Tydra: An Efficient Hybrid Model for Tabular Data

Mieszko Komisarczyk[1]*, **Saurabh Mathur**[1]*, **Maurice Kraus**[1], **Sriraam Natarajan**[2], **Kristian Kersting**[1,3,4]

[1] Department of Computer Science, Technical University of Darmstadt, Germany
[2]Department of Computer Science, The University of Texas at Dallas, USA
[3] Hessian Center for Artificial Intelligence (hessian.ai), Darmstadt, Germany
[4]German Research Center for AI (DFKI)

Abstract

Transformer-based tabular foundation models such as TabPFN achieve strong predictive performance but incur quadratic computational cost with context length. On the other hand, subquadratic SSM-based alternatives such as Hydra trade away accuracy for efficiency. To balance both, we introduce Tydra, a hybrid Transformer–State Space Model (SSM) architecture for tabular in-context learning that interleaves attention and SSM layers. Across 30 OpenML datasets, Tydra reduces inference time by 30% relative to TabPFN while retaining much of its predictive performance. Tydra also outperforms an approximately ten-times-larger Hydra model while providing faster inference. The results indicate that hybrid architectures are a promising direction for tabular foundation models.

Introduction

Tabular foundation models such as TabPFN have achieved strong predictive performance on several tabular prediction tasks without task-specific training (Hollmann et al. 2025). However, TabPFN's Transformer backbone scales quadratically with context length, making inference prohibitively expensive on large-scale or long-context tabular tasks. This cost is especially restrictive for institutions that cannot use server-hosted inference at all. Hospitals and other regulated institutions are often barred from sending patient records to an external server for data protection reasons and must instead run inference locally on hardware with limited compute resources. While the Hydra architecture reduces this cost with a subquadratic State Space Model (SSM) architecture, its predictive performance falls short of TabPFN (Hwang et al. 2024; Koch et al. 2025). This gap persists even as Hydra is scaled up; shrinking TabPFN to match Hydra's efficiency degrades its accuracy. Neither pure architecture achieves both high accuracy and low inference cost.

Hybrid architectures that interleave attention and SSM layers have recently improved this accuracy–efficiency trade-off for language models (Merrill et al. 2026), though such architectures remain unexplored for tabular foundation models. Tabular data are permutation-invariant over rows and columns, unlike the fixed-order sequences of language for these hybrid architectures were designed. To this end, we introduce Tydra, which interleaves TabPFN's attention layers with Hydra's SSM layers. We show that this hybridization **matches TabPFN's accuracy at substantially lower inference cost on small-to-medium-scale data**, making strong tabular in-context learning practical in settings where server-hosted inference is not an option.

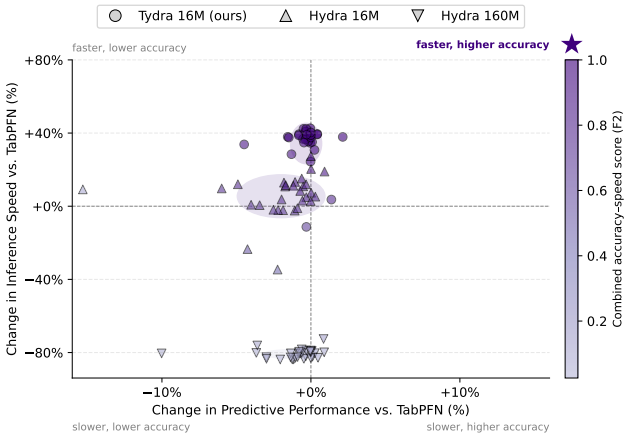

Figure 1: **Tydra is fast on tabular data without sacrificing predictive performance.** Inference speed and predictive performance of Tydra compared to TabPFN and Hydra on OpenML datasets. Color encodes a combined accuracy–speed score (F2, weighted toward speed). Shaded ellipses span one standard deviation around each model's mean. The star marks the ideal corner: fastest and most accurate. Hydra 16M is fast but trails substantially in predictive performance, while TabPFN and Hydra 160M are accurate but slow; Tydra achieves predictive performance close to TabPFN and Hydra 160M at over 2× their inference speed.

To summarize, we make the following contributions:

(1) We introduce Tydra, the first hybrid Transformer–SSM architecture for tabular in-context learning.

(2) We show that Tydra matches TabPFN's accuracy at up to 30% lower inference time on 30 OpenML datasets, while substantially outperforming Hydra.

(3) We provide an extensive study on the architecture family of Tydras showing the advantages of different combinations and ratios of Hydra and TabPFN layers.

We proceed as follows. We start off by discussing related work. We then present the Tydra architecture. Before concluding, we present empirical results.

*These authors contributed equally.

Related Work

Tydra is related to several lines of works, namely, TabPFN, state space models, and hybrid models.

TabPFN

TabPFN (Tabular Prior-Data Fitted Network) is a tabular foundation model that performs tabular classification via in-context learning. Given an entire training set and test queries, it makes predictions via a single forward pass. Its architecture is based on the Transformer, with self-attention adapted for permutation invariance across rows and columns. This adapted Transformer is meta-trained offline on millions of synthetic classification tasks, yielding strong predictive accuracy across a wide range of tabular datasets without any dataset-specific training. However, since self-attention scales quadratically with the number of rows in a tabular dataset, inference can be computationally expensive.

