Tabular Foundation Models

From Tree-based Methods to TabPFN

Sungwoo Park

Uncertainty Quantification Lab
Seoul National University

May 4, 2026

Overview

1. Motivation
2. The Tabular Landscape
3. TabPFN: How and Why
4. TabICL: Scaling ICL via Architectural Surgery
5. Where TabPFN-style FMs Break
6. Closing

Contents

Why this talk?

Tabular data is everywhere — healthcare, finance, manufacturing, science — yet on `openml.org` 76% of datasets have fewer than 10K rows, and gradient-boosted decision trees (GBDTs) have dominated this regime for two decades.

Two questions drive this seminar:

1. How did deep learning finally start to compete on tabular data, and what did it have to change?
2. Where does the new TabPFN line still break, and what does that say about the design space of tabular foundation models?

Contents

Why tabular data is hard for deep learning

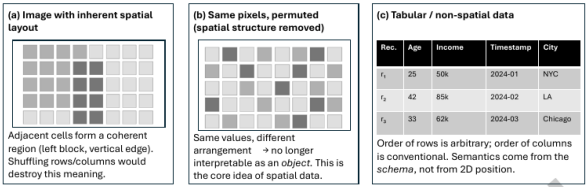


Figure 1: Spatial vs. tabular structure [Somvanshi et al., 2026].

Three structural obstacles:

1. Heterogeneous types: numerical, categorical, ordinal, missing all in one row
2. Non-spatial / non-sequential: column order is arbitrary; semantics live in the schema, not in adjacency
3. Small samples + irregular interactions: many real datasets have less then 10K rows and require non-local, axis-aligned splits

Why tabular data is hard for deep learning

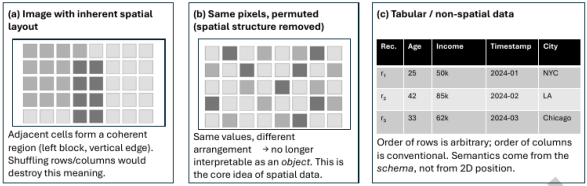


Figure 2: Spatial vs. tabular structure [Somvanshi et al., 2026].

CNN/RNN inductive biases (locality) give *nothing* here. When using traditional DL models, we must create structure rather than exploit it.

The 20-year baseline: GBDTs

XGBoost [Chen and Guestrin, 2016], LightGBM [Ke et al., 2017], CatBoost [Prokhorenkova et al., 2018].

- Each tree fits the residual via axis-aligned splits on Gini/entropy
- Histogram-based split finding (LightGBM), ordered target statistics (CatBoost)
- Native handling of missing values, mixed types, high-cardinality categoricals
- Interpretable via SHAP, fast to train, robust to monotonic transformations

Empirical dominance

On medium-sized benchmarks (less than 10K samples), GBDTs win on $\sim 89\%$ of datasets in standardized comparisons [Grinsztajn et al., 2022, McElfresh et al., 2023]. The gap narrows on large or high-cardinality data, but tree models remain the pragmatic default.

Why GBDTs win on tabular: four inductive biases

1. **Hierarchical, axis-aligned partitioning** — automatic discovery of high-order interactions through recursive splits
2. **Piecewise-constant approximation** — robust to irregular, non-smooth target functions
3. **Native heterogeneity** — mixed types and missing values without preprocessing
4. **Sample efficiency** — strong performance even at $n \sim 10^3$, where MLPs overfit

Deep learning had to either copy these biases (TabNet, NODE, DeepGBM) or replace them with much stronger priors (TabPFN, TabICL).

TDL Wave 1: regularizing the MLP

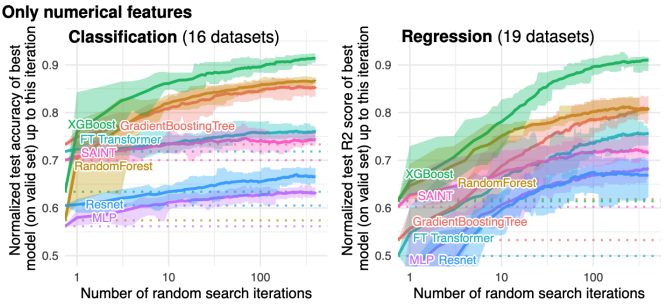


Figure 3: Normalized test score vs. random-search iterations on medium-sized ($\leq$10K samples) numerical classification benchmarks [Grinsztajn et al., 2022]. MLP/Resnet stay well below GBT/XGBoost across the entire tuning budget.

