loops on the TALENT classification datasets; the global-gate configuration retains the same model parameterization but does not receive task-specific support statistics, making α invariant across datasets. Figure 3(b) shows that the task-specific gate achieves a consistently lower mean rank than the global gate toward the end of training, with the gap becoming more pronounced as training progresses. This result demonstrates that support-derived task statistics provide useful signals for adapting the strength of contextual refinement across datasets. Appendix C.12 provides a frozen-checkpoint intervention against a support-independent coefficient and $\alpha = 1$.

Further Analysis of the Task-Conditioned Looped Transformer. We first vary the number of loops jointly during training and inference. Figure 3(c) shows that Taffy-4L obtains the lowest average ranks on TALENT and OpenML, whereas Taffy-3L performs best on PFN and TabArena. Increasing the matched depth from two to four loops improves the rank from 4.70 to 3.95 on TALENT and from 6.18 to 5.62 on OpenML, while PFN and TabArena favor three loops. Thus, additional trained loops can improve prediction, although the preferred depth varies across benchmark suites.

We next fix training at two loops and vary only the number of inference loops. Figure 3(d) shows that a third inference loop improves the OpenML rank from 4.22 to 3.97 and the PFN rank from 4.20 to 4.09, but does not improve TALENT or TabArena. Across the 428 benchmark entries shared by all three inference configurations, two and three loops both obtain aggregate ranks that round to 4.00, whereas four loops worsen the rank to 4.23. Additional inference-time repetition therefore does not reliably reproduce the benefit of training with greater loop depth.

What Does the Task-Specific Gate Capture? To understand what the task-specific gate captures, we compute correlations between α and task properties, including feature composition, numerical distribution shape, label structure, and missingness, across TALENT tasks.

Feature composition. Across all three depths, a higher fraction of categorical input features is associated with lower α (Table 2), controlling for support rows, effective feature columns, class count, and support-label entropy. The relationship remains consistent across preprocessing views and within datasets containing both numerical and categorical features.

Table 2: Spearman correlations between α and task attributes. * indicates corrected $q < 0.05$.

Property	2L	3L	4L
Categorical fraction	$-.533^*$	$-.453^*$	$-.452^*$
Excess kurtosis	$+.230^*$	$+.326^*$	$+.323^*$
Label entropy	$+.084$	$+.010$	$-.011$
Missing ratio	$+.021$	$+.042$	$+.031$

Numerical distributions. Controlling also for categorical fraction, median excess kurtosis across numerical features correlates positively with α. Both survive multiple-testing correction, suggesting the gate responds to feature composition and numerical distribution shape.

Uncaptured task properties. The support-label entropy and the raw weighted missing-value ratio show no stable association with α. Specifically, the correlations with support-label entropy remain close to zero across all three depths. A more complex conditioning architecture is needed to exploit such dataset-specific information more fully, which is an interesting future direction.

5.4 Pretraining Data Efficiency

Figure 1(d) compares pretraining data efficiency on TALENT classification datasets. The horizontal axis measures cumulative maximum feature elements, $\mathcal{B}(s) = \sum_{t=1}^{s} B_t R_t^{\max} d_t^{\max}$, where B_t is the global task batch size, $R_t^{\max}$ is the maximum number of support and query rows per task, and $d_t^{\max}$ is the maximum number of feature columns, excluding labels. Budgets accumulate across pretraining stages without multiplying by loop count (Appendix C.9). These results reveal two patterns. First, over the shared budget range, Taffy-3L achieves data efficiency comparable to TabICLv2 and continues to improve after TabICLv2 reaches its best result, while Taffy-4L attains higher accuracy at comparable budgets. This suggests that Taffy can extract additional predictive gains from more pretraining data after the baseline has largely saturated. Second, Taffy-4L $>$ Taffy-3L $>$ Taffy-2L holds at checkpoints at the same training steps. This consistent ordering shows that increasing the trained loop depth improves the predictive performance obtained from a fixed data budget.

6 Conclusion

We presented Taffy to improve how tabular foundation models learn and refine predictive relationships from context. Rather than treating task construction and context processing separately, Taffy coordinates them: the *In-Context Diversity Prior* places related environments within a pretraining task to expose informative variation, while the *Task-Conditioned Looped Transformer* repeatedly integrates this evidence and uses support statistics to regulate the final refinement. Controlled comparisons favor in-context diversity, task-specific gating, and greater trained loop depth. Across six classification and five regression suites, the reported classification and regression configurations attain the lowest average rank on all eleven benchmarks under the stated protocols. Overall, the results identify context composition and task-adaptive iterative refinement as complementary design axes for tabular foundation models.

AI USE STATEMENT

We did not use generative AI tools for any task requiring disclosure under the ICLR 2027 AI policy; we used Claude and ChatGPT only to polish the writing, assist with auxiliary code (e.g., plotting and debugging), and edit figures, all of which were reviewed and verified by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Sercan Ö. Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021. doi: 10.1609/aaai.v35i8.16826. URL https://ojs.aaai.org/index.php/AAAI/article/view/16826.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, pp. 34282–34327, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/54d6a55225cebbdc16fbb0e45c5bdf2b-Abstract-Conference.html.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron C. Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 96572–96617, 2025b. doi: 10.52202/085713-3229. URL https://papers.nips.cc/paper_files/paper/2025/hash/8b08bbf8b420faa6eeb4020720582ec7-Abstract-Conference.html.

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? understanding inference dynamics in tabular foundation models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2605.06510. URL https://arxiv.org/abs/2605.06510.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning*, 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c7e1249ffc03eb9ded908c236bd1996d-Abstract-round2.html.

Mohamed Bouadi, Nassim Bouarour, Varun Kulkarni, Shivam Dubey, Aditya Tanna, and Vinay Kumar Sankarapu. Shaping the prior: How synthetic task distributions determine tabular foundation model quality, 2026. URL https://arxiv.org/abs/2605.18971.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Philippe Brouillard, Sébastien Lachapelle, Alexandre Lacoste, Simon Lacoste-Julien, and Alexandre Drouin. Differentiable causal discovery from interventional data. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21865–21877, 2020.

Jintai Chen, Kuanlun Liao, Yanwen Fang, Danny Z. Chen, and Jian Wu. TabCaps: A capsule neural network for tabular data classification with BoW routing. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OgbtSLESnI.

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. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28241–28259, 2025.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019.

Felix den Breejen, Sangmin Bae, Stephen Cha, and Se-Young Yun. Fine-tuned in-context learning transformers are excellent tabular data classifiers, 2024. URL https://arxiv.org/abs/2405.13396.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Publishing, 1978.

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. URL https://arxiv.org/abs/2003.06505.

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. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0519. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/1697e3fb412da11dc9488249f9e7bbc9-Abstract-Datasets_and_Benchmarks_Track.html.

Sebastian Felix Fischer, Liana Harutyunyan, Matthias Feurer, and Bernd Bischl. OpenML-CTR23: A curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023. URL https://openreview.net/pdf?id=HebAOoMm94.