Since the original release, several extensions have targeted TabPFN's scope and scaling. TabPFN v2 (Hollmann et al. 2025) broadened the original classification-only model (Hollmann et al. 2022) to regression and increased model capacity, while later revisions, TabPFN v2.5 and v3 (Grinsztajn et al. 2026) roughly doubled transformer depth and raised the supported class count. These successive versions improve accuracy and task coverage, but retain the same attention mechanism and therefore inherit its quadratic cost in the number of rows. Recognizing this bottleneck, a separate line of work has proposed *post-hoc* strategies to extend TabPFN's practical reach without retraining or architectural change: subsampling and divide-and-conquer schemes that partition large datasets into TabPFN-sized chunks and aggregate predictions (Ye, Liu, and Chao 2026), and domain-specific adaptations such as TabPFN-TS for time-series forecasting (Hoo et al. 2025). These efforts confirm that the scaling limitation is widely recognized, but they work around the transformer's row-scaling behavior rather than removing it. In contrast, our approach addresses the bottleneck at the architectural level.

State Space Models

State Space Models (SSMs) have emerged as an effective approach to overcoming the limitations of both RNNs and Transformers. Derived from continuous-time dynamical systems, they achieve near-linear complexity in sequence length (Somvanshi et al. 2025). Mamba (Gu and Dao 2023) is a prominent example, introducing a selective state space mechanism that adaptively filters information across the sequence; this selectivity allows Mamba to achieve linear-time inference while matching or exceeding Transformer performance across several domains (Gu and Dao 2023). Hydra (Hwang et al. 2024) extends Mamba to non-causal settings via quasiseparable matrix mixers, enabling bidirectional context aggregation while retaining Mamba's efficiency — a property particularly relevant for tabular data, where rows have no natural sequence order. Hydra has previously been adapted to the tabular setting (Koch et al. 2025). While TabPFN outperforms Hydra in both inference speed and predictive accuracy on standard small-to-medium-scale datasets, Hydra becomes the better choice at larger dataset sizes, where TabPFN's quadratic cost becomes prohibitive and Hydra's subquadratic scaling allows it to remain practical. This suggests that SSMs offer a genuine efficiency advantage over attention-based tabular foundation models but at the cost of predictive performance, especially in standard small-to-medium scale datasets.

Hybrid models

Hybrid architectures that combine transformer layers with more efficient, subquadratic layers have emerged as an effective solution for the efficiency–accuracy tradeoff. Hybrid language models have achieved promising results across common reasoning, math, and coding tasks (Merrill et al. 2026; Li et al. 2026). Several hybridization strategies have been proposed, including interleaving transformer and state-space blocks (Lieber et al. 2024), reusing a small number of attention blocks across an otherwise recurrent backbone (Glorioso et al. 2024), and computing recurrent and attention branches in parallel over the same input (Dong et al. 2025). However, hybridization has so far been explored almost exclusively for language modeling, with only limited extensions to other modalities (Zhu et al. 2024), and, to our knowledge, no prior work has introduced a hybrid architecture for tabular foundation models.

Tydra precisely addresses this gap, adopting the interleaved strategy to combine TabPFN's attention layers with Hydra's state-space layers.

The Tydra Family of Architectures

We now present *Tydra*, a family of hybrid architectures for tabular in-context learning that interleaves attention layers with Hydra's state-space (SSM) layers. This section describes Tydra's architecture, model initialization, and training.

Hybrid Architecture

Figure 2 illustrates Tydra[1], our hybrid prior-fitted architecture. Without losing of generality we refer to the fully interleave Tydra {4 HT} model as Tydra. Other variants of Tydra can be represented by modifying the backbone part.

Given training features X_{train}, training labels Y_{train}, and test features X_{test}, Tydra estimates

$$p(Y_{\text{test}} \mid X_{\text{train}}, Y_{\text{train}}, X_{\text{test}}) .$$

Tydra combines the efficient bidirectional sequence mixing of Hydra with the content-dependent interactions provided by self-attention.

Each table row is represented by one token. Features and labels are embedded separately using linear encoders,

$$E_x : \mathbb{R}^n \to \mathbb{R}^m, \qquad E_y : \mathbb{R} \to \mathbb{R}^m,$$

where in our particular case $n = 10$ and $m = 512$. For a labeled training row (x_i, y_i), the initial representation is formed by adding the two embeddings,

$$z_i^{(0)} = E_x(x_i) + E_y(y_i),$$

[1]We provide the code and implementation details in the Appendix.

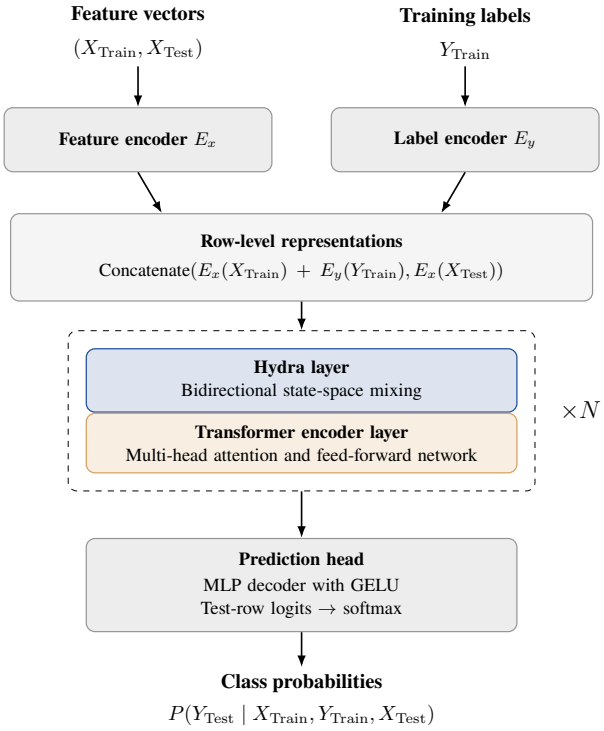

Figure 2: **Tydra's Hybrid Architecture.** Feature vectors are projected into row-level embeddings, with label embeddings added for training rows. The resulting sequence passes through four Hydra–Transformer encoder pairs (eight layers total), each ordered as Hydra followed by attention. An MLP decoder produces class logits for the test rows, which are converted into class probabilities by softmax.

whereas an unlabeled test row is represented only by its feature embedding, $z_i^{(0)} = E_x(x_i)$. Thus, training and test rows share the same m-dimensional token space while labels are revealed only for the training rows.

