Tabular Foundation Model : TabPFN

Based on Noah et al., “Transformer can do Bayesian Inference”
“TabPFN 2.5 : Advancing the SOTA in Tabular Foundation Model” and related works

이상형

26.05.14

Introduction



Noah Hollmann



Samuel Müller



Frank Hutter

PRIOR Since 2024

The Next Chapter for Prior Labs





Frank Hutter - Noah Hollmann - Sauraj Gambhir
04 May 2026 • 3 min read

When we started Prior Labs, we had a conviction that ran against the grain of where most of the AI world was looking. While the field poured its energy into language, we believed an untapped opportunity in AI sat somewhere else entirely: in the tables, spreadsheets, and structured records that actually run the world's businesses and scientific discoveries.

Today we're announcing the next chapter. Prior Labs signed a definitive agreement to be acquired by SAP to become the next frontier AI lab. The terms are not disclosed and the deal is subject to regulatory approval. We will continue to operate as an independent entity, retaining our brand, our mission, our open-source commitments, our customers and our research-led identity. SAP will invest more than 1 billion EUR over the next four years and support us with long-term investment and a direct path to productization across the SAP portfolio.

Introduction

Capabilities

Classification
Solve binary or multi-class classification problems with calibrated probabilities.

Regression
Estimate continuous values with uncertainty-aware outputs and minimal preprocessing.

Forecasting
Model time series (via TabPFN for forecasting) to predict future values and trends.

Anomaly Detection
Detect rare and anomalous samples using TabPFN.

Data Generation
Generate realistic synthetic tabular data with TabPFN.

Fine Tuning
Optimize TabPFN models to your own data with fine-tuning.

Models

Compare TabPFN model versions — capabilities, limits, and availability.

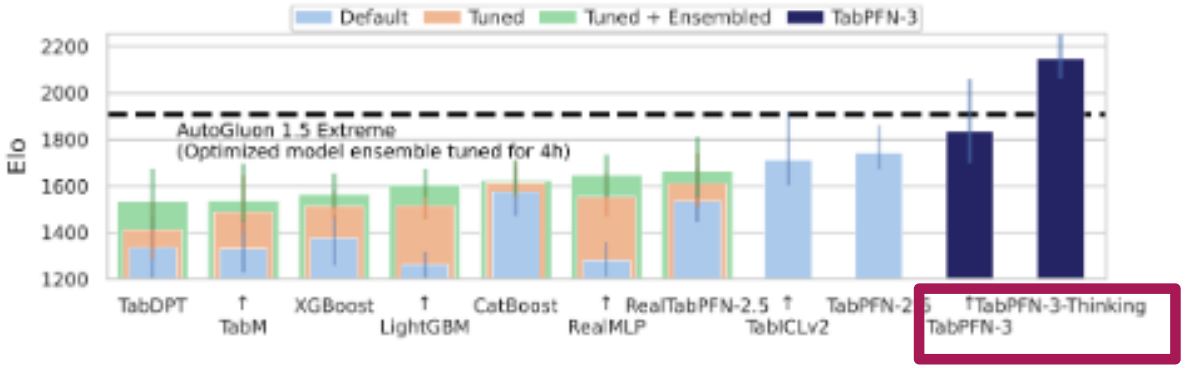
Model	Description	Limits	Publication
TabPFN-3-Plus	All in TabPFN-3, plus thinking mode and native text feature support. Available through the API and enterprise deployments.	1M rows × 200 features (trade-off: 100K × 2K, 1K × 20K) 160 classes	—
TabPFN-3	Latest foundation model. Up to 1M rows in-context with a row/feature trade-off.	1M rows × 200 features (trade-off: 100K × 2K, 1K × 20K) 160 classes	—
TabPFN-2.6	Good general-purpose choice for datasets up to 100K rows. Previous default model.	100K rows × 2K features 10 classes	—
TabPFNv2	Published in Nature. First TabPFN to handle real-world messy data — mixed types, missing values, categorical features. Best suited for smaller datasets.	10K rows × 500 features 10 classes	Nature (2025)

https://docs.priorlabs.ai/models

Introduction

TabPFN-3: Technical Report

May 12, 2026 Prior Labs Team



Introduction

Advancing the sota in Tabular FM (26.2)

V2.5

- Accurate predictions on small data with a tabular foundation model (2025) — V2
  Nature
  Noah Hollmann*, **Samuel Müller***, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, **Frank Hutter**
- The tabular foundation model TabPFN outperforms specialized time series forecasting models based on simple features (2024)
  NeurIPS - Table Representation Learning Workshop
  Shi Bin Hoo, **Samuel Müller**, David Salinas, **Frank Hutter**
- Drift-resilient tabPFN: In-context learning temporal distribution shifts on tabular data (2024)
  NeurIPS - Table Representation Learning Workshop
  Kai Helli*, David Schnurr*, **Noah Hollmann, Samuel Müller, Frank Hutter**
- Bayes' Power for Explaining In-Context Learning Generalizations (2024)
  arXiv 2024.10 – ICLR 2025 reject
  Samuel Müller, Noah Hollmann, Frank Hutter
- Large language models for automated data science: Introducing caafe for context-aware automated feature engineering (2023)
  NeurIPS
  Noah Hollmann, Samuel Müller, Frank Hutter
- Efficient bayesian learning curve extrapolation using prior-data fitted networks (2023)
  NeurIPS
  Steven Adriaensen*, Herilalaina Rakotoarison*, **Samuel Müller, Frank Hutter**
- PFNs4BO: In-Context Learning for Bayesian Optimization (2023)
  ICML
  Samuel Müller, Matthias Feurer, **Noah Hollmann, Frank Hutter**
- TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second (2022) — V1
  NeurIPS Oral
  Noah Hollmann*, **Samuel Müller***, Katharina Eggensperger, **Frank Hutter**
- Transformers Can Do Bayesian Inference (2022) — V0
  ICLR
  Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, **Frank Hutter**

Introduction

Prior Fitted Network(PFN)
: trained with only synthetic data
: generation

Tabular data adaptation
: modeling

In context learning(ICL)
: train and inference

V_0

TRANSFORMERS CAN DO BAYESIAN INFERENCE

Samuel Müller [1] **Noah Hollmann** [2] **Sebastian Pineda** [1] **Josif Grabocka** [1] **Frank Hutter** [1,3]
[1] University of Freiburg, [2] Charité Berlin, [3] Bosch Center for Artificial Intelligence
Correspondence to Samuel Müller: muellesa@cs.uni-freiburg.de

Preliminary : Gaussian process(GP)

Gaussian + Process

Stochastic process : Set of random variable
Gaussian : Structure of process

Preliminary : Gaussian process(GP)

