THINKING IN DEPTH: RETROSPECTIVE INFERENCE FOR TABULAR FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) are pretrained across diverse tabular tasks and make predictions on a new table at inference time using its labeled examples as context. Most recent TFMs perform such in-context prediction with stacked Transformer layers, repeatedly transforming how examples are represented and compared. By tracing individual queries through several strong TFMs, we find that predictive refinement is highly uneven across depth and is often concentrated in later layers. This uneven refinement motivates us to reconsider how intermediate representations are constructed and reused throughout the network. We introduce RETRO, a tabular foundation model based on *retrospective inference*, where later stages can explicitly revisit and recombine intermediate information produced earlier in the network. RETRO organizes this process around two complementary operations: *which intermediate information to revisit, and how the resulting contextual update should be shaped for each query*. Attention Residuals address the former by adaptively reweighting contributions from different depths, while query-conditioned Gated Attention addresses the latter by modulating the attention output element-wise across representation dimensions. Our analysis shows that RETRO shifts predictive refinement earlier and more broadly across depth, with different stages revising different subsets of queries in a pattern suggestive of multi-view refinement. Across TabArena, TALENT, and RelArena, RETRO ranks among the top three and lies on the Pareto frontier. These results indicate that directly reusing intermediate representations provides a practical way to better exploit depth in TFMs.

1 INTRODUCTION

Tabular foundation models (TFMs) are reshaping tabular learning from training a separate predictor for each dataset toward reusing a pretrained prediction procedure across heterogeneous tables. By pretraining over diverse tasks, TFMs can adapt to a new table at inference time using its labeled examples as context, without requiring a task-specific model to be learned from scratch (Hollmann et al., 2023; 2025; Qu et al., 2025). Recent models have demonstrated increasingly strong performance across a broad range of tabular tasks (Qu et al., 2026; Grinsztajn et al., 2026; Eo et al., 2026), establishing TFMs as a promising general-purpose paradigm for tabular prediction.

In most recent in-context TFMs, labeled support examples and unlabeled queries pass through a stack of Transformer layers before producing the final prediction. Several works have examined what happens along this layered computation. Ye et al. (2025a) show that representations formed within a TFM contain task-conditioned predictive structure and can support downstream prediction,

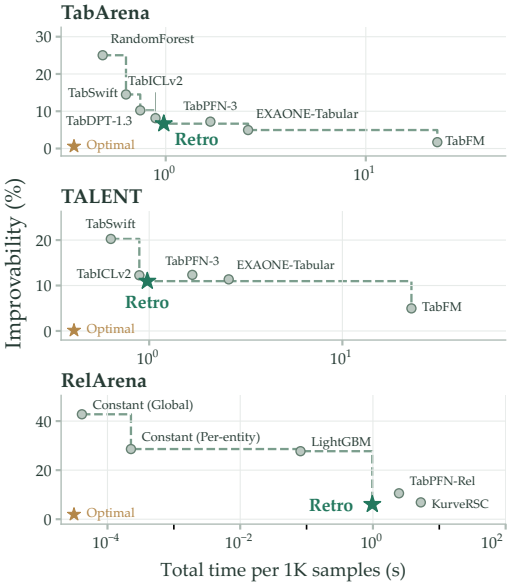

Figure 1: RETRO lies on the Pareto frontier of three mainstream benchmarks.

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.

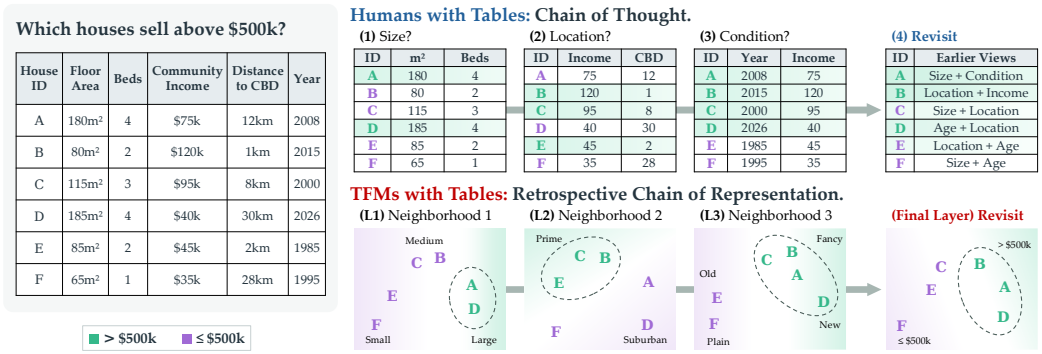

Figure 2: **A housing example of complementary neighborhood views and retrospective integration. Top:** a person considers size, location, and condition, then revisits these assessments to reach a decision. **Bottom:** the TFM schematic places the same homes in different neighborhoods across representations and integrates earlier information in the final representation. This illustrates a possible organization of tabular inference.

while Balef et al. (2026) reveal substantial redundancy and non-uniform contributions across layers through probing and layer interventions. These studies begin to characterize how predictions emerge through the internal computation of TFMs rather than only at their final output.

We take a complementary, sample-level view by tracing how individual queries change as they pass through several strong TFMs. We find that predictive refinement is highly uneven across depth and, in some models, is strongly concentrated in later layers. Different queries are also revised at different stages, revealing a much less uniform process than final aggregate performance alone would suggest. These observations suggest that the issue may not simply be what each layer computes, but also how information produced along the way remains available to later prediction. This motivates a different organization of intermediate computation, in which earlier representations can be reconsidered rather than influencing subsequent stages only through the latest accumulated state.

We call this organization *retrospective inference*. Instead of viewing prediction as a sequence in which each representation simply gives way to the next, retrospective inference keeps intermediate information available as the prediction evolves. The goal is to let later computation selectively draw on representations formed at different stages, so that useful information can be retained and recombined as different queries are progressively refined.

This perspective leads to two coupled design requirements. First, when several intermediate representations remain available, the model must determine which of them are relevant to the current sample. Second, after contextual information is computed from these representations, the model must determine how that update should modify the current query representation. Accordingly, we design RETRO with a retrospective aggregation path that adaptively reweights contributions from different depths, and a query-conditioned gating mechanism that modulates newly computed contextual information across representation dimensions. We instantiate the former with Attention Residuals (Kimi Team, 2026) and the latter with Gated Attention (Qiu et al., 2025). Together, they allow the model to adapt both the historical information it reuses and the way new contextual information is incorporated for each query.

Our analysis shows that RETRO shifts predictive refinement earlier and distributes it more broadly across depth, with different stages revising different subsets of queries in a pattern suggestive of multi-view refinement. Across TabArena (Erickson et al., 2025), TALENT (Ye et al., 2024), and RelArena (Hayler et al., 2026), RETRO ranks among the top three among the evaluated methods, while lying on the estimated performance–cost Pareto frontier of each benchmark. These results highlight the value of treating intermediate representations as reusable resources for subsequent inference. Our contributions are threefold:

- We characterize sample-level predictive dynamics in strong TFMs, complementing prior representation- and layer-level analyses, and show that refinement is highly uneven across depth and often concentrated in later stages.
- We propose *retrospective inference* and instantiate it in RETRO through adaptive reuse of intermediate representations and query-conditioned modulation of contextual updates, leading to earlier and more distributed prediction refinement across stages.

- We evaluate RETRO on three benchmarks spanning tabular classification, regression, and relational prediction, obtaining top-three overall Elo rankings among the evaluated methods.

2 RELATED WORK

Tabular machine learning. Tabular supervised learning predicts a target for each row from a set of heterogeneous feature columns, typically including numerical and categorical variables. Most dataset-specific tabular predictors fall broadly into two families. Tree-based methods, particularly gradient-boosted decision trees, include XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). Deep tabular methods learn task-specific predictors with neural architectures, including FT-Transformer (Gorishniy et al., 2021), RealMLP (Holzmüller et al., 2024), ModernNCA (Ye et al., 2025b), TabM (Gorishniy et al., 2025), and TabPack (Gorishniy et al., 2026). The relative strengths of these methods have been systematically examined in large-scale empirical studies and benchmarks (Grinsztajn et al., 2022; McElfresh et al., 2023; Liu et al., 2025; Erickson et al., 2025). Broader surveys further organize the architectures, learning paradigms, and generalization regimes of tabular learning methods (Borisov et al., 2024; Jiang et al., 2026).

Tabular foundation models. Tabular foundation models (TFMs) move beyond dataset-specific fitting by pretraining a prediction procedure across many tasks and applying it to unseen tables using labeled examples as context (van Breugel & van der Schaar, 2024). TabPFN (Hollmann et al., 2023) established this paradigm with a Transformer that performs in-context prediction over labeled and query examples. Subsequent TFMs mainly differ in how they represent a table and exchange information across features and samples. One family retains fine-grained feature or cell states while exchanging information across feature and sample axes, including TabPFN v2 (Hollmann et al., 2025), LimiX (Zhang et al., 2025a), and EXAONE-Tabular (Eo et al., 2026). Another first aggregates feature information into row representations and then performs row-level in-context learning, including TabICL (Qu et al., 2025), TabICLv2 (Qu et al., 2026), TabPFN-3 (Grinsztajn et al., 2026), and TabFM (Kong & Das, 2026). Efficient direct row-wise modeling provides another design, as exemplified by TabSwift (Liu & Ye, 2026). Orthogonal to architecture, richer synthetic priors and downstream adaptation improve transfer across heterogeneous tasks, as explored by TabForestPFN (den Breejen et al., 2024), Mitra (Zhang et al., 2025b) and Mitra-v2 (Tao et al., 2026). RETRO follows the feature-to-row, row-level ICL hierarchy, but redesigns representation flow within the ICL stack so that intermediate contributions remain directly reusable by later stages.

Recent work has also begun to examine how TFMs arrive at their predictions beyond final predictive accuracy. Ye et al. (2025a) analyze task-conditioned representations in TabPFN v2 and study the predictive utility and reuse of intermediate representations. Balef et al. (2026) investigate layer-wise inference dynamics across multiple TFMs using probing and layer interventions, revealing substantial redundancy across depth. Bilos et al. (2026) use causal interventions to study similarity-based in-context prediction mechanisms across contemporary TFMs. In our work, such analysis serves as a design motivation rather than the endpoint: we trace sample-level prediction changes, observe uneven refinement across depth, and use this phenomenon to motivate retrospective reuse of intermediate computation.

Residual connections. The study of inference across depth also raises questions about how intermediate information is passed between layers. Standard residual connections (He et al., 2016) facilitate optimization through additive updates, allowing earlier computations to contribute to later states. Subsequent designs introduce learned gates (Srivastava et al., 2015) or residual scaling (Bachlechner et al., 2021) to control the propagation of information and improve training. Hyper-Connections (Zhu et al., 2024) extend this design to multiple interacting residual streams, while Manifold-Constrained Hyper-Connections (mHC) (Xie et al., 2025) constrain their mixing matrices to stabilize signal propagation across depth. A complementary direction provides explicit access to earlier computations. DenseNet (Huang et al., 2017) concatenates earlier feature maps, while Attention Residuals (Kimi Team, 2026) use input-dependent weights to aggregate contributions from different depths. Delta Attention Residuals (Luo et al., 2026) further investigate the representations used as sources for this aggregation. In tabular modeling, Xiaomi-TabLDM (Wang et al., 2026) incorporates lightweight Attention Residuals into its architecture. RETRO builds on block Attention Residuals to organize information flow within the ICL stack, and examines how retrospective access relates to sample-level prediction changes across depth.

3 PRELIMINARIES

Tabular in-context learning. Let $S = \{(\mathbf{x}_j, y_j)\}_{j=1}^{n_s}$ be a labeled support set and $Q = \{\mathbf{x}_i^q\}_{i=1}^{n_q}$ a set of unlabeled queries from the same table. The target space is $\mathcal{Y} = \{1, \ldots, K\}$ for classification and $\mathcal{Y} = \mathbb{R}$ for regression. A tabular foundation model (TFM) is pretrained across tasks with different feature schemas and targets. At inference time, it predicts $p_\theta(y \mid \mathbf{x}_i^q, S)$ using observed support labels while keeping its pretrained parameters θ fixed; query labels are withheld. In architectures such as TabICLv2 (Qu et al., 2026), this process consists of two stages: constructing a representation for each row, then performing in-context prediction over these row representations.

Row representations. Following TabICLv2, we use column-wise embedding and row-wise feature aggregation to construct the inputs to the ICL stack. The column encoder produces context-dependent feature representations, and the row encoder aggregates them into a fixed-width vector $h_{0i} \in \mathbb{R}^d$ for each sample. For $n = n_s + n_q$ support and query rows, these vectors form $H_0 \in \mathbb{R}^{n \times d}$. Observed support labels are incorporated into the representations used for ICL, enabling the subsequent layers to condition predictions on the current task.

Multi-layer in-context prediction. An L-block Transformer processes H_0 using the labeled support set as context. Let $H_\ell \in \mathbb{R}^{n \times d}$ collect the row representations after block ℓ, with row i denoted by $h_{\ell i}$. Each block applies attention followed by a feed-forward network, both with residual connections. Before attention, layer normalization produces $Z_\ell = \mathrm{LN}(H_{\ell-1})$, where LN denotes layer normalization applied independently to each row. Let Z_ℓ^S contain the normalized support rows. For clarity, we write the attention computation for a single head and omit layer indices on its parameters:

$$
\begin{aligned}
O_\ell &= \mathrm{softmax}\left(\frac{(Z_\ell W_Q^\top)(Z_\ell^S W_K^\top)^\top}{\sqrt{d_k}}\right) Z_\ell^S W_V^\top, \\
A_\ell &= O_\ell W_O^\top + \mathbf{b}_O^\top.
\end{aligned}
\tag{1}
$$

Here W_Q, W_K, and W_V are learned query, key, and value projections; d_k is the query/key dimension. The softmax normalizes each receiving row's scores over the support rows. W_O and $\mathbf{b}_O$ are the output projection and its bias, shared across rows. In multi-head attention, O_ℓ concatenates the outputs of all heads before this projection. The update A_ℓ is added to $H_{\ell-1}$, followed by a pre-normalized feed-forward residual update. After L blocks, a prediction head maps the final query representations to outputs.

4 HOW DO TFMS REFINE PREDICTIONS ACROSS DEPTH?

Following Ye et al. (2025a), we extract support embeddings in a held-out query role and fit a linear readout on their final-layer representations. Applying the same readout to $h_{\ell i}$ across depth lets us track each query's predictive state and true-class probability margin. We analyze TabICLv2, TabPFN-3, and EXAONE on the first official split of 38 TabArena classification datasets (Erickson et al., 2025). Extraction, margin definitions, and visualization details are given in Appendix A.