Anurag Garg, Muhammad Ali, Noah Hollmann, Lennart Purucker, Samuel Müller, and Frank Hutter. Real-TabPFN: Improving tabular foundation models via continued pre-training with real-world data. *arXiv preprint arXiv:2507.03971*, 2025.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15130–15152. PMLR, 2024. URL https://proceedings.mlr.press/v235/gatmiry24b.html.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, pp. 41340–41391, 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63:3–42, 2006. doi: 10.1007/s10994-006-6226-1. URL https://doi.org/10.1007/s10994-006-6226-1.

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*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4ef594af0d9a519db8fb292452c461fa-Abstract-Conference.html.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c1ba41c694834aeef91ae161711d4939-Abstract-Conference.html.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

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

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, Dominik Safaric, Jake Robertson, Benjamin Jäger, Simone Alessi, Adrian Hayler, Vladyslav Moroshan, Lennart Purucker, Philipp Singer, Alan Arazi, Julien Siems, Jan Hendrik Metzen, Georg Grab, Nick Erickson, Siyuan Guo, Eliott Kalfon, Simon Bing, David Salinas, Clara Cornu, Lilly Charlotte Wehrhahn, Diana Kriuchkova, Kursat Kaya, Lydia Sidhoum, Marie Salmon, Jerry Chen, Madelon Hulsebos, Yann LeCun, Samuel Müller, Bernhard Schölkopf, Sauraj Gambhir, Noah Hollmann, and Frank Hutter. TabPFN-3: Technical report, 2026. URL https://arxiv.org/abs/2605.13986.

Kai Helli, David Schnurr, Noah Hollmann, Samuel Müller, and Frank Hutter. Drift-resilient TabPFN: In-context learning temporal distribution shifts on tabular data. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

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. doi: 10.1038/s41586-024-08328-6.

Biwei Huang, Kun Zhang, Jiji Zhang, Joseph Ramsey, Ruben Sanchez-Romero, Clark Glymour, and Bernhard Schölkopf. Causal discovery from heterogeneous/nonstationary data. *Journal of Machine Learning Research*, 21(89):1–53, 2020a. URL https://www.jmlr.org/papers/v21/19-232.html.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. TabTransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020b. URL https://arxiv.org/abs/2012.06678.

Aapo Hyvärinen, Hiroaki Sasaki, and Richard E. Turner. Nonlinear ICA using auxiliary variables and generalized contrastive learning. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 859–868. PMLR, 2019. URL https://proceedings.mlr.press/v89/hyvarinen19a.html.

Alan Jeffares, Tennison Liu, Jonathan Crabbé, Fergus Imrie, and Mihaela van der Schaar. TANGOS: Regularizing tabular neural networks through gradient orthogonalization and specialization. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=n6H86gW8u0d.

Ilyes Khemakhem, Diederik P. Kingma, Ricardo Pio Monti, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2207–2217. PMLR, 2020. URL https://proceedings.mlr.press/v108/khemakhem20a.html.

Mykhailo Koshil, Matthias Feurer, and Katharina Eggensperger. In-context learning of soft nearest neighbor classifiers for intelligible tabular machine learning. In *Proceedings of the 4th Table Representation Learning Workshop*, pp. 182–191, 2025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, Huai-Hong Yin, Tao Zhou, Jun-Peng Jiang, and Han-Jia Ye. TALENT: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226):1–16, 2025. URL http://jmlr.org/papers/v26/25-0512.html.

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. In *Advances in Neural Information Processing Systems*, volume 38, pp. 172692–172722. Curran Associates, Inc., 2025. doi: 10.52202/085713-5748. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/fc0e3f908a2116ba529ad0a1530a3675-Abstract-Conference.html.

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? In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-3337. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f06d5ebd4ff40b40dd97e30cee632123-Abstract-Datasets_and_Benchmarks.html.

Ratmir Miftachov, Bruno Charron, and Simon Valentin. Interpretable tabular foundation models via in-context kernel regression. *arXiv preprint arXiv:2602.02162*, 2026.

Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. CoTFormer: A chain of thought driven architecture with budget-adaptive computation cost at inference. In *International Conference on Learning Representations*, pp. 11503–11520, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1eaa5146756be028ad6fff1efcc8e6bd-Abstract-Conference.html.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2 edition, 2009. doi: 10.1017/CBO9780511803161. URL https://www.cambridge.org/core/books/causality/B0046844FAE10CBF274D4ACBDAEB5F5B.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: Identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016. doi: 10.1111/rssb.12167. URL https://doi.org/10.1111/rssb.12167.

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. URL https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

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 of *Proceedings of Machine Learning Research*, 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. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.11139.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models, 2024. URL https://arxiv.org/abs/2404.02258.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, pp. 14855–14881, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2676109d49d1eb26d6bc584a8f556305-Abstract-Conference.html.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

Burak Varici, Emre Acartürk, Karthikeyan Shanmugam, and Ali Tajer. General identifiability and achievability for causal representation learning. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2314–2322. PMLR, 2024. URL https://proceedings.mlr.press/v238/varici24a.html.

Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. Self-supervised learning with data augmentations provably isolates content from style. In *Advances in Neural Information Processing Systems*, volume 34, pp. 16451–16467. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/8929c70f8d710e412d38da624b21c3c8-Abstract.html.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. LimiX-2M: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. In *International Conference on Machine Learning*, 2026. doi: 10.48550/arXiv.2606.04485. URL https://arxiv.org/abs/2606.04485.

Jing Wu, Suiyao Chen, Qi Zhao, Renat Sergazinov, Chen Li, Shengjie Liu, Chongchao Zhao, Tianpei Xie, Hanqing Guo, Cheng Ji, Daniel Cociorva, and Hakan Brunzell. SwitchTab: Switched autoencoders are effective tabular learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 15924–15933, 2024. doi: 10.1609/aaai.v38i14.29523. URL https://ojs.aaai.org/index.php/AAAI/article/view/29523.

Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z. Chen, and Jian Wu. T2G-Former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10720–10728, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26272.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b8402301e7f06bdc97a31bfaa653dc32-Abstract-Conference.html.

Kun Zhang, Mingming Gong, and Bernhard Schölkopf. Multi-source domain adaptation: A causal view. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 3150–3157. AAAI Press, 2015. doi: 10.1609/aaai.v29i1.9542. URL https://ojs.aaai.org/index.php/AAAI/article/view/9542.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, Ningbo Dai, Renzhe Xu, Shuyang Li, Tianyang Zhang, Yue He, Yuanrui Wang, Yunjia Zhang, Zijing Xu, Dongzhe Li, Fang Gao, Hao Zou, Jiandong Liu, Jiashuo Liu, Jiawei Xu, Kaijie Cheng, Kehan Li, Linjun Zhou, Qing Li, Shaohua Fan, Xiaoyu Lin, Xinyan Han, Xuanyue Li, Yan Lu, Yuan Xue, Yuanyuan Jiang, Zimu Wang, Zhenlei Wang, and Peng Cui. LimiX: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025a. doi: 10.48550/arXiv.2509.03505. URL https://arxiv.org/abs/2509.03505.

