RECURSIVE TABULAR FOUNDATION MODELS EXHIBIT BAYESIAN MODEL-SELECTION BEHAVIOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Tabular foundation models (TFMs) have recently achieved strong performance on new prediction tasks through in-context learning without dataset-specific training. However, most existing TFMs perform inference in one computational pass, which may limit their ability to capture complex relationships in the context. We study latent recursive reasoning to scale test-time computation, developing recursive TFMs that repeatedly revisit the context and progressively refine predictions. On regression tasks, we characterize this refinement with attention-induced kernel predictors that closely reproduce the networks' predictions and analyze how the corresponding Gaussian-process (GP) marginal likelihood evolves across recursive steps. We find that recursive refinement typically improves marginal likelihood by changing the balance between data fit and model complexity. These findings provide empirical evidence for Bayesian model-selection behavior in recursive TFMs, although neither predictive pre-training nor the kernel predictors' prediction-matching calibration explicitly optimizes marginal likelihood.

1 INTRODUCTION

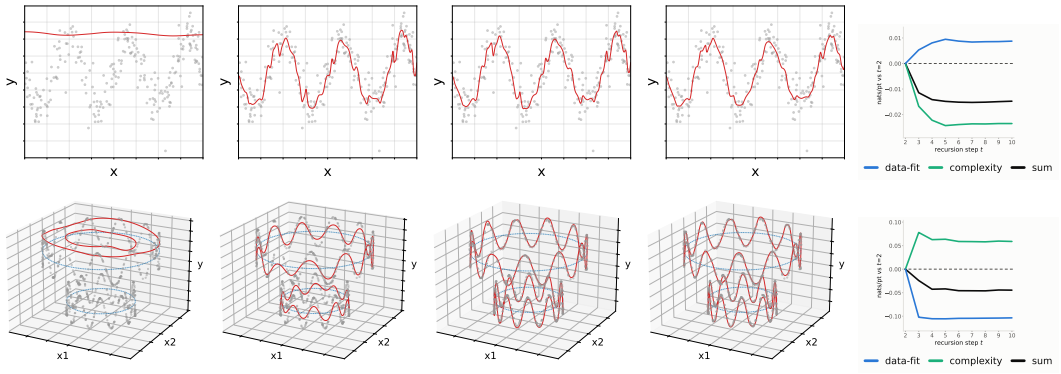
Tabular foundation models (TFMs) such as TabPFN (Hollmann et al., 2023), TabICL (Qu et al., 2025), and TabDPT (Ma et al., 2025) have established a new paradigm for tabular learning. Unlike classical models requiring dataset-specific training, TFMs solve new tasks through in-context learning: labeled training samples provide context, test samples form queries, and the model infers a prediction rule directly from the context at inference time. Despite strong performance, existing TFMs typically perform in-context learning in one computational pass. Because tabular datasets vary substantially in complexity, noise, and feature relationships, one pass over the context may be insufficient and yield suboptimal predictions. Scaling test-time computation in TFMs remains largely unexplored.

In other domains, particularly large language models, techniques such as chain-of-thought reasoning have proven highly effective for scaling test-time computation. Beyond autoregressive language models, recursive architectures such as looped Transformers (Yang et al., 2024) and TRM (Jolicoeur-Martineau, 2025) have demonstrated effective test-time reasoning by iteratively refining internal representations. Recursive reasoning provides a natural form of test-time computation for tabular in-context learning but raises a fundamental question: if additional computation allows a TFM to progressively refine its inferred prediction rule, how does that rule evolve across steps? *We find that this evolution exhibits Bayesian model-selection behavior.*

We study recursive reasoning to scale test-time computation in TFMs, developing variants based on looped Transformers (Yang et al., 2024) and TRMs (Jolicoeur-Martineau, 2025). Standard TFMs process the context and query data once through a fixed Transformer stack before prediction. Our recursive TFMs instead repeatedly apply the same Transformer stack while carrying an evolving internal state across steps, then predict after several iterations. At each step, the model processes the same in-context data and query points using its current state, allowing it to further refine its task representation, a capability standard TFMs lack.

To understand how recursive computation refines the prediction rule, we develop a kernel-based view across steps. At each step, we empirically verify that a kernel predictor constructed from the model's internal representations can closely reproduce the model's predictions, characterizing recursion as a sequence of evolving model-induced kernels. Interpreting these kernels through Gaussian-process

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.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 6$ (e) GP tracking

Figure 1: Recursive refinement by our TRM model on two toy regression tasks. (a)–(d) show context samples (gray) and model predictions (red) at recursion steps $t = 1, 2, 3, 6$. (e) shows changes relative to $t = 2$ in the induced GP's data-fit term (blue), complexity term (green), and their sum, NLML (black). $t = 1$ is omitted from the GP analysis as its predictions are nearly constant.

(GP) marginal likelihood reveals that the model progressively adjusts the trade-off between fitting observed context data and controlling model complexity. The recursive TFM thus appears to follow an implicit Bayesian model-selection process, although it is never explicitly trained to optimize this objective during pre-training.

Figure 1 illustrates this behavior for our TRM model on two toy regression tasks. On a noisy one-dimensional sine function (top row), the prediction is nearly constant at the first step; by recursive step $t = 2$ it already follows the periodic structure, and further steps yield a smoother fit. The GP decomposition (panel e) shows that this refinement mainly reduces model complexity while accepting a worse data fit. On a more complex two-dimensional task whose targets oscillate along stacked circles (bottom row), the opposite occurs: the $t = 2$ prediction captures only coarse structure and underfits the higher-frequency oscillations, which later steps recover. Here, recursion improves data fit at the cost of higher complexity. These examples illustrate how recursion can refine predictions by changing the balance between data fit and model complexity.

In summary, our main contributions are as follows:

- We introduce recursive reasoning to scale test-time computation in TFMs and develop variants based on looped Transformers and Tiny Recursive Models.
- Using model-induced kernels, we characterize prediction-rule evolution across recursive steps. Their GP-style marginal-likelihood trajectories provide empirical evidence for Bayesian model-selection behavior.
- Extensive experiments show that recursive computation improves predictive performance, with particularly strong gains on regression and challenging tabular tasks, and that performance evolves, saturates, and may eventually degrade across recursive steps.