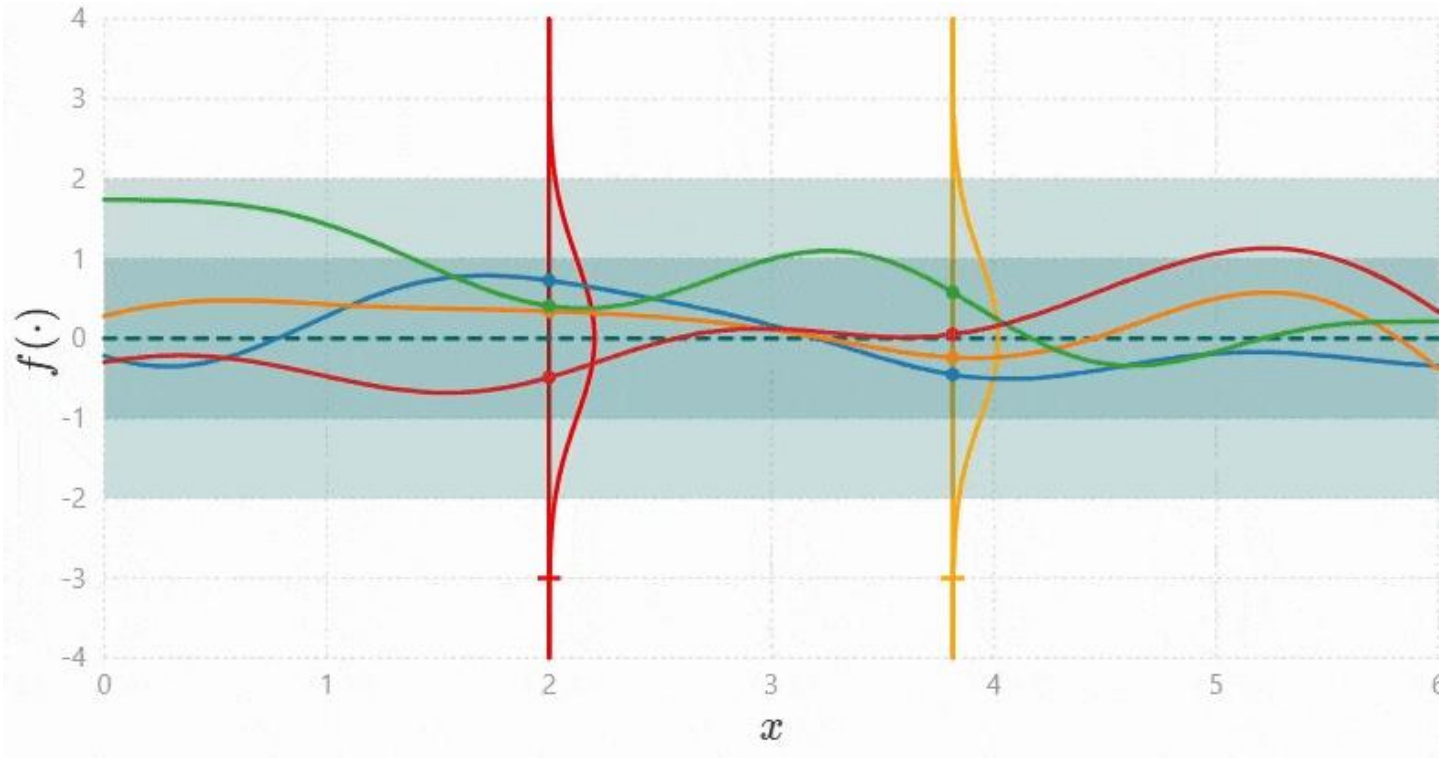
$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix} \right)$$

-> (like Infinite expansion) $f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big)$

Where $\operatorname{cov}\big(f(\mathbf{x_p}), f(\mathbf{x_q})\big) = k(\mathbf{x_p}, \mathbf{x_q})$

-> Subset of Process or marginalization $\mathbf{f} \sim \mathcal{N}\big(m(\boldsymbol{x}), K(X, X)\big)$

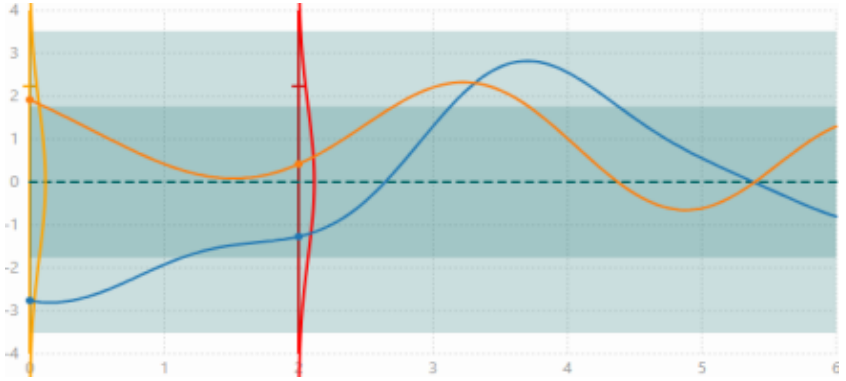
GP : function space

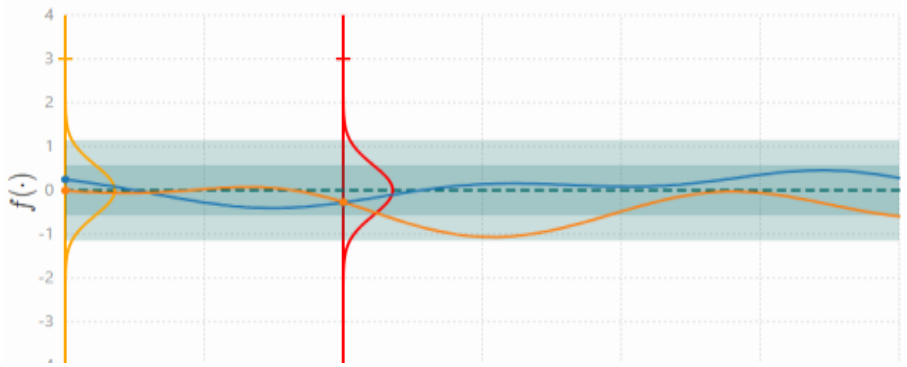


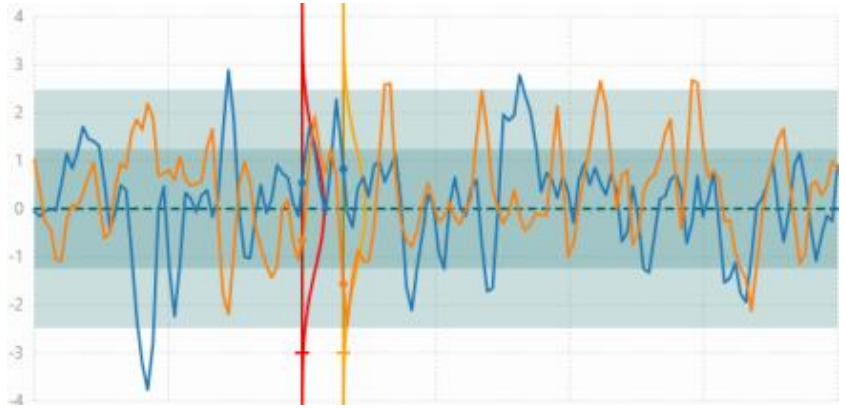
GP : kernel & hyperparameter make different prior