Xiyuan Zhang, Danielle Maddix Robinson, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. 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. doi: 10.52202/085713-0535. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/177d68f4adef163b7b123b5c5adb3c60-Abstract-Conference.html.

A SYNTHETIC TASK GENERATION

This section details the In-Context Diversity Prior of Section 4.2. All environments of a multi-environment task are derived from one base structural causal model. Three distribution shifts change how rows are sampled from this model while keeping its graph and mechanisms fixed; two interventions change the model itself before data are generated. Table 3 summarizes the five operators. Appendix A.4 describes how the resulting environments are assembled into a single support–query task.

Table 3: Environment operators of the In-Context Diversity Prior. The three distribution shifts reweight rows sampled from the unchanged base model, with strength fixed by $\mathrm{KL}(p_e \,\|\, p_0) = 0.1$. The two interventions modify the mechanisms of 5% of the non-label nodes and regenerate the data.

Operator	Implementation	Preserved	Changed	Strength
Distribution shifts (row sampling from $\mathcal{M}_0$)				
Label shift	Reweight rows by $a_e(y) = \exp\big(\tau_e u_e(y)\big)$	$p(x \mid y)$	$p(y)$, hence $p(y \mid x)$	$\mathrm{KL} = 0.1$
Covariate shift	Reweight rows by $b_e(x) = \exp\big(-\tau_e \|x - \mu_e\|_2\big)$	$p(y \mid x)$	$p(x)$	$\mathrm{KL} = 0.1$
Conditional shift	Reweight rows within each class by $c_e(x, y) = \exp\big(-\tau_e \|x - \mu_{e,y}\|_2\big)$	$p(y)$	$p(x \mid y)$, hence $p(y \mid x)$	$\mathrm{KL} = 0.1$
Interventions (mechanism changes in $\mathcal{M}_0$)				
Hard intervention	Set $z_j := v_{e,j}$ with $v_{e,j} \sim \mathcal{U}[\ell_j, u_j]$ for $j \in I_e$	Other mechanisms	z_j and its descendants	5% of non-label nodes
Soft intervention	Replace g_j by $g_{e,j}$ from the same function family with resampled parameters, for $j \in I_e$	Graph G, other mechanisms	Mechanisms in I_e and their descendants	5% of non-label nodes

A.1 BASE CAUSAL PROTOTYPE

Let $\mathcal{M}_0$ denote a structural causal model sampled from the causal prior of Hollmann et al. (2023), with an acyclic graph G over nodes $V = \{1, \ldots, m\}$ and structural equations

$$z_j = g_j\big(z_{\mathrm{pa}_G(j)}, \epsilon_j\big), \qquad j = 1, \ldots, m, \tag{7}$$

where $\epsilon = (\epsilon_1, \ldots, \epsilon_m)$ follows the sampled exogenous-noise distribution. A readout selects the feature nodes and the label node $y \in V$; nodes that are not selected remain unobserved. Evaluating Equation (7) in topological order and applying the readout induces the base joint distribution $p_0(x, y)$, from which the base table is drawn. We use p for a probability density or mass function, as appropriate. All environments of a task share G, the node ordering, and the readout; they differ only in their sampling distributions or in selected mechanisms.

A.2 DISTRIBUTION SHIFTS

Candidate pool and weights. For a distribution-shift environment e, we draw a fresh candidate pool $\mathcal{C}_e = \{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{M}_0$ without modifying its graph, mechanisms, noise distribution, or readout. The empirical base distribution $\hat{p}_0$ places mass $1/n$ on each row. For label and covariate shift, each row receives a normalized sampling weight of the exponential-tilting form

$$w_{e,i}(\tau) = \frac{\exp\big(\tau\, s_e(x_i, y_i)\big)}{\sum_{k=1}^n \exp\big(\tau\, s_e(x_k, y_k)\big)}, \qquad \tau \geq 0, \tag{8}$$

where the score s_e depends on the shift type and is specified below. Conditional shift applies the same form separately within each class (Appendix A.2.3). The rows of environment e are then drawn with replacement from $\mathcal{C}_e$ according to $w_e(\tau_e)$, which defines the shifted distribution p_e. Setting $\tau = 0$ gives uniform weights and recovers the base distribution.

Shift strength. We control the strength of each distribution shift through its Kullback–Leibler divergence from the base distribution. For a reweighted distribution $p_e = r_e\, p_0$ with density ratio r_e, $\mathrm{KL}(p_e \,\|\, p_0) = \mathbb{E}_{p_e}[\log r_e]$. On the candidate pool this becomes

$$\widehat{\mathrm{KL}}_e(\tau) = \sum_{i=1}^{n} w_{e,i}(\tau) \log\big(n\, w_{e,i}(\tau)\big), \tag{9}$$

computed with the natural logarithm. We set a common target $\kappa = 0.1$ nats for all three shift types and choose τ_e by bisection so that $\widehat{\mathrm{KL}}_e(\tau_e) = \kappa$ up to a numerical tolerance. The bisection is well posed because $\widehat{\mathrm{KL}}_e(0) = 0$, $\widehat{\mathrm{KL}}_e$ is continuous, and its derivative

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\widehat{\mathrm{KL}}_e(\tau) = \tau\, \mathrm{Var}_{w_e(\tau)}\big[s_e(X,Y)\big] \geq 0 \tag{10}$$

makes it nondecreasing in τ. For conditional shift, the same argument applies within each class, so the class-averaged divergence is also nondecreasing. As $\tau \to \infty$, $\widehat{\mathrm{KL}}_e(\tau)$ approaches $\log(n/n_e^\star)$, where $n_e^\star$ is the number of rows attaining the maximal score, so the target is attainable whenever $\log(n/n_e^\star) > \kappa$. If the target is not attainable, we discard the environment and sample a new one.

Because each shift multiplies p_0 by a factor that depends only on the modified component, the joint divergence equals the divergence of that component. For label shift, $\mathrm{KL}(p_e \,\|\, p_0) = \mathrm{KL}\big(p_e(y) \,\|\, p_0(y)\big)$; for covariate shift, $\mathrm{KL}(p_e \,\|\, p_0) = \mathrm{KL}\big(p_e(x) \,\|\, p_0(x)\big)$; and for conditional shift, $\mathrm{KL}(p_e \,\|\, p_0) = \mathbb{E}_{p_0(y)}\big[\mathrm{KL}\big(p_e(x \mid y) \,\|\, p_0(x \mid y)\big)\big]$. A common κ therefore gives the three shift types comparable magnitudes, measured on the component each one changes. The value $\kappa = 0.1$ corresponds to a mild shift; for example, it moves a balanced binary label distribution from $50/50$ to approximately $72/28$.