We find that predictive refinement is often concentrated in a small portion of the network. In Figure 3 (left), TabICLv2 shows little visible predictive change through much of its depth, followed by a pronounced late transition. Across datasets, the final third accounts for 84.4% of its absolute margin movement, compared with 51.5% for TabPFN-3 and 61.1% for EXAONE. The latter models also exhibit earlier changes (Appendix C.1); the shared finding is an uneven distribution of refinement, not an identical trajectory across architectures.

These observations motivate organizing depth around more differentiated contributions to prediction. Successive representations could emphasize different sample relationships, while keeping their results available for later integration. The diagnostic measures readable predictive progress, rather than proving that earlier representations contain no useful information. Our design question is how to make intermediate results explicitly reusable as the focus of subsequent computation changes.

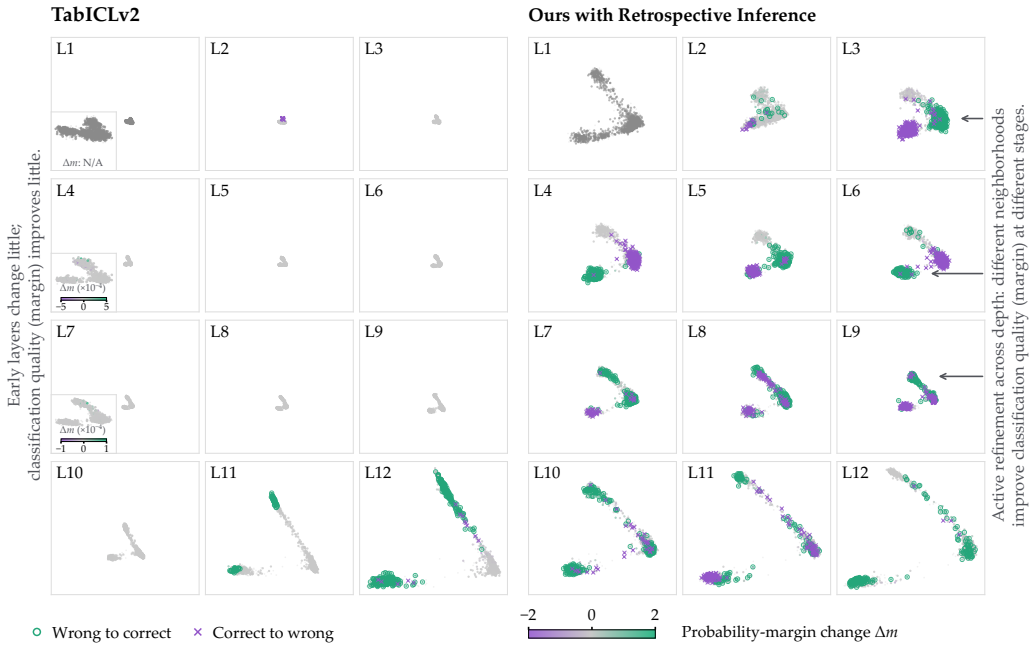

Figure 3: **Layer-wise changes in predictive margin for TabICLv2 and a retrospective variant on SDSS17.** The left and right grids show query representations across all 12 layers of TabICLv2 and our retrospective variant, respectively. Within each model, representations are projected onto a fixed PCA basis fitted to its final-layer support embeddings; coordinates are therefore comparable across layers within a model, but not between models. Color indicates the change in true-class probability margin from the preceding layer (Δm), with green denoting an increase and purple a decrease. Circles mark queries changing from incorrect to correct, and crosses mark the reverse. L1 is shown in gray because no preceding layer is measured. Insets enlarge L1, L4, and L7; the L4 and L7 insets use separately labeled, narrower color scales to reveal small early-layer changes. TabICLv2 exhibits limited directly readable changes through much of its depth, whereas the retrospective variant shows changes for different queries across more stages. Full four-model visualizations and aggregate results over 38 datasets appear in Appendices A.4 and C.1.

5 Retrospective Inference for Tabular Foundation Models

The findings in Section 4 raise a question about representation learning: how does the way intermediate information is reused affect the contributions learned across depth? We introduce *retrospective inference*, which makes earlier contributions directly available to later computation. Attention Residuals implement this reuse, while Gated Attention modulates the contextual updates produced within the resulting stack.

5.1 Retrospective Reuse with Attention Residuals

In a standard residual stack, each sublayer reads a state that accumulates preceding updates. An early contribution can influence subsequent prediction, but only as part of this evolving state; it is no longer available as a separate source that later layers can selectively weight. Consequently, its predictive value depends on how subsequent transformations retain and use it. This organization permits layer-wise specialization, but does not explicitly support selecting an earlier contribution independently of those accumulated after it.

Retrospective inference changes this relationship between intermediate computation and prediction. By retaining earlier contributions as separate sources, it allows a stage to provide information that remains useful when selected later, even if that information is insufficient for prediction on its own. We hypothesize that this reuse pathway favors complementary contributions across depth: different stages can refine different sample relationships without requiring their results to remain separately recoverable from a single accumulated state. For tabular ICL, this means that a representation useful

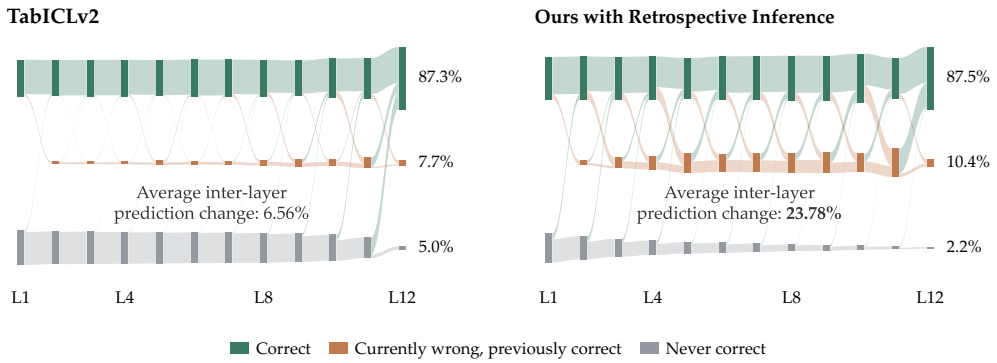

Figure 4: **Sample-state flows across depth.** Frozen-readout trajectories for TabICLv2 (left) and our ungated retrospective variant (right), averaged equally over 38 classification datasets. *Correct* merges always-correct and previously-wrong but currently-correct queries; the two wrong states distinguish whether a query was ever correct. Ribbon widths show query fractions across all 12 layers. The displayed average measures correctness switches over L1→L2 through L10→L11; only this summary excludes the final transition.

for comparing some queries with their support examples can remain accessible while later stages refine other comparisons. The objective is useful complementarity, rather than representational dissimilarity alone.

We realize this access through block Attention Residuals (Kimi Team, 2026). The source bank contains the initial row encoding, sums of residual updates from completed two-layer groups, and the current group's partial sum. At reading site t, let $\mathcal{B}_t$ denote the available sources and b_i the representation of row i in source b. A learned vector w_t produces row-dependent weights:

$$\alpha_{tbi} = \frac{\exp(\langle w_t, \mathrm{RMSNorm}_t(b_i)\rangle)}{\sum_{b' \in \mathcal{B}_t} \exp(\langle w_t, \mathrm{RMSNorm}_t(b'_i)\rangle)}, \qquad \widetilde{h}_{ti} = \sum_{b \in \mathcal{B}_t} \alpha_{tbi} b_i. \tag{2}$$

The scorer is shared across rows but learned separately at each reading site. Weights are recomputed from each row's candidate contributions. RMS normalization applies only to scoring; the values remain unnormalized. Independent readers precede every feed-forward sublayer and every attention sublayer after the first. Sublayer outputs accumulate within the current group, and completed groups remain available to later readers. The final reader combines the initial encoding and completed groups before prediction. This retains grouped update contributions rather than every full hidden state; Appendix B gives the recurrence.

Representation dynamics. We examine the resulting trajectories with the fixed final-support readout used in Section 4. In Figure 3 (right), predictive changes appear earlier and involve different queries at successive stages. The aggregate state flows in Figure 4 show revisions distributed across depth, including both corrections and regressions. The retrospective model in this aggregate comparison does not use gating.

These observations are consistent with a less concentrated organization of predictive refinement, in which successive representations contribute changes for different subsets of queries. The observed variation across queries and stages is suggestive of multi-view refinement, without imposing predefined views or semantic roles on individual layers. Neither increased state turnover nor the cross-checkpoint comparison alone establishes that historical access causes greater complementarity. The measured changes concern predictions under a shared readout, not all information available in each representation. Additional model comparisons, a regression case, and an additional classification case are reported in Figures 13 and 14 and Appendices C.4 and C.5.

5.2 Shaping Contextual Updates with Gated Attention

Historical reads determine the input to a sublayer; its output becomes a new contribution available for later reuse. We use Gated Attention (Qiu et al., 2025) to modulate this contextual update for each receiving row. Let $z_{\ell i}$ be the normalized attention-reader output and $o_{\ell i}$ the concatenated

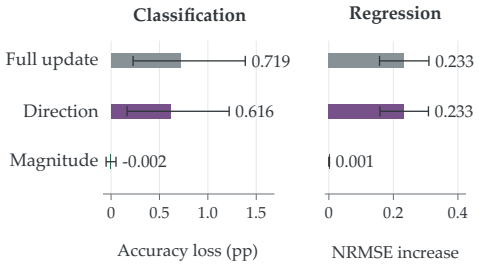

Figure 5: **Gated-update geometry.** Mean-gate replacement effects on 38 classification and 13 regression datasets in TabArena. Whiskers: 95% dataset-bootstrap intervals.

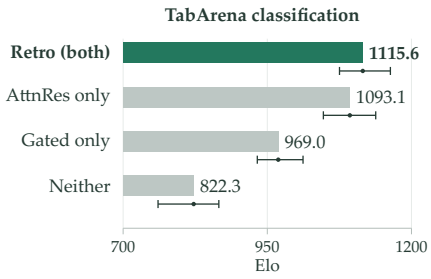

Figure 6: **Component ablation.** TabArena classification Elo for the four component configurations. Whiskers: 95% confidence intervals.

attention-head outputs. Following the row-vector convention of Equation (1),

$$g_{\ell i} = \sigma(z_{\ell i} W_{g,\ell}^{\top} + \mathbf{b}_{g,\ell}^{\top}),$$
$$A_{\ell i} = (g_{\ell i} \odot o_{\ell i}) W_{O,\ell}^{\top} + \mathbf{b}_{O,\ell}^{\top}. \tag{3}$$

Here σ is the sigmoid. Each layer learns a gate projection shared across support and query rows. Ordinary attention is already query-dependent; gating additionally modulates its output channels before projection. It can therefore alter both the direction and magnitude of the update, rather than only its overall strength. The gated update enters the current group's accumulation, shaping the information retained for subsequent reads.

Direction and magnitude. We analyze fixed gated models on 38 classification and 13 regression datasets, using the first official split, one view without ensembling, and native final prediction heads. A counterpart update replaces each query's gate with the channelwise mean over the unlabeled query batch. We substitute either this full update, its direction while retaining the learned norm, or its norm while retaining the learned direction. Replacements occur after output projection at every ICL layer, on query rows only. All normalization layers remain in place; the updates are recomputed at each intervened state (Appendix D.1).

Replacing direction reduces mean classification accuracy by 0.616 percentage points, compared with -0.002 for replacing magnitude (Figure 5); the paired difference is 0.618 points (95% interval: [0.173, 1.217]). The corresponding regression NRMSE increases are 0.2328 and 0.0011, with NRMSE defined as query RMSE divided by the support-target standard deviation. Thus, under these replacements, the trained models depend more on the learned update direction than on its norm. This supports a role for gating in shaping contextual information, beyond attenuating its strength. The experiment measures fixed-model dependence, not the training benefit of gating; replacement distances are unmatched, and normalization may contribute to the observed scale tolerance. Dataset-level variation and further controls appear in Appendix D.

6 RETRO: MODEL, TRAINING, AND EVALUATION

We combine retrospective access and selective contextual incorporation in RETRO, a tabular foundation model built on a hierarchical table encoder and a row-level ICL stack. We describe its architecture and pretraining procedure, then evaluate its predictive performance, component contributions, and computational cost.

Architecture. As illustrated in Figure 7, the table encoder consists of three column-processing blocks and three row-processing blocks, both with a hidden dimension of 128 and eight attention heads. The column stage uses 128 inducing vectors, while the row stage aggregates feature representations through four CLS tokens. Their outputs are concatenated into a 512-dimensional representation for each row. We retain TabICLv2's repeated feature grouping, target-aware embeddings, and query-aware scalable softmax (QASSMax). The row representations, augmented with observed support labels, enter a 12-layer ICL stack with eight attention heads and a feed-forward expansion ratio of two. Within this stack, retrospective aggregation constructs the inputs to attention and feed-forward sublayers, and gated attention modulates contextual outputs before projection (Equations (2) and (3)).

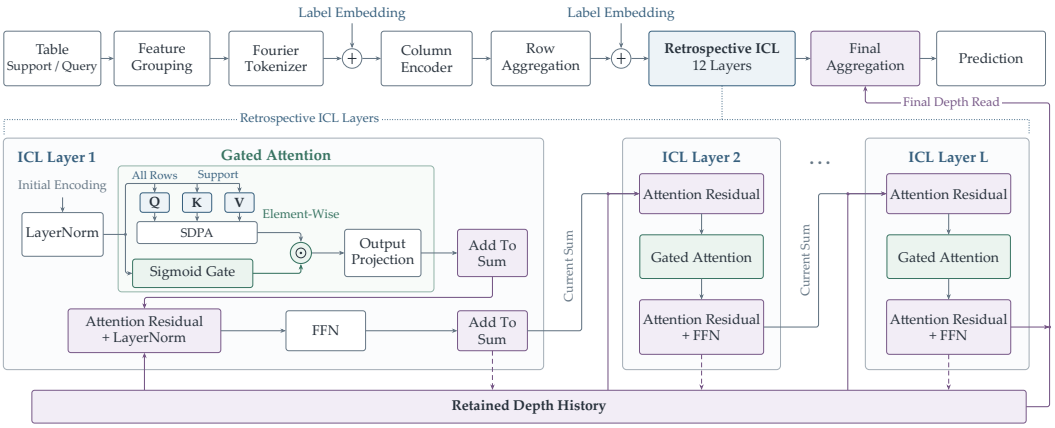

Figure 7: **The RETRO architecture.** A table encoder produces row representations for a retrospective ICL stack. Within each layer, separate aggregation modules construct attention and feed-forward inputs from available historical contributions. An input-conditioned gate modulates attention outputs before projection. Completed two-layer update groups remain available to subsequent layers and the final aggregation used for prediction.