2 Related Work

Tabular Foundation Models. Prior-data fitted networks (Müller et al., 2022) showed that Transformers pre-trained on tasks sampled from a prior can approximate Bayesian inference through in-context learning. Building on this, TabPFN (Hollmann et al., 2023) introduced a tabular foundation model and showed that synthetic-data pre-training with in-context learning can outperform task-specific tabular models. Later versions, TabPFN-2 (Hollmann et al., 2025), TabPFN-2.5 (Grinsztajn et al., 2025), and TabPFN-3 (Grinsztajn et al., 2026), further improved scalability and capacity. TabICL (Qu et al., 2025) introduced a two-stage architecture using column-wise then row-wise attention to compress representations and improve scalability. TabFlex (Zeng et al., 2025) replaced softmax attention with linear attention for greater efficiency. Mitra (Zhang et al., 2025) studied synthetic priors and proposed a curated mixture of task distributions. TabDPT (Ma et al., 2025)

incorporated real tabular data into pre-training and used k-nearest-neighbor context selection. Other work explored post-hoc adaptation and dataset-specific fine-tuning, including LoCalPFN (Thomas et al., 2024), MixturePFN (Xu et al., 2025), TuneTables (Feuer et al., 2024), TabPFN-ICD (Ma et al., 2024), BETA (Liu & Ye, 2025), test-time divide-and-conquer (Ye et al., 2025), and fine-tuning strategies (Rubachev et al., 2025).

Most directly related to our work, TabPFN-3's Plus Thinking feature (Grinsztajn et al., 2026) scales inference-time computation. However, it is available only through the enterprise API, and, as of this writing, no details of its mechanism have been released. Relatedly, Balef et al. (2026) studied inference across Transformer layers in existing TFMs, finding substantially overlapping computations across depths. As a small-scale proof of concept for model compression, they showed that a looped, single-layer nanoTabPFN could match a six-layer counterpart with fewer parameters. In contrast, we study recursion to *scale test-time computation* and improve predictions beyond a fixed-depth model. Importantly, we characterize this refinement through a GP marginal-likelihood view.

Recursive Reasoning and Looped Transformers. Looped Transformers reuse shared Transformer blocks across iterations (Yang et al., 2024). Theoretical studies connect looping and chain-of-thought to multi-step gradient descent in linear regression (Gatmiry et al., 2024; Huang et al., 2025). Fan et al. (2026) combined looping with chain-of-thought supervision for efficient latent reasoning. Wang et al. (2025) introduced the Hierarchical Reasoning Model (HRM), which uses two coupled recurrent modules operating at different timescales to iteratively refine latent states. Building on HRM, Jolicoeur-Martineau (2025) introduced the Tiny Recursive Model (TRM), which uses a single shared network to repeatedly update a latent reasoning state and refine an answer representation. With deep supervision, TRM achieved stronger generalization than HRM on Sudoku, maze solving, and abstract reasoning tasks. We adapt looped Transformers and TRM to TFMs and analyze how recursive refinement can improve predictions beyond one forward pass.

Kernel-Based Views of In-Context Learning. Several works interpret Transformer in-context learning through kernel methods. Han et al. (2025) developed a kernel-regression interpretation of in-context learning in pretrained LLMs on text classification tasks, such as sentiment detection. Shen et al. (2026) studied in-context regression of Hölder-continuous functions on low-dimensional manifolds, explicitly constructing a Transformer that implements Nadaraya–Watson regression with a Gaussian kernel over inputs and deriving generalization bounds. Miftachov et al. (2026) introduced KernelICL, replacing TabICL's prediction head with explicit kernel-weighted averages of context labels and fine-tuning the model to predict through this kernel mechanism for interpretability. In contrast to these studies, we construct attention-induced kernel ridge predictors that closely approximate pre-trained TFM predictions without imposing a kernel prediction mechanism. We use these predictors to track refinement across recursive steps through their GP marginal likelihood.

3 Preliminaries: Tabular Foundation Models

Given a labeled training set $\mathcal{D}_{\mathrm{tr}} = (X_{\mathrm{tr}}, y_{\mathrm{tr}})$ and test samples X_{te}, the objective is to predict the unknown test targets y_{te}. Tabular foundation models (TFMs) formulate this task as in-context learning (Hollmann et al., 2023): labeled training samples provide context, $(X_c, y_c) = (X_{\mathrm{tr}}, y_{\mathrm{tr}})$, while test samples form the query set, $X_q = X_{\mathrm{te}}$. The model predicts the query targets $\hat{y}_{\mathrm{q}}$ in one forward pass:

$$\hat{y}_{\mathrm{q}} = f_\theta(X_{\mathrm{c}}, y_{\mathrm{c}}, X_{\mathrm{q}}). \tag{1}$$

TFMs are pre-trained on a large prior of synthetic or real tabular tasks $\left(X_{\mathrm{c}}, y_{\mathrm{c}}, X_{\mathrm{q}}\right) \mapsto y_{\mathrm{q}}$ using supervised prediction loss $\mathcal{L}(\hat{y}_{\mathrm{q}}, y_{\mathrm{q}})$. This meta-learning-style pre-training enables generalization to unseen tabular tasks.

A TFM has three stages: encoding, computation, and decoding. Typical row-token architectures (Hollmann et al., 2023; Ma et al., 2025) use:

$$x = \left[E_X(X_c) + E_y(y_c); E_X(X_q)\right], \qquad s = F_\theta(x), \qquad \hat{y}_q = D(s_q). \tag{2}$$

In *encoding*, the feature encoder E_X processes context and query features, while the target encoder E_y processes observed context targets. The resulting representations are concatenated into x, with labeled context samples followed by unlabeled query samples. In *computation*, the Transformer backbone F_θ performs in-context learning on this sequence, producing context and query representations $s = [s_c; s_q]$. Context samples attend only to one another, while query samples attend only

to the context. Finally, in *decoding*, D maps query representations s_q to predicted targets $\hat{y}_q$. For simplicity, we omit encoder- and decoder-specific parameters but retain θ for the backbone. This description mainly follows TabPFN-v1 (Hollmann et al., 2023) and TabDPT (Ma et al., 2025); other TFM architectures may differ.