A.2.1 Label Shift

Label shift uses a score that depends only on the label, $s_e(x, y) = u_e(y)$, where u_e assigns each class k an independent score $u_{e,k}$ drawn from a uniform distribution. Because τ_e is calibrated to the KL target, the range of this uniform distribution does not affect the resulting weights. For regression, the continuous label is first discretized into 10 bins, each bin is treated as a class, and label shift and conditional shift are then constructed exactly as for classification. The resulting label weight $a_e(y) = \exp\big(\tau_e u_e(y)\big)$ defines

$$p_e^{\mathrm{lab}}(x, y) = \frac{a_e(y) p_0(x, y)}{Z_e^{\mathrm{lab}}}, \qquad p_e^{\mathrm{lab}}(y) = \frac{a_e(y) p_0(y)}{Z_e^{\mathrm{lab}}}, \qquad Z_e^{\mathrm{lab}} = \mathbb{E}_{p_0}[a_e(Y)]. \tag{11}$$

Since the weight depends only on the label, $p_e^{\mathrm{lab}}(x \mid y) = p_0(x \mid y)$ wherever $p_e^{\mathrm{lab}}(y) > 0$. The class proportions change, and $p(y \mid x) \propto a_e(y)\, p_0(y \mid x)$ changes accordingly. Classes with larger $u_e(y)$ are over-represented, and τ_e is calibrated so that $\mathrm{KL}\big(p_e^{\mathrm{lab}}(y) \,\|\, \hat{p}_0(y)\big) = 0.1$.

A.2.2 Covariate Shift

Covariate shift uses a score that depends only on the features. Features are standardized before distances are computed. We draw a center μ_e uniformly at random within the range of the standardized features and set $s_e(x, y) = -\|x - \mu_e\|_2$, so that $b_e(x) = \exp\big(-\tau_e \|x - \mu_e\|_2\big)$ and

$$p_e^{\mathrm{cov}}(x, y) = \frac{b_e(x) p_0(x, y)}{Z_e^{\mathrm{cov}}}, \qquad p_e^{\mathrm{cov}}(x) = \frac{b_e(x) p_0(x)}{Z_e^{\mathrm{cov}}}, \qquad Z_e^{\mathrm{cov}} = \mathbb{E}_{p_0}[b_e(X)]. \tag{12}$$

Rows closer to μ_e receive larger weights, so the environment concentrates on a random region of the feature space. The conditional label distribution satisfies $p_e^{\mathrm{cov}}(y \mid x) = p_0(y \mid x)$ on the support of the shifted feature distribution, because selection depends on the features alone.

A.2.3 Conditional Shift

Conditional shift changes the feature distribution within each class while keeping the class proportions fixed (Zhang et al., 2015). For regression, the label bins of Appendix A.2.1 serve as classes, so the distribution over bins is preserved. For each class k, we draw a separate center $\mu_{e,k}$ in the same

way as for covariate shift and set $s_e(x, y) = -\|x-\mu_{e,y}\|_2$, giving $c_e(x, y) = \exp\big(-\tau_e\|x-\mu_{e,y}\|_2\big)$. The weights are normalized separately within each class,

$$q_e(x \mid y) = \frac{c_e(x, y)p_0(x \mid y)}{Z_e^{\text{cond}}(y)}, \qquad Z_e^{\text{cond}}(y) = \mathbb{E}_{p_0(X|y)}[c_e(X, y)], \tag{13}$$

and the environment distribution is $p_e^{\text{cond}}(x, y) = p_0(y)\, q_e(x \mid y)$. On the candidate pool, row i receives weight $\hat{p}_0(y_i)\, c_e(x_i, y_i)/\sum_{l:\, y_l=y_i} c_e(x_l, y_l)$. Within-class normalization is essential: a single global normalizer would generally change the label marginal as well. Consequently, $p_e^{\text{cond}}(y) = p_0(y)$, while $p(x \mid y)$ and hence $p(y \mid x)$ change. The temperature τ_e is shared across classes and calibrated so that the class-averaged divergence $\sum_k \hat{p}_0(k)\, \mathrm{KL}\big(q_e(\cdot \mid k) \,\|\, \hat{p}_0(\cdot \mid k)\big)$ equals 0.1. The invariances in this subsection hold for the environment distributions; finite sampled datasets need not satisfy their empirical counterparts exactly.

A.3 Interventions

Intervention targets. Interventions are applied directly to the causal graph and mechanisms of $\mathcal{M}_0$, rather than to a completed table (Pearl, 2009). For each interventional environment e, we sample the target set I_e uniformly at random without replacement from the non-label nodes $V \setminus \{y\}$, with

$$|I_e| = \big\lceil 0.05\,(m-1) \big\rceil, \tag{14}$$

so that at least one node is intervened on. Targets may be observed feature nodes or unobserved nodes. The label node is never intervened on directly, although the label can change through intervened ancestors. After the mechanisms in I_e are modified, we draw fresh exogenous noise, evaluate the modified model in topological order, and apply the base readout. Intervention effects therefore propagate to all descendants of I_e, including the label when an intervened node is one of its ancestors. Unlike the distribution shifts, interventions are not calibrated to a KL target; their effect is determined by the 5% budget and by the positions of the targets in G.

A.3.1 Hard Interventions

A hard intervention replaces the structural equation of each target with a constant:

$$z_j = \begin{cases} v_{e,j}, & j \in I_e, \\ g_j(z_{\mathrm{pa}_G(j)}, \epsilon_j), & j \notin I_e, \end{cases} \qquad v_{e,j} \sim \mathcal{U}[\ell_j, u_j], \tag{15}$$

where ℓ_j and u_j are the minimum and maximum values of node j in the base table. The value $v_{e,j}$ is drawn once per environment and shared by all of its rows. Each target no longer depends on its parents, so its incoming edges are removed in the intervened graph. Its outgoing effects remain: descendants are generated from their original structural equations using the intervened value. The exogenous-noise distribution and all non-target mechanisms are unchanged. When a target is a feature node, that feature is constant within the environment, which introduces a point mass into the pooled context.

A.3.2 Soft Interventions

A soft intervention replaces the mechanism of each target while retaining its parental dependence:

$$z_j = \begin{cases} g_{e,j}(z_{\mathrm{pa}_G(j)}, \epsilon_j), & j \in I_e, \\ g_j(z_{\mathrm{pa}_G(j)}, \epsilon_j), & j \notin I_e. \end{cases} \tag{16}$$

The new mechanism $g_{e,j}$ belongs to the same function family as g_j in the causal prior, with its parameters resampled independently for environment e. The graph G, the parent set of each target, and the noise distribution are unchanged, so a soft intervention changes how the parents determine the target without removing their influence (Brouillard et al., 2020). All other mechanisms retain their base definitions.

A.4 Context Assembly and Model Inputs

For a multi-environment task, we first sample the number of environments D uniformly from $\{2, 3, 4\}$. The environments consist of the unmodified base environment and $D-1$ further environments, each constructed with one operator sampled uniformly from the five candidates in Table 3.

Given the support and query budgets n_s and n_q, every environment contributes an equal number of rows, $|S_e| \approx n_s/D$ and $|Q_e| \approx n_q/D$, so that $\sum_{e=1}^{D} |S_e| = n_s$ and $\sum_{e=1}^{D} |Q_e| = n_q$. These datasets are combined as in Equation (3). Because each environment contributes to the support and query sets in the same proportion, both follow the same environment mixture; the construction introduces heterogeneity within a task rather than a shift between its support and query data. Half of the pretraining tasks use the unmodified single-environment process, and the remainder use this multi-environment construction.