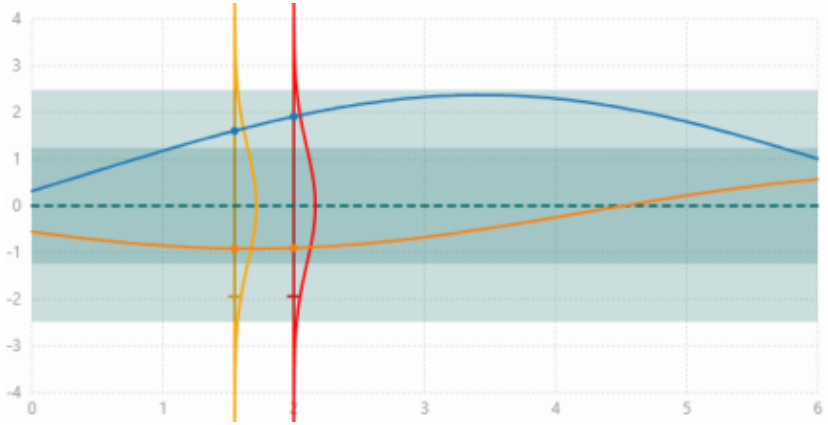
Kernel : Symmetric & positive semi-definite

$$\text{Squared exponential(SE)}: \; k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right)$$



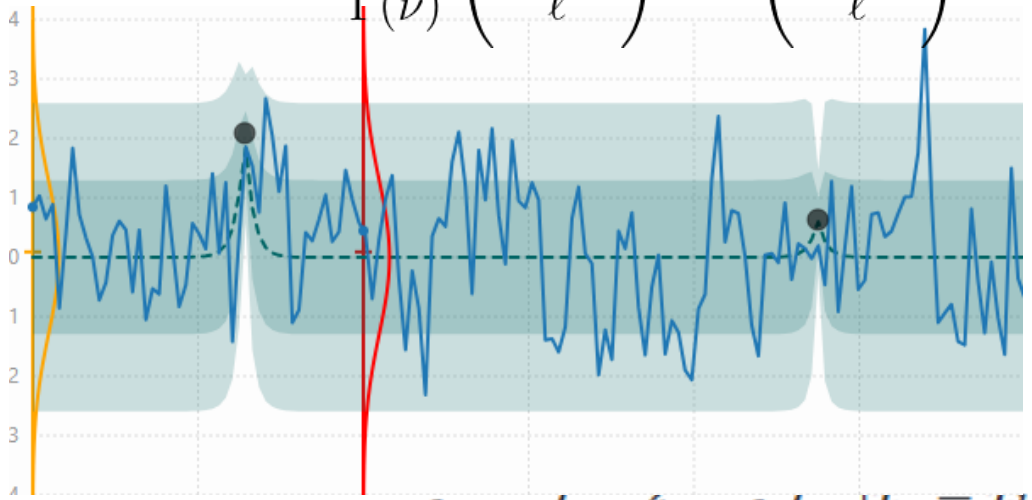




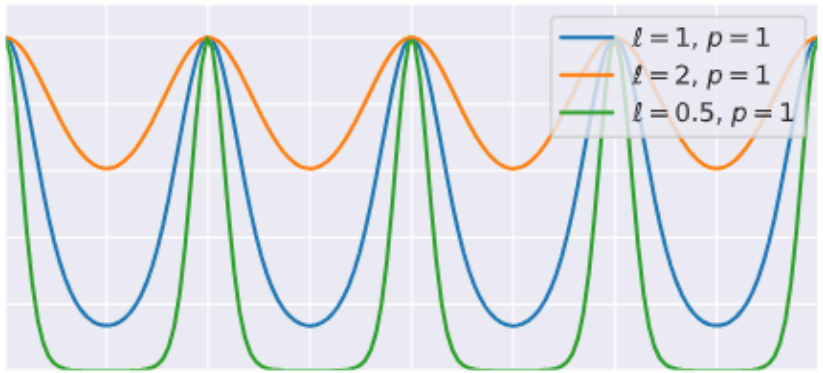


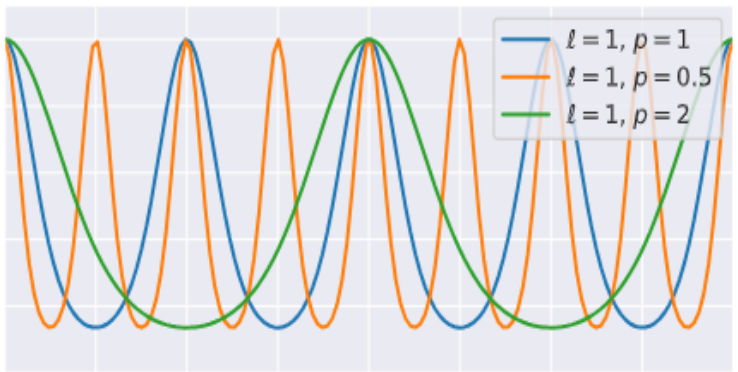
GP : kernel & hyperparameter make different prior

$$k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{\ell} \right)$$



$$k(x_a, x_b) = \sigma^2 \exp\left(-\frac{2}{\ell^2} \sin^2\left(\pi \frac{|x_a - x_b|}{p} \right) \right)$$





GP PPD inference

training : $\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, K(X, X)\right)$