4 Recursive Tabular Foundation Models

Standard TFMs perform in-context learning in a single pass, processing encoded context through a Transformer stack before prediction. However, extracting richer context structure may require deeper computation and, often, repeated visits to the same data, neither of which a single-pass model naturally provides. We address this by incorporating recursive reasoning into TFMs to scale test-time computation. A recursive TFM retains a standard TFM's encoder and decoder but replaces its single-pass backbone with a shared Transformer stack that repeatedly processes x while updating state s across recursive steps. This allows the model to progressively refine the inferred prediction rule. We instantiate this framework with two existing recursive architectures: looped Transformers (Yang et al., 2024) and Tiny Recursive Models (TRMs) (Jolicoeur-Martineau, 2025), described with pseudocode in Appendix Figure 4.

Looped Transformer. The looped Transformer repeatedly applies its backbone over T recursive steps. At each step, the backbone receives fixed encoded input x, containing context and query samples, and a same-dimensional latent state $s_t = [s_{c,t}; s_{q,t}]$. The state is initialized as $s_0 = [0; 0]$ and updated each step. After T steps, the final query state $s_{q,T}$ is decoded into the prediction:

$$s_{t+1} = F_\theta(s_t, x), \qquad \hat{y}_q = D(s_{q,T}). \tag{3}$$

TRM. TRM employs nested recursion. Instead of one latent state, it maintains a pair $s = (z, v)$: reasoning state z and solution state v, both with the same dimensionality as x. The inner loop performs n recursive updates to z, then one to v. The outer loop repeats this process T times to obtain v_T. The query portion of v_T is decoded as $\hat{y}_\mathrm{q} = D(v_{q,T})$.

Both architectures also use recursive computation during pre-training with the standard TFM prediction loss $\mathcal{L}(\hat{y}_\mathrm{q}, y_\mathrm{q})$.

5 Characterizing Recursive Refinement

The recursive TFMs evolve through a sequence of intermediate states:

$$s_0 \to s_1 \to \cdots \to s_T. \tag{4}$$

This raises a central question: *What effective learning rule does the model implement at each recursive step?* We first develop a kernel-based view of its predictions, and then analyze recursive evolution through the induced kernels. The model appears to perform Bayesian model selection across steps, a behavior that appears to emerge from TFM meta-learning-style pre-training rather than explicit training for that objective. We focus on regression, where the model's predictions can be naturally characterized by kernel regression.

5.1 A Kernel-Based View of the Model at Each Step

Prior work motivates a similarity-based view of TFMs, where predictions can be modeled as weighted combinations of context labels, with weights reflecting learned query–context similarities (Shaheen et al., 2026; Miftachov et al., 2026). We therefore hypothesize that a kernel predictor using K_t to capture the model's internal sample similarity can approximate its prediction at recursive step t. For regression, this approximation is:

$$\hat{y}_t \;\approx\; K_{qc,t}\big(K_{cc,t} + \lambda_t I\big)^{-1} y_c. \tag{5}$$

This standard kernel ridge regression (KRR) predictor is also the posterior mean of a Gaussian process (GP). Here, K_t induces context–context matrix $K_{cc,t}$ and query–context matrix $K_{qc,t}$. The KRR regularization coefficient λ_t corresponds to the GP noise-to-signal variance ratio. Equation 5 requires K_t to reflect the recursive TFM's similarity structure at step t.

Model-Induced Attention Kernels. We use attention to construct a measure of the model's notion of sample similarity: two samples are similar when they attend to the context similarly. Let sample i's attention profile $a_{t,i}$ at recursive step t be its attention weights over context samples, averaged across all Transformer layers. The model-induced kernel is the cosine similarity between two samples' profiles:

$$K_t(i,j) = \langle a_{t,i},\, a_{t,j} \rangle \big/ \big(\|a_{t,i}\|\, \|a_{t,j}\| \big). \tag{6}$$

Thus, samples are similar when their relative context-attention weights are similar. We construct query and context profiles differently:

- For each query sample, we use its attention weights over the full labeled context from the standard forward pass.
- For each context sample, directly using its profile would leak its observed label through its encoded representation. We therefore temporarily treat it as a query and compute its attention over the remaining labeled context, without direct access to its own label.

These profiles construct $K_{cc,t}$ and $K_{qc,t}$. Because a separate forward pass per context sample is expensive, we partition context samples into fixed groups and treat each group as queries in one pass. Remaining samples are labeled as context. Each profile therefore excludes other samples in its held-out group; we set their attention weights to zero when we map the profile to full context.

Validation of the Kernel Hypothesis. We empirically validate the approximation by comparing the model-induced attention kernel with null kernels that omit the model's internal similarity structure: a raw-input kernel from input features and an encoded-input kernel from initial representations. The attention kernel more closely approximates recursive TFM predictions than these alternatives, supporting our hypothesis.

5.2 Recursion as Implicit Bayesian Model Selection

The kernel-based view maps the recursive states in Equation 4 to model-induced kernel estimators:

$$(K_1, \lambda_1) \to (K_2, \lambda_2) \to \cdots \to (K_T, \lambda_T). \tag{7}$$

Under the Gaussian-process (GP) interpretation, marginal likelihood is a principled Bayesian criterion balancing fit to observed context targets against model complexity. The GP negative log marginal likelihood is:

$$\mathrm{NLML}_t \;=\; \underbrace{\frac{1}{2} y_c^\top \left(K_{cc,t} + \lambda_t I\right)^{-1} y_c}_{\text{Data fit } \mathcal{D}_t} \;+\; \underbrace{\frac{1}{2} \log\det\left(K_{cc,t} + \lambda_t I\right)}_{\text{Occam factor (complexity) } \mathcal{C}_t} \;+\; \text{const.} \tag{8}$$

The data-fit term $\mathcal{D}_t$ measures fit to observed context targets, while the Occam factor $\mathcal{C}_t$ accounts for model complexity. These terms generally oppose each other, yielding the Bayesian data fit–complexity trade-off. A priori, the induced kernels could evolve in any direction because training uses only supervised prediction loss and never exposes the model to this decomposition. Empirically, we find that recursion typically lowers the induced kernels' NLML. Rather than simply fitting the context more aggressively, it adjusts the data fit–complexity trade-off to improve marginal likelihood. This suggests that recursive refinement implements implicit Bayesian model selection emerging from TFM meta-learning-style pre-training.