Residual updates are grouped over pairs of layers. The final aggregation combines the initial encoding and six completed update groups, and a two-layer MLP with hidden dimension 1,024 produces query predictions. Following TabICLv2, classification and regression use separately trained models with task-specific target embeddings and output heads.

Pretraining. We use the synthetic task generator of TabICLv2, which samples structural causal models with diverse graph structures and functional relationships to generate heterogeneous numerical and categorical features. Each task is divided into labeled support examples and held-out queries. The classification model learns to predict query class labels, while the regression model predicts 999 quantiles at levels $\{0.001, 0.002, \ldots, 0.999\}$ using the summed pinball loss. Pretraining follows a three-stage curriculum that progressively increases the number of samples per task, with up to 100 features and a batch size of 64 tasks. The first stage uses 500K updates on tasks with 1,024 samples, allocating 30–90% to the support set. The second and third stages use 40K and 10K updates, respectively, with task sizes sampled log-uniformly from 400–10,240 and 400–60,000 samples and an 80% support fraction. We use Muon for matrix-valued parameters and AdamW for one-dimensional parameters, following the optimization setup of TabICLv2. We adopt cosine learning-rate schedules with peak rates of 8×10^{-4}, 10^{-4}, and 2×10^{-5} across the three stages, and a weight decay of 0.01. Further details of synthetic data generation, filtering, and the remaining optimization settings follow Qu et al. (2026).

Evaluation protocol and predictive performance. At inference time, pretrained parameters remain fixed and labeled support examples provide task-specific context. Classification predictions are obtained from the classification head; regression point predictions are computed by averaging the predicted quantiles. We retain the preprocessing and inference procedures of TabICLv2, with benchmark-specific configurations reported in Appendix E.

We evaluate on TabArena, TALENT, and RelArena using the comparison pools listed in Appendix E. All results use the trained prediction heads rather than the auxiliary probes used for the layer-wise analysis. We report separate classification and regression Elo rankings, together with overall ratings recomputed from pooled results across both task types. Overall Elo ratings are recomputed from pooled classification and regression results rather than averaged across the two task types.

Figure 8 shows the five highest-rated methods in each ranking. Among the evaluated methods, RETRO ranks third on TabArena, second on TALENT, and first on RelArena by overall Elo. It also ranks above TabICLv2 in the overall rankings on TabArena and TALENT, indicating competitive performance relative to the backbone on which our architecture is based.

The task-specific rankings reveal variation across evaluation settings. On TabArena, RETRO ranks fourth for classification and second for regression. On TALENT, it ranks second for both task types, although its overall Elo is close to those of several competing TFMs. On RelArena, it ranks first

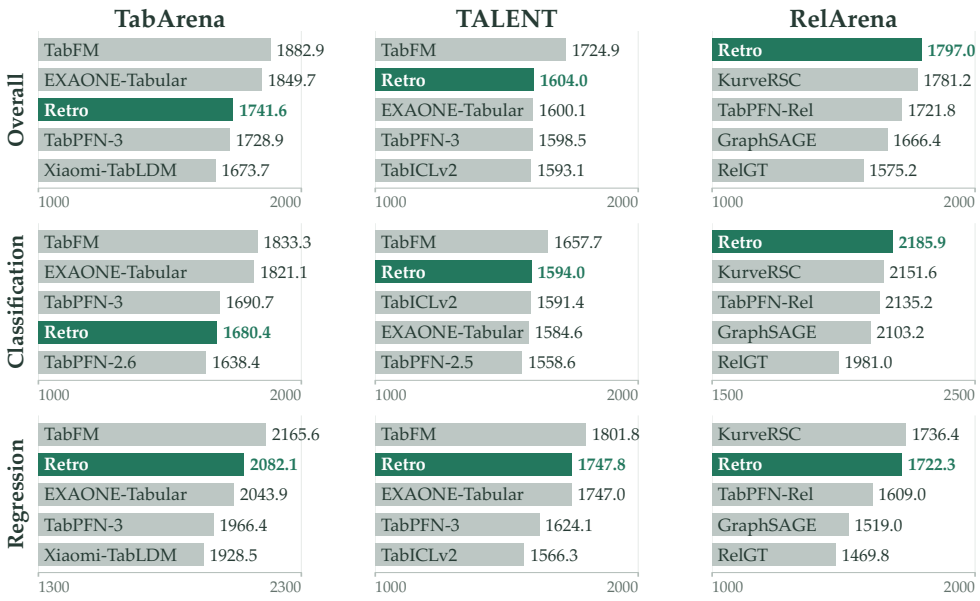

Figure 8: **Elo rankings across three benchmarks.** Each panel shows the five highest-rated methods. Candidate sets and scoring protocols differ, so ratings should be compared within panels. Complete rankings appear in Appendix E.

for classification and second for regression. These results establish competitive performance across classification and regression, while the relative advantages depend on the benchmark. Complete rankings and uncertainty estimates are provided in Appendix E.

Component ablation. On TabArena classification (Figure 6), the combined model has the highest Elo point estimate (1115.6) in this ablation, while both single-mechanism variants score above the model with neither.

Efficiency. RETRO retains the backbone's width and depth, adding only historical aggregation modules and gate projections within the ICL stack. The aggregation modules introduce 24,576 parameters. With a d-dimensional gate in each of the L layers, gate projections add $L(d^2 + d)$ parameters, approximately 3.15M for $L = 12$ and $d = 512$. For n rows and B available historical sources, each aggregation requires $\mathcal{O}(Bnd)$ computation, while each gate projection requires $\mathcal{O}(nd^2)$. These operations complement the existing $\mathcal{O}(nn_sd)$ support-conditioned attention and $\mathcal{O}(nd^2)$ feed-forward computation, where n_s is the support-set size. The performance–runtime trade-off in Figure 1 shows that RETRO improves predictive performance over TabICLv2 with nearly unchanged inference time. Relative to the other evaluated methods, it lies on the Pareto frontier: no compared method achieves both better predictive performance and lower runtime under the reported evaluation setting. This positions RETRO as a competitive choice when both predictive quality and inference cost matter.

7 DISCUSSION AND CONCLUSION

We studied how individual queries are refined across the depth of contemporary TFMs and found that predictive changes are uneven, often concentrating in later layers. This observation motivates *retrospective inference*: retaining access to earlier contributions while adapting how newly computed contextual information modifies each query. RETRO instantiates these two operations through Attention Residuals and query-conditioned, element-wise Gated Attention.

Retrospective variants exhibit earlier and more distributed prediction changes, with different stages revising different subsets of queries. Fixed-weight gating interventions further reveal sensitivity to update direction beyond changes in magnitude alone. Native-prediction comparisons place RETRO among the top three methods by overall Elo on TabArena, TALENT, and RelArena within the evaluated candidate sets. Together, these findings support organizing intermediate computation as reusable information rather than only as transient states passed from layer to layer.

Reproducibility Statement

We describe the model architecture, pretraining setup, and evaluation protocols, with additional implementation details in the appendices. For the layer-wise analyses, we specify representation extraction, probe fitting, and prediction readout procedures. For the intervention experiments, we define the modified quantities and controls, and distinguish fixed-parameter interventions from comparisons between separately trained models. Settings inherited from TabICLv2 are identified and referenced explicitly.

We plan to release the implementation, model and training configurations, checkpoint identifiers, and benchmark predictions to support reproduction of the reported results.

Ethics Statement

This work uses existing benchmark datasets and does not involve new data collection or human-subject experiments. Our use and redistribution of datasets and associated artifacts are subject to their respective licenses and access restrictions.

Tabular prediction models may be used in consequential applications, where aggregate benchmark performance does not establish fairness, robustness, or suitability for deployment. Such applications require evaluation of subgroup performance, potential biases, and domain-specific risks. Likewise, the layer-wise visualizations and interventions in this work provide limited evidence about model behavior; they do not constitute complete explanations of individual predictions. We use retrospective inference to describe a computational mechanism for reusing intermediate representations, without implying human-like reasoning or deliberation.

Statement on the Use of AI Tools

Generative AI tools assisted with literature retrieval and synthesis, conceptual framing, hypothesis refinement, mathematical formulation, interpretation of experimental results, code inspection, preparation of figure-generation scripts, and manuscript drafting and editing. They also assisted with checks of scientific claims, references, and supporting artifacts. The reported numerical results were computed from experimental outputs rather than generated by AI as hypothetical observations.

The authors are responsible for verifying the accuracy of the scientific claims, references, mathematical formulations, code, and reported results, including all AI-assisted content, and retain full responsibility for the submitted work.

References

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Gary Cottrell, and Julian J. McAuley. Rezero is all you need: fast convergence at large depth. In *UAI*, 2021.

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? understanding inference dynamics in tabular foundation models. In *ICML*, 2026.

Marin Bilos, James T. Wilson, Anderson Schneider, and Yuriy Nevmyvaka. A mechanistic study of tabular foundation models. In *ICML*, 2026.

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

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, 2016.

Felix den Breejen, Sangmin Bae, Stephen Cha, and Se-Young Yun. Why in-context learning transformers are tabular data classifiers. *CoRR*, abs/2405.13396, 2024.

Moonjung Eo, Min-Kook Suh, Hye-Seung Cho, Jiwon Kim, Seoyoon Kim, Sangjun Nam, and Soonyoung Lee. EXAONE Tabular 1.0: Technical report. *CoRR*, abs/2608.25774, 2026.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. TabArena: A living benchmark for machine learning on tabular data. In *NeurIPS*, 2025.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, 2021.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *ICLR*, 2025.

Yury Gorishniy, Akim Kotelnikov, Ivan Rubachev, and Artem Babenko. Tabpack: Efficient hyperparameter ensembles for tabular deep learning. In *ICML*, 2026.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? In *NeurIPS*, 2022.

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. *CoRR*, abs/2605.13986, 2026.

Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, Jonas M. Kübler, Siyuan Guo, Oscar Key, Jan Hendrik Metzen, Rylee Grace, David Salinas, Arthur Cahu, Simon Bing, Benjamin Jäger, Tuana Çelik, Mihir Manium, Vitor Monteiro, Jake Robertson, Jerry Chen, Eliott Kalfon, Tomás Pereda, Lilly Charlotte Wehrhahn, Dominik Safaric, Tobias Schroeder, Georg Grab, Diana Kriuchkova, Clara Cornu, Philipp Singer, Nick Erickson, Vahid Balazadeh, Marie Salmon, Simone Alessi, Kürşat Kaya, Philipp Jund, Léo Grinsztajn, Yann LeCun, Bernhard Schölkopf, Madelon Hulsebos, Lennart Purucker, Sauraj Gambhir, Frank Hutter, and Noah Hollmann. Advancing open and reproducible relational learning: Relarena-α, tabpfn-rel and RPI. *CoRR*, abs/2608.16319, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *ICLR*, 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(8044):319–326, 2025.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. In *NeurIPS*, 2024.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Jun-Peng Jiang, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and Han-Jia Ye. Representation learning for tabular data: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(6):6488–6508, 2026.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

Kimi Team. Attention residuals. *CoRR*, abs/2603.15031, 2026.

Weihao Kong and Abhimanyu Das. Introducing TabFM: A zero-shot foundation model for tabular data. Google Research Blog, 2026.

Si-Yang Liu and Han-Jia Ye. TabSwift: An efficient tabular foundation model with row-wise attention. In *ICML*, 2026.

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.

Cheng Luo, Zefan Cai, and Junjie Hu. Delta attention residuals. *CoRR*, abs/2605.18855, 2026.

Duncan C. 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 *NeurIPS*, 2023.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. In *NeurIPS*, 2018.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In *NeurIPS*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.

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

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

Yefan Tao, Xiyuan Zhang, Xinyi Liu, Boran Han, Danielle Maddix, Haoyang Fang, Zhen Han, Jiading Gai, Xuanqing Liu, Michael Bohlke-Schneider, Yuyang Wang, Gerald Friedland, Kevan Mah, Chris Lee, and Chris Kong. Mitra-v2 technical report. *CoRR*, abs/2609.04540, 2026.

Boris van Breugel and Mihaela van der Schaar. Position: Why tabular foundation models should be a research priority. In *ICML*, 2024.

Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Zhenwei Xin, Ping Hou, Jie Yu, Chunxiao Liu, Erli Meng, and Bin Wang. Xiaomi-TabLDM: A tabular foundation model technical report. *CoRR*, abs/2609.03880, 2026.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections. *CoRR*, abs/2512.24880, 2025.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning methods on tabular datasets. *CoRR*, abs/2407.00956, 2024.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *NeurIPS*, 2025a.

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

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. *CoRR*, abs/2509.03505, 2025a.

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 Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *NeurIPS*, 2025b.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. *CoRR*, abs/2409.19606, 2024.

A LAYER-WISE REPRESENTATION ANALYSIS

This appendix specifies the diagnostic used in Sections 4 and 5 and reports aggregate depth statistics. Appendix B describes the retrospective computation, and Appendix C presents the full-depth cases. The fixed-readout analysis here is distinct from the native-head gating interventions in Appendix D and the benchmark evaluations in Appendix E.

A.1 DATA AND QUERY-ROLE REPRESENTATIONS

We analyze the first official repeat/fold ($r0/f0$) of each of the 38 TabArena classification datasets (Erickson et al., 2025). Models use the same outer query rows, a single preprocessing view, and no ensemble. Inference is performed in FP32 with TF32 disabled.

Following Ye et al. (2025a), support representations are obtained by cross-fitting within the outer support set. Each held-out support fold is encoded as unlabeled queries conditioned on the remaining support rows. The number of stratified folds is $\min(10, n_{\min})$, where $n_{\min}$ is the smallest support-class count; the random seed is 42. Outer queries are encoded against the full outer support set. Their labels enter neither representation extraction nor readout fitting. Cross-fitting removes the support row's own label from its encoding, although the cross-fitted and outer-query contexts have different sizes.

Capture points. For TabICLv2, each intermediate representation is the input to the next block's pre-attention normalization, with the final state taken from the encoder output. For the retrospective variants, this point includes the next attention reader; the final state includes the output reader. It is therefore the accessible mixture, not the raw update accumulator. Both sequences contain 12 states of width 512. TabPFN-3 supplies 24 ICL block outputs of width 512 before final RMS normalization. EXAONE-Tabular supplies 12 target-cell CAST outputs of width 192 after block normalization. A depth index denotes a native block, not equal computation across models.