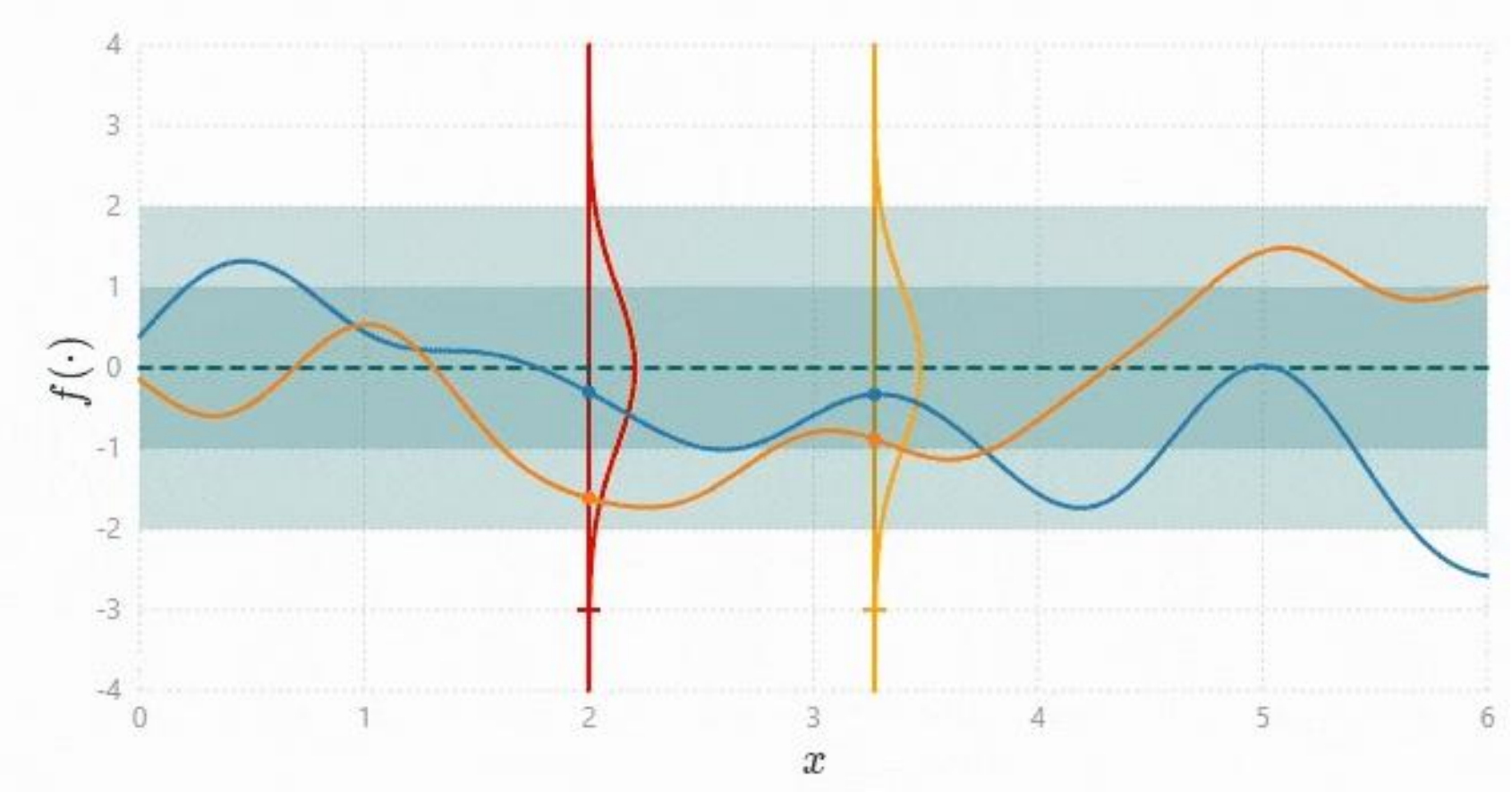
Test prior : $\mathbf{f}_* \sim \mathcal{N}\left(\mathbf{0}, K(X_*, X_*)\right)$

MVN : $$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

Posterior : $$\mathbf{f}_* | X_*, X, \mathbf{f} \sim \mathcal{N}\big(K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big)$$

V_0(Transformer can do BI)

GP : PPD baseline



Training framework : Prior data fitted Network

Sample prior datasets $D^{(i)} \sim p(\mathcal{D})$

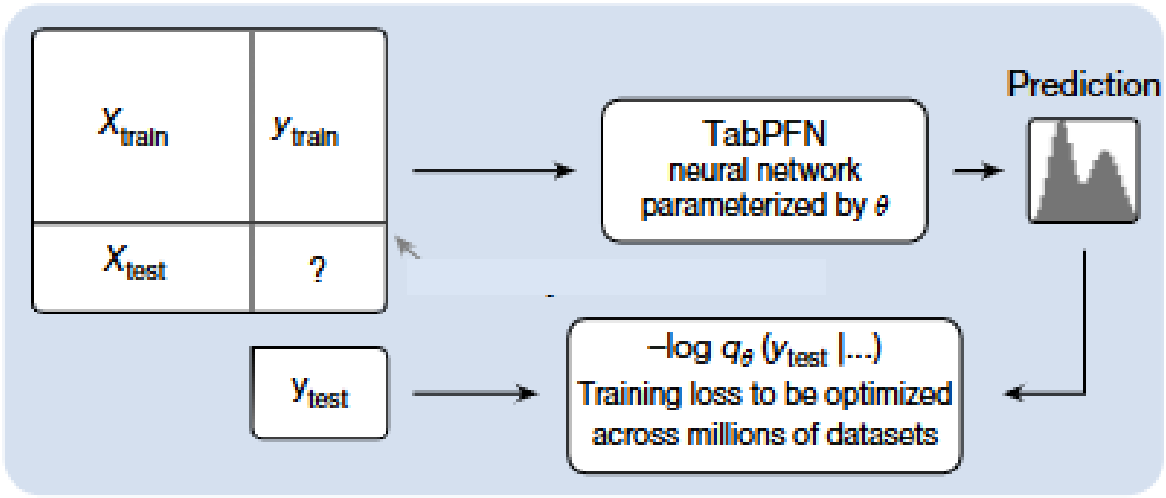
$$D^{(1)} = D^{(1)}_{train} \cup \{(x^{(1)}_{test}, y^{(1)}_{test})\}$$

$$\vdots$$

$$D^{(K)} = D^{(K)}_{train} \cup \{(x^{(K)}_{test}, y^{(K)}_{test})\}$$

Input : train D (x,y)
& test(query) x

TabPFN is trained on synthetic data to take entire datasets as inputs and predict in a forward pass