Plain MLPs underperform GBDTs on $\sim$10K-sample datasets, and the gap *persists* even after extensive hyperparameter tuning [Grinsztajn et al., 2022].

TDL Wave 1: regularizing the MLP

First fixes (2019–2021). Each model copies *one* GBDT inductive bias into a shallow MLP.

- **DNF-Net** (copies rule-based decisions) — 3-layer differentiable gates: soft thresholds (*literals*) $\rightarrow$ *AND* $\rightarrow$ *OR*. Output is OR-of-ANDs = disjunctive normal form, the same shape as a tree's leaf rules.

$$
\begin{aligned}
\mathrm{DNNF}(x) &= \mathrm{OR}\big(\big[\mathrm{AND}_{c_1}(L(x)),\ \mathrm{AND}_{c_2}(L(x)),\ \ldots,\ \mathrm{AND}_{c_k}(L(x))\big]\big) \\
L(x) &= \tanh(Wx + b), \\
\mathrm{AND}_c(z) &= \tanh\Big(\textstyle\sum_{i \in c} z_i - |c| + 1.5\Big), \\
\mathrm{OR}(d) &= \tanh\Big(\textstyle\sum_{j=1}^{k} d_j + k - 1.5\Big)
\end{aligned}
$$

TDL Wave 1: regularizing the MLP

First fixes (2019–2021). Each model copies *one* GBDT inductive bias into a shallow MLP.

- **MLP+** (copies feature-wise treatment + heterogeneity) — Leaky Gates: a learnable scalar in front of each input column for soft feature selection. Ghost BatchNorm: split each batch into smaller chunks, normalize separately — stable on mixed-distribution rows.
- **Regularization cocktails** [Shwartz-Ziv and Armon, 2022] (copies sample efficiency) — no architectural change; per-dataset random search over 13 regularizers (Mixup, Dropout, Weight Decay, Stochastic Depth, ...). Plain MLPs match GBDTs once overfit is controlled.

Lesson

It is not the *depth* but the *inductive bias* that matters. Each model rebuilds one piece of what GBDTs get for free.

TDL Wave 2: tree-inspired deep architectures

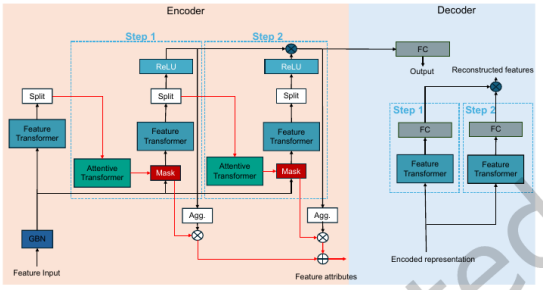


Figure 4: TabNet encoder–decoder [Somvanshi et al., 2026].

TabNet [Arik and Pfister, 2021]: sequential attention with sparse Entmax masks — per-step instance-wise feature selection. Local + global interpretability. Higher cost than GBDTs.

TDL Wave 3: transformers for tables

Treat features (and sometimes samples) as tokens.

- **TabTransformer [Huang et al., 2020]**: multi-head self-attention over categorical embeddings; numerical features bypass attention
- **FT-Transformer [Gorishniy et al., 2021]**: tokenize *every* feature (cat + num) with a learned embedding, then apply standard transformer blocks. Strong on 100–1000 feature regimes
- **SAINT [Somepalli et al., 2021]**: row-attention *and* sample-attention — "borrow strength" from similar rows

Trade-offs

Better interaction modeling and transferability $\Leftrightarrow$ quadratic memory in features (FT-T) or samples (SAINT), reduced interpretability, and sensitivity to hyperparameter tuning.