Model variants. The retrospective representation analysis in Figure 3 and Table 1 uses an ungated checkpoint from the end of the third training stage (10,000 updates). A separate ungated stage-2 classifier is used for the state-flow comparisons in Figures 4, 13 and 14 and the additional classification case in Appendix C.5. The regression case in Appendix C.4 uses its ungated stage-2 regression counterpart. These stage-2 models average weights from updates 39,901–40,000. Gated models are evaluated separately in Appendix D.1.

The representation-analysis checkpoint and TabICLv2 have the same backbone dimensions: width 128 in three column and three row blocks, four row CLS tokens, and 12 ICL blocks of width 512 with eight heads. Shared pretraining data, optimization, and initialization are not established. The comparison describes representation dynamics rather than a training-controlled estimate of the effect of historical access.

A.2 FIXED READOUTS AND PREDICTIVE CHANGES

A StandardScaler and L2-regularized logistic classifier are fitted to the final-layer cross-fitted support representations. We use $C = 1$ and LBFGS with tolerance 10^{-6}, allowing up to 20,000 iterations when the initial 3,000-iteration limit is insufficient for convergence. The scaler and classifier remain fixed across depth.

Let r_L denote this readout and $p_{\ell i} = r_L(h_{\ell i})$. For query i with true class y_i^q, define

$$m_{\ell i} = p_{\ell i, y_i^q} - \max_{c \neq y_i^q} p_{\ell i, c}, \qquad c_{\ell i} = \mathbf{1}\left[\arg\max_c p_{\ell i, c} = y_i^q\right]. \tag{4}$$

The probability margin and correctness indicator provide a common decision reference across depth. They are not predictions from the pretrained model's native head.

Transitions. At layer $\ell > 1$, correction and deterioration fractions are $C_\ell = \mathbb{E}_i[(1 - c_{\ell-1,i})c_{\ell i}]$ and $D_\ell = \mathbb{E}_i[c_{\ell-1,i}(1 - c_{\ell i})]$. Writing $a_\ell = \mathbb{E}_i[c_{\ell i}]$ gives

$$a_\ell - a_{\ell-1} = C_\ell - D_\ell, \qquad R_\ell = C_\ell + D_\ell, \qquad V_\ell = \mathbb{E}_i|m_{\ell i} - m_{\ell-1,i}|. \tag{5}$$

Model	Any revision (%)	Revision / step (%)	$\mathbb{E}\lvert\Delta m\rvert$	Early motion (%)
TabICLv2	63.8	9.82	0.197	15.6
TabPFN-3	66.8	7.55	0.149	48.5
EXAONE-Tabular	71.8	16.12	0.319	38.9
RETRO (ours)	**79.5**	**24.66**	**0.488**	**60.7**

Table 1: **Descriptive frozen-readout dynamics, equally averaged over 38 datasets (one official split each).** Bold marks the largest value in each column, not a performance ranking. A revision is a change in correctness, not a claim of useful reasoning. Early motion is the share of absolute probability-margin change before the final third of depth. TabPFN-3 has 24 layers; the other models have 12. Native-step averages complement the depth-dependent any-revision statistic, but do not equate compute or depth granularity across architectures.

A correctness switch does not include a change between two wrong classes. The early-motion share is $\sum_{\ell=2}^{\lfloor 2L/3 \rfloor} V_\ell / \sum_{\ell=2}^{L} V_\ell$; its complement is the final-third share in Section 4. The early range contains seven of eleven transitions for 12-layer models and fifteen of twenty-three for TabPFN-3. All query rows contribute to each dataset's statistics, followed by an equal-weight average across datasets.

State-flow plots use three histories: *Correct*, *Currently wrong, previously correct*, and *Never correct*. The second refers to any earlier correct state, not only the preceding layer; the third includes the current layer. The two currently-correct histories of the underlying four-state partition are merged by summing their node counts and transition counts. Every query and native layer is retained. The scalar summary printed on the flow plots averages R_ℓ over $\ell = 2, \ldots, L-1$, excluding the final transition. In contrast, the per-step statistics in Table 1 include all $L-1$ transitions.

A.3 VISUALIZATION AND REGRESSION

For each model and dataset, PCA is fitted to at most 1,200 deterministically selected final-layer support embeddings. Up to 1,200 query rows are shown. Every layer uses the same basis and axis limits within a model; coordinates are not aligned across models. Readouts operate in the original embedding space, so the plots are not two-dimensional decision surfaces.

In classification plots, green and purple denote positive and negative adjacent margin changes. Rings mark corrections and crosses mark deteriorations. The first measured layer is gray because it has no measured predecessor. The main SDSS17 figure includes separately labeled inset scales to resolve small early-layer changes. The remaining classification plates share the same main color scale.

Regression. The superconductivity case in Appendix C.4 contains 14,175 support and 7,088 query rows. Support extraction uses shuffled ten-fold cross-fitting with seed 42. A StandardScaler and Ridge regressor ($\alpha = 1$, with intercept) are fitted to final-layer cross-fitted support representations and frozen across depth. Colors encode

$$\delta e_{\ell i} = \frac{|\widehat{y}_{\ell-1,i} - y_i^q| - |\widehat{y}_{\ell i} - y_i^q|}{s_S}, \tag{6}$$

where s_S is the outer-support target standard deviation. Positive values indicate error reduction. The four models share a symmetric-log scale spanning $[-10, 10]$; no classification-state partition is assigned to regression.

A.4 QUANTITATIVE DEPTH DIAGNOSTICS

Table 1 summarizes the representation-analysis checkpoint specified in Appendix A.1. Mean correctness turnover per transition is 24.66% for the retrospective variant and 9.82% for TabICLv2; 79.5% and 63.8% of queries, respectively, change correctness at least once. The early-motion shares are 60.7% and 15.6%, with the retrospective share larger on 36 of 38 datasets. Turnover and absolute margin movement are larger on 33 of 38 datasets. Final frozen-readout accuracies are 87.24% and 87.28%, respectively. Thus, greater activity across depth is not itself a gain in final accuracy.

These all-transition statistics should not be equated with the scalar summaries in Figure 14: those summaries omit the final transition and use the separate stage-2 retrospective checkpoint. The case

studies in Appendix C complement both summaries by locating changes within individual query trajectories.

A.5 Interpretation

The diagnostic measures compatibility with a fixed final-support readout, not all predictive information contained in each layer. A representation change in the readout's nullspace leaves its logits unchanged; conversely, an invertible coordinate change can preserve information while changing the frozen readout's predictions. PCA can suppress further distinctions. The observed trajectories therefore support descriptive comparisons of refinement, not claims that early layers contain no useful information or that each revision improves the final prediction. The displayed cases were selected by visual inspection and are exploratory; the aggregate analysis uses all 38 classification datasets.

B Retrospective ICL Implementation

This appendix specifies the grouped accumulation underlying Equations (2) and (3) and the architecture in Figure 7. The table encoder and pretraining procedure are described in Section 6.

B.1 Historical reads and grouped updates

Let P be the current group accumulator and M the ordered bank of completed sources. Initially $P = H_0$ and M is empty. For layer ℓ, the attention input is

$$U_\ell = \begin{cases} P, & \ell = 1, \\ \mathrm{Read}_{a,\ell}(M, P), & \ell > 1, \end{cases} \tag{7}$$

where each Read applies Equation (2) independently to every row. At the start of a two-layer group ($\ell = 1, 3, \ldots, 11$), this read is followed by appending P to M and resetting P to zero. Thus, the initial encoding becomes the first stored source. Attention and feed-forward updates then proceed as

$$\begin{aligned} P &\leftarrow P + A_\ell(\mathrm{LN}(U_\ell); S), \\ V_\ell &= \mathrm{Read}_{f,\ell}(M, P), \\ P &\leftarrow P + \mathrm{FFN}_\ell(\mathrm{LN}(V_\ell)). \end{aligned} \tag{8}$$

Here A_ℓ includes the attention output projection and, when enabled, the gate in Equation (3). Both support and query rows receive updates, but only support rows supply attention keys and values. The attention read occurs before a group is stored; the feed-forward read occurs after the attention update is added.

After the last layer, an independent output reader combines (M, P). For 12 layers grouped in pairs, it reads seven sources: the initial encoding and six group sums. The capture points in Appendix A.1 include these learned reads rather than only the accumulator.

B.2 Normalization, initialization, and gating

Each reader has its own length-d scoring vector and RMS gain, shared across rows. The scorer is initialized to zero and the gain to one, giving uniform source weights initially. RMS normalization, scoring, softmax, and weighted reduction are evaluated in FP32; the result is cast to the input dtype. Only scoring uses normalized sources. The attention and feed-forward sublayers retain their pre-LayerNorms.

The scorer is a learned site-specific vector, not a projection of the latest state. Consequently, weights vary with the row's source values and the reader's parameters, without inheriting a previous reader's weights. Gating is computed from the normalized attention input and multiplies the concatenated head outputs before their output projection. It modulates latent channels but does not skip attention or feed-forward computation. Its experimental role is examined in Appendix D.

There are $(L-1)+L+1 = 2L$ historical readers, adding $4Ld$ parameters, or 24,576 at $L = 12, d = 512$. The L gate projections add $L(d^2 + d)$ parameters. These counts cover the additional modules, not the total model size or measured runtime.

C Full-Depth Representation and State-Flow Analyses

The following case studies extend Sections 4 and 5 using the extraction and readout protocol in Appendices A.1 and A.2. We first compare full-depth SDSS17 representations and state flows, then report benchmark-average flows and additional regression and classification cases. All measured layers are retained. Within each model, every layer uses the final-support PCA basis; neither coordinates nor computation per layer are aligned across models. The cases are exploratory illustrations, while Appendix A.4 reports aggregate classification statistics.

C.1 SDSS17: full-depth representations

Figures 9–12 extend Figure 3 to all four models on the same SDSS17 split. TabPFN-3 has 24 layers; the other models have 12. The fixed readouts reveal when and where predictions change without refitting a classifier at each layer. Model variants and capture points are specified in Appendix A.1.

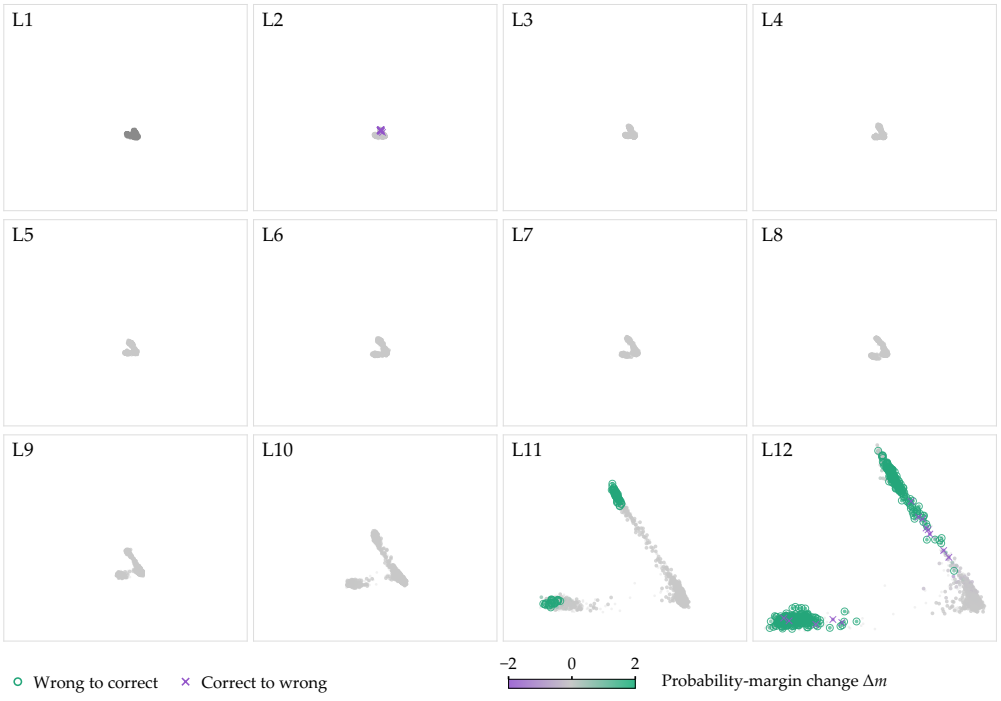

Figure 9: **TabICLv2 on SDSS17.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

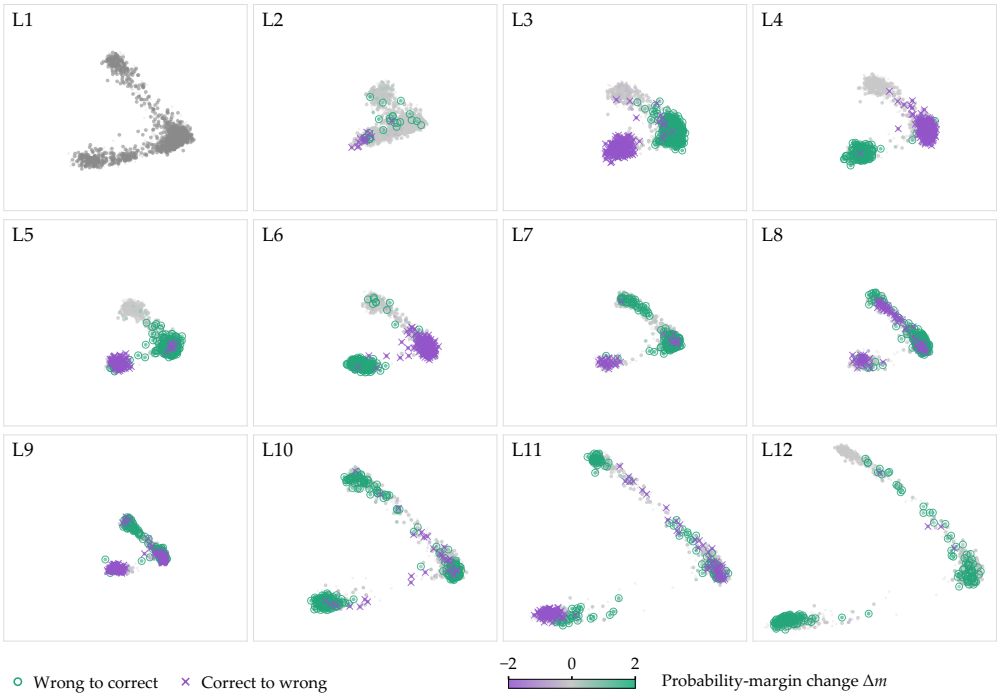

Figure 10: **Our retrospective model on SDSS17.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