6 Experiments

We first evaluate whether our attention-induced kernel predictors faithfully approximate model predictions and characterize how their GP marginal likelihood evolves across recursion. We then examine predictive performance across recursive steps, including beyond the trained horizon, and report overall benchmark performance to contextualize the models' predictive capabilities under our training protocol.

We study recursion in TFMs using TabDPT (Ma et al., 2025) as the backbone because its open-source pre-training pipeline fits our computational budget. We develop **TRM-TabDPT** and **Looped-TabDPT** and compare them with our primary baseline, **Standard TabDPT**. Our main comparison

Table 1: Agreement between model and kernel predictions across regression datasets and recursive or depth-wise steps. Results are averaged over steps, two folds, and five seeds per dataset; across 35 datasets, we report average RMSE rank and median [Q1, Q3] R^2.

Kernel	TRM-TabDPT		Looped-TabDPT		Standard TabDPT	
	avg rank ↓	Med [Q1, Q3] R^2 ↑	avg rank ↓	Med [Q1, Q3] R^2 ↑	avg rank ↓	Med [Q1, Q3] R^2 ↑
attn_cos	**1.26**	**0.98** [0.97, 0.99]	**1.35**	**0.98** [0.96, 0.99]	**1.39**	**0.97** [0.94, 0.98]
enc_x_rbf	2.49	0.94 [0.87, 0.97]	2.29	0.95 [0.89, 0.97]	2.59	0.94 [0.86, 0.96]
raw_x_rbf	3.09	0.92 [0.83, 0.97]	3.26	0.94 [0.87, 0.96]	3.03	0.92 [0.86, 0.96]
enc_x_lin	4.89	0.79 [0.60, 0.88]	4.78	0.83 [0.63, 0.91]	4.78	0.81 [0.68, 0.89]
enc_x_cos	4.90	0.79 [0.59, 0.88]	4.78	0.83 [0.63, 0.91]	4.76	0.81 [0.68, 0.89]
raw_x_lin	5.09	0.79 [0.58, 0.88]	5.25	0.81 [0.60, 0.87]	5.37	0.81 [0.65, 0.87]
raw_x_cos	6.27	0.72 [0.57, 0.84]	6.29	0.75 [0.59, 0.85]	6.07	0.74 [0.65, 0.85]

uses 6 Transformer layers in each recursive model and 16 in Standard TabDPT; we also report 8-layer recursive models and a 12-layer standard model. For a controlled comparison, we retrain all three models on identical pre-training data, batch size, sequence length, and context-sampling procedure. Each receives one NVIDIA A100 40GB GPU for six days. For both recursive models, we set $T = 6$: six loop iterations for Looped-TabDPT and six outer-loop iterations for TRM-TabDPT. Additional implementation details and an ablation of training recursion settings are provided in Appendix Sections A and B, respectively.

Evaluation Datasets and Metrics. Following TabDPT, we evaluate on two folds of each of the 72 OpenML-CC18 classification datasets (Bischl et al., 2017) and on the 35 OpenML-CTR23 regression datasets (Fischer et al., 2023). We report AUC and accuracy for classification, and Pearson correlation and R^2 for regression. The kernel and GP analyses use the CTR23 regression datasets.

Inference Setting. For kernel and GP analyses in Subsection 6.1, we subsample larger datasets to 1,500 context and 500 test samples, giving all test samples the same context; we repeat this for five seeds. Elsewhere, as in TabDPT, we select the nearest 2,048 samples as context for each test sample. We use one ensemble member in all experiments except in Subsection 6.3, where we use eight.

6.1 Empirical Evidence for Bayesian Model-Selection Behavior

We analyze whether recursive refinement moves the induced GP predictors toward higher marginal likelihood and how its data-fit and complexity terms evolve. For comparison, we construct a depth-wise Standard TabDPT trajectory to examine whether recursion and increasing Transformer depth exhibit different data fit–complexity trade-offs. Separate decoder heads on representations from layers 8, 10, 12, 14, and 16 produce five intermediate predictions characterizing Standard TabDPT across depth. Before analyzing GP marginal likelihood and its terms, we verify that the kernel predictors faithfully approximate the predictions of all three models along their trajectories.

Validating the Model-Induced Kernel. At every trajectory point, we measure agreement between the kernel predictor in Equation 5 and the model's predictions on the same query samples. This uses no ground-truth query labels; it measures only how faithfully each kernel reproduces the model's prediction. For each kernel, a grid search selects λ to minimize RMSE against model predictions. We compare the model-induced attention kernel with two null-kernel families that omit trajectory-dependent internal representations: kernels from preprocessed raw input features (*raw_x*) or initial feature-encoder representations (*enc_x*). We use RBF, linear, and cosine similarities for both null families. The attention kernel uses cosine similarity because it has a simple interpretation as attention-profile alignment and yields a fixed kernel trace.