TDL Wave 4: in-context learning foundation models

A different paradigm:

- No per-dataset training. The whole training set is fed as *context* to a frozen model
- Pre-trained *once* on millions of *synthetic* tabular tasks
- One forward pass = one fitted model

Examples: **TabPFN** [Hollmann et al., 2023, Hollmann et al., 2025], **TabICL** [Qu et al., 2025], **Mitra** [Zhang et al., 2025], **Real-TabPFN** [Garg et al., 2025], TabDPT.

This is the focus of the rest of the talk.

Contents

Prior-data Fitted Networks: the core idea

Training objective. Across millions of synthetic datasets $\mathcal{D}_i \sim p(\mathcal{D})$:

$$\min_{\theta} \; \mathbb{E}_{\mathcal{D},(X_{\text{tr}},y_{\text{tr}},X_{\text{te}},y_{\text{te}})}[-\log q_\theta(y_{\text{te}} \mid X_{\text{tr}}, y_{\text{tr}}, X_{\text{te}})]$$

- Treat the entire training set as context
- Learn a function-from-data, not a function-from-input
- Theoretically: q_θ approximates $p(y_{\text{test}} \mid X_{\text{test}}, X_{\text{train}}, y_{\text{train}})$, the posterior predictive under the prior $p(\mathcal{D})$ [Müller et al., 2024]

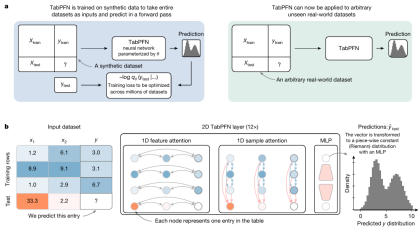


Figure 5: TabPFN training and use [Hollmann et al., 2025].

The synthetic prior: structural causal models

Generative pipeline per training task:

1. Sample DAG hyperparameters (#nodes, density, complexity)
2. Inject noise into root nodes
3. Propagate through edges — each edge is one of: small NN, decision tree, or discretization
4. Read off feature columns F and target column T at random graph positions
5. Post-process: warping (Kumaraswamy), quantization, missing values, outliers

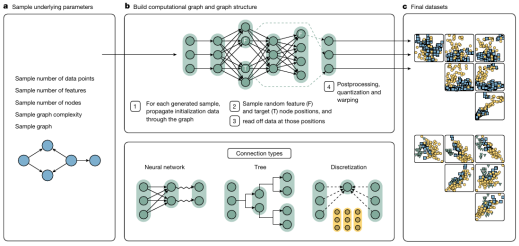


Figure 6: Causal prior [Hollmann et al., 2025].

TabPFN v2 architecture: 2D alternating attention

Two key design choices.

- Each cell of the table gets its own representation; the input is a 2D grid, not a flat sequence
- Each layer alternates feature attention (across columns) and sample attention (across rows), then an MLP
- Randomized attribute tokens added per column — handle variable column count without dataset-specific embeddings

Why 2D attention matters

- Permutation invariant over both axes by construction
- Train state can be cached once and reused for many test samples (fit/predict separation)
- Generalizes to more samples and features than seen at training

TabPFN v2 — per-layer tensor walkthrough

Notation: $R = n_{\text{train}} + n_{\text{test}}$ rows, F features, $D = 192$ hidden. (Batch axis omitted)

Input & embedding

- $X \in \mathbb{R}^{R \times F}$ projected to $[R, F, D]$
- $+$ randomized attribute tokens (per-column, resampled each forward) $\rightarrow [R, F, D]$
- y_{train} embedded and broadcast-added to every token of train rows: $[R, F, D]$

One of 12 TabPFN blocks

- Feature attention: reshape to $[R, F, D]$, self-attention over F (one sequence per row, R in parallel)
- Sample attention: reshape to $[F, R, D]$, self-attention over R (one sequence per column, F in parallel)
- Position-wise MLP back to $[R, F, D]$

Readout

- Slice test rows, take the dummy-y token at $[n_{\text{test}}, D] \rightarrow$ MLP $\rightarrow$ logits / piecewise-constant density

Inference economics

One forward pass replaces a tuning loop.

- Default TabPFN: 2.8 s on a single GPU for classification ($\leq$10K samples)
- Tuned XGBoost / CatBoost ensembles: 4 hours of CV-based hyperparameter search

What this changes

TabPFN reframes "fit a model" as a *frozen forward pass*. The expensive work was paid once during pre-training; downstream users never run gradient descent.