The resulting sequence is processed by $K = 4$ ordered Hydra–TabPFN pairs, giving eight layers in total:

$$\underbrace{[\,\text{Hydra} \to \text{TabPFN}\,] \times K}_{\text{2K layers}}.$$

Every layer preserves the m-dimensional representation, allowing Hydra and attention layers to be interleaved without additional projection layers. The Hydra layers use the bidirectional state-space mixer of Hwang et al. (2024), following its adaptation to tabular prior-fitted networks by Koch et al. (2025). Each Transformer layer contains multi-head self-attention followed by a two-layer feed-forward network with hidden dimension $2m$. Both sublayers use residual connections and layer normalization.

Finally, the representations corresponding to the test rows are passed through a two-layer MLP,

$$\mathbb{R}^m \to \mathbb{R}^{2m} \to \mathbb{R}^C,$$

with a GELU activation, where C denotes number of classes. The resulting class logits are normalized with a softmax to obtain predictive class probabilities.

Bidirectional State-Space Mixing

Tabular data has no canonical row order, *that is,* permuting the rows of X_{Train} should not change the model's predictions. This *permutation invariance* distinguishes language from tabular in-context learning. Standard SSMs process a sequence left-to-right. So, naively substituting an SSM with TabPFN's set-transformer layers would break permutation invariance. Hydra layers resolve this by replacing the standard SSM with a bidirectional state-space mixer that lets every row attend to every other row, in either direction, while still admitting linear-time evaluation.

Let L denotes sequence length and D number of channels. Formally, a causal selective SSM (as used in unidirectional Mamba) applied to $\mathbf{X} \in \mathbb{R}^{L\times D}$ acts as a matrix mixer $\mathbf{Y} = \mathbf{M}\mathbf{X}$, where entry m_{ts} of the mixer matrix is

$$m_{ts} = \mathbf{c}_t^\top \Big(\textstyle\prod_{k=s+1}^{t} \mathbf{A}_k \Big) \mathbf{b}_s, \qquad t \geq s, \tag{1}$$

with data-dependent $\mathbf{A}_k \in \mathbb{R}^{N\times N}$ and $\mathbf{b}_k, \mathbf{c}_k \in \mathbb{R}^N$ the discretized state, input, and output matrices at position k, and N the SSM state dimension. Such $\mathbf{M}$ are N-*semiseparable*: every submatrix taken from the lower triangle has rank at most N, which is exactly what permits an $O(L)$ recurrent evaluation, but also forces $\mathbf{M}$ to be strictly causal (zero above the diagonal).

Hydra (Hwang et al. 2024) lifts this restriction by parameterizing the mixer as a *quasiseparable* matrix,

$$m_{ts} = \begin{cases} \mathbf{c}_t^\top \big(\prod_{k=s+1}^{t} \mathbf{A}_k \big) \mathbf{b}_s, & t > s, \\ \delta_t, & t = s, \\ \overleftarrow{\mathbf{c}}_t^\top \big(\prod_{k=t}^{s-1} \overleftarrow{\mathbf{A}}_k \big) \overleftarrow{\mathbf{b}}_s, & t < s, \end{cases} \tag{2}$$

i.e., a lower-triangular semiseparable block identical to a causal SSM, an independently parameterized upper-triangular semiseparable block built from a second, reversed set of state matrices $\overleftarrow{\mathbf{A}}_k$, $\overleftarrow{\mathbf{b}}_k$, $\overleftarrow{\mathbf{c}}_k$, and free diagonal terms δ_t. Crucially, the rank bound defining quasiseparable matrices holds only for the strictly upper- and lower-triangular submatrices, not across the diagonal as for semiseparable matrices – so this parameterization is strictly more expressive than naive bidirectional SSMs that tie the forward and backward passes together through a shared diagonal, while still admitting sub-quadratic evaluation.

In practice, the quasiseparable mixer is never materialized directly; instead it is realized by combining two ordinary causal SSM (semiseparable) passes, $\mathrm{QS}(\mathbf{X}) =$

$$\mathrm{shift}\big(\mathrm{SS}(\mathbf{X})\big) + \mathrm{flip}\Big(\mathrm{shift}\big(\mathrm{SS}(\mathrm{flip}(\mathbf{X}))\big)\Big) + \mathbf{D}\mathbf{X}, \tag{3}$$

where $\mathrm{SS}(\cdot)$ denotes a standard causal SSM scan, $\mathrm{flip}(\cdot)$ reverses the input sequence, $\mathrm{shift}(\cdot)$ shifts it one position forward with zero padding at the start, and $\mathbf{D} = \mathrm{diag}(\delta_1, \ldots, \delta_L)$ collects the learned diagonal terms. Any semiseparable SSM can play the role of $\mathrm{SS}(\cdot)$; following Hwang et al. (2024), we instantiate it with SSD (structured