In this experiment, we use two folds and five seeds, yielding ten fold–seed evaluations per dataset. In each evaluation, we compare kernels at every recursive or depth-wise trajectory point, excluding the first TRM-TabDPT step and the first two Looped-TabDPT steps because their predictions are nearly constant; Section 6.2 examines these early steps. At each evaluated point, we rank the seven kernels by prediction RMSE. We then average each kernel's rank and R^2 over trajectory points and the ten evaluations to obtain one value per dataset. Across the 35 CTR23 regression datasets, Table 1

Table 2: Net changes in the GP marginal-likelihood decomposition. For each of the 35 CTR23 regression datasets, endpoint changes are averaged over two folds and five seeds before counting datasets that satisfy each condition; brackets give the [min, max] counts across the ten fold–seed evaluations. D and C denote the data-fit and complexity terms, and Δ is the final minus initial value over each interval. Conditions overlap and describe endpoint changes, not trajectory monotonicity.

Observed endpoint change	TRM-TabDPT $t = 2 \to 6$	Looped-TabDPT $t = 3 \to 6$	Standard $\ell = 8 \to 16$
NLML decreases: $\Delta(D + C) < 0$	29/35 [29,31]	29/35 [26,29]	28/35 [26,29]
Data-fit term decreases: $\Delta D < 0$	19/35 [18,23]	29/35 [27,30]	35/35 [35,35]
Complexity term decreases: $\Delta C < 0$	28/35 [27,29]	18/35 [17,18]	0/35 [0,0]
NLML decreases despite an increasing data-fit term: $\Delta(D + C) < 0,\ \Delta D > 0$	13/35 [12,15]	6/35 [5,8]	0/35 [0,0]

reports mean rank and median [Q1, Q3] R^2. The attention kernel has the lowest mean rank for all three models: 1.26, 1.35, and 1.39 for TRM, Looped, and Standard TabDPT, respectively. Paired one-sided Wilcoxon tests show that our *attn_cos* achieves higher R^2 than the strongest null kernel, *enc_x_rbf*, across all three models (Holm-adjusted $p < 10^{-7}$).

The fitted λ_t exhibits dataset-specific values but changes little across recursive steps after $t = 2$ for TRM-TabDPT and $t = 3$ for Looped-TabDPT (Appendix Figure 5). Thus, the subsequent evolution is driven primarily by changes in the induced kernel rather than λ_t. For the following GP analysis, we therefore fix λ for each dataset at its value at the first evaluated step ($t = 2$ and $t = 3$, respectively).

Evolution of Data Fit and Complexity. Table 2 summarizes endpoint changes from the initial evaluated step to the trained endpoint for both recursive models, and from the initial to final evaluated depth for Standard TabDPT. For each dataset, we compute the endpoint change in each GP term for each of the two folds and five seeds, then average the ten changes before counting how many datasets satisfy each condition. The bracketed [min, max] values give the smallest and largest dataset counts obtained by applying each condition separately to the ten fold–seed evaluations.

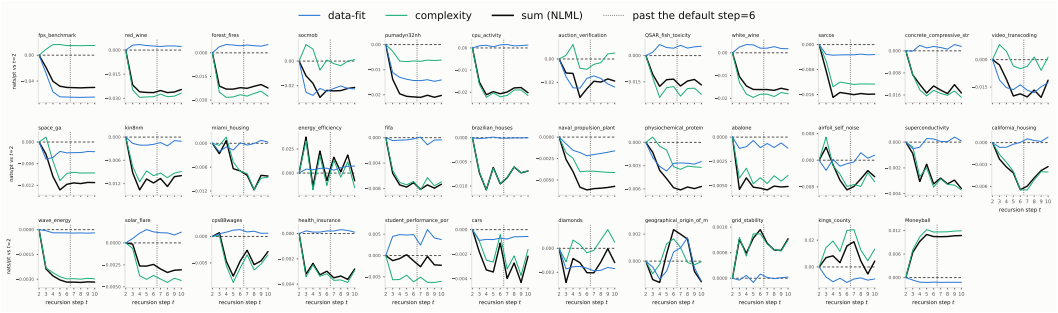
Both recursion and increasing standard-model depth frequently improve the induced GP marginal likelihood, but their decompositions differ. For Standard TabDPT, the data-fit term decreases on all 35 datasets while complexity increases on all 35. TRM-TabDPT instead reduces complexity on 28 datasets; on 13, this reduction more than offsets an increased data-fit term, lowering total NLML. Looped-TabDPT more often improves data fit (29 datasets), but also reduces complexity on 18 datasets and lowers NLML despite worse data fit on 6. These results provide empirical evidence for a Bayesian model-selection interpretation of recursive refinement: improvements in the induced GP score are not driven by better data fit alone. Figure 2 shows intermediate reversals and dataset-dependent behavior, so endpoint improvements do not imply monotonic optimization at every step.

6.2 Predictive Performance across Recursion Steps

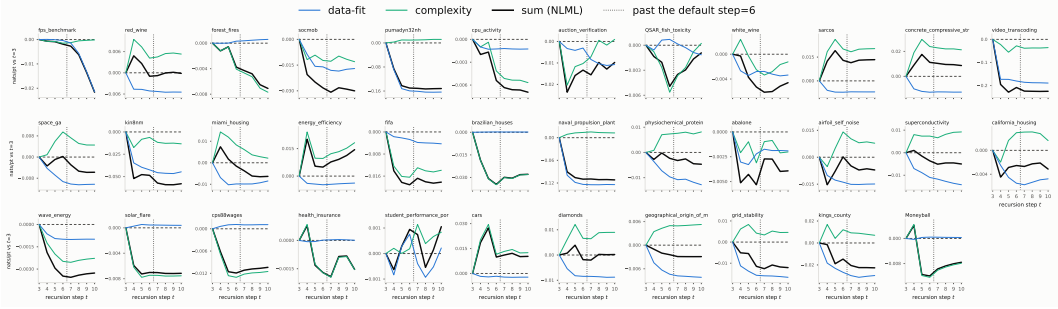
Table 3 reports the interquartile mean (IQM) of each metric across benchmark datasets for TRM and Looped-TabDPT at each recursive step. Predictions are initially poor at step 1 for TRM-TabDPT and steps 1–2 for Looped-TabDPT. Performance then rises sharply and largely plateaus by step 6, the pre-training horizon. Additional inference-time steps offer little benefit and slightly reduce some metrics, making the training horizon a suitable inference point. Refinement varies across datasets. Figure 3 shows three classification and three regression examples for TRM-TabDPT. On the two binary classification datasets, most improvement occurs from steps 1 to 2, with little or no later benefit; the five-class gesture dataset improves over several steps. Regression examples generally improve early and stabilize around step 6. Thus, some datasets benefit mainly from early steps, while others continue to benefit through all six. Appendix Figure 7 shows Looped-TabDPT.