Pre-training: 8$\times$RTX 2080 Ti, $\sim$2 weeks, $\sim$130M synthetic datasets.

TabPFN-2.5 [Prior Labs Team, 2025]

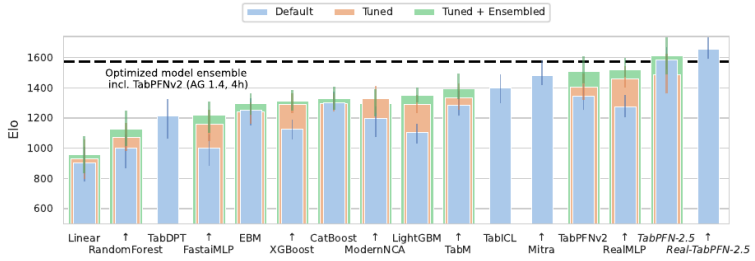


Figure 7: TabArena-lite Elo [Prior Labs Team, 2025].

- Depth: 12 → 18 (regression) / 24 (classification)
- Feature group size 2 → 3 — fewer feature tokens to attend over, faster inference
- +64 "thinking" rows appended to each input as learned attention sinks (LLM-inspired)
- Scaled to 50K samples × 2K features
- Real-TabPFN-2.5: continued pre-training on 43 curated real datasets [Garg et al., 2025]
- Hyperparameters and training recipe: not public

Why does TabPFN work? — three working hypotheses

1. Prior coverage: SCM-based synthetic tasks span a large region of the function-from-data space. The model meta-learns *the algorithm*, not the function.
2. 2D attention biases for tables: alternating row/column attention efficiently learns "borrow strength" across rows + interactions across features — the right structure for tabular data.
3. Bayesian-like calibration: training as posterior approximation gives well-calibrated uncertainty by construction, mitigating overfit on small data.

Recent reinforcement (Ye et al. 2025) [Ye et al., 2025]

Even with *randomized* attribute tokens, TabPFN v2 internally infers attribute relationships through ICL. Freezing TabPFN and using it as a feature extractor with a linear probe nearly matches its full ICL accuracy — the embeddings are highly linearly separable.

Is TabPFN actually Bayesian?

Bayesian inference

Given a prior $p(\theta)$ over an unknown latent θ (parameter, function, or data-generating mechanism) and a likelihood $p(D \mid \theta)$, combine them into the *posterior* $p(\theta \mid D) \propto p(D \mid \theta)\, p(\theta)$, and predict new x_* via the posterior predictive distribution

$$p(y_* \mid x_*, D) = \int p(y_* \mid x_*, \theta)\, p(\theta \mid D)\, d\theta.$$

The training objective is Bayesian.

[Müller et al., 2024] show that minimizing the cross-entropy of $q_\theta(y_{\text{te}} \mid X_{\text{tr}}, y_{\text{tr}}, X_{\text{te}})$ over datasets $\mathcal{D} \sim p(\mathcal{D})$ is equivalent to minimizing

$$\mathbb{E}_{X_{\text{te}}, D_{\text{tr}}}\left[\, D_{\text{KL}}\big(\, p(y_{\text{te}} \mid X_{\text{te}}, D_{\text{tr}}) \,\big\|\, q_\theta(\cdot)\,\big)\right].$$

Is TabPFN actually Bayesian?

- Recovering the true posterior predictive requires three idealizations: infinite training data, a network with sufficient capacity to represent $p(y \mid x)$, and optimization that finds the global optimum.
- [Hollmann et al., 2025] hedge: TabPFN *"can be viewed as approximating Bayesian prediction for a prior defined by the synthetic datasets."*

Why TabPFN is not purely Bayesian — architectural violations

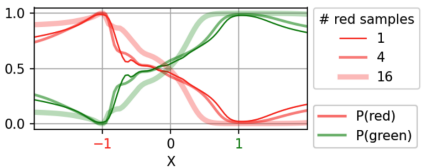


Figure 8: Sample-duplication asymmetry [McCarter, 2024].

The trained network does not satisfy basic structural properties that any true posterior predictive distribution (PPD) must obey.