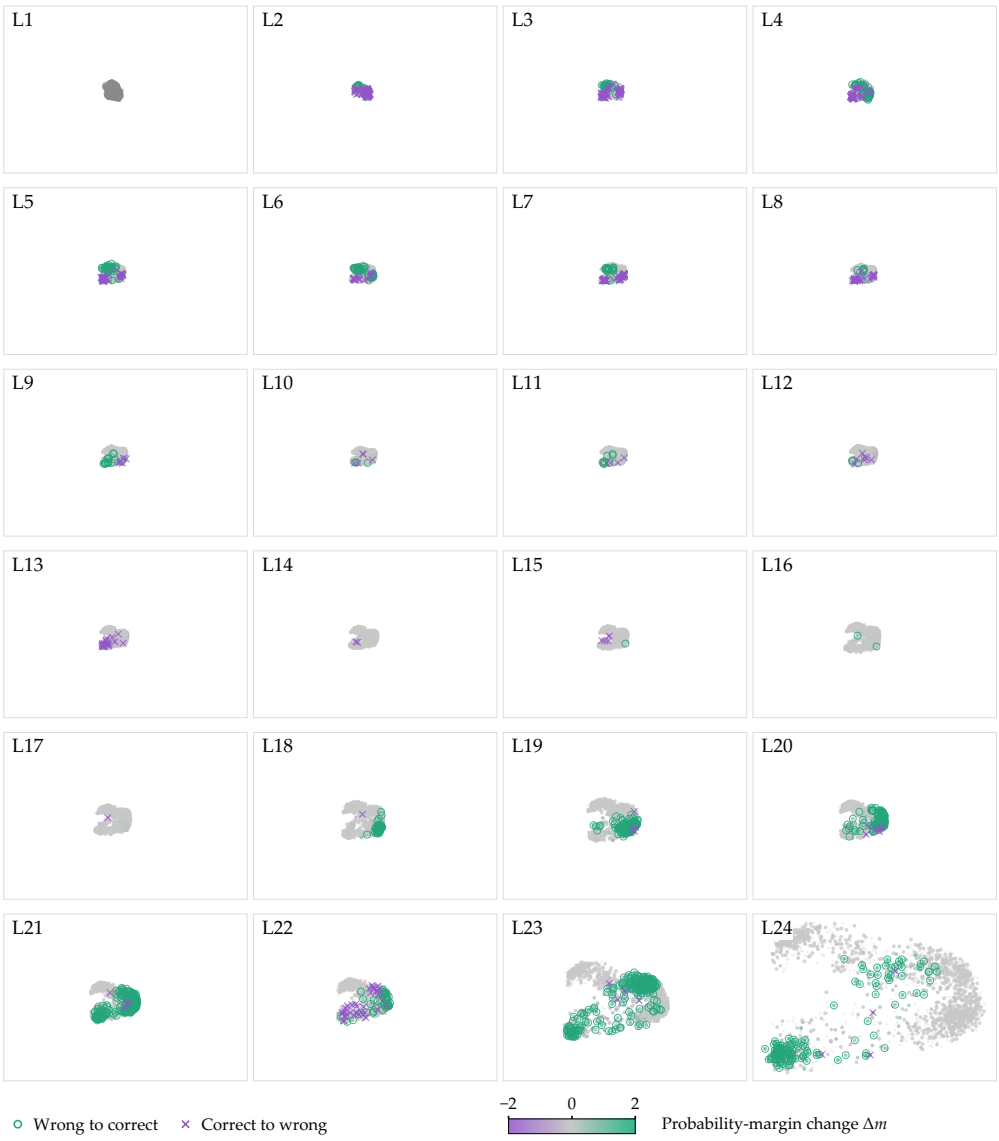

Figure 11: **TabPFN-3 on SDSS17.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

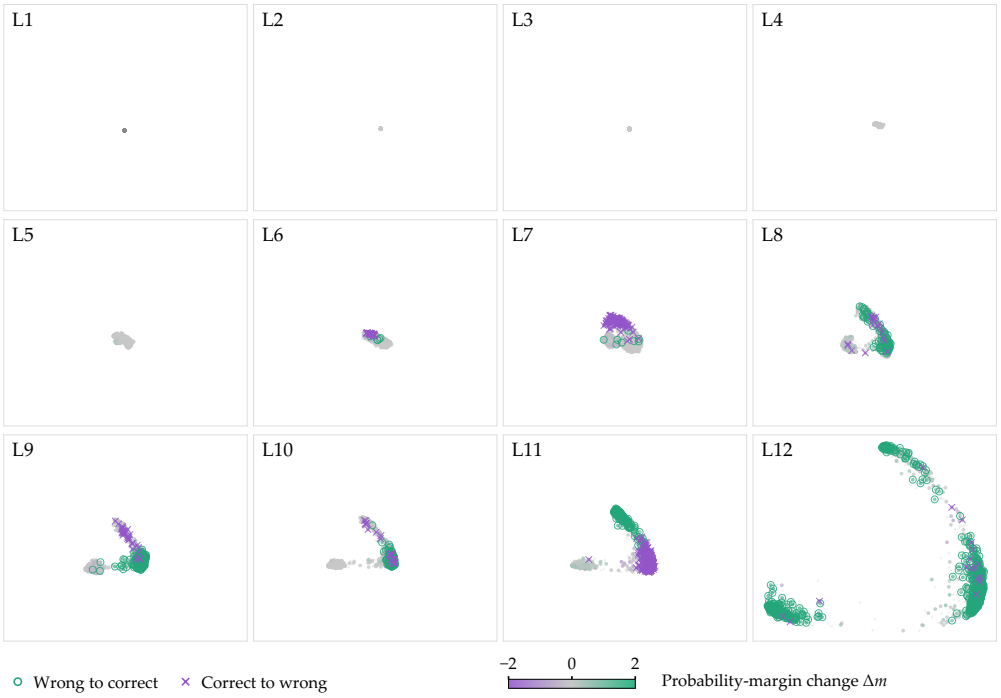

Figure 12: **EXAONE-Tabular on SDSS17.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

C.2 SDSS17: SAMPLE-STATE FLOWS

Figure 13 tracks all 26,018 SDSS17 queries using the three-state partition in Appendix A.2. Unlike the sampled PCA displays, these flows retain every query. The retrospective flow uses the stage-2 classifier specified in Appendix A.1, rather than the representation-analysis checkpoint used in Figure 10.

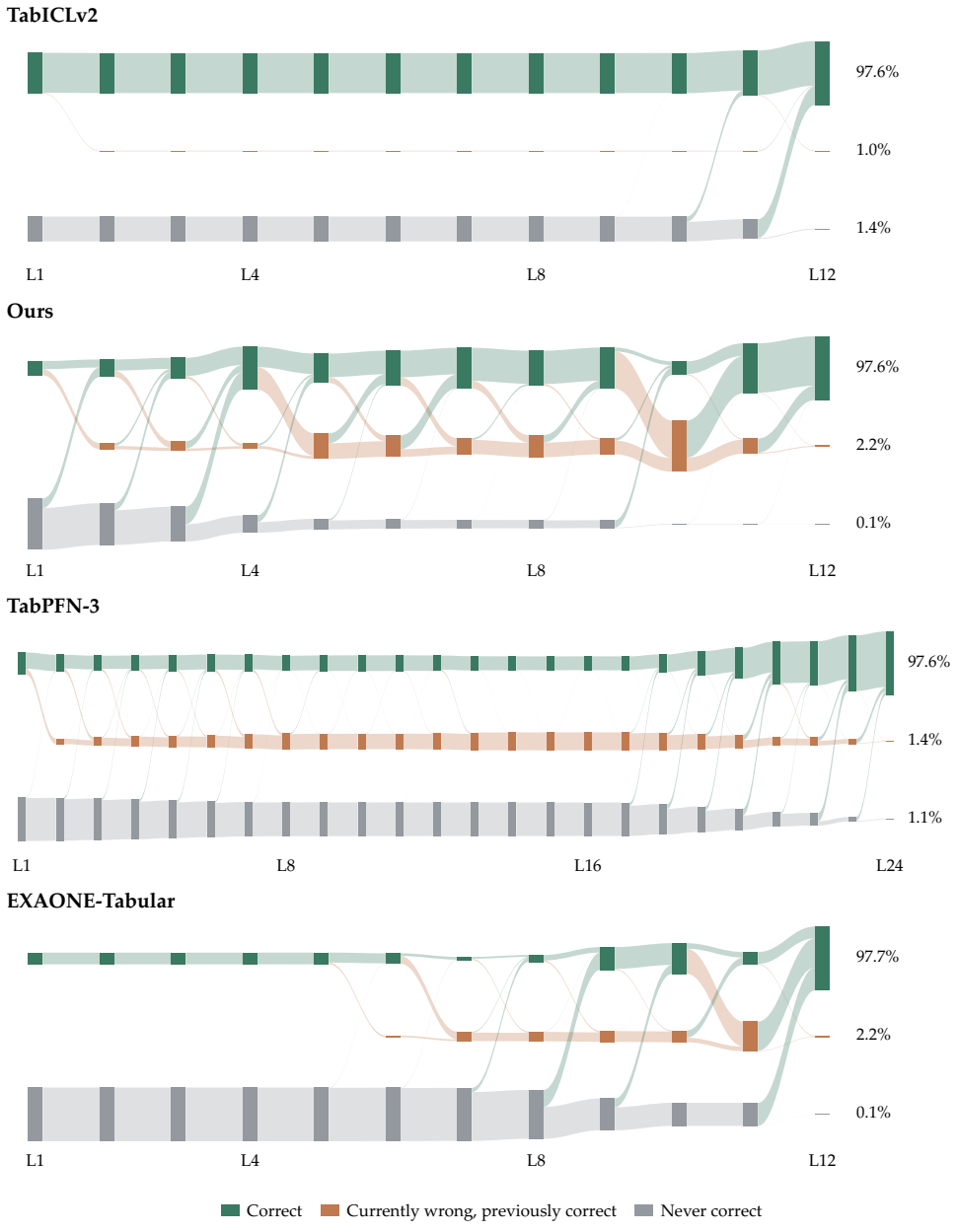

Figure 13: **Four-model sample-state flows on SDSS17.** Node heights and ribbon widths are percentages of all outer queries. *Correct* merges both currently-correct histories; the two wrong states distinguish whether a query was ever correct at an earlier layer. All native layers are retained. These are frozen-readout trajectories, not intermediate native-model accuracies.

C.3 BENCHMARK-AVERAGE SAMPLE-STATE FLOWS

Figure 14 extends Figure 4 to all four models over the 38 classification datasets. Each dataset contributes equally, regardless of query-set size. The plotted summary excludes the final transition, whereas the per-step statistics in Table 1 include it; Appendices A.2 and A.4 explains this distinction and the retrospective checkpoint difference.

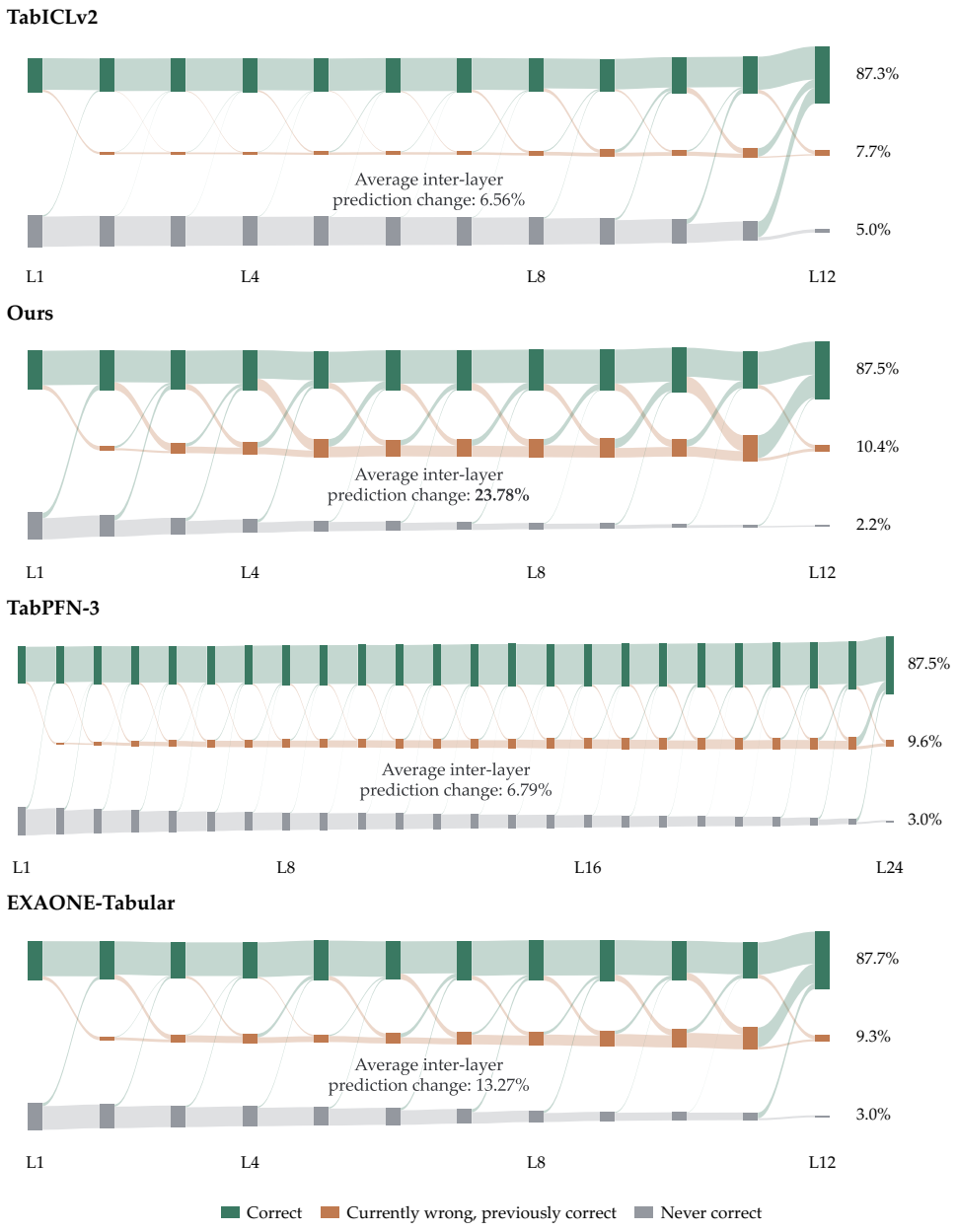

Figure 14: **Four-model sample-state flows averaged over 38 classification datasets.** Node heights and ribbon widths are equal-weight means of per-dataset query percentages, using the same three states as Figure 4. All native layers are shown. The average inter-layer prediction change excludes only each model's final transition from the summary, not from the plotted flows. These are frozen-readout trajectories, not intermediate native-model accuracies.