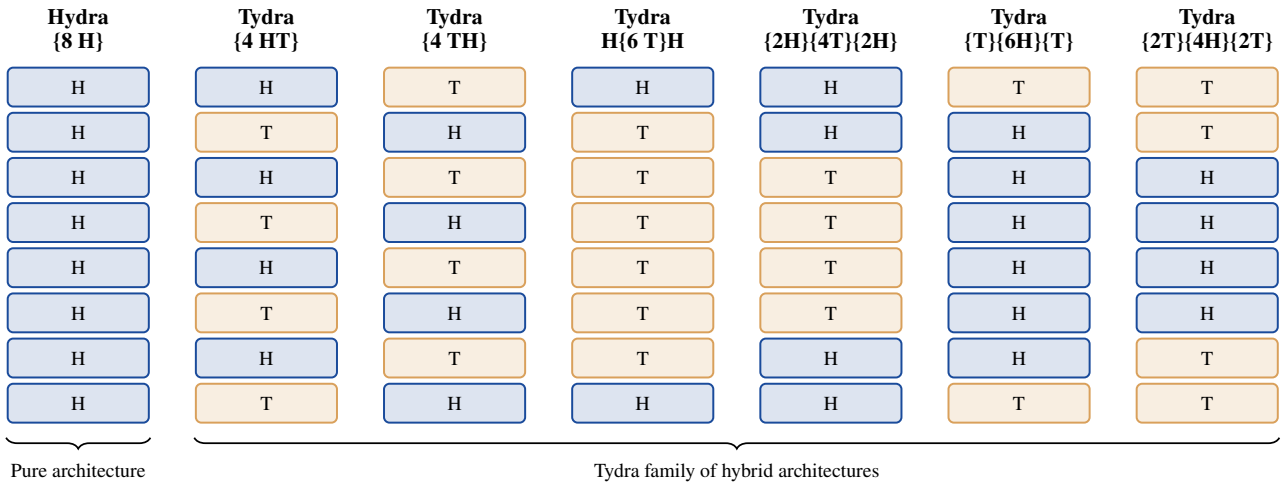

Figure 3: **The Tydra Family of Architectures.** Architectures of the evaluated Hydra and Tydra models. Blocks are ordered from the input layer at the bottom to the output layer at the top. T denotes a Transformer attention layer, and H denotes a Hydra layer.

state-space duality, Dao and Gu 2024), chosen for its linear-time, hardware-efficient scan. Concretely, each Hydra layer in Tydra runs a forward SSD scan over $\mathbf{X}$ and a second SSD scan over $\mathrm{flip}(\mathbf{X})$, then re-aligns the reversed output via $\mathrm{flip} \circ \mathrm{shift}$ before summing both directional contributions with the diagonal (residual) term. This lets every row attend to every other row in the table, matching the expressivity of full self-attention while retaining $O(L)$ time and memory in the number of rows L.

Training

We meta-train Tydra offline using the prior-data fitted network pipeline adopted by Koch et al. (2025). Training is performed on synthetic tasks generated on the fly using the neural-network component of the TabPFN prior framework. A synthetic task

$$\mathcal{T}_t = (\mathbf{X}_t, \mathbf{y}_t) \tag{4}$$

is one complete, randomly generated tabular classification dataset, where $\mathbf{X}_t \in \mathbb{R}^{n_t \times d_t}$ contains its feature values and $\mathbf{y}_t \in \{1, \ldots, K_t\}^{n_t}$ contains its class labels.

Let $\boldsymbol{\phi}_t$ collect the hyperparameters of the neural-network generator used to produce task t, including its architecture, activation functions, dropout probability, weight-initialization scale, and noise level. We use $p_{\mathrm{HP}}(\boldsymbol{\phi})$ to denote the configured hyperprior from which these generator hyperparameters are sampled. Conditional on $\boldsymbol{\phi}_t$, the distribution $p_{\mathrm{NN}}(\mathcal{T} \mid \boldsymbol{\phi}_t)$ describes the datasets produced by the resulting neural-network generator. Task generation can therefore be written as

$$\boldsymbol{\phi}_t \sim p_{\mathrm{HP}}(\boldsymbol{\phi}), \qquad \mathcal{T}_t \sim p_{\mathrm{NN}}\left(\mathcal{T} \mid \boldsymbol{\phi}_t\right). \tag{5}$$

For each sampled task $\mathcal{T}_t$, the observations are divided into a labeled context set

$$\mathcal{C}_t = \{(\mathbf{x}_{t,j}, y_{t,j})\}_{j=1}^{N_t} \tag{6}$$

and a query set

$$\mathcal{Q}_t = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^{M_t}. \tag{7}$$

Given the context and query features, Tydra produces query logits

$$\mathbf{z}_{t,i} = f_\theta(\mathcal{C}_t, \mathbf{x}_{t,i}). \tag{8}$$

For a minibatch of B tasks, we minimize the unweighted multiclass cross-entropy $\mathcal{L}(\theta) =$

$$-\frac{1}{B}\sum_{t=1}^{B}\frac{1}{M_t}\sum_{i=1}^{M_t}\log\left[\mathrm{softmax}(\mathbf{z}_{t,i})\right]_{y_{t,i}}. \tag{9}$$

Thus, the loss is averaged over the query rows of each task and over the tasks in the minibatch.

At inference time, the labeled training data form the context $\mathcal{C}$, while the test features form the queries. Using the fixed meta-trained parameters $\theta^\star$, the predictive distribution for a test example is

$$p_{\theta^\star}(y_i = k \mid \mathbf{x}_i, \mathcal{C}) = \left[\mathrm{softmax}\left(f_{\theta^\star}(\mathcal{C}, \mathbf{x}_i)\right)\right]_k, \tag{10}$$

and the predicted class is

$$\hat{y}_i = \arg\max_k p_{\theta^\star}(y_i = k \mid \mathbf{x}_i, \mathcal{C}). \tag{11}$$

We largely follow the training protocol of Koch et al. (2025), retaining its synthetic-task generation and classification objective as well as AdamW optimization with a learning rate of 10^{-4}, an effective batch size of 64, and gradient-norm clipping at 1.0, while replacing the original Transformer backbone with Tydra's alternating Hydra–Transformer backbone.We additionally tighten the synthetic MLP-prior distributions by reducing the upper bound on the sampled mean of the weight-initialization scale from 10.0 to 1.5 and that of the noise standard deviation from 0.3 to 0.1. We clamp synthetic-prior activation values to $[-10^4, 10^4]$.