The synthetic generator may use labels, including query labels, to construct a task. The prediction network receives only support features, support labels, and query features; query labels provide supervision. The task-conditioning branch computes its feature and label summaries from support data only. Environment identifiers, sampling weights, intervention targets, the causal graph, and its mechanisms are not provided to the prediction network.

The complete procedure for one multi-environment task is summarized below.

1. Sample $\mathcal{M}_0$ from the causal prior and draw the base table.
2. Sample D uniformly from $\{2, 3, 4\}$ and, for each of the $D - 1$ non-base environments, one operator uniformly from Table 3.
3. For a distribution shift, draw a candidate pool from $\mathcal{M}_0$, compute scores s_e, find τ_e by bisection so that $\widehat{\mathrm{KL}}_e(\tau_e) = 0.1$, and sample rows with replacement according to $w_e(\tau_e)$; if the target is not attainable, discard the environment and sample a new one.
4. For an intervention, sample I_e with $|I_e| = \lceil 0.05(m-1) \rceil$, modify the mechanisms in I_e according to Equation (15) or (16), and regenerate data in topological order.
5. Split each environment's rows into S_e and Q_e with equal per-environment budgets, and concatenate them as in Equation (3).

B Model Details

B.1 Support Statistics

Table 4 summarizes the 51-dimensional descriptor used by the task-specific support gate. The scalar H denotes the input feature width before grouping, including padding, while $d \leq H$ counts valid features. Each of the ten per-feature summaries is aggregated across valid features by its mean, standard deviation, minimum, and maximum. Both standard deviations use the population definition. The near-zero, near-integer, and near-binary tests are $|x| < 10^{-6}$, $|x - \mathrm{round}(x)| < 10^{-4}$, and $\min(|x|, |x - 1|) < 10^{-4}$, respectively. Summaries use finite support values; a valid feature with none receives zero summaries. Padded features are excluded, and nonfinite descriptor entries are replaced by zero.

Table 4: Statistics used by the task-specific support gate. The descriptor contains 51 entries. Here, H is the input feature width before grouping, including padding, and $d \leq H$ is the number of valid features. Feature summaries are aggregated across valid features by mean, standard deviation, minimum, and maximum.

Category	Statistics	Dim.
Size and validity	$\log(1+d)$, $\log(1+n_s)$, $\log(1+N)$, n_s/N, d/H, $1-d/H$, and the fraction of finite support values over valid features.	7
Support labels	Observed class count divided by $K_{\max}$; label entropy divided by $\log K_{\max}$; largest and smallest proportions among observed classes, with $K_{\max} = 20$.	4
Support features	Per-feature mean, standard deviation, mean absolute value, minimum, maximum, and range.	6×4
	Per-feature fractions of zero, near-zero, near-integer, and near-binary values.	4×4
Total		**51**

C Experimental Details and Additional Results

C.1 Baseline Roster

The 20 baselines in Table 1 are organized into four groups. Only available results under the matching evaluation protocol are reported; inclusion in this roster does not indicate that every method has a completed result on every suite.

Tree-based methods. These comprise CatBoost (Prokhorenkova et al., 2018), XGBoost (Chen & Guestrin, 2016), random forests (RF) (Breiman, 2001), and extremely randomized trees (ET) (Geurts et al., 2006).

Deep tabular methods. We include SwitchTab (Wu et al., 2024), T2G-Former (Yan et al., 2023), TANGOS (Jeffares et al., 2023), TabCaps (Chen et al., 2023), TabM (Gorishniy et al., 2025), TabNet (Arik & Pfister, 2021), TabR (Gorishniy et al., 2024), and TabTransformer (Huang et al., 2020b). TabCaps is evaluated only for classification.

Tabular foundation models. We include TabICLv1 (Qu et al., 2025), LimiX-2M (Wang et al., 2026), TabPFN2 (Hollmann et al., 2025), Mitra (Zhang et al., 2025b), LimiX-16M (Zhang et al., 2025a), TabICLv2 (Qu et al., 2026), and TabPFN3 (Grinsztajn et al., 2026). TabICLv1 is evaluated only for classification.

Ensemble methods. AutoGluon (Erickson et al., 2020) represents automated ensembling in the comparison roster.

C.2 Data Splits and Preprocessing

We retain the provided train–test splits. TALENT training and validation datasets are combined as support, BCCO uses its published splits, and OpenML-derived tasks use repeat 0 and fold 0. Numerical missing values are imputed with support-dataset medians. Categorical encodings are fitted on support examples, assigning unseen query categories a value of -1. Regression-label transformations are also fitted on support data, and predictions are mapped back to the original label units before evaluation. Benchmark suites may overlap.

C.3 Training and Inference Settings

Classification pretraining. Classification pretraining uses 64 AMD MI210 GPUs, a global batch size of 1,024 synthetic tasks, and a maximum context length of 4,096 rows, including support and

query examples. The configured schedule comprises 25,000 steps with the Muon optimizer, a peak learning rate of 6×10^{-4}, weight decay of 0.01, gradient clipping at 10, and label smoothing of 0.02. The learning rate warms up for 500 steps and then follows cosine decay.

Regression model and pretraining. Following TabICLv2 (Qu et al., 2026), we train a separate TAFFY model for regression; it shares no parameters with the classification model. The regression model retains the column and row encoders and the task-conditioned looped Transformer of Sections 4.3 and 4.4, with the regression-specific components of TabICLv2. Continuous support labels are embedded with linear layers instead of class lookup tables, both in the column encoder and before the looped ICL learner. The prediction head outputs 999 quantiles at probability levels $\{0.001, 0.002, \ldots, 0.999\}$ instead of class probabilities, is trained with the pinball loss of Section 3, and produces a point prediction by averaging the predicted quantiles. In the task-conditioning branch, the four class-specific support-label statistics are set to zero (Section 4.4). Regression pretraining uses the In-Context Diversity Prior of Appendix A and the same hardware, global batch size, maximum context length, number of steps, and optimizer settings as classification pretraining, with approximately the same training time (Section 5.1); label smoothing applies only to classification.

Primary-comparison model. In Table 1, classification uses Taffy-4L on all suites; regression uses the separately trained regression model. No per-dataset model selection is performed.

Inference. In-context foundation models predict query labels from the labeled support dataset without updating pretrained parameters. The recorded cross-suite classification protocol uses 32 estimators and FP32 inference for the TabICL-based classifiers, with both identity and power-transformed feature representations. For regression, we use 8 estimators, and the ensemble construction and the feature and target preprocessing follow the TabICLv2 regression configuration: ensemble members differ in random column permutations and feature preprocessing, target transformations are fitted on the support data (Appendix C.2), and the predictions of the ensemble members are averaged. TabPFN and LimiX retain their model-specific inference configurations, including ensemble and retrieval settings. Thus, frozen-parameter inference does not imply an identical inference-compute budget across models.