Training framework

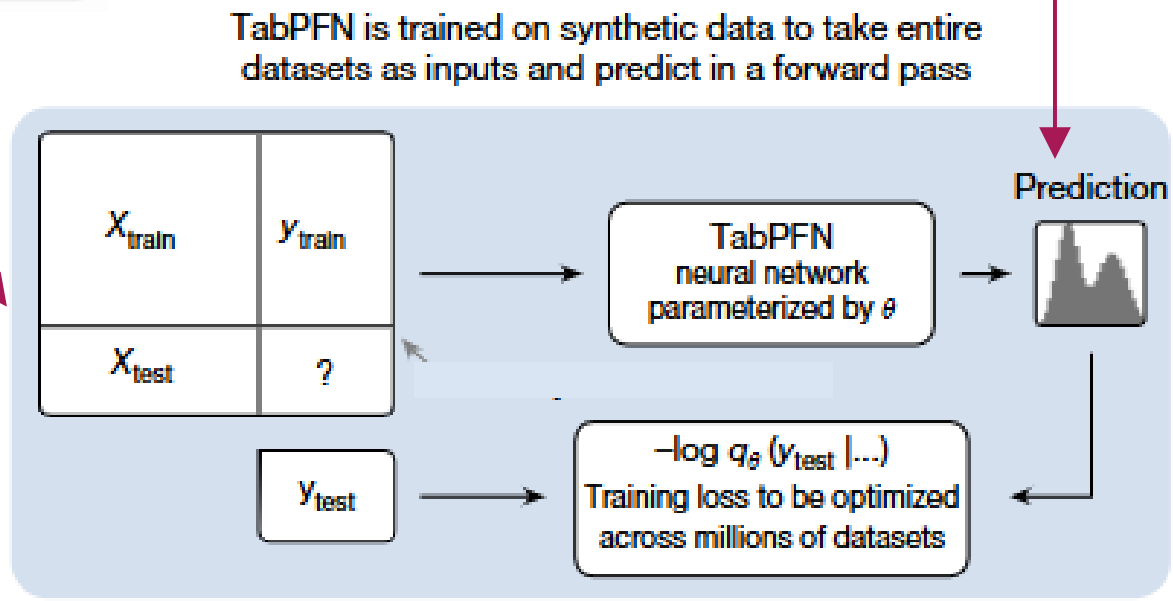
Sample prior datasets $D^{(i)} \sim p(\mathcal{D})$

$$D^{(1)} = D^{(1)}_{train} \cup \{(x^{(1)}_{test}, y^{(1)}_{test})\}$$

$$\vdots$$

$$D^{(K)} = D^{(K)}_{train} \cup \{(x^{(K)}_{test}, y^{(K)}_{test})\}$$

output : dist q on y



Training algorithm

Algorithm 1: Training a PFN model by Fitting Prior-Data

Input : A prior distribution over datasets $p(\mathcal{D})$, from which samples can be drawn and the number of samples K to draw

Output : A model q_θ that will approximate the PPD

Initialize the neural network q_θ;

for $j \leftarrow 1$ **to** K **do**

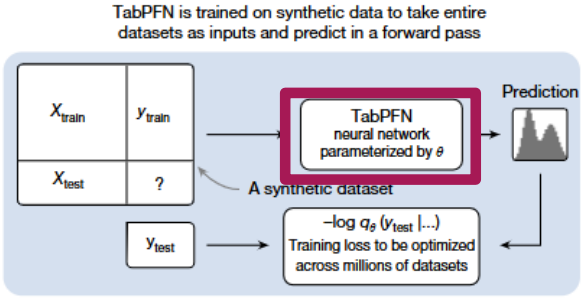
  Sample $D \cup \{(x_i, y_i)\}_{i=1}^{m} \sim p(\mathcal{D})$;

  Compute stochastic loss approximation $\bar{\ell}_\theta = \sum_{i=1}^{m}(-\log q_\theta(y_i|x_i, D))$;

  Update parameters θ with stochastic gradient descent on $\nabla_\theta \bar{\ell}_\theta$;

end

Architecture_body : encoder-only transformer



-input encoding (w/o positional encode, revised after)

Train : $\text{Linear}_x(x_i) + \text{Linear}_y(y_i)$

Test : $\text{Linear}_x(x_{test}) + \mathbf{0}$

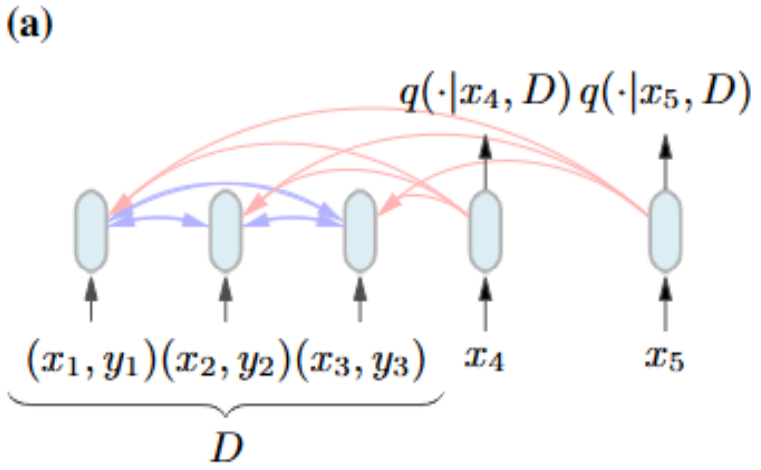
-> d * N (N=n+m)

-encoder-only transformer

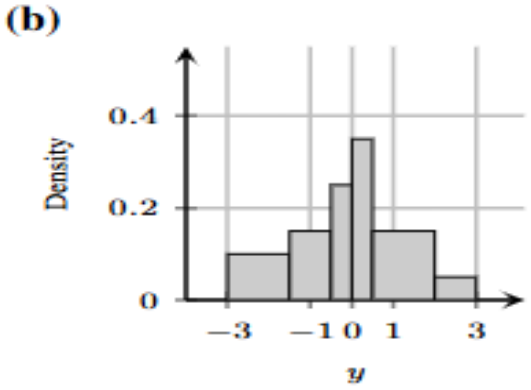
-attention mask

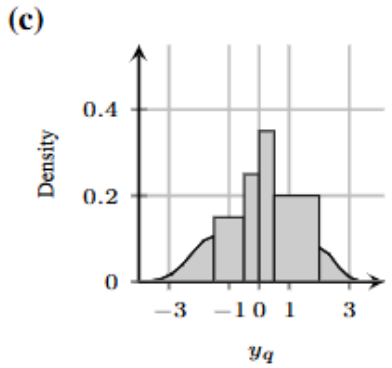
: query(test) : can not attend to other quer

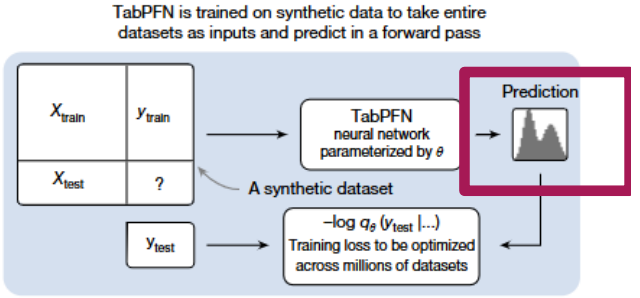
: train : attend to train and queries



Head : Piece-wise constant(Riemann) distribution







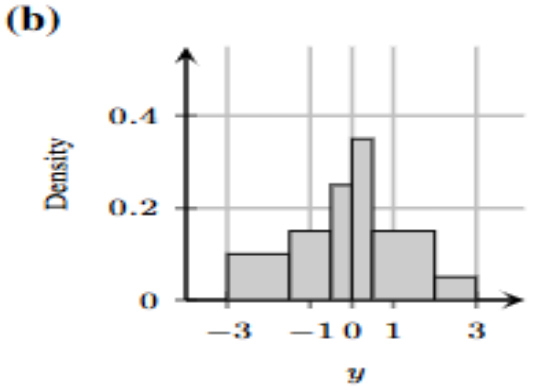
$$[\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{z}_{n+1}, \ldots, \mathbf{z}_{n+m}] \longrightarrow \begin{matrix} \mathrm{Head}(\mathbf{z}_{n+1}) \\ \ldots \end{matrix}$$