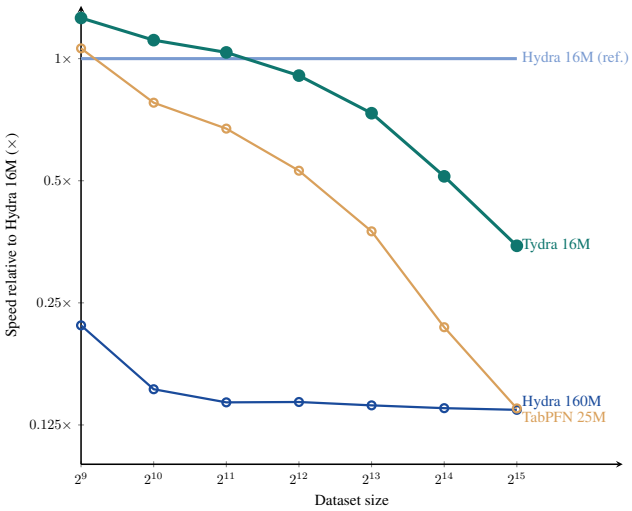

Figure 4: **Tydra scales far better than TabPFN and Hydra 160M, staying close to Hydra 16M's speed well into the large-context regime.** Inference speed of Tydra, Hydra 160M, and TabPFN relative to Hydra 16M on the synthetic benchmark. Tydra is competitive with or faster than Hydra 16M up to 2^{12} samples, then falls increasingly behind at larger scales, reaching roughly a third of Hydra 16M's speed at 2^{15} – still far ahead of TabPFN and Hydra 160M. Hydra 160M is substantially slower than Hydra 16M across the entire range, while TabPFN degrades from being competitive at small sizes (up to $1.06\times$ faster) to roughly to roughly the same speed as Hydra 160M at 2^{15}.

Empirical Evaluation

We aim to investigate the efficiency of tabular hybrid models empirically. To this end, we conduct experiments to tackle the following questions:

(Q1) Does Tydra match TabPFN's predictive performance at lower inference cost?

(Q2) Does naively scaling the pure SSM Hydra architectures narrow the accuracy–efficiency gap?

(Q3) How does Tydra's performance vary across different Transformer–SSM interleaving ratios?

Experimental Setup

Datasets. We evaluated Tydra in two regimes. First, we evaluated it on the 30 binary and multiclass classification datasets from OpenML-CC-18 (Bischl et al. 2017) containing at most 2,000 observations, 100 features, and 10 classes, following Koch et al. (2025). Second, we evaluated its inference speed at longer context lengths using synthetic tabular classification datasets with table sizes ranging from 512 to 32,768 samples. Unless stated otherwise, each dataset contains 10 numerical features drawn independently from a standard normal distribution and two balanced classes assigned independently of the features. Samples are jointly shuffled using a fixed seed to ensure reproducibility. Each table is divided equally into a context set and a query set: a table of size N contains $N/2$ context samples and $N/2$ samples for which predictions are produced. The synthetic datasets contain no categorical features or missing values and are used solely to measure inference speed and memory scaling.

Methods. We compare Tydra with pure Transformer and pure SSM baselines. As a Transformer baseline, we use the 25.8M-parameter TabPFN-style architecture following Koch et al. (2025). As an SSM baseline, we use Hydra at two scales: 16M and 160M parameters. Finally, we evaluate a family of Tydra hybrid models that vary the ratio and placement of attention and Hydra layers (Fig 3). All models are trained using the same prior-fitting procedure and synthetic-data prior introduced by Koch et al. (2025).

Metrics. We evaluated all models in terms of predictive performance and inference speed. We measured predictive performance using the area under the receiver operating characteristic curve (AUROC). For each dataset, we evaluated five deterministic 50/50 train-test splits in which both partitions contain all observed classes. We report the mean AUROC in the main results and the corresponding standard deviation across splits in the appendix.

We define inference speed as the total number of test predictions divided by the total synchronized model inference time. Following two warm-up runs, we measured inference time over 75 trials, consisting of 15 repetitions for each of the five splits.

For the long-context experiment, we measured synchronized end-to-end evaluation latency. For each model and table size, we performed two warm-up runs followed by 10 timed repetitions and report the mean latency in seconds. These measurements included data preparation, model execution, postprocessing, and metric computation, but excluded model loading and synthetic dataset generation.

The corresponding standard deviations are reported in the appendix. Together, the two evaluation regimes characterize model throughput on real-world datasets and end-to-end computational scaling as the context length increases.

(Answer Q1)

To evaluate whether Tydra matches TabPFN's predictive performance at lower inference cost, we compared Tydra against TabPFN across 30 datasets from OpenML benchmark and a synthetic benchmark isolating runtime scaling. Fig. 1 presents the accuracy–speed trade-off the OpenML datasets, showing Tydra achieves inference speedups of up to 29.9% (mean +24.8%) while matching TabPFN's predictive performance, with differences remaining within a narrow band (mean $|\Delta\text{AUROC}| = 0.006$) on all but one dataset. Fig. 5 breaks these results down per dataset, confirming the speedup is consistent across nearly the full suite rather than driven by a few outliers. For example, dataset 23381 is very small, consisting of 98 datapoints in total, which are split into 49 training points and 49 query points. Thus, Tydra matches TabPFN's predictive performance while substantially reducing inference cost.