C.4 SUPERCONDUCTIVITY REGRESSION: FULL-DEPTH REPRESENTATIONS

Figures 15–18 extend the analysis to superconductivity. Colors encode the normalized reduction in absolute error defined in Equation (6), using one scale across models. This reveals improvements and deteriorations without discretizing a continuous prediction into correctness states.

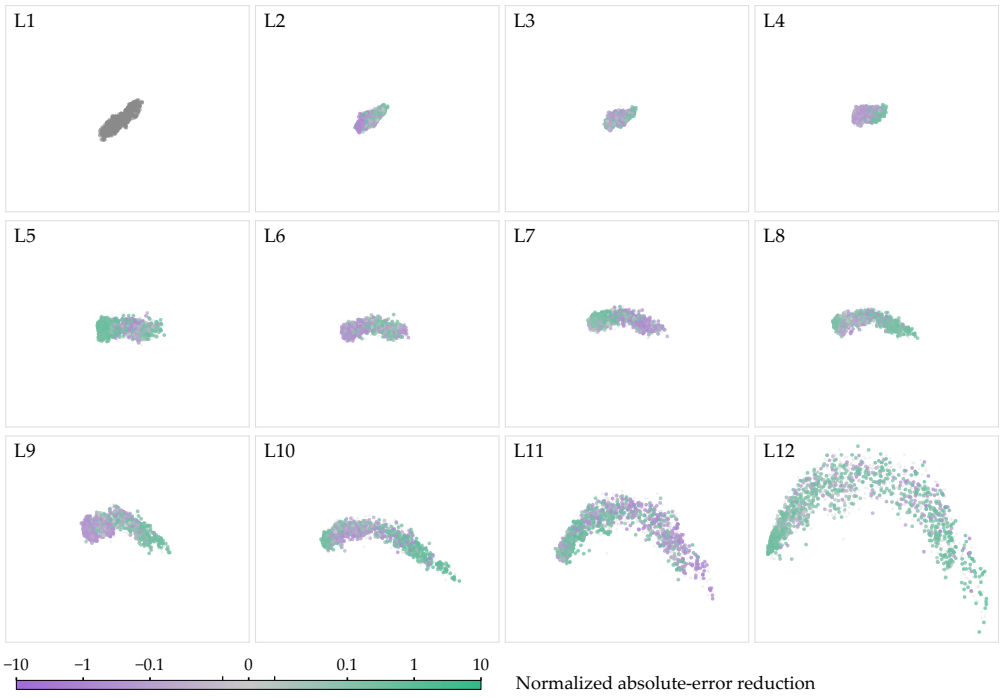

Figure 15: **TabICLv2 on superconductivity.** All native layers in the fixed final-support PCA basis. Green/purple indicate reductions/increases in absolute error under the frozen final-support Ridge readout, normalized by the support-target standard deviation. All four models share the same color scale. The first layer is gray; no binary correctness states are assigned.

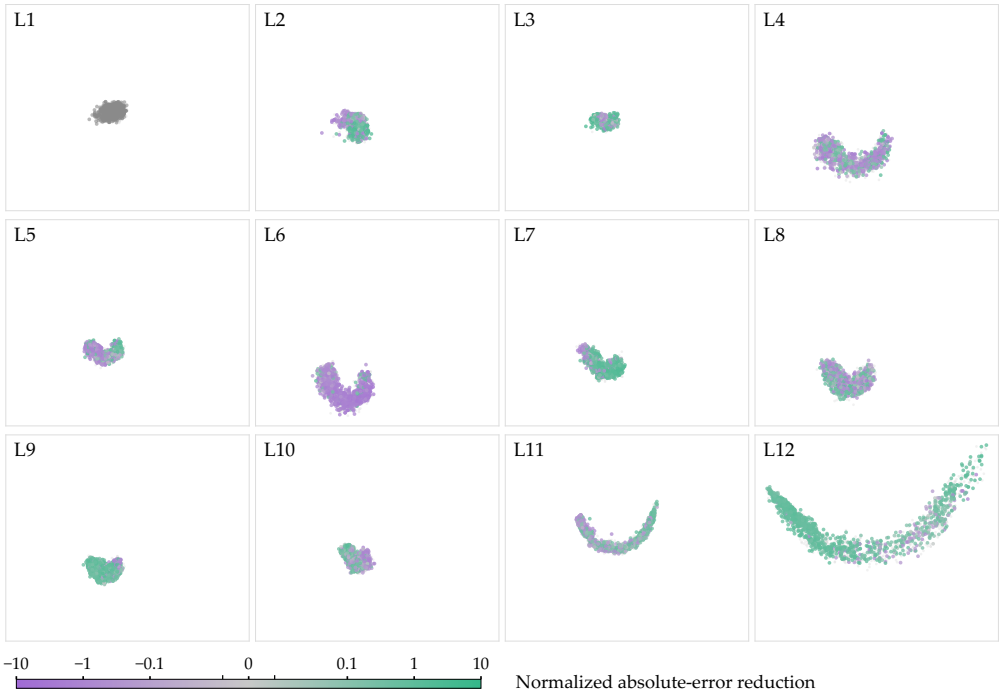

Figure 16: **Our retrospective model on superconductivity.** All native layers in the fixed final-support PCA basis. Green/purple indicate reductions/increases in absolute error under the frozen final-support Ridge readout, normalized by the support-target standard deviation. All four models share the same color scale. The first layer is gray; no binary correctness states are assigned.



Figure 17: **TabPFN-3 on superconductivity.** All native layers in the fixed final-support PCA basis. Green/purple indicate reductions/increases in absolute error under the frozen final-support Ridge readout, normalized by the support-target standard deviation. All four models share the same color scale. The first layer is gray; no binary correctness states are assigned.

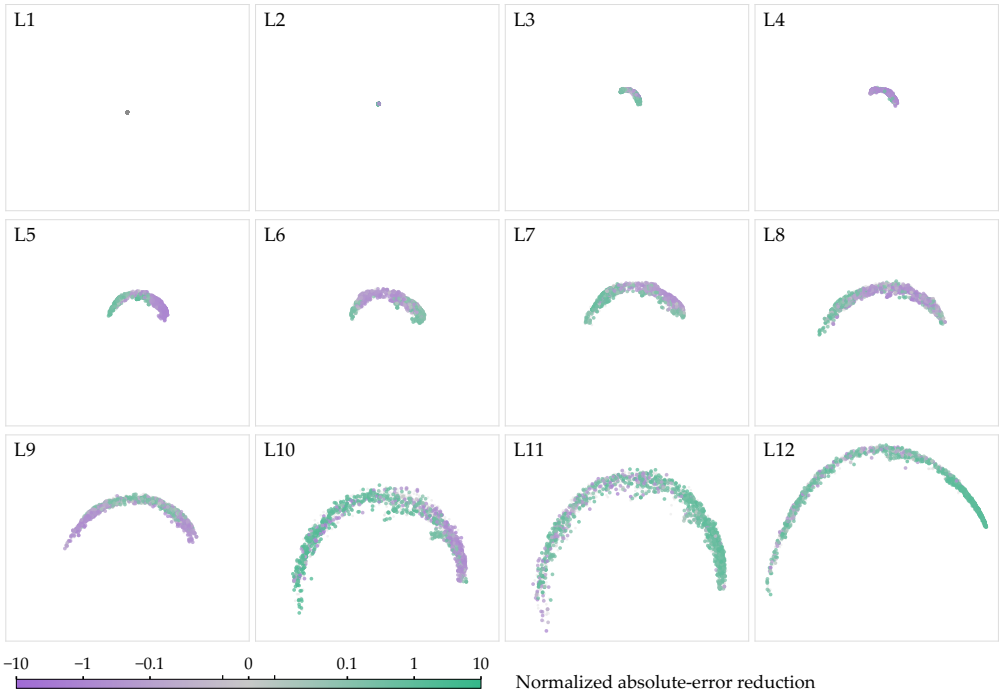

Figure 18: **EXAONE-Tabular on superconductivity.** All native layers in the fixed final-support PCA basis. Green/purple indicate reductions/increases in absolute error under the frozen final-support Ridge readout, normalized by the support-target standard deviation. All four models share the same color scale. The first layer is gray; no binary correctness states are assigned.

C.5 Additional classification case: Is-this-a-good-customer

Figures 19–22 provide an additional classification case using the stage-2 retrospective classifier. Corrections can be followed by deteriorations, illustrating the non-monotone trajectories that underlie the state-flow summaries. As discussed in Appendix A.5, such changes do not by themselves establish a benefit to the final prediction.

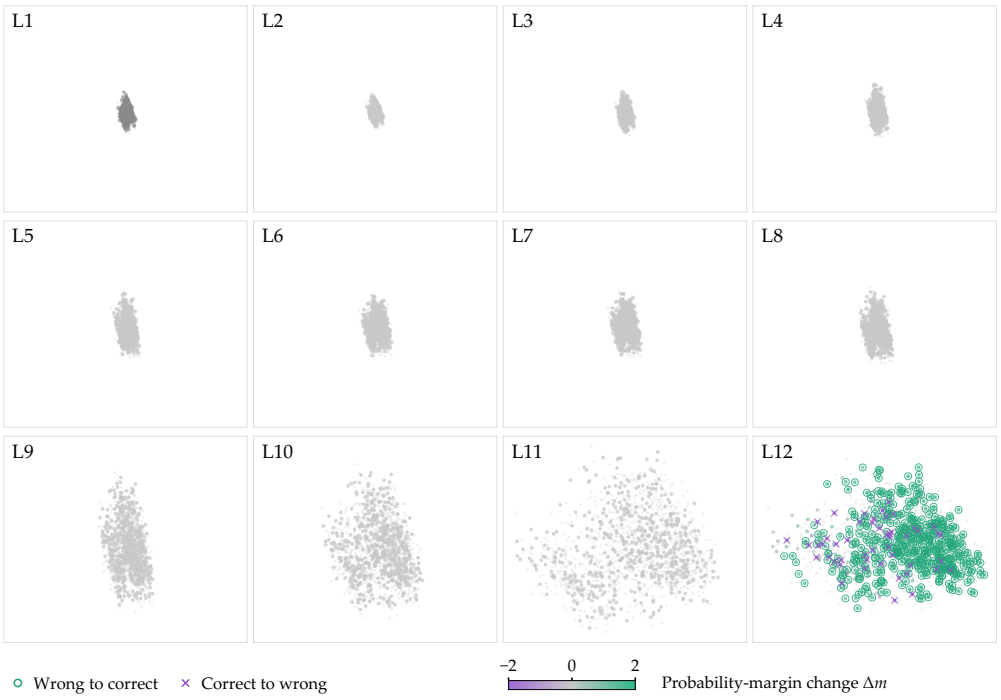

Figure 19: **TabICLv2 on Is-this-a-good-customer.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

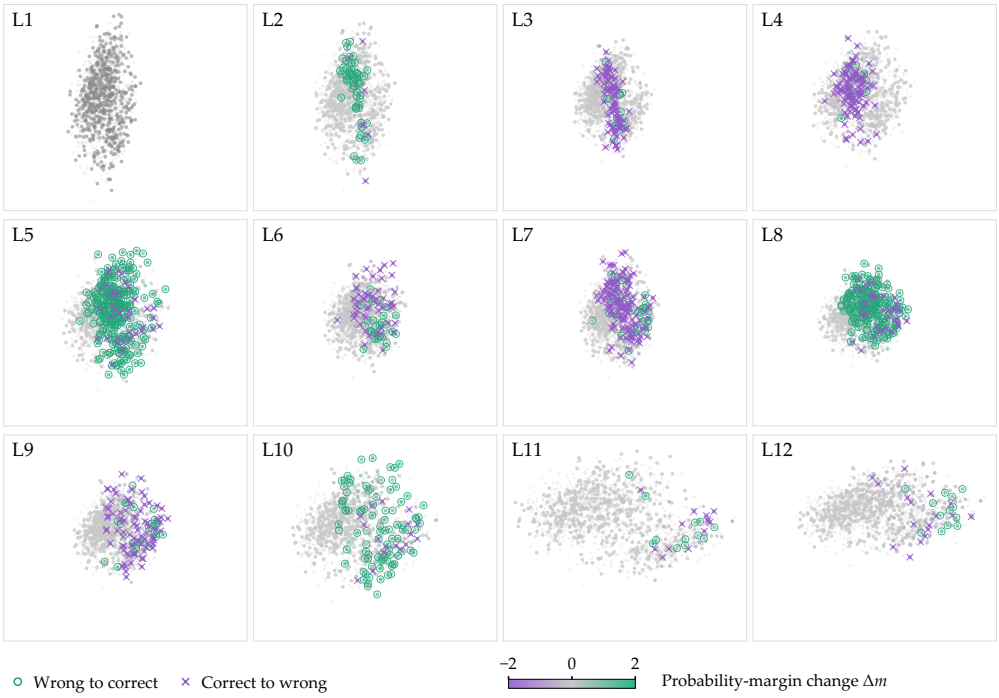

Figure 20: **Our retrospective model on Is-this-a-good-customer.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