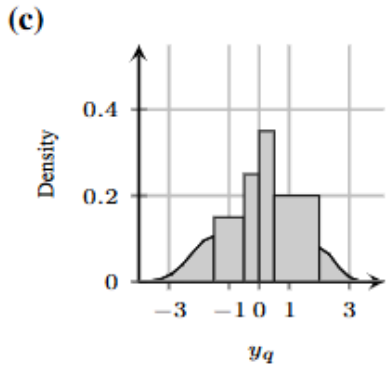
Head : q_theta = Rieman (softmax(z)) = $p_{b(y)}/w(b(y))$

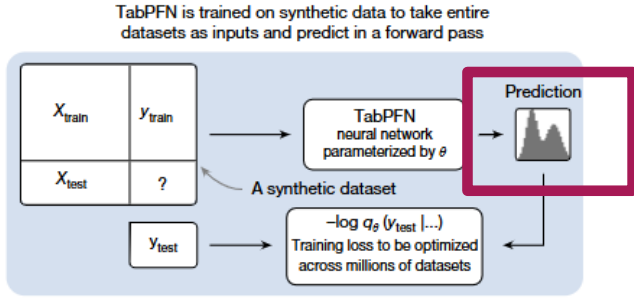
Bucket (default 1000)

Probability density
: discretized continuous distribution
: uniform dist on y in bucket b_i

Head : Piece-wise constant(Riemann) distribution







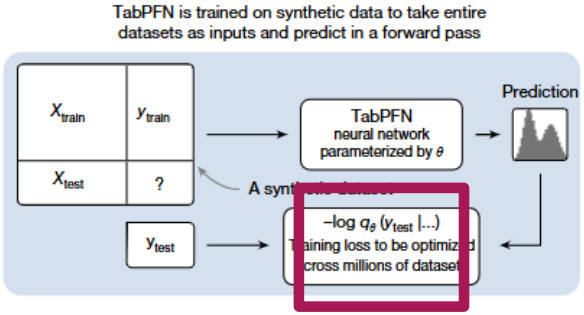
Head : q_theta = Rieman (softmax(z)) = $p_{b(y)}/w(b(y))$

Bucket (default 1000)

Theorem 1. *A Riemann distribution q_{B} with finite support, where B are the buckets, can model any distribution with a finite, almost everywhere continuous (Riemann-integrable) density function and full support up to arbitrary precision. That is,* $\forall p \forall \epsilon \exists \mathrm{B}. KL(q_{\mathrm{B}}, p) \leq \epsilon$. *Proof in App. C.*

Loss : Negative Log likelihood

$$\ell_\theta = \mathbb{E}_{D \cup \{x,y\} \sim p(\mathcal{D})}[-\log q_\theta(y|x, D)]$$



Conditional dist

$$\ell_\theta = -\int_{D,x,y} p(x,y,D) \log q_\theta(y|x,D) = -\int_{D,x} p(x,D) \int_y p(y|x,D) \log q_\theta(y|x,D)$$

$$= \int_{D,x} p(x,D) \mathrm{H}(p(\cdot|x,D), q_\theta(\cdot|x,D)) = \mathbb{E}_{x,D\sim p(\mathcal{D})}[\mathrm{H}(p(\cdot|x,D), q_\theta(\cdot|x,D))]$$

-not calculated based on ppd of prior

Transformer can do BI

Loss : NLL

$$\ell_\theta = -\int_{D,x,y} p(x,y,D)\log q_\theta(y|x,D) = -\int_{D,x} p(x,D)\int_y p(y|x,D)\log q_\theta(y|x,D)$$

$$= \int_{D,x} p(x,D)\mathrm{H}(p(\cdot|x,D),q_\theta(\cdot|x,D)) = \mathbb{E}_{x,D\sim p(\mathcal{D})}[\mathrm{H}(p(\cdot|x,D),q_\theta(\cdot|x,D))]$$

$\mathbb{E}_{x,D}[KL(p(\cdot|x,D),q_\theta(\cdot|x,D))]$: Equivalent to minimize KL divergenc

$$= -\mathbb{E}_{x,D}\left[\int_y p(y|x,D)\log\frac{q_\theta(y|x,D)}{p(y|x,D)}\right]$$

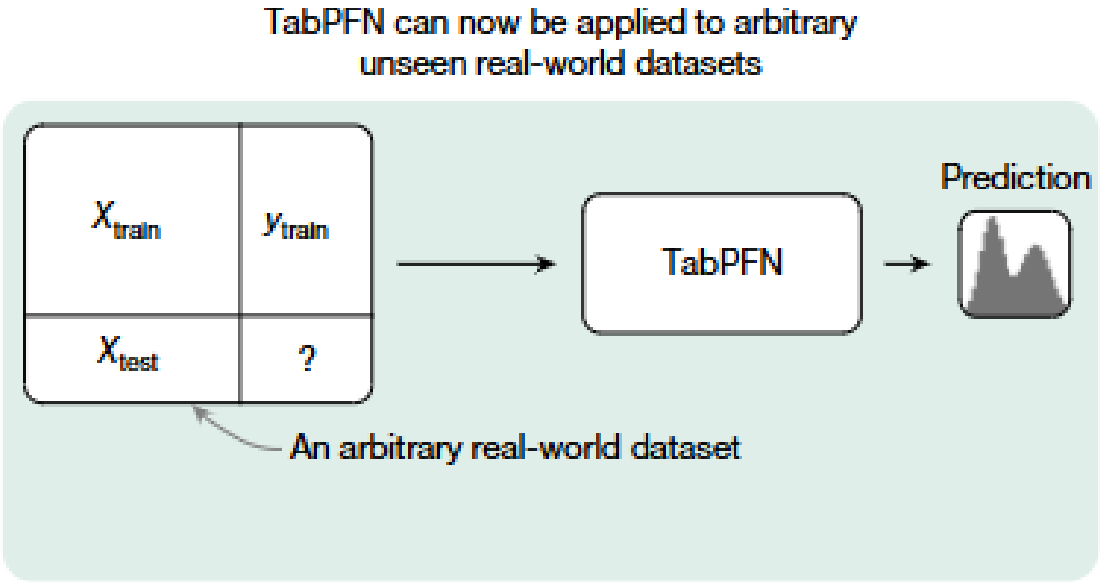
$$= -\mathbb{E}_{x,D}\left[\int_y p(y|x,D)\log q_\theta(y|x,D)\right] + \mathbb{E}_{x,D}\left[\int_y p(y|x,D)\log p(y|x,D)\right]$$