- Sample-duplication asymmetry[McCarter, 2024]. Duplicating examples of *one* class shifts the decision boundary in directions that a likelihood product $\prod_i p(x_i, y_i \mid \ell)$ *cannot* produce.

  $\Rightarrow$ TabPFN does not internally factorize the data into a likelihood times a prior — the basic Bayesian decomposition is absent.

Why TabPFN is not purely Bayesian — architectural violations

- Periodicity not learned (McCarter 2024, Fig. 6). On periodic 1D targets, predictions *collapse to* 0.5 as the number of cycles grows; the response is also asymmetric under left/right reflection.

  $\Rightarrow$ Permutation invariance, which is required of any PPD over an exchangeable dataset, is broken.

Diagnosis

TabPFN is a *Bayesian-motivated amortized estimator*, not a Bayesian method. The architecture cannot represent a true PPD. The Bayesian framing is useful for design, not for interpretation of the trained model.

Contents

Why TabPFN v2 hits a wall around 10K rows

Per-layer cost in TabPFN v2: $\mathcal{O}(R^2FD + F^2RD)$ — *both* axes are quadratic in attention.

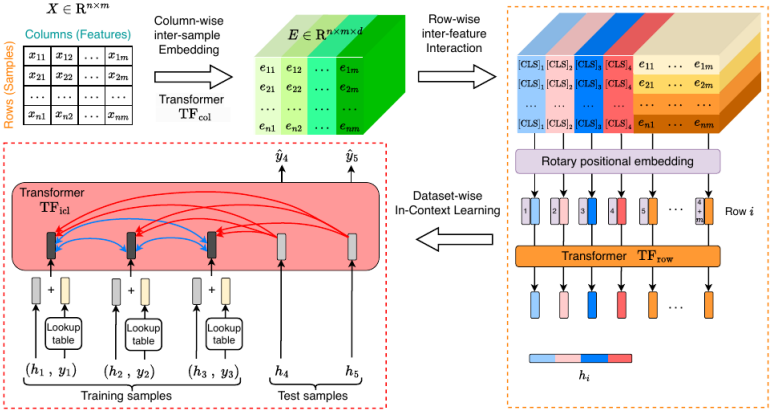
Two consequences as R grows:

1. Compute / memory: at $R = 50\text{K}$ and $F = 100$, the row-attention term $R^2FD \approx 5 \times 10^{11}$ ops per layer — prohibitive
2. Softmax fading (next slide): even if you can afford the compute, attention degenerates as R grows

TabICL's strategy [Qu et al., 2025]

Compress rows first. Process each cell column-wise, pool features into a fixed-dim row embedding, then run ICL on the row sequence only. The expensive $\mathcal{O}(\cdot)$ on rows happens *once*, not per layer.

TabICL high-level architecture



Three transformers: $\mathrm{TF}_{\mathrm{col}}$ (distribution-aware column encoder) $\rightarrow$ $\mathrm{TF}_{\mathrm{row}}$ (row interaction with [CLS]) $\rightarrow$ $\mathrm{TF}_{\mathrm{icl}}$ (dataset-wise ICL, where labels y are finally injected).

$\mathrm{TF}_{\mathrm{row}}$: context-aware row interaction

Input. $E \in \mathbb{R}^{n \times m \times d}$. Per row, prepend 4 learnable [CLS] tokens: $\mathbb{R}^{n \times (m+4) \times d}$.

3-layer 8-head transformer, self-attention over the $(m + 4)$ axis, batched across n rows.

Output. Concatenate the 4 [CLS] tokens of each row $\rightarrow h_i \in \mathbb{R}^{4d} = \mathbb{R}^{512}$.

$$H \in \mathbb{R}^{n \times 4d}$$

Key consequence

After $\mathrm{TF}_{\mathrm{row}}$, each row is a single fixed-dim vector, regardless of the original feature count. The rest of the network never sees individual cells again.

Also: y has not been injected yet. The row embedding is purely a function of X.

$\mathrm{TF}_{\mathrm{icl}}$: dataset-wise ICL where labels finally enter

Label injection. For train rows: $H_{\text{train}}[i] = h_i + \mathrm{Lookup}(y_i)$ (one-hot via lookup table). For test rows: $H_{\text{test}}[i] = h_i$ (no y).