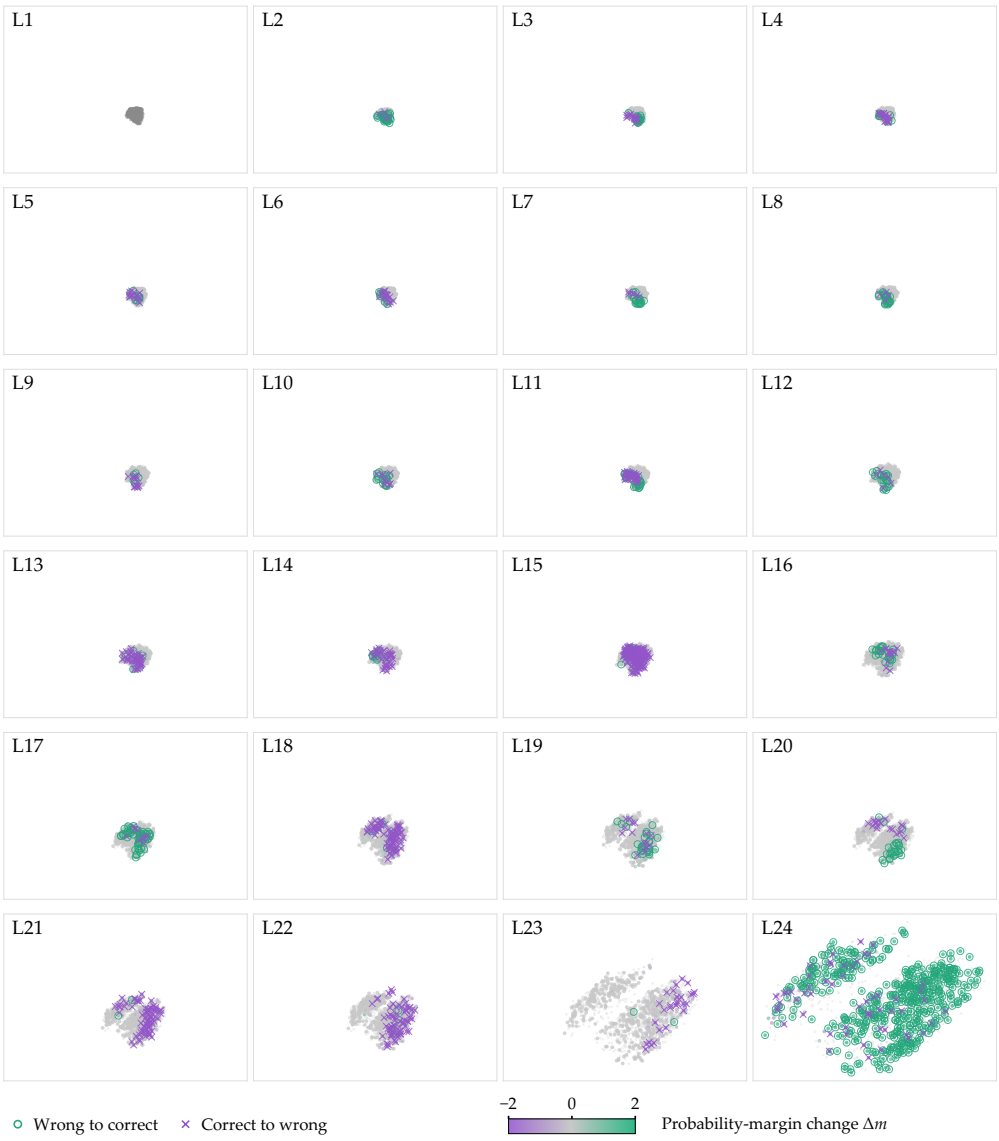

Figure 21: **TabPFN-3 on Is-this-a-good-customer.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

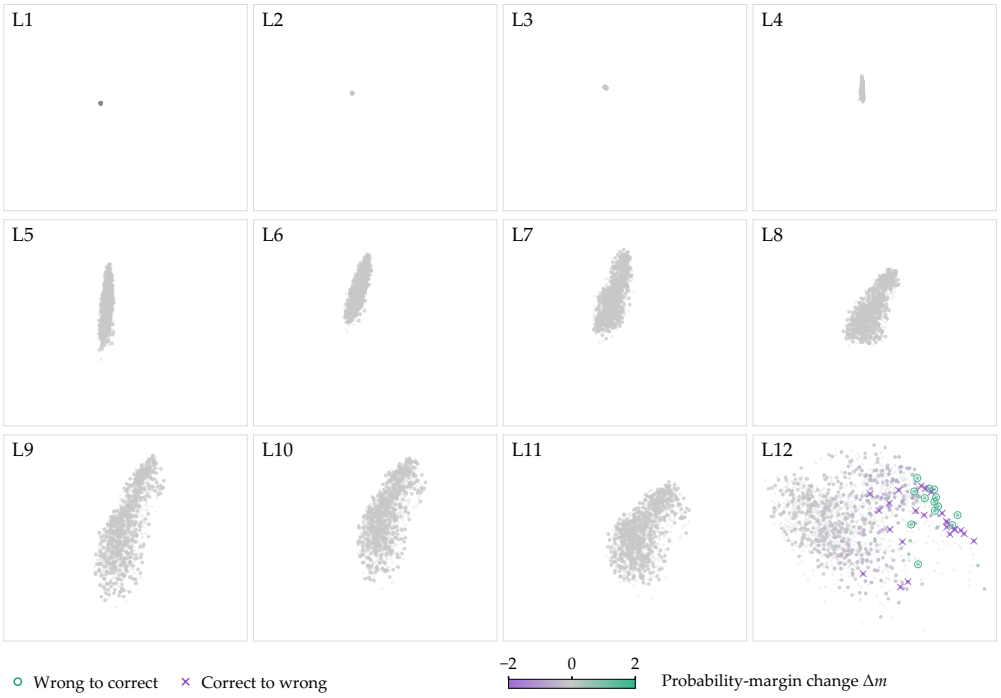

Figure 22: **EXAONE-Tabular on Is-this-a-good-customer.** All native layers in the fixed final-support PCA basis. Colors show adjacent probability-margin change under the frozen readout; rings and crosses mark corrections and deteriorations. The first layer is gray. Extraction and display conventions are given in Appendices A.1 and A.3.

D Analysis of Gated Contextual Updates

This appendix extends the update-geometry analysis in Section 5.2. We first test which aspects of the learned update matter to a fixed predictor, then examine whether gates are more interchangeable among nearby samples. A separate checkpoint comparison distinguishes these interventions from training with or without gating. All results use native final prediction heads, unlike the frozen-readout trajectories in Appendix A.

D.1 Direction and magnitude interventions

Protocol. We evaluate gated classification and regression models with 12 ICL layers, width 512, and two-layer history groups. Each checkpoint averages model weights from updates 39,901–40,000 of the second pretraining stage. The study covers the first official split of all 51 TabArena datasets (38 classification and 13 regression), using every support and query row, one view, and FP32 inference without ensembling. Query labels are used only for evaluation. Dataset means receive equal weight; 95% intervals use 20,000 paired dataset-bootstrap resamples.

Geometric controls. At each intervened state, let $o_{\ell i}$ and $g_{\ell i}$ be the concatenated attention output and learned gate for query i. Using the row-vector convention of Equation (3), define

$$\begin{aligned} \bar{g}_\ell &= |Q|^{-1} \sum_{i \in Q} g_{\ell i}, \\ u_{\ell i} &= (g_{\ell i} \odot o_{\ell i}) W_{O,\ell}^\top + \mathbf{b}_{O,\ell}^\top, \\ v_{\ell i} &= (\bar{g}_\ell \odot o_{\ell i}) W_{O,\ell}^\top + \mathbf{b}_{O,\ell}^\top. \end{aligned} \tag{9}$$

The counterpart $v_{\ell i}$ uses the channelwise mean gate of the current unlabeled query batch. We compare replacing the update by $v_{\ell i}$ with two partial replacements:

$$u_{\ell i}^{\mathrm{dir}} = v_{\ell i} \frac{\|u_{\ell i}\|_2}{\|v_{\ell i}\|_2}, \qquad u_{\ell i}^{\mathrm{mag}} = u_{\ell i} \frac{\|v_{\ell i}\|_2}{\|u_{\ell i}\|_2}. \tag{10}$$

The first changes direction while preserving the learned norm; the second changes the norm while preserving direction. Both act on the projected update, including its bias, rather than on the gate vector itself. Only query updates are replaced, at all 12 layers. Support updates, feed-forward computation, and normalization remain unchanged. The pair (u, v) is recomputed along each intervened trajectory.

Results. Figure 23 expands Figure 5 with dataset-level effects. Mean-gate replacement reduces classification accuracy by 0.719 percentage points. Replacing direction loses 0.616 points, versus -0.002 for replacing magnitude. The paired contrast is 0.618 points (95% interval: [0.173, 1.217]), but its median is 0.126: 23 datasets favor preserving direction, seven favor preserving magnitude, and eight tie. On credit-g, direction replacement is less harmful by 1.497 points. The effect is therefore heterogeneous across classification tasks.

For regression, NRMSE is query RMSE divided by the support-target standard deviation. Mean-gate, direction, and magnitude replacement increase NRMSE by 0.2326, 0.2328, and 0.0011, respectively. The direction-minus-magnitude contrast is 0.2317 (95% interval: [0.1579, 0.3070]), positive on all 13 datasets. Under these controls, preserving the learned direction is less disruptive than preserving only update size.

Interpretation. The interventions identify dependence on update geometry in trained models; they do not isolate the benefit of adding gating during training. Replacement distances differ, later layers operate on altered states, and normalization may confer tolerance to magnitude changes. This does not make residual-update scaling an invariance of the model. The evidence supports query-conditioned shaping of contextual updates, not layer skipping or the identification of semantic feature rules.

For MIC, missing-column masks are aligned with the encoded feature order. Excluding MIC leaves the classification direction contrast at 0.625 percentage points. This handling applies to the gating study and does not alter the earlier representation diagnostic in Appendix A.

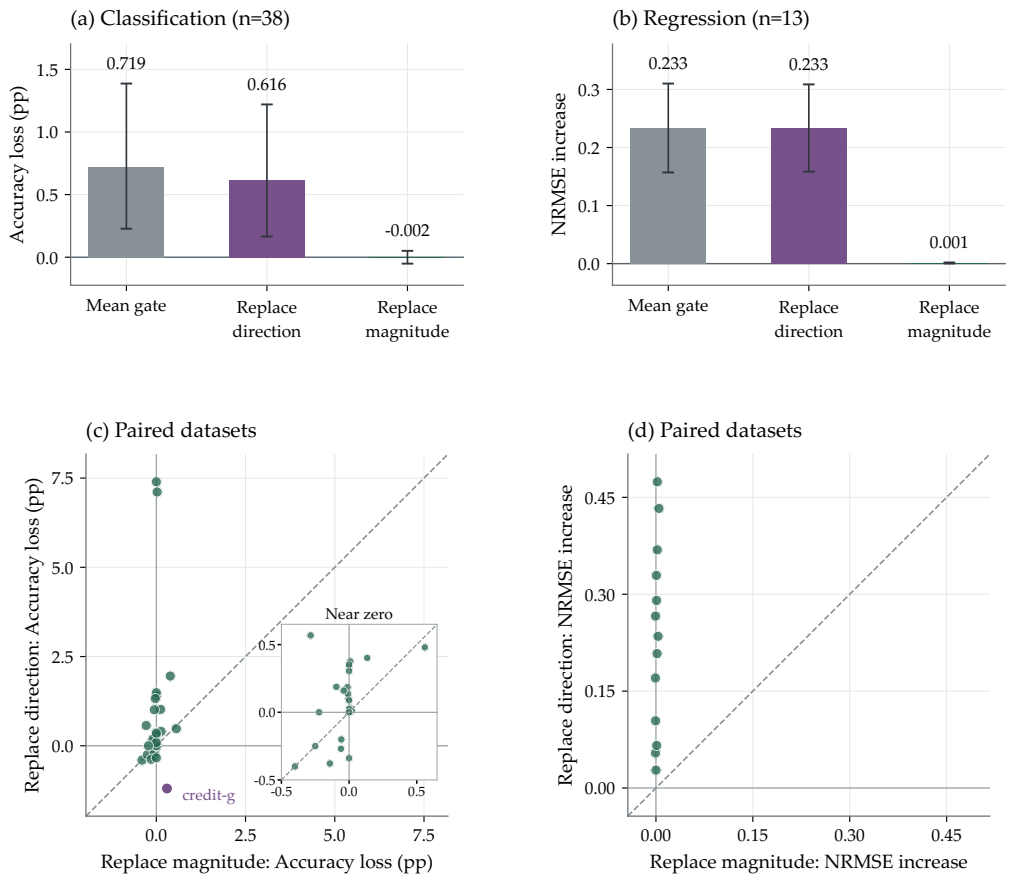

Figure 23: **Sensitivity to update direction and magnitude.** Top: mean degradation relative to learned gating, with 95% dataset-bootstrap intervals. Bottom: dataset-level effects; points above the diagonal indicate greater harm from replacing direction. The classification inset enlarges the region near zero. Positive values mean lower accuracy or higher NRMSE. Replacement distances are not matched.

D.2 Gate exchange within sample neighborhoods

We next examine whether gates can be exchanged with less disruption between nearby samples. Normalized first-layer attention inputs are standardized using support statistics. Eight MiniBatchKMeans clusters are fitted to support rows, and queries are assigned without their labels. Complete query gate vectors are cyclically exchanged within these groups. The control uses random groups with exactly the same sizes. Both conditions preserve the gate multiset, moved-query count, and singleton count; support gates remain unchanged. Groups are fixed across layers, and results are averaged over three exchange seeds within each dataset.

In Figure 24, neighborhood-constrained exchange has a mean advantage of 0.441 accuracy points (95% interval: [0.057, 1.031]) and 0.0810 NRMSE ([0.0415, 0.1289]). Classification has 18 improvements, 14 reversals, and six ties, with median advantage zero; all 13 regression datasets favor neighborhood-constrained exchange.

Local exchanges also have smaller gate displacement on every dataset. Mean gate RMS displacement is 0.0839 versus 0.0991 for classification and 0.1163 versus 0.1322 for regression. The result is consistent with local continuity of sample-conditioned updates, but does not separate continuity from semantic specialization. These neighborhoods are fitted from support-role inputs, not the cross-fitted query-role representations in Appendix A.1.

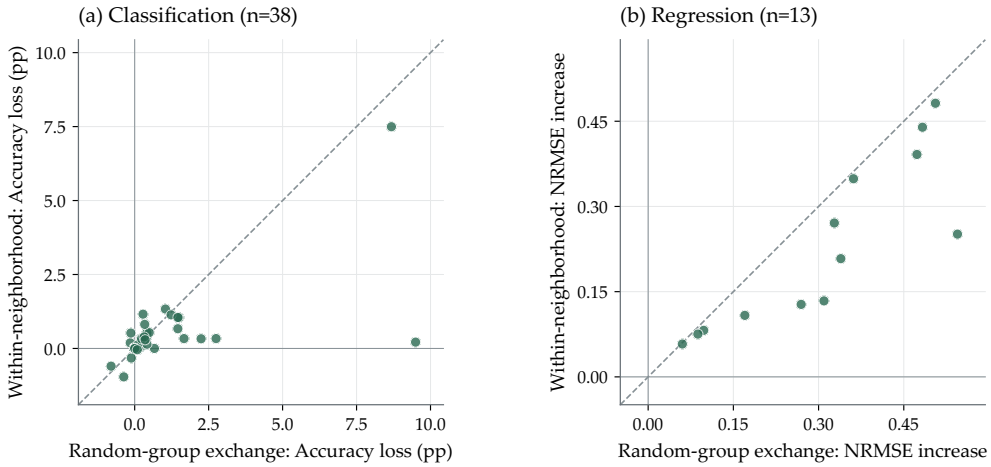

Figure 24: **Neighborhood-constrained versus random gate exchange.** Each point is a dataset, averaging three exchange seeds. Below the diagonal, exchange within support-derived neighborhoods is less harmful. Group sizes and moved-row counts are matched, but gate displacement is not.