Log operation

$$= \mathbb{E}_{x,D}[\mathrm{H}(p(\cdot|x,D),q_\theta(\cdot|x,D))] - \mathbb{E}_{x,D}[\mathrm{H}(p(\cdot|x,D))]]$$

$$= \ell_\theta + C,$$

Inference framework (ICL)



- D(x,y) : context

- Single forward pass
- No extra tuning

Data generation

Sample prior datasets $D^{(i)} \sim p(\mathcal{D})$

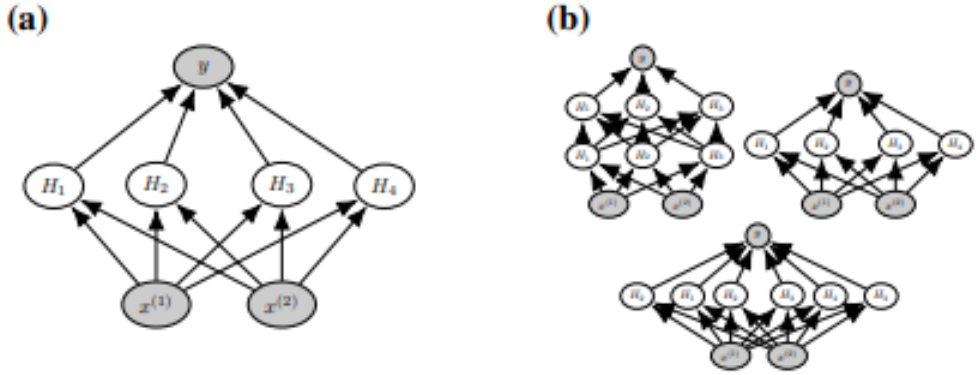
$$D^{(1)} = D^{(1)}_{train} \cup \{(x^{(1)}_{test}, y^{(1)}_{test})\}$$