12-layer 4-head transformer, self-attention along the row axis only:

- Train rows attend to other train rows
- Test rows attend to train rows only — no test-to-test or test-to-itself peeking (causal-style mask)

Readout. 2-layer MLP on $H_{\text{test}} \to$ class probabilities.

TabPFN v2 vs. TabICL — side by side

	TabPFN v2	**TabICL**
Attention pattern	alternating feature/sample, 12 layers	column-then-row (3+3 layers) + ICL (12 layers)
Token shape	$[R, F, D]$ throughout	$[n, m, d] \to [n, 4d]$ before ICL
y injection	broadcast onto every train-row token early	lookup-added *only* at $\mathrm{TF}_{\mathrm{icl}}$
Position encoding	randomized attribute tokens	RoPE on column index
Hidden dim	$D = 192$	$d = 128$ (col/row), $4d = 512$ (ICL)
Permutation invariance	true (token randomization)	approximate (RoPE breaks it; column-perm ensemble restores)
Per-layer cost	$\mathcal{O}(R^2FD + F^2RD)$ both quadratic	col $\mathcal{O}(nkmd)$, row $\mathcal{O}(m^2nd)$, ICL $\mathcal{O}(n^2d)$
Practical scale	$\leq$ 10K rows $\times$ 500 feat.	$\leq$ 500K rows $\times$ 500 feat.
Pre-training	2 weeks, 8$\times$ RTX 2080 Ti	20 days, 3$\times$ A100 + 3-stage curriculum

Why TabICL scales — the softmax-fading argument

Standard scaled dot-product attention:

$$\alpha_i = \frac{\exp(q^\top k_i/\sqrt{d})}{\sum_{j=1}^{n} \exp(q^\top k_j/\sqrt{d})}$$

- Denominator: n terms, grows linearly $\Rightarrow \max_i \alpha_i \to 0$ as $n \to \infty$
- Attention degenerates to uniform — "attention fading" [Nakanishi, 2025]

Why this hurts tabular more than language:

- In tabular ICL, context length *is the training set size*
- Tabular rows have no notion of locality — the sliding-window tricks that rescue LLMs do not apply
- Keeping attention sharpness constant in n requires query scaling $\Theta(\log n)$ — this motivates QASSMax in TabICLv2 [Chen et al., 2025]

This is one root cause of TabPFN v2's collapse beyond $\sim$ 10K samples.

TabICL pretraining recipe — curriculum + tree SCM

Three-stage curriculum (gradually increase n):

1. Micro-batch $N_B = 4$, fixed size 1,024, for 160K steps
2. $N_B = 1$, 1K–40K log-uniform, 2K steps (activation checkpointing on)
3. $N_B = 1$, 40K–60K, 50 steps, training only $\mathrm{TF}_{\mathrm{icl}}$

Mixed synthetic prior:

- 70% MLP-based SCM (TabPFN v2 style)
- 30% tree-based (Grinsztajn 2024 inspired): each parent→child edge fits an XGBoost regressor on fake targets, then uses its prediction as the child variable
- Activation pool extended to 15 functions (vs. 4 in TabPFN v1)

TabICLv2 [Qu et al., 2026]

1. QASSMax: a query-aware log n–scaled softmax, applied to $\mathrm{TF}_{\mathrm{col}}$ and $\mathrm{TF}_{\mathrm{icl}}$. Fixes the attention-fading problem that limited TabICL at $n \gtrsim 30\mathrm{K}$. *(Next slide.)*
2. Repeated feature grouping: features with similar marginal distributions are grouped *before* $\mathrm{TF}_{\mathrm{col}}$.
3. Quantile regression head: replaces the binning / piecewise-constant density head used by TabPFN v2, TabPFN-2.5, and Mitra. The model emits a vector of quantile predictions trained with pinball loss; smoothing (spline center, decay tails) reconstructs a full predictive density.

QASSMax — query-aware scalable softmax

Problem. Standard softmax: $\alpha_i = \exp(q^\top k_i/\sqrt{d})/\sum_j \exp(q^\top k_j/\sqrt{d})$ has $\max_i \alpha_i \to 0$ as $n \to \infty$. Chen et al. (2025): keeping sharpness constant requires query scaling $\Theta(\log n)$ [Chen et al., 2025].