D.3 Separately trained ungated models

For each task type, we compare the gated model with a separately trained ungated counterpart, also averaging updates 39,901–40,000. The recorded architecture configurations differ only in the ICL gating flag; both retain pre-normalization and their native prediction heads. Evaluation uses the same splits and single-view settings as Appendix D.1.

E BENCHMARK PROTOCOLS AND COMPLETE RANKINGS

This appendix provides the evaluation scope and complete rankings underlying Section 6 and Figure 8. These results evaluate native predictors, not the intermediate readouts in Appendix A or the single-view interventions in Appendix D.

E.1 EVALUATION SCOPE

TabArena includes 38 classification and 13 regression datasets, comprising 594 and 222 official splits, respectively. TALENT includes 198 classification and 98 regression datasets. RelArena includes 12 classification and nine regression tasks. The comparison pools contain 37/38/39 methods for TabArena (overall/classification/regression), 11 for each TALENT ranking, and 10 for each RelArena ranking.

E.2 RANKING AND UNCERTAINTY

All tables are ordered by decreasing Elo, with RETRO highlighted in bold. Average Rank is lower-is-better. The benchmark-specific Elo procedures and comparison pools are retained, so Elo values are not calibrated across benchmarks.

Elo+ and Elo− are the distances from the point estimate to the upper and lower interval endpoints. TabArena uses 200 dataset-bootstrap replicates and RelArena 100 task-bootstrap replicates, with 2.5th and 97.5th percentile endpoints. TALENT uses dataset-level aggregated scores, averaging evaluation seeds. Its Elo is averaged over 30 random permutations of dataset order. The resulting interval reflects order variability, not evaluation-seed uncertainty or a bootstrap confidence interval.

E.3 TABARENA

Tables 2–4 report the complete overall, classification, and regression comparisons for the TabArena panels of Figure 8.

Table 2: **TabArena: overall ranking.** All 37 selected methods on 51 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM (default)	1882.9	109.9	99.5	4.31
EXAONE-Tabular (default)	1849.7	72.8	49.7	4.89
RETRO	**1741.6**	**84.3**	**72.5**	**7.26**
TabPFN-3 (default)	1728.9	76.1	55.1	7.58
Xiaomi-TabLDM (default)	1673.7	74.2	66.0	9.11
TabPFN-2.6 (default)	1669.5	62.7	44.9	9.24
TabICLv2 (default)	1650.9	67.6	57.6	9.80
TabDPT-1.3 (default)	1601.4	60.6	40.1	11.39
RealMLP (tuned + ensembled)	1558.3	46.1	44.7	12.88
TabDPT-Turbo (default)	1502.2	59.1	50.6	14.93
TabM (tuned + ensembled)	1501.3	49.8	45.7	14.97
LightGBM (tuned + ensembled)	1483.0	32.0	33.5	15.66
RealMLP (tuned)	1470.7	46.5	50.4	16.13
CatBoost (tuned + ensembled)	1467.6	33.1	39.8	16.25
CatBoost (tuned)	1452.1	36.5	35.0	16.85
TabM (tuned)	1437.1	49.6	43.0	17.43
ModernNCA (tuned + ensembled)	1437.1	67.8	52.1	17.43
LightGBM (tuned)	1427.2	31.4	26.7	17.82
XGBoost (tuned + ensembled)	1417.9	32.0	31.9	18.18
CatBoost (default)	1410.3	47.6	43.1	18.48
ModernNCA (tuned)	1394.9	41.1	40.3	19.08
XGBoost (tuned)	1387.8	30.3	30.9	19.36
TabSwift (default)	1387.8	66.3	53.8	19.36
TabM (default)	1328.5	48.2	47.5	21.65
ModernNCA (default)	1279.4	39.2	40.3	23.50
ExtraTrees (tuned + ensembled)	1254.1	53.4	55.9	24.42
RealMLP (default)	1253.7	36.1	45.8	24.43
XGBoost (default)	1234.3	37.6	41.9	25.12
ExtraTrees (tuned)	1218.7	58.2	68.1	25.66
RandomForest (tuned + ensembled)	1207.4	61.8	58.3	26.04
LightGBM (default)	1202.6	43.4	36.9	26.20
RandomForest (tuned)	1163.2	62.9	69.3	27.48
Linear (tuned + ensembled)	1032.9	65.0	106.2	31.10
ExtraTrees (default)	1019.3	56.9	86.3	31.42
RandomForest (default)	1000.0	55.0	66.1	31.86
Linear (tuned)	996.6	70.0	126.9	31.93
Linear (default)	897.1	76.7	148.5	33.81

Table 3: **TabArena: classification ranking.** All 38 selected methods on 38 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM (default)	1833.3	118.3	107.4	4.76
EXAONE-Tabular (default)	1821.1	82.0	56.0	5.00
TabPFN-3 (default)	1690.7	74.8	68.0	8.12
RETRO	**1680.4**	**79.1**	**70.8**	**8.42**
TabPFN-2.6 (default)	1638.4	56.6	50.2	9.70
Xiaomi-TabLDM (default)	1631.5	70.7	60.1	9.93
TabICLv2 (default)	1627.6	78.4	68.7	10.05
TabDPT-1.3 (default)	1566.9	83.6	61.4	12.16
RealMLP (tuned + ensembled)	1516.8	50.0	41.7	14.05
TabM (tuned + ensembled)	1495.9	57.1	45.0	14.88
LightGBM (tuned + ensembled)	1463.0	51.0	35.2	16.21
TabDPT-Turbo (default)	1454.3	72.5	60.1	16.57
TabICL (default)	1447.4	49.0	56.2	16.85
CatBoost (tuned + ensembled)	1442.0	50.1	42.7	17.07
TabM (tuned)	1437.8	55.9	43.0	17.25
RealMLP (tuned)	1434.3	56.6	54.2	17.40
CatBoost (tuned)	1430.9	49.0	44.7	17.54
LightGBM (tuned)	1413.3	41.1	34.8	18.27
XGBoost (tuned + ensembled)	1407.7	42.0	47.0	18.51
CatBoost (default)	1407.2	48.3	47.6	18.53
ModernNCA (tuned + ensembled)	1391.6	71.9	65.9	19.18
ModernNCA (tuned)	1385.3	51.3	35.9	19.44
XGBoost (tuned)	1376.6	38.1	46.0	19.81
TabSwift (default)	1366.2	64.6	61.1	20.25
TabM (default)	1331.4	53.4	48.0	21.69
ModernNCA (default)	1265.9	45.2	48.3	24.34
RealMLP (default)	1251.2	39.7	45.2	24.92
XGBoost (default)	1242.9	43.7	50.1	25.24
ExtraTrees (tuned + ensembled)	1241.7	63.5	62.3	25.28
RandomForest (tuned + ensembled)	1210.6	77.4	77.2	26.45
ExtraTrees (tuned)	1207.0	67.6	73.2	26.58
LightGBM (default)	1192.7	57.8	55.7	27.10
RandomForest (tuned)	1166.1	76.0	80.3	28.03
Linear (tuned + ensembled)	1088.1	91.1	98.4	30.51
Linear (tuned)	1053.4	91.8	106.5	31.48
RandomForest (default)	1000.0	78.3	78.3	32.81
ExtraTrees (default)	997.5	90.5	87.7	32.87
Linear (default)	956.3	97.9	123.9	33.76

Table 4: **TabArena: regression ranking.** All 39 selected methods on 13 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM (default)	2165.6	254.1	73.4	3.56
RETRO	**2082.1**	**332.6**	**117.5**	**4.83**
EXAONE-Tabular (default)	2043.9	251.0	83.9	5.52
TabPFN-3 (default)	1966.4	320.5	122.8	7.15
Xiaomi-TabLDM (default)	1928.5	348.6	176.1	8.05
Nori-30M (default)	1895.0	254.2	76.1	8.90
TabPFN-2.6 (default)	1866.7	252.9	51.6	9.66
TabICLv2 (default)	1835.8	292.9	124.4	10.53
Nori (default)	1825.1	242.2	65.6	10.84
TabDPT-1.3 (default)	1817.0	299.2	111.4	11.07
RealMLP (tuned + ensembled)	1786.9	204.5	47.5	11.97
TabDPT-Turbo (default)	1762.6	251.9	83.4	12.72
ModernNCA (tuned + ensembled)	1684.5	206.6	99.9	15.24
RealMLP (tuned)	1678.3	189.1	78.3	15.45
CatBoost (tuned + ensembled)	1628.3	212.6	76.2	17.13
LightGBM (tuned + ensembled)	1623.3	202.9	90.5	17.30
CatBoost (tuned)	1597.8	219.6	88.8	18.17
TabM (tuned + ensembled)	1593.0	240.3	96.3	18.34
LightGBM (tuned)	1546.3	227.3	87.7	19.94
TabSwift (default)	1536.7	212.2	140.0	20.27
XGBoost (tuned + ensembled)	1523.8	195.5	77.9	20.71
ModernNCA (tuned)	1509.2	199.9	105.4	21.21
TabM (tuned)	1505.1	248.6	102.7	21.34
XGBoost (tuned)	1497.4	217.6	87.9	21.61
CatBoost (default)	1491.9	167.0	126.8	21.79
ModernNCA (default)	1393.3	192.3	94.7	25.01
TabM (default)	1384.0	222.6	125.6	25.30
ExtraTrees (tuned + ensembled)	1354.7	168.4	107.1	26.20
RealMLP (default)	1325.6	172.1	125.8	27.05
ExtraTrees (tuned)	1316.4	169.4	113.4	27.31
LightGBM (default)	1288.8	168.9	90.4	28.08
XGBoost (default)	1259.9	191.5	124.5	28.84
RandomForest (tuned + ensembled)	1246.1	118.0	116.6	29.19
RandomForest (tuned)	1199.1	119.6	129.5	30.31
ExtraTrees (default)	1121.8	133.6	157.9	31.91
RandomForest (default)	1000.0	107.5	137.5	33.82
Linear (tuned + ensembled)	491.6	142.7	2134.7	37.27
Linear (tuned)	389.4	185.5	2245.4	37.71
Linear (default)	121.4	220.6	2245.2	38.69

E.4 TALENT

Tables 5–7 report the corresponding TALENT comparisons. Their interval columns use the dataset-order variability described in Appendix E.2.

Table 5: **TALENT: overall ranking.** All 11 selected methods on 296 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM	1724.9	210.7	271.0	3.41
RETRO	**1604.0**	**229.3**	**265.6**	**4.65**
EXAONE-Tabular	1600.1	282.4	271.5	4.56
TabPFN-3	1598.5	225.1	157.0	4.88
TabICLv2	1593.1	251.2	276.6	5.27
TabPFN-2.5	1485.4	170.7	188.4	6.00
TabPFN-2.6	1478.9	260.5	275.3	5.99
TabPFN-2	1414.5	274.9	209.0	7.46
LimiX-16M	1399.1	268.9	307.6	7.60
TabSwift	1333.6	301.7	245.3	7.87
LimiX-2M	1267.9	314.3	220.0	8.31

Table 6: **TALENT: classification ranking.** All 11 selected methods on 198 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM	1657.7	233.4	328.7	3.61
RETRO	**1594.0**	**216.4**	**228.6**	**5.12**
TabICLv2	1591.4	236.9	298.9	5.15
EXAONE-Tabular	1584.6	169.4	258.9	4.97
TabPFN-2.5	1558.6	223.9	273.8	5.83
TabPFN-3	1540.9	231.0	283.8	5.15
TabPFN-2.6	1488.1	198.8	179.7	6.23
TabPFN-2	1415.7	254.2	218.6	7.08
LimiX-16M	1405.8	236.6	237.4	7.19
TabSwift	1341.6	344.6	270.8	7.82
LimiX-2M	1321.5	377.3	267.3	7.84

Table 7: **TALENT: regression ranking.** All 11 selected methods on 98 datasets.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
TabFM	1801.8	213.8	225.1	3.01
RETRO	**1747.8**	**218.1**	**323.0**	**3.70**
EXAONE-Tabular	1747.0	171.6	191.1	3.71
TabPFN-3	1624.1	249.2	308.7	4.33
TabICLv2	1566.3	183.8	244.2	5.50
TabPFN-2.6	1556.6	236.9	175.3	5.52
TabPFN-2.5	1446.5	245.4	194.1	6.34
LimiX-16M	1295.3	282.4	281.4	8.43
TabSwift	1279.3	269.6	187.3	7.98
TabPFN-2	1246.7	425.2	227.0	8.21
LimiX-2M	1188.7	302.7	228.5	9.27

E.5 RELARENA

Tables 8–10 report the RelArena comparisons. Overall results pool all 21 tasks, while the two task-specific tables retain their respective subsets.

Table 8: **RelArena: overall ranking.** All 10 selected methods on 21 tasks.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
RETRO	**1797.0**	**78.4**	**60.4**	**2.90**
kurversc	1781.2	92.8	93.6	3.05
TabPFN-rel-local	1721.8	116.2	80.2	3.62
GraphSAGE	1666.4	90.8	78.6	4.19
RelGT	1575.2	126.8	103.3	5.17
RDBLearn	1555.1	89.6	93.1	5.38
RelGNN-ES	1521.4	90.6	81.6	5.74
LightGBM	1356.2	90.2	120.8	7.33
constant-per-entity	1260.3	130.4	150.4	8.10
constant-global	1000.0	70.9	130.1	9.52

Table 9: **RelArena: classification ranking.** All 10 selected methods on 12 tasks.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
RETRO	**2185.9**	**266.1**	**45.3**	**3.08**
kurversc	2151.6	340.0	111.5	3.42
TabPFN-rel-local	2135.2	315.9	122.3	3.58
GraphSAGE	2103.2	358.6	135.7	3.92
RelGT	1981.0	352.6	124.4	5.25
RelGNN-ES	1958.1	292.1	106.1	5.50
RDBLearn	1958.1	292.1	89.5	5.50
LightGBM	1770.2	250.6	106.4	7.33
constant-per-entity	1754.8	331.8	185.6	7.46
constant-global	1000.0	186.4	1952.8	9.96

Table 10: **RelArena: regression ranking.** All 10 selected methods on 9 tasks.

Method	Elo ↑	Elo+	Elo−	Average Rank ↓
kurversc	1736.4	321.6	143.4	2.56
RETRO	**1722.3**	**302.3**	**130.4**	**2.67**
TabPFN-rel-local	1609.0	247.2	169.5	3.67
GraphSAGE	1519.0	103.4	94.2	4.56
RelGT	1469.8	208.6	185.5	5.06
RDBLearn	1453.4	158.5	188.5	5.22
RelGNN-ES	1369.7	162.9	197.9	6.06
LightGBM	1228.9	103.2	211.6	7.33
constant-per-entity	1000.0	184.3	353.1	8.94
constant-global	1000.0	63.7	147.0	8.94