(Answer Q2)

To evaluate whether naively scaling pure SSM architectures closes the accuracy–efficiency gap, we compare three Hydra configurations (16M and 160M parameters) with TabPFN

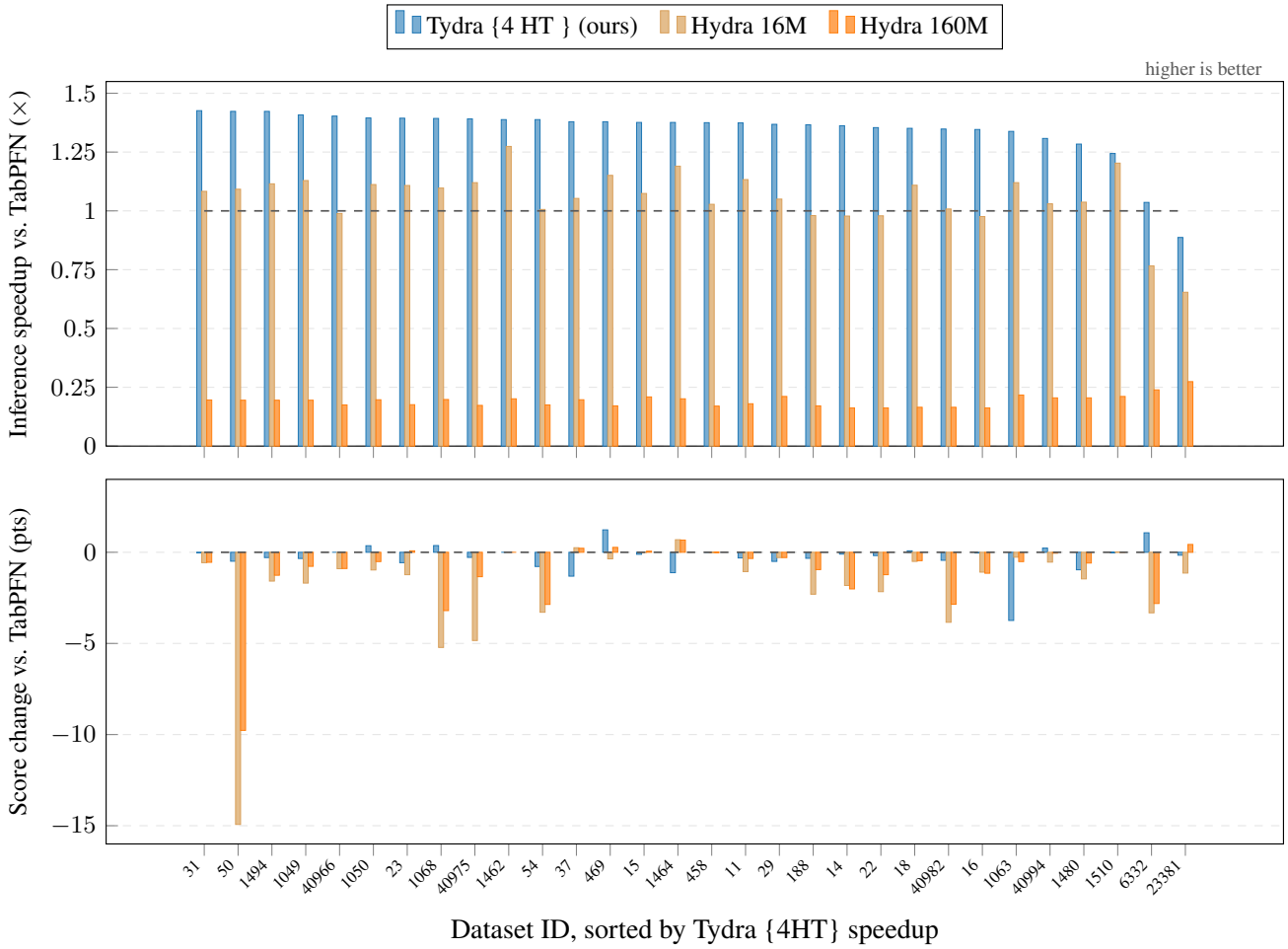

Figure 5: **Tydra matches TabPFN's accuracy at substantially lower inference cost.** Inference speedup relative to TabPFN (top) and change in predictive performance (ΔAUROC) relative to TabPFN (bottom), for Tydra 4HT and two Hydra baselines (16M, 160M), across 30 OpenML datasets. Datasets are sorted by Tydra's inference speedup.

and Tydra, and isolate runtime scaling on a synthetic benchmark of up to 2^{15} rows. Fig. 1 positions all four models on accuracy and inference speed jointly, showing Hydra 16M is fast but accuracy-poor, while Hydra 160M is slower than TabPFN itself. Figure 4 isolates runtime scaling on the synthetic benchmark, showing that Tydra scales substantially better than TabPFN and Hydra 160M, while remaining competitive with the fastest model, Hydra 16M, up to 2^{13} rows before falling behind at the largest scales; Hydra 16M's speed, however, comes at a substantial cost in predictive performance, as shown in Figs. 1 and 5. No single Hydra scale or size achieves both TabPFN-level accuracy and a meaningful efficiency advantage. Therefore, naively scaling Hydra fails to close the accuracy–efficiency gap.

(Answer Q3)

To investigate how the Transformer-SSM composition affects the accuracy–speed trade-off, we evaluate six eight-layer Tydra variants with different layer ratios and interleaving patterns, as illustrated in Figure 3. Table 1 summarizes the results of the evaluation. Hybridization generally yields models faster than TabPFN without sacrificing too much predictive power. The $\{4\,\mathrm{HT}\}$ architecture combines both types of layers in a 1:1 ratio and provides the best accuracy–speed trade-off. The $T\{6H\}T$ achitecture, with 6 hydra layers and 2 transformer layers, sacrifices the most predictive power on average at 1.24 points worse than TabPFN. However, disaggregating these metrics across the OpenML datasets (Figure 6) shows that that the drop in performance is driven by Dataset 50, on which $T\{6H\}T$ performs 15 AUC-ROC points worse than TabPFN. Overall, this confirms that hybrid architectures are fast without sacrificing much predictive power, with hybrids closer to 1:1 achieving a better tradeoff.