C.4 PRETRAINING COST

Table 5 reports the pretraining cost of the single-pass backbone and the three TAFFY variants on 64 AMD MI210 GPUs. The single-pass backbone uses the TabICLv2 architecture with one pass of the ICL stack. We measure the wall-clock time per training step and project the total time of the 25,000-step schedule from it; GPU-hours multiply this projected time by the 64 GPUs. Pretraining cost increases with loop depth: Taffy-4L requires $1.80\times$ the backbone time per step, while regression has comparable cost under the same setup (Appendix C.3).

Table 5: Pretraining cost on 64 AMD MI210 GPUs. Time per step is measured; total time and GPU-hours are projected for the 25,000-step schedule. *Relative* is the time per step divided by that of the single-pass backbone.

Model	Time/step (s)	Relative	25k steps (h)	GPU-hours
Single-pass backbone	7.01	1.00×	48.66 (2.03 days)	3,114
Taffy-2L	8.79	1.25×	61.03 (2.54 days)	3,906
Taffy-3L	10.62	1.51×	73.74 (3.07 days)	4,719
Taffy-4L	12.62	1.80×	87.67 (3.65 days)	5,611

C.5 METRICS AND AGGREGATION

We report test accuracy for classification and RMSE for regression. Methods are ranked on each dataset, with ties assigned their average rank, and these ranks are averaged within each benchmark suite; lower average rank is better. Missing results are neither imputed nor treated as losses. Elo ratings are computed from observed dataset-level pairwise outcomes as detailed in Appendix C.10.

Table 6: Results on TALENT classification datasets with more than ten classes. Only methods with observed results on all such datasets are included.

Method	Avg. rank ↓	Elo ↑
RF	8.42	1000.00
TabICLv1	5.17	1595.23
LimiX-2M	5.50	1552.44
TabPFN2	5.17	1595.23
Mitra	8.25	1044.05
LimiX-16M	5.12	1600.49
TabICLv2	2.62	1923.15
TabPFN3	2.62	1923.15
Taffy	**2.12**	**2000.81**

C.6 Primary Evaluation Pool

TabICLv2 (Qu et al., 2026) reports excluding development tasks but does not provide a machine-readable list. We therefore use an operational reconstruction from public materials: classification tasks identified in TabICL Appendix F (Qu et al., 2025) and regression tasks matched between the TabPFNv2 supplementary development roster (Hollmann et al., 2025) and TALENT by name, row count, and feature count. Matching dataset identities are removed from other suites as well; name matches with different source identities and table structures are retained. No missing result is interpolated.

The reconstructed classification exclusions are `allbp`, `baseball`, `delta_ailerons`, `eye_movements`, `eye_movements_bin`, `heloc`, `hill-valley`, `JapaneseVowels`, `led24`, `national-longitudinal-survey-binary`, `page-blocks`, `ringnorm`, `rl`, `thyroid-ann`, and `waveform-5000`. The regression exclusions are `analcatdata_supreme`, `cpu_act`, `cpu_small`, `debutanizer`, `delta_elevators`, `mauna-loa-atmospheric-co2`, `puma32H`, `stock_fardamento02`, `treasury`, `weather_izmir`, and `wind`.

After these exclusions, the primary evaluation uses the remaining memberships in each suite. Classification additionally removes TALENT tasks with more than ten classes from Table 1 and reports them separately below.

C.7 Many-Class TALENT Tasks

For TALENT classification datasets with more than ten classes, we apply hierarchical classification following TabICL (Qu et al., 2025) together with mixed-radix ensembling (Qu et al., 2026). To avoid imputation, Table 6 includes only the nine methods with observed results on all such datasets. Taffy has the best average rank and Elo in this matched comparison.

C.8 LimiX-Compatible Evaluation Pool

Following the dataset-eligibility criteria of LimiX-2M (Wang et al., 2026), we retain datasets with at most 50,000 training examples and 10,000 features, and for classification at most 10 classes. We further remove `eucalyptus` memberships with known evaluation-sample mismatches. Table 7 recomputes accuracy- and RMSE-based ranks using the complete applicable-method subset in each suite. This analysis adopts the LimiX-2M dataset filter, but not its multi-metric aggregation. Taffy has the lowest overall average rank in this pool.

C.9 Pretraining Data Budget

We measure the configured input-data budget by cumulative maximum feature elements, $\mathcal{B}(s) = \sum_{t=1}^{s} B_t R_t^{\max} d_t^{\max}$, where B_t is the global number of training tasks per step, $R_t^{\max}$ is the maximum number of support and query rows, and $d_t^{\max}$ is the maximum number of input-feature columns, excluding labels. Multistage budgets include the initializing checkpoint's accumulated budget. Loop count does not multiply this input-data proxy, which is not a measure of computational cost or

Table 7: Average ranks on the LimiX-2M-eligible, sample-compatible evaluation pool (lower is better). Rank aggregation uses the complete applicable-method subset within each suite; dashes denote incomplete task-type coverage.

Model	Classification						Regression					Average
	BCCO	OpenML	PFN	TALENT	TabArena	TabZilla	BCCO	CTR23	PFN	TALENT	TabArena	
Ensemble methods												
AutoGluon	10.53	8.29	9.88	10.00	9.98	8.52	5.86	5.85	5.96	6.00	6.15	7.91
Tree-based methods												
CatBoost	10.47	9.88	9.30	10.72	10.78	11.02	7.12	7.39	7.25	6.99	7.77	8.97
XGBoost	11.68	10.52	9.96	11.71	11.03	11.09	11.28	10.55	10.93	10.29	10.92	10.91
ET	12.03	11.56	10.93	13.23	14.77	11.80	10.14	10.03	10.29	9.70	9.54	11.27
RF	10.80	13.46	11.39	13.06	10.48	13.15	10.60	11.48	10.32	10.06	10.54	11.39
Deep tabular methods												
SwitchTab	16.71	16.45	16.16	17.45	19.00	16.02	16.90	16.36	16.39	16.56	17.23	16.84
T2G-Former	16.49	17.70	19.00	16.94	14.95	17.06	13.18	13.88	14.43	13.41	14.23	15.57
TANGOS	14.26	16.16	16.93	15.82	14.39	14.93	12.10	11.18	12.43	12.80	13.00	14.00
TabCaps	14.78	14.02	14.46	14.04	14.84	13.48	—	—	—	—	—	14.27
TabM	12.31	11.67	12.30	11.30	13.00	11.37	16.00	15.52	15.18	15.61	17.08	13.76
TabNet	18.92	18.34	19.34	18.70	18.88	18.93	16.20	15.58	16.96	16.41	16.38	17.69
TabR	11.99	11.09	11.14	11.00	12.12	10.54	10.66	10.67	12.93	11.20	12.31	11.42
TabTransformer	16.54	13.67	13.34	15.83	16.14	14.09	18.56	17.55	16.57	17.57	17.92	16.16
Tabular foundation models												
TabICLv1	7.74	8.16	7.93	7.63	7.06	8.65	—	—	—	—	—	7.86
LimiX-2M	6.88	6.56	6.14	6.65	5.72	6.20	10.02	8.88	7.75	9.24	7.15	7.38
TabPFN2	7.30	7.87	6.68	6.40	6.44	7.33	5.14	6.30	5.50	6.05	6.15	6.47
Mitra	9.71	12.60	12.20	11.34	10.31	12.09	9.26	10.39	8.93	10.13	9.92	10.63
LimiX-16M	6.09	6.03	7.00	5.94	6.03	6.67	8.16	7.55	7.57	7.99	6.46	6.86
TabICLv2	5.50	6.20	5.57	4.70	5.88	6.41	3.72	3.73	3.39	4.12	3.23	4.77
TabPFN3	5.83	6.10	6.45	4.68	4.73	6.48	3.12	4.18	3.93	3.39	2.69	4.69
TAFFY	**4.44**	**4.68**	**4.89**	**3.85**	**4.45**	**5.19**	**1.98**	**2.94**	**3.29**	**2.47**	**1.31**	**3.59**

wall-clock throughput. Figure 1(d) compares measured checkpoints at matched or lower input-data budgets; inference settings are not fully controlled across models.