SSMax (Nakanishi 2025) [Nakanishi, 2025]: $\tilde{q}_{hi} = q_{hi} \cdot s_h \log n$ with one learnable scalar s_h per head. Simple, FlashAttention-compatible, but rigidly head-uniform.

QASSMax (TabICLv2): per-element scaling *and* a query-dependent gate.

$$\tilde{q}_{hi} = q_{hi} \cdot \underbrace{\mathrm{MLP}_{\mathrm{base}}(\log n)_{hi}}_{\text{element-wise log-n scaling}} \cdot \underbrace{\big(1 + \tanh(\mathrm{MLP}_{\mathrm{gate}}(q_h)_i)\big)}_{\text{query-aware gate} \in (0,2)}$$

Both MLPs have hidden dim 64 with GELU. Pre-rescaling preserves FlashAttention compatibility (unlike entmax-style replacements).

QASSMax is applied only to $\mathrm{TF}_{\mathrm{col}}$ and $\mathrm{TF}_{\mathrm{icl}}$ — $\mathrm{TF}_{\mathrm{row}}$ attends over the feature axis m, which does not blow up with dataset size.

Contents

Critical observations from the literature

[McCarter, 2024]

- On 1d/2d toy problems, sample duplication has asymmetric effects (duplicating only one class shifts the boundary in unintuitive ways)
- No evidence of learning periodic patterns; predictions trend toward uniform as cycles increase
- Ensembling appears to do double duty: variance reduction *and* (approximate) permutation invariance enforcement

[Liu and Ye, 2025] & **[Ye et al., 2025]**

- Hard scalability ceilings: sample size, feature dim, class count

Common diagnosis: prior-coverage gap. Synthetic SCMs do not span all function classes the model is asked to fit.

Binning regression head $\Rightarrow$ no extrapolation

The regression heads of TabPFN v2, TabPFN-2.5, and Mitra all use a piecewise-constant density (Bar Distribution): bin boundaries are set at the quantiles of the context y_{train}, and a probability is output for each bin.

- Bin boundaries are defined only within the min/max of y_{train} — values outside are not representable
- When the test target falls outside the training range, predictions saturate at the outermost bin — $\hat{y}$ is clipped to the bound

TabICLv2's quantile regression head with spline/decay smoothing avoids strict saturation by smoothly connecting predictions near the boundary, but extrapolation beyond the quantile range seen during training remains weak.

Robustness, calibration, and open questions

Permutation robustness. Not robust to row permutation in the strict sense. Variance is small relative to true residual, but grows with train-set noise. When train and test have different sampling distributions (e.g., $y = x$ trained on $\mathcal{N}(0, 1)$, tested on uniform $[-0.5, 0.5]$), residuals explode.

Open issues for the TabPFN line:

1. Prior coverage — multiplication, periodic patterns, high-order symmetric interactions
2. Output head — binning $\Rightarrow$ no extrapolation beyond train y-range
3. Calibration metrics — ECE / Brier / proper scoring rules under-reported; ROC-AUC dominates
4. Reproducibility — TabPFN hyperparameters and curriculum are not public

Contents

Take-aways

1. **GBDTs remain the default** for typical tabular workloads. Deep learning earns its keep on large, high-cardinality, multimodal, or transfer-heavy regimes.
2. **TabPFN reframes tabular ML.** A forward pass over a meta-learned posterior approximator beats 4-hour-tuned ensembles on $\leq$10K-sample data.
3. **TabPFN v2, TabICL, and TabPFN-2.5** progress along *different axes*: 2D attention biases, attention scaling for long contexts, and depth + curated priors respectively.
4. **The next frontier is the prior.** Multiplication, periodicity, and high-order symmetry expose what synthetic SCMs do not yet cover.

Our research line

Two directions, both built on a clean re-implementation we control end-to-end.