Conclusions and Future Work

We introduced Tydra, a hybrid Hydra-Transformer architecture for tabular in-context learning that achieves upto 1.43 × faster inference than TabPFN while retaining comparable predictive performance across OpenML benchmarks. These results suggest that combining subquadratic sequence mixers

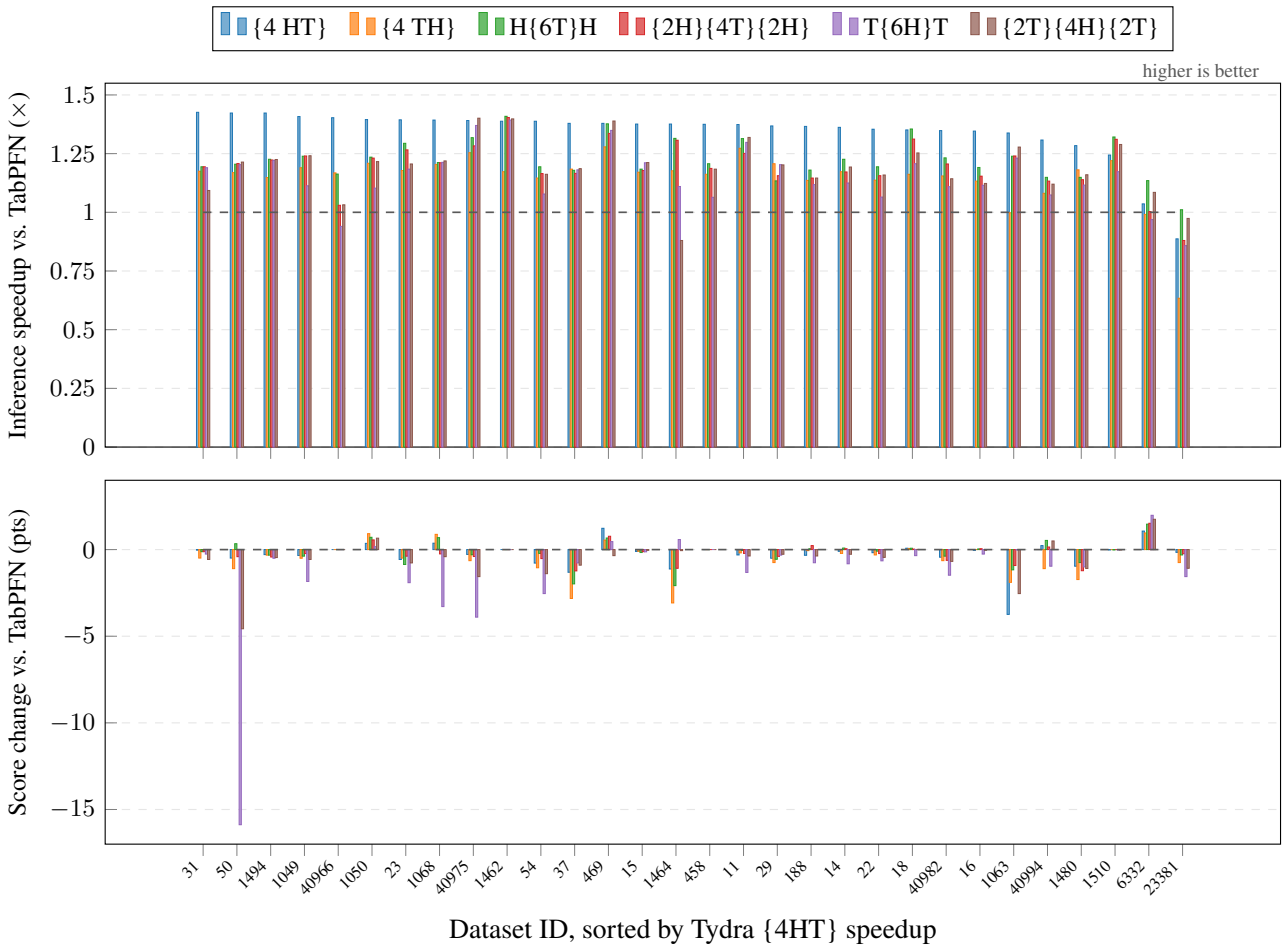

Figure 6: **The Tydra family consistently speeds up inference over TabPFN, with balanced hybrid variants also preserving predictive performance.** Inference speedup (top) and AUC-ROC change (bottom) relative to TabPFN across 30 OpenML datasets, for six Tydra configurations that vary the ratio and placement of attention (T) and Hydra (H) layers. All achieve similar inference speedups over TabPFN (up to 1.4×), but the one that concentrate Hydra layers together, such as T{6H}T, show substantially larger accuracy drops on individual datasets (e.g., -15.7 points on dataset 50) than more balanced interleavings. Datasets are sorted by Tydra {4HT}'s inference speedup.

with PFN-style learners is a promising direction for scaling tabular foundation models.

Tydra opens up several directions for future work. First, other subquadratic sequence mixers, such as gated linear attention or Gated DeltaNet (Yang, Kautz, and Hatamizadeh 2025; Merrill et al. 2026), could be adapted for tabular data and interleaved with attention layers in place of Hydra, potentially offering a different point on the accuracy–efficiency trade-off. Second, extending Tydra to large-scale, long-context tabular data remains an important direction. Third, looped transformers (Dehghani et al. 2018; Balef, Koshil, and Eggensperger 2026), which repeatedly apply a shared set of layers rather than stacking distinct ones, offer an orthogonal route to efficient inference, and could be combined with hybridization. Together with these directions, the Tydra family of models points towards a broader design space for efficient hybrid tabular foundation models.