6.3 Overall Benchmark Performance

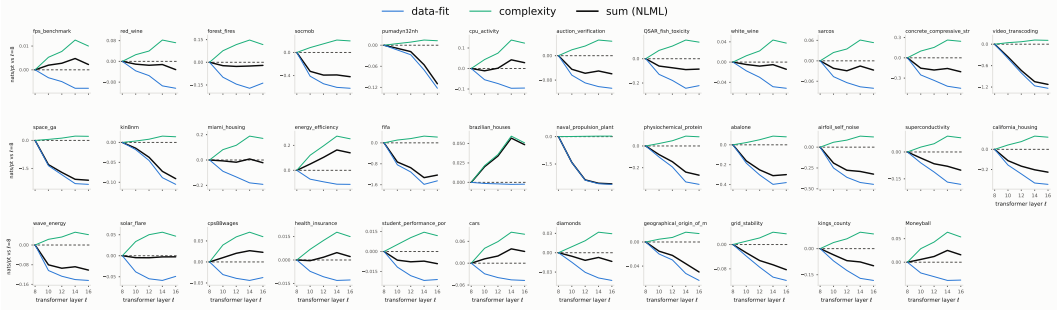
Table 4 reports the IQM of each metric across benchmark datasets. The upper blocks compare recursive and standard TabDPT under matched pre-training; the bottom block includes other models



(a) **TRM-TabDPT.** Changes over $t = 2, \ldots, 10$ relative to $t = 2$.



(b) **Looped-TabDPT.** Changes over $t = 3, \ldots, 10$ relative to $t = 3$.



(c) **Standard TabDPT.** Changes over trained decoder heads at layers $\ell \in \{8, \ldots, 16\}$ relative to $\ell = 8$.

Figure 2: GP marginal-likelihood trajectories for 35 CTR23 regression datasets (fold 0, seed 0). Each plot shows changes in NLML (black), data fit (blue), and complexity (green); lower NLML indicates higher GP marginal likelihood. For recursive models, the dotted vertical line marks the pre-training horizon $T = 6$; later steps extend recursion beyond it. Values are in nats per context point; small per-point changes can correspond to substantial differences over the full context.

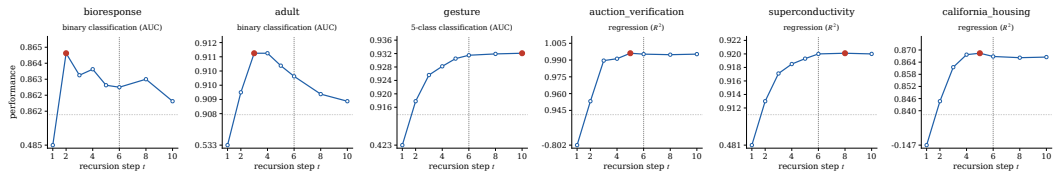


Figure 3: Performance across recursion steps for TRM-TabDPT on example datasets.

for reference. Both recursive variants match or exceed the stronger 16-layer standard model on every metric. Gains are largest on CTR23 regression: the six-layer TRM improves correlation from 0.896 to 0.917 and R^2 from 0.807 to 0.843. The six-layer Looped model reaches the same correlation 0.917 and nearly the same R^2 0.842, while attaining the highest classification accuracy

Table 3: Performance (IQM) at each recursion step t, using one ensemble member. Models are trained with $T = 6$; $t = 8, 10$ extend recursion at inference. Bold marks each column's best value.

	TRM-TabDPT				Looped-TabDPT			
	CC18		CTR23		CC18		CTR23	
Step	AUC	Acc.	Corr.	R^2	AUC	Acc.	Corr.	R^2
$t = 1$	0.5146	0.3381	0.2522	-0.4703	0.4953	0.1992	0.4921	0.1726
$t = 2$	0.9704	0.9087	0.8938	0.8039	0.5640	0.2878	0.7752	0.4734
$t = 3$	0.9724	0.9137	0.9129	0.8359	0.9695	0.9011	0.8887	0.7933
$t = 4$	0.9731	0.9171	0.9147	0.8396	0.9723	0.9142	0.8968	0.8078
$t = 5$	**0.9735**	0.9177	0.9148	0.8398	**0.9734**	0.9169	0.9109	0.8317
$\mathbf{t = 6}$	**0.9735**	**0.9180**	**0.9152**	**0.8404**	**0.9734**	0.9186	0.9151	0.8401
$t = 8$	0.9733	0.9174	0.9150	0.8400	0.9729	0.9179	**0.9158**	**0.8415**
$t = 10$	0.9732	0.9175	**0.9152**	0.8403	0.9727	**0.9189**	0.9154	0.8405

Table 4: Benchmark performance (IQM) of recursive and standard TabDPT pre-trained under matched settings, using eight ensemble members. Other models, trained under different settings, are included for reference.