(1) UQ plugins for TabPFN-style models. Compose classical uncertainty machinery on top of a frozen ICL forward pass:

- Conformal Prediction on the PPD — distribution-free coverage guarantees
- Bootstrap / context resampling — subsetting train rows to recover frequentist intervals without retraining

(2) Theoretical grounding of the SCM prior. What function class does the synthetic prior actually span, and how does it overlap with real tables?

- Coverage diagnostics on the SCM generator — which interaction orders, multiplicative and periodic terms are reachable under current hyperparameters
- Causal-structure faithfulness — DAG-ordered features vs. random column permutation; collider-bias diagnosis under topological ordering
- Hyperparameter sensitivity of the SCM prior — number of causes, activation pool, edge mixture

References I

Arik, S. Ö. and Pfister, T. (2021).
Tabnet: Attentive interpretable tabular learning.
In *AAAI*.

Chen et al. (2025).
Critical attention scaling in long-context transformers.
arXiv preprint.

Chen, T. and Guestrin, C. (2016).
Xgboost: A scalable tree boosting system.
In *KDD*.

Garg, A. et al. (2025).
Real-tabpfn: Improving tabular foundation models via continued pre-training with real-world data.
arXiv preprint.

References II

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021).
Revisiting deep learning models for tabular data.
In *NeurIPS*.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022).
Why do tree-based models still outperform deep learning on typical tabular data?
In *NeurIPS*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023).
Tabpfn: A transformer that solves small tabular classification problems in a second.
In *ICLR*.

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, 637:319–326.

References III

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. (2020).
Tabtransformer: Tabular data modeling using contextual embeddings.
arXiv preprint arXiv:2012.06678.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017).
Lightgbm: A highly efficient gradient boosting decision tree.
In *NeurIPS*.

Liu, S.-Y. and Ye, H.-J. (2025).
Tabpfn unleashed: A scalable and effective solution to tabular classification problems.
arXiv preprint.

McCarter, C. (2024).
What exactly has tabpfn learned to do?
ICLR Blogposts Track.

References IV


McElfresh, D., Khandagale, S., Valverde, J., Prasad C., V., Ramakrishnan, G., Goldblum, M., and White, C. (2023).
When do neural nets outperform boosted trees on tabular data?
In *NeurIPS*.


Müller, S., Feurer, M., Hollmann, N., and Hutter, F. (2024).
Bayes' power for explaining in-context learning generalizations.
arXiv preprint.


Nakanishi, K. M. (2025).
Scalable softmax is superior for attention.
arXiv preprint arXiv:2501.19399.


Prior Labs Team (2025).
Tabpfn-2.5: Advancing the state of the art in tabular foundation models.
Technical report, Prior Labs.
arXiv:2511.08667.

References V


Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018).
Catboost: Unbiased boosting with categorical features.
In *NeurIPS*.


Qu, J., Holzmüller, D., Varoquaux, G., and Le Morvan, M. (2025).
Tabicl: A tabular foundation model for in-context learning on large data.
In *ICML*.


Qu, J., Holzmüller, D., Varoquaux, G., and Le Morvan, M. (2026).
Tabiclv2: A better, faster, scalable, and open tabular foundation model.
arXiv preprint arXiv:2602.11139.


Shwartz-Ziv, R. and Armon, A. (2022).
Tabular data: Deep learning is not all you need.
Information Fusion.

References VI

Somepalli, G., Goldblum, M., Schwarzschild, A., Bayan Bruss, C., and Goldstein, T. (2021).
Saint: Improved neural networks for tabular data via row attention and contrastive pre-training.
arXiv preprint arXiv:2106.01342.

Somvanshi, S., Das, S., Javed, S., Antariksa, G., and Hossain, A. (2026).
A survey on tabular data: From tree-based methods to tabular deep learning.
ACM Computing Surveys.

Ye, H.-J., Liu, S.-Y., and Chao, W.-L. (2025).
A closer look at tabpfn v2: Understanding its strengths and extending its capabilities.
arXiv preprint arXiv:2502.17361.

Zhang, X., Maddix, D. C., Yin, J., Erickson, N., Ansari, A. F., Han, B., Zhang, S., Akoglu, L., Faloutsos, C., Mahoney, M. W., et al. (2025).
Mitra: Mixed synthetic priors for enhancing tabular foundation models.
arXiv preprint arXiv:2510.21204.

Thank You