Table 8: Corresponding observed-only Elo ratings under the operational development-set reconstruction (higher is better).

Method	Classification						Regression					Mean
	TALENT	BCCO	OpenML	PFN	TabArena	TabZilla	TALENT	BCCO	CTR23	PFN	TabArena	
AutoGluon	1139.13	1001.45	1217.39	1064.14	1022.49	1187.80	1241.40	1295.80	1310.03	1257.72	1332.51	1188.17
CatBoost	1105.99	1015.31	1150.20	1088.56	986.67	1086.68	1171.15	1210.85	1219.76	1178.34	1204.78	1128.94
XGBoost	1056.39	957.92	1123.70	1061.09	975.45	1084.46	981.73	956.97	1050.70	964.64	970.83	1016.72
ET	986.81	934.06	1080.91	1017.65	800.16	1053.96	1016.33	1027.80	1078.09	1002.07	1074.19	1006.55
RF	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
SwitchTab	769.70	718.50	854.24	767.09	522.28	869.73	573.18	490.49	677.30	590.18	322.86	650.50
T2G-Former	806.69	735.87	779.84	555.29	790.55	812.06	803.46	827.86	860.26	743.84	682.11	763.44
TANGOS	866.91	844.88	870.70	719.58	819.03	921.98	846.05	909.75	1016.50	874.66	800.31	862.76
TabCaps	960.88	822.62	974.46	859.48	795.37	985.93	–	–	–	–	–	899.79
TabM	1067.57	935.32	1075.44	959.47	885.79	1072.59	652.53	576.99	746.94	689.38	344.10	818.74
TabNet	683.62	567.24	734.24	519.59	532.55	680.37	583.67	572.26	742.28	535.86	434.53	598.75
TabR	1087.05	947.49	1098.94	1009.99	926.06	1105.94	939.75	992.01	1044.23	843.41	860.32	986.84
TabTransformer	873.54	725.87	990.23	912.61	726.10	959.55	467.88	201.40	555.86	573.90	218.03	655.00
TabICLv1	1250.62	1125.39	1223.07	1148.49	1159.11	1182.42	–	–	–	–	–	1181.52
LimiX-2M	1287.21	1157.56	1295.61	1232.14	1229.12	1290.29	1048.98	1051.50	1139.16	1148.70	1251.78	1193.82
TabPFN2	1310.96	1142.44	1235.63	1206.00	1190.78	1238.43	1223.98	1361.42	1282.47	1288.05	1332.51	1255.70
Mitra	1077.65	1039.40	1036.81	964.20	1007.72	1043.47	983.63	1079.13	1058.78	1080.25	1045.86	1037.90
LimiX-16M	1329.06	1199.91	1321.67	1190.80	1212.16	1268.48	1114.45	1153.87	1211.35	1159.22	1306.92	1224.35
TabICLv2	1416.24	1232.42	1313.12	1261.44	1220.58	1280.59	1365.45	1480.73	1458.84	1447.78	1647.59	1374.98
TabPFN3	1414.45	1216.29	1317.99	1216.32	1286.42	1277.10	1422.65	1543.54	1423.03	1402.27	1731.77	1386.53
Taffy	**1476.85**	**1292.81**	**1394.88**	**1298.76**	**1304.18**	**1341.96**	**1529.52**	**1678.27**	**1529.66**	**1457.50**	**2047.85**	1486.57

C.10 Elo Comparison

We report Elo ratings under both the primary evaluation pool and the LimiX-2M eligibility rule. The former follows the development-set and many-class controls used for Table 1; the latter uses the stricter LimiX-2M dataset filter. Taffy has the highest mean Elo under both evaluation pools.

Rating protocol. Each dataset supplies one comparison per method pair. Higher accuracy wins for classification and lower RMSE wins for regression; ties count as half a win. Accuracy differences within 10^{-12} are ties; RMSE ties require equality of stored values. We jointly fit an unpenalized Bradley–Terry model per benchmark, with equal dataset weights. The fitted strengths θ_i give order-independent Elo ratings,

$$E_i = 1000 + \frac{400}{\ln 10}(\theta_i - \theta_{\mathrm{RF}}), \qquad p_{ij} = \frac{1}{1 + 10^{(E_j - E_i)/400}}. \tag{17}$$

RF is anchored at 1000, following TabArena's calibration convention (Erickson et al., 2025). Average is the mean of all eleven unrounded ratings, not a pooled Elo fit; it is omitted for methods without regression results. The evaluation settings are those used for the reported results; this is not a compute-matched comparison or a test of statistical significance.

Primary evaluation. Table 8 applies the same operational development-set exclusions as Table 1; classification additionally excludes TALENT tasks with more than ten classes.

LimiX-compatible evaluation. Table 9 uses the same LimiX-compatible evaluation pool as Table 7. Missing scores are neither imputed nor treated as losses.

C.11 Gate Analysis Details

Protocol and cohorts. We analyze one late-training checkpoint of each of Taffy-2L, Taffy-3L, and Taffy-4L. We reconstruct α on CPU from the saved FP32 gate parameters and the implementation's statistics of the full processed training-plus-validation support dataset. Input-shape metadata retain the full query count. For each dataset, we average the coefficients from the `none` and `power` preprocessing views. These are reconstructed coefficients, not recorded GPU traces. The analysis uses TALENT datasets with at most 20 classes and at most one million support cells. Categorical fraction requires matching raw metadata, while the kurtosis analysis additionally requires numerical features and finite metadata.

Descriptors and adjustment. Categorical fraction is the fraction of original input columns that are categorical. Numerical excess kurtosis is the median across raw numerical columns of $\mathrm{mean}[(x -$

Table 9: Elo ratings on the LimiX-2M-eligible, sample-compatible evaluation pool (higher is better; RF = 1000 per benchmark). Only observed pairwise outcomes are used; missing scores are not imputed, and dashes denote incomplete task-type coverage.