Model	CC18-AUC	CC18-Acc.	CTR23-Corr.	CTR23-R^2
TRM-TabDPT (8-layer)	0.975	0.925	0.916	0.840
TRM-TabDPT (6-layer)	0.975	0.926	0.917	0.843
Looped-TabDPT (8-layer)	0.976	0.925	0.914	0.837
Looped-TabDPT (6-layer)	0.976	0.927	0.917	0.842
Standard TabDPT (16-layer)	0.975	0.924	0.896	0.807
Standard TabDPT (12-layer)	0.974	0.920	0.894	0.796
TabDPT v1.1	0.976	0.928	0.920	0.847
TabDPT v1.0	0.972	0.917	0.911	0.831
TabPFN v2	0.972	0.917	0.917	0.841
TabPFN	0.939	0.852	N/A	N/A
TabR	0.967	0.923	0.909	0.825
MLP-PLR	0.967	0.914	0.907	0.827
MLP	0.915	0.865	N/A	N/A
XGBoost	0.965	0.910	0.904	0.820
LightGBM	0.964	0.906	0.900	0.809
CatBoost	0.964	0.908	0.897	0.802

among the matched models 0.927. The recursive variants are also competitive with public TabDPT checkpoints, although these are *not* direct baselines because their pre-training differs substantially. For example, TabDPT v1.1 was trained on eight GPUs, enabling a much larger pre-training context size than our single-GPU setup. Increasing the standard model from 12 to 16 layers improves all four metrics, whereas six layers match or slightly outperform eight for both recursive variants. The stronger results with six-layer recursive models suggest that a larger backbone is not necessary to benefit from recursion, consistent with Jolicoeur-Martineau (2025)'s finding that a smaller recursive model can generalize better.

7 Conclusion

We introduced recursive reasoning TFMs that revisit the context to refine predictions across inference steps. Both TRM-TabDPT and Looped-TabDPT matched or exceeded Standard TabDPT across classification and regression benchmarks, with the largest gains on regression. Using attention-induced kernel predictors that closely reproduce the models' predictions, we found that recursive refinement typically improves the induced GP marginal likelihood by adjusting the balance between data fit and model complexity. These findings provide evidence for Bayesian model-selection behavior emerging from predictive pre-training, without explicit marginal-likelihood optimization. Developing a theoretical understanding of how predictive pre-training gives rise to this behavior is a promising direction for future work.

AI USE STATEMENT

We used ChatGPT to improve the writing and provide feedback on research ideas. We used Claude Code to assist with code. We reviewed the AI-assisted text and code. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The source code for all experiments is submitted in the Supplementary Material. The main text describes the models, kernel analysis, and evaluation protocol; Appendix A provides additional implementation details. The benchmark datasets are publicly available through the cited references.

REFERENCES

Amir Rezaei Balef, Mykhailo Koshil, and Katharina Eggensperger. Is one layer enough? understanding inference dynamics in tabular foundation models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=IxpVGcg39J.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017.

Ying Fan, Anej Svete, and Kangwook Lee. Bridging the gap between latent and explicit reasoning with looped transformers. *arXiv preprint arXiv:2606.31779*, 2026.

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

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

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? *arXiv preprint arXiv:2410.08292*, 2024.

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

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models, 2025. URL https://arxiv.org/abs/2511.08667.

Chi Han, Ziqi Wang, Han Zhao, and Heng Ji. Understanding emergent in-context learning from a kernel regression perspective. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=6rD50Q6yYz.

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*, 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*, 01 2025. doi: 10.1038/s41586-024-08328-6. URL https://www.nature.com/articles/s41586-024-08328-6.

Jianhao Huang, Zixuan Wang, and Jason D Lee. Transformers learn to implement multi-step gradient descent with chain of thought. *arXiv preprint arXiv:2502.21212*, 2025.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Siyang Liu and Han-Jia Ye. TabPFN unleashed: A scalable and effective solution to tabular classification problems. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=5DD3RCcVcT.

Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony Caterini. In-context data distillation with tabpfn. *arXiv preprint arXiv:2402.06971*, 2024.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Jesse C Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L Caterini, and Maksims Volkovs. Tabdpt: Scaling tabular foundation models on real data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

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

Nour Shaheen, Junwei Ma, Alex Labach, Frank Hutter, Valentin Thomas, and Anthony L Caterini. Understanding the surprising generalization properties of tabular foundation models. *arXiv preprint arXiv:2608.17957*, 2026.

Zhaiming Shen, Alexander Hsu, Rongjie Lai, and Wenjing Liao. Understanding in-context learning on structured manifolds: Bridging attention to kernel methods. In *International Conference on Learning Representations*, volume 2026, pp. 42067–42103, 2026.

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

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Derek Qiang Xu, F Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2fojNANZSv.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International conference on learning representations*, volume 2024, pp. 42195–42214, 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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yuchen Zeng, Tuan Dinh, Wonjun Kang, and Andreas C Mueller. Tabflex: Scaling tabular learning to millions with linear attention. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=d60cmFf89H.

Xiyuan Zhang, Danielle C. Maddix, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael W. Mahoney, Cuixiong Hu, Huzefa Rangwala, George Karypis, and Bernie Wang. Mitra: Mixed synthetic priors for enhancing tabular foundation models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=t8YRsWY6HM.

A Additional Implementation Details

Figure 4 summarizes the recursive inference procedures for Looped-TabDPT and TRM-TabDPT.

For pre-training Looped-TabDPT, we set $T = 6$ and use a gradient window of four, backpropagating through only the last four iterations. For TRM-TabDPT, we use $n = 3$ inner updates, two deep-recursion steps, and three deep-supervision steps, yielding $T = 2 \times 3 = 6$ outer steps. This nesting applies only during training; inference uses a flat sequence of six outer steps. We do not use early stopping because varying context sizes across batches complicate replacing halted tasks with new ones. For both recursive models, replacing TabDPT's pre-norm structure with a post-norm structure using RMSNorm improves training stability. This change did not improve Standard TabDPT, so we retain its default normalization. We evaluate recursive backbones with six and eight Transformer layers. All models use a training seed of 0, a batch size of 32, a sequence length of 1,536, and a context size sampled uniformly between 50 and 1,024 for each batch. We train recursive models for 2,000 epochs and standard models for 10,000 epochs. Since recursive epochs are more computationally expensive, these schedules yield approximately equal training time under the same six-day, single-GPU budget. Except for the architectural changes and training schedules described above, all settings follow the public TabDPT implementation and are shared across models. For attention-kernel construction, we partition the context into 8 groups.