Table 1: **Hybrid Model Comparison vs. TabPFN.** Mean test AUC-ROC and mean inference time (per batch, in milliseconds) across all OpenML datasets. 5/6 hybrid variants achieve faster inference without sacrificing much accuracy.

Method	**AUROC (%)**	**Time (ms)**
TabPFN	88.90	27.12
Tydra {4 HT}	88.61 (-0.29)	19.99 (-7.13)
Tydra {4 TH }	88.41 (-0.49)	23.41 (-3.71)
Tydra H{6T}H	88.73 (-0.17)	21.98 (-5.14)
Tydra {2H}{4T}{2H}	88.71 (-0.19)	22.57 (-4.55)
Tydra T{6H}T	87.66 (-1.24)	23.59 (-3.63)
Tydra {2T}{4H}{2T}	88.37 (-0.53)	22.78 (-4.36)

Acknowledgement

We gratefully acknowledge support from the BMFTR project XEI (grant number 16IS24079B), the Cluster of Excellence

"Reasonable AI" funded by the German Research Foundation (DFG) under Germany's Excellence Strategy (EXC-3057), the DYNAMIC center funded by the LOEWE program of the Hessian Ministry of Science and Arts (grant number LOEWE1/16/519/03/09.001(0009)/98), German Federal Ministry for Economic Affairs and Energy (BMWE) through EU-SAI: Souveräne KI für Europa (grant number 13IPC040G), and the US Army Research Office (award W911NF2010224).

References

Balef, A. R.; Koshil, M.; and Eggensperger, K. 2026. Is One Layer Enough? Understanding Inference Dynamics in Tabular Foundation Models. *arXiv preprint arXiv:2605.06510*.

Bischl, B.; Casalicchio, G.; Feurer, M.; Hutter, F.; Lang, M.; Mantovani, R. G.; Van Rijn, J. N.; and Vanschoren, J. 2017. OpenML benchmarking suites and the OpenML100. *stat*, 1050(11): 97.

Dao, T.; and Gu, A. 2024. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, Ł. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819*.

Dong, X.; Fu, Y.; Diao, S.; Byeon, W.; Chen, Z.; Mahabaleshwarkar, A.; Liu, S.-Y.; Chen, M.-H.; Suhara, Y.; Lin, Y. C.; et al. 2025. Hymba: A hybrid-head architecture for small language models. In *International Conference on Learning Representations*, volume 2025, 97729–97755.

Glorioso, P.; Anthony, Q.; Tokpanov, Y.; Whittington, J.; Pilault, J.; Ibrahim, A.; and Millidge, B. 2024. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*.

Grinsztajn, L.; Flöge, K.; Key, O.; Birkel, F.; Jund, P.; Roof, B.; Manium, M.; Hoo, S. B.; Bühler, M.; Garg, A.; et al. 2026. Tabpfn-3: Technical report. *arXiv preprint arXiv:2605.13986*.

Gu, A.; and Dao, T. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Hollmann, N.; Müller, S.; Eggensperger, K.; and Hutter, F. 2022. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*.

Hollmann, N.; Müller, S.; Purucker, L.; Krishnakumar, A.; Körfer, M.; Hoo, S. B.; Schirrmeister, R. T.; and Hutter, F. 2025. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045): 319–326.

Hoo, S. B.; Müller, S.; Salinas, D.; and Hutter, F. 2025. From Tables to Time: Extending TabPFN-v2 to Time Series Forecasting. *arXiv preprint arXiv:2501.02945*.

Hwang, S.; Lahoti, A.; Puduppully, R.; Dao, T.; and Gu, A. 2024. Hydra: Bidirectional state space models through generalized matrix mixers. *Advances in Neural Information Processing Systems*, 37: 110876–110908.

Koch, F.; Wever, M.; Raisch, F.; and Tischler, B. 2025. State-Space Models for Tabular Prior-Data Fitted Networks. *arXiv preprint arXiv:2510.14573*.

Li, Y.; Xie, R.; Yang, Z.; Sun, X.; Li, S.; Han, W.; Kang, Z.; Wang, D.; and Cheng, Y. 2026. Transmamba: A sequence-level hybrid transformer-mamba language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, 31823–31833.

Lieber, O.; Lenz, B.; Bata, H.; Cohen, G.; Osin, J.; Dalmedigos, I.; Safahi, E.; Meirom, S.; Belinkov, Y.; Shalev-Shwartz, S.; et al. 2024. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*.

Merrill, W.; Li, Y.; Romero, T.; Svete, A.; Costello, C.; Dasigi, P.; Groeneveld, D.; Heineman, D.; Kuehl, B.; Lambert, N.; et al. 2026. Olmo hybrid: From theory to practice and back. *arXiv preprint arXiv:2604.03444*.

Somvanshi, S.; Islam, M. M.; Mimi, M. S.; Polock, S. B. B.; Chhetri, G.; Dutta, A.; Rafe, A.; and Das, S. 2025. Advancing intelligent sequence modeling: evolution, trade-offs, and applications of state-space architectures from S4 to Mamba. *arXiv preprint arXiv:2503.18970*.

Yang, S.; Kautz, J.; and Hatamizadeh, A. 2025. Gated delta networks: Improving mamba2 with delta rule, 2025. *URL https://arxiv. org/abs/2412.06464*, 10.

Ye, H.-J.; Liu, S.-Y.; and Chao, W.-L. H. 2026. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. *Advances in Neural Information Processing Systems*, 38: 135605–135637.

Zhu, Q.; Cai, Y.; Fang, Y.; Yang, Y.; Chen, C.; Fan, L.; and Nguyen, A. 2024. Samba: Semantic segmentation of remotely sensed images with state space model. *Heliyon*, 10(19).