Model	Classification						Regression					Average
	BCCO	OpenML	PFN	TALENT	TabArena	TabZilla	BCCO	CTR23	PFN	TALENT	TabArena	
Ensemble methods												
AutoGluon	1011.38	1217.39	1064.14	1138.25	1021.54	1187.80	1303.99	1310.03	1257.72	1236.25	1332.51	1189.18
Tree-based methods												
CatBoost	1013.58	1150.20	1088.56	1106.17	973.79	1086.68	1216.33	1219.76	1178.34	1174.86	1204.78	1128.46
XGBoost	962.97	1123.70	1061.09	1061.68	976.48	1084.46	957.85	1050.70	964.64	986.98	970.83	1018.31
ET	948.06	1080.91	1017.65	991.99	809.15	1053.96	1028.09	1078.09	1002.07	1020.32	1074.19	1009.50
RF	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
Deep tabular methods												
SwitchTab	724.60	854.24	767.09	768.92	521.75	869.73	480.01	677.30	590.18	556.05	322.86	648.43
T2G-Former	737.05	779.84	555.29	799.91	800.02	812.06	831.16	860.26	743.84	800.85	682.11	763.85
TANGOS	849.72	870.70	719.58	863.75	827.08	921.98	905.28	1016.50	874.66	840.19	800.31	862.70
TabCaps	825.54	974.46	859.48	954.11	792.33	985.93	—	—	—	—	—	898.64
TabM	935.27	1075.44	959.47	1079.86	880.95	1072.59	586.18	746.94	689.38	640.91	344.10	819.19
TabNet	570.05	734.24	519.59	675.82	528.89	680.37	564.30	742.28	535.86	569.60	434.53	595.96
TabR	949.47	1098.94	1009.99	1093.77	922.90	1105.94	996.32	1044.23	843.41	935.41	860.32	987.34
TabTransformer	734.37	990.23	912.61	863.60	723.74	959.55	182.08	555.86	573.90	444.82	218.03	650.80
Tabular foundation models												
TabICLv1	1131.13	1223.07	1148.49	1249.15	1152.41	1182.42	—	—	—	—	—	1181.11
LimiX-2M	1170.90	1295.61	1232.14	1298.69	1198.36	1290.29	1035.39	1139.16	1148.70	1045.70	1251.78	1191.52
TabPFN2	1151.09	1235.63	1206.00	1311.45	1182.70	1238.43	1358.90	1282.47	1288.05	1233.01	1332.51	1256.39
Mitra	1045.63	1036.81	964.20	1078.11	1007.40	1043.47	1081.58	1058.78	1080.25	996.04	1045.86	1039.83
LimiX-16M	1210.05	1321.67	1190.80	1336.53	1203.14	1268.48	1149.40	1211.35	1159.22	1116.35	1306.92	1224.90
TabICLv2	1240.08	1313.12	1261.44	1409.58	1211.18	1280.59	1483.45	1458.84	1447.78	1370.48	1647.59	1374.92
TabPFN3	1222.74	1317.99	1216.32	1410.83	1273.94	1277.10	1546.03	1423.03	1402.27	1433.48	1731.77	1386.86
TAFFY	**1301.29**	**1394.88**	**1298.76**	**1467.72**	**1290.81**	**1341.96**	**1697.50**	**1529.66**	**1457.50**	**1530.52**	**2047.85**	**1487.13**

$\bar{x})^4]/\sigma^4 - 3$, using the population standard deviation and assigning zero to constant columns. The raw numerical summaries use finite values from a deterministic sample of at most 10,000 original training rows. Datasets without numerical columns are excluded from the kurtosis analysis. Raw metadata and processed support statistics therefore describe different stages of the data pipeline. We compute partial Spearman correlations by residualizing the midranks of each descriptor and α against an intercept and the midranks of log support rows, log effective columns, class count, and support-label entropy $H(y)/\log 20$. Kurtosis fits also control for categorical fraction.

Uncertainty and multiple testing. Table 10 reports pointwise 95% percentile intervals from 2,000 dataset bootstrap resamples, recomputing ranks and residuals each time. Categorical-fraction q-values use 9,999 residual permutations and Benjamini–Hochberg correction over 15 tests: three descriptors across five late-training checkpoints of the three variants. Kurtosis q-values use approximate two-sided t tests and a separate 18-test correction covering three additional descriptors, each tested against three gate coefficients and three fixed-checkpoint inference-depth gains. Across the three model depths, Table 10 shows negative associations with categorical fraction and positive associations with numerical excess kurtosis.

Table 10: Adjusted associations between α and TALENT dataset properties. Categorical fraction and numerical excess kurtosis use separate multiple-testing corrections.

Descriptor	Model	Partial ρ [95% CI]	q
Categorical fraction	Taffy-2L	-0.533 $[-0.652, -0.394]$	0.0003
	Taffy-3L	-0.453 $[-0.580, -0.299]$	0.0003
	Taffy-4L	-0.452 $[-0.579, -0.296]$	0.0003
Numerical excess kurtosis	Taffy-2L	0.230 $[0.057, 0.383]$	0.0338
	Taffy-3L	0.326 $[0.167, 0.466]$	0.000703
	Taffy-4L	0.323 $[0.167, 0.463]$	0.000703

C.11.1 Support Rows and Effective Columns

Across TALENT datasets, α is positively associated with support-row count but shows no consistent association with effective column count (Table 11).

Table 11: Unadjusted associations between α and support size on TALENT datasets. All q-values use Benjamini–Hochberg correction over the original 216 tests.

	Support rows		Effective columns	
Model	ρ	q	ρ	q
Taffy-2L	0.192	0.0374	0.138	0.199
Taffy-3L	0.297	2.31×10^{-4}	0.053	0.660
Taffy-4L	0.397	3.44×10^{-7}	0.022	0.881

C.12 Frozen-Checkpoint Gate Intervention

We examine task-specific gating with separately trained two-loop models in Figure 3(b) and a frozen-checkpoint intervention that holds all model parameters fixed.

Starting from a Taffy-4L checkpoint, we rerun the complete inference path under three conditions: the native task-specific coefficient α_S, the checkpoint's support-independent coefficient obtained by setting $b_S = 0$, and $\alpha = 1$, which bypasses the learned modulation and retains the final recurrent update at full strength. All conditions use identical parameters, four passes, preprocessing, ensemble construction, and query batches; only the update coefficient changes. Complete outputs for all three conditions are available on matched TALENT classification datasets, so we compute the comparison without imputing missing conditions. On each dataset, the three conditions are ranked by classification performance, with average ranks assigned to ties, before averaging across datasets.

Table 12: Frozen-checkpoint gate intervention on matched TALENT classification datasets. Mean rank among the three inference conditions; lower is better.

Inference condition	Update coefficient	Average rank ↓
Task-specific gate	α_S	**1.57**
Support-independent gate	$\tanh(a)$	1.88
Gate bypass (full update)	1	2.55

On the matched TALENT datasets, the task-specific gate has the lowest average rank among the three inference conditions (Table 12).

D Limitations

Task conditioning in Taffy uses a single task-level scalar gate applied to the final refinement. Although this keeps the mechanism lightweight, it cannot assign different refinement strengths across examples, features, layers, or earlier passes. Comparing this design with finer-grained gating under matched parameter and compute budgets is an important extension.