(a) Looped Transformer

$s \leftarrow s_0$
for $t = 1, \ldots, T$ **do**
  $s \leftarrow F_\theta(s,\, x)$ ▷ refine state
$\hat{y}_{\mathrm{q}} \leftarrow D(s_q)$ ▷ decode the query portion

(b) Tiny Recursive Model

$z,\, v \leftarrow z_0,\, v_0$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $z \leftarrow F_\theta(z,\, v,\, x)$ ▷ reasoning update
  $v \leftarrow F_\theta(z,\, v)$ ▷ solution update
$\hat{y}_{\mathrm{q}} \leftarrow D(v_q)$ ▷ decode the query portion

Figure 4: Looped Transformer (Yang et al., 2024) and TRM (Jolicoeur-Martineau, 2025).

B Ablation of Training Recursion Settings

Tables 5 and 6 provide an ablation on different training recursion settings. A moderate amount of recursion yields strong performance, whereas increasing the number of steps further does not consistently improve the results. Our objective is not to optimize the recursion settings for the best possible benchmark performance, but rather to show that recursive reasoning can improve predictive performance and that its dynamics exhibit behavior consistent with Bayesian model selection.

Table 5: Ablation of training recursion settings for TRM-TabDPT. Parenthesized values indicate the configuration (deep-recursion steps $\times$ deep-supervision steps), whose product gives the number of outer steps. We use three inner steps and six outer steps (2×3) in the main experiments.

Setting	CC18-AUC	CC18-Acc.	CTR23-Corr.	CTR23-R^2
inner:1, outer:6 (2×3)	0.9750	0.9258	0.9148	0.8366
inner:3, outer:1 (1×1)	0.9757	0.9283	0.9136	0.8357
inner:3, outer:3 (1×3)	0.9751	0.9257	0.9154	0.8391
inner:3, outer:4 (2×2)	0.9757	0.9278	0.9175	0.8433
inner:3, outer:6 (2×3)	0.9751	0.9255	0.9169	0.8433
inner:3, outer:9 (3×3)	0.9754	0.9249	0.9133	0.8375

C Additional Results

Fitted λ_t. Figure 5 shows the fitted λ_t across recursive steps and standard-model depths.

Looped-TabDPT Refinement. Figures 6 and 7 complement the TRM results with toy regression examples and dataset-level predictive trajectories for Looped-TabDPT. In Figure 6, $t = 3$ is still

Table 6: Ablation of training recursion settings for Looped-TabDPT. We use six loop steps in the main experiments.

Setting	CC18-AUC	CC18-Acc.	CTR23-Corr.	CTR23-R^2
n_loop:4	0.9760	0.9290	0.9137	0.8386
n_loop:6	0.9758	0.9270	0.9171	0.8415
n_loop:10	0.9757	0.9266	0.9154	0.8394

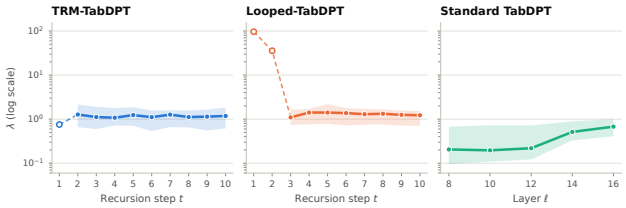
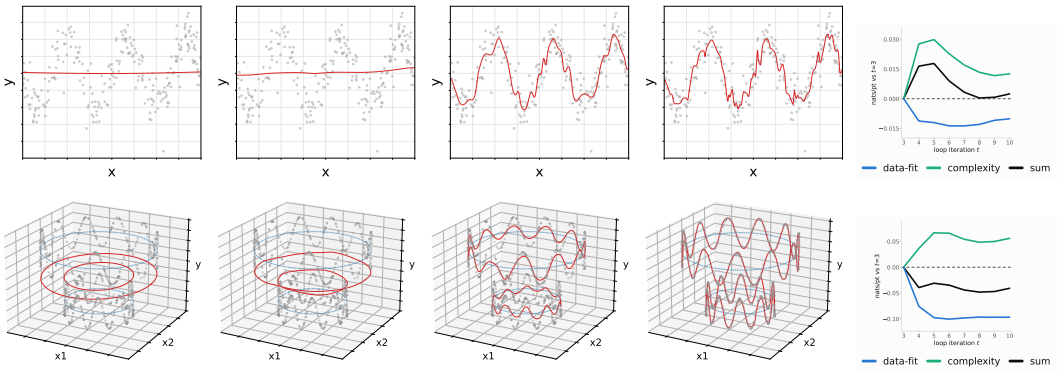


Figure 5: λ across recursive steps or layers. For each dataset, $\log_{10} \lambda$ is averaged over two folds and five seeds; lines show the median across datasets and bands the interquartile range. Dashed segments with hollow markers mark the initial steps. After the initial steps, λ stays near 1 in both recursive models, whereas in Standard TabDPT it rises about $3.7\times$ from layer 8 to 16.

transitional: from $t = 3$ to $t = 4$, data fit improves sharply while complexity rises by more, so NLML temporarily increases as the prediction rule forms. From $t = 4$ onward, NLML declines, so marginal-likelihood improvement begins once the prediction rule has formed. As noted previously, we do not claim that NLML improves at every step.



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 6$ (e) GP tracking

Figure 6: Recursive refinement by Looped-TabDPT on two toy regression tasks. (a)–(d) show context samples (gray) and model predictions (red) at recursion steps $t = 1, 2, 3, 6$. (e) shows changes relative to $t = 3$ in the induced GP's data-fit term (blue), complexity term (green), and their sum, NLML (black). $t = 1, 2$ are omitted from the GP analysis as their predictions are nearly constant.

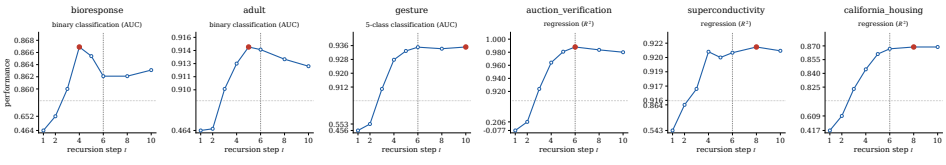


Figure 7: Performance across recursion steps for Looped-TabDPT on example datasets.