$$\vdots$$

$$D^{(K)} = D^{(K)}_{train} \cup \{(x^{(K)}_{test}, y^{(K)}_{test})\}$$

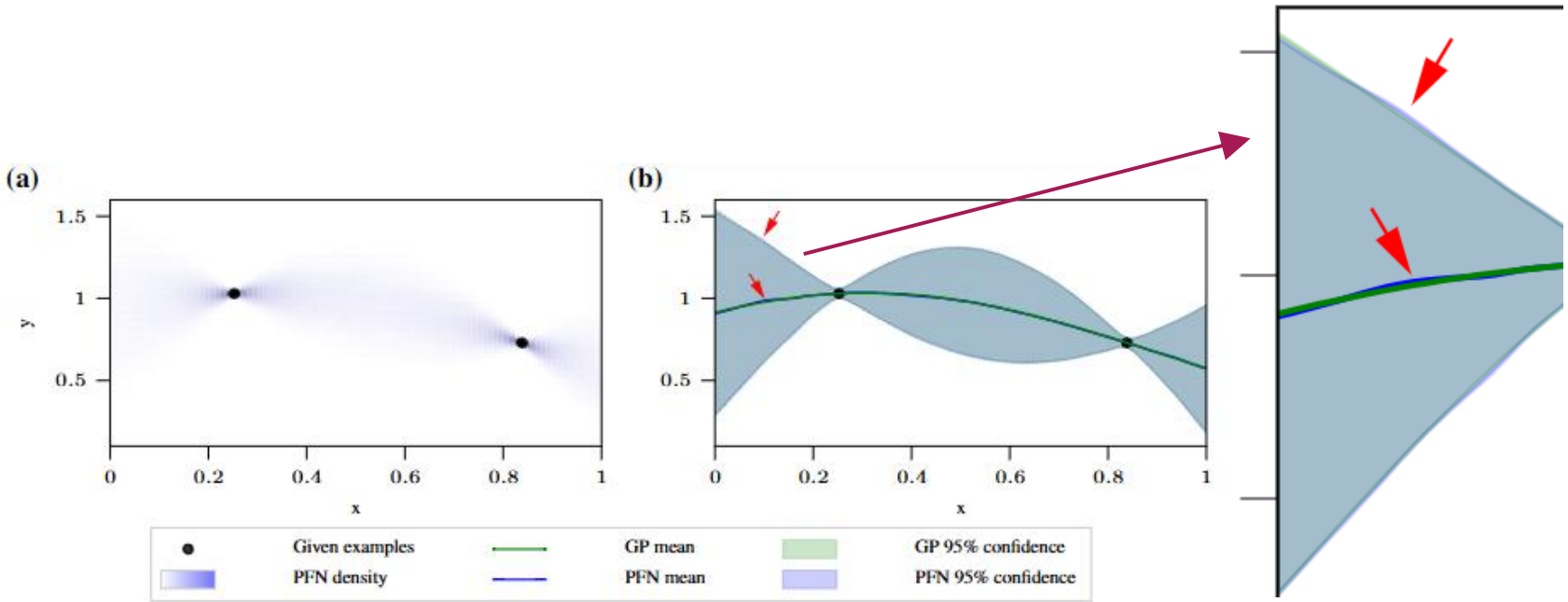
- GP

- Bavesian Neural Network

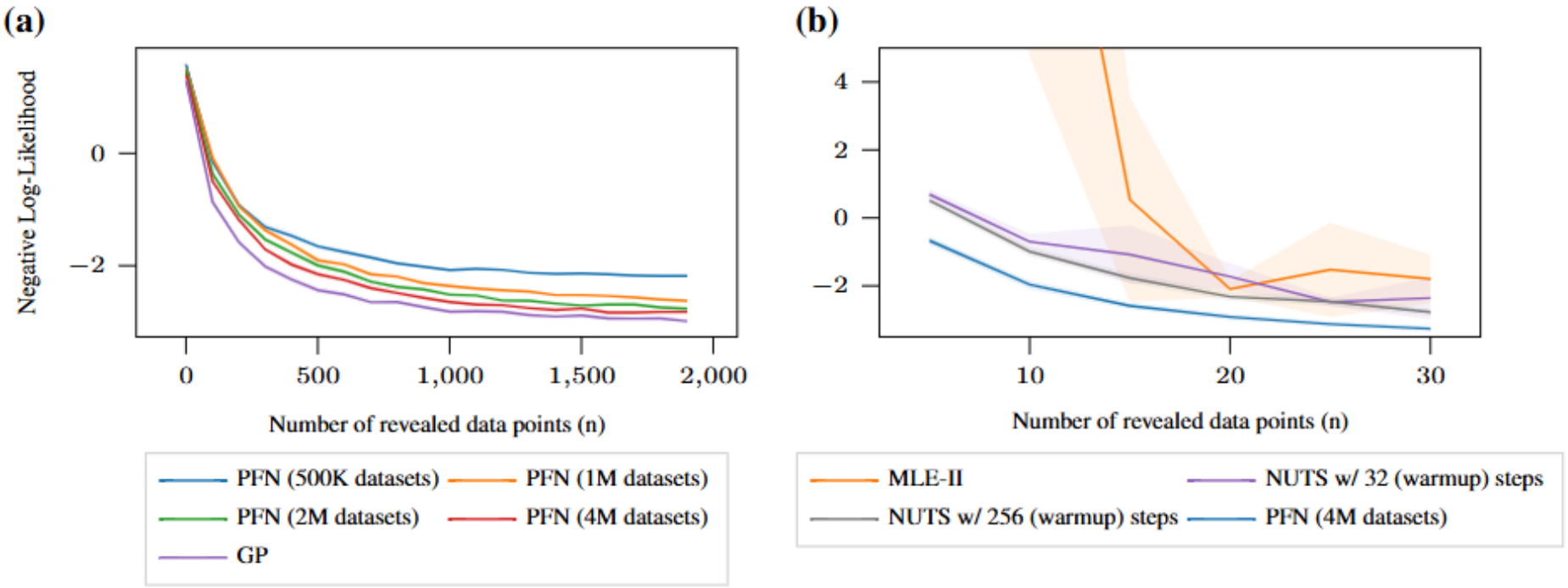


Transformer can do BI

mimic GP PPD



Approximate GP PPD



ICL. No need to tune (BNN PPD)

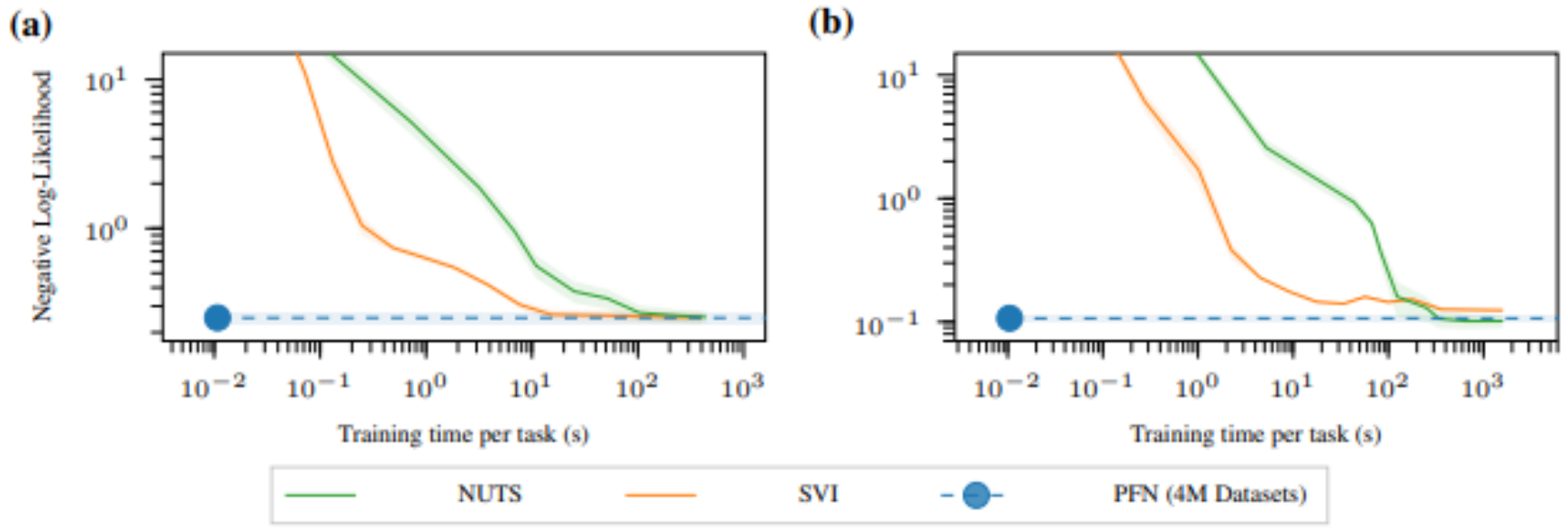


Figure 5: Time spent for Bayesian Inference per dataset compared to Prior-Data NLL. For NUTS we scale the number of warmup and sampling steps, for SVI, we scale the number of training steps and averaged samples. PFN is evaluated at only one setting (blue dot). We show two different BNN architectures (a) a BNN with 3 features, 2 layers and a hidden dimensionality of 5 and (b) a BNN with 8 features, 2 layers and a hidden dimensionality of 64.

Small dataset - classification

Metric	PFN-BNN	PFN-GP	Log. Reg.	GP	KNN	BNN	Catboost	XGB
Mean rank ROC AUC	**2.786**	3.833	4.690	5.286	6.214	5.000	4.833	3.357
Loss/Tie/Win vs PFN-BNN	–	14/1/6	16/1/4	17/0/4	17/1/3	17/1/3	13/1/7	12/2/7
Wilcoxon p. vs PFN-BNN	–	3.6e-13	3.6e-13	3.9e-13	3.6e-13	3.6e-13	1.9e-12	1.6e-06
Expected Calibration Error	**0.025**	0.067	0.157	0.095	0.093	0.089	0.157	0.066
Benchmark Time	GPU: **0:00:13** CPU: 0:04:23		0:09:56	0:24:30	0:00:34	12:04:41	2:05:20	20:59:46

V_0 : Just Proof of concept

V_1

TABPFN: A TRANSFORMER THAT SOLVES SMALL TABULAR CLASSIFICATION PROBLEMS IN A SECOND

Noah Hollmann[*,1,2] **Samuel Müller**[*,1] **Katharina Eggensperger**[1] **Frank Hutter**[1,3]
[1] University of Freiburg, [2] Charité University Medicine Berlin
[3] Bosch Center for Artificial Intelligence [*] Equal contribution.
Correspondence to noah.hollmann@charite.de & muellesa@cs.uni-freiburg.de

"Tabular data classification problems are still dominated by Gradient-Boosted Decision Trees(GBDT)"

TabPFN v_1

Slight changes from v_0

항목	기존 PFN (2022)	TabPFN (2023)
Number of class	Binary	2 ~ 10 개 (Multi class)
Class distribution	Balanced	Imbalanced도 가능
Output	스칼라 → Binary	스칼라 → Multi class 구간화
Prior	BNN based	BNN + SCM 기반, 클래스 샘플링 구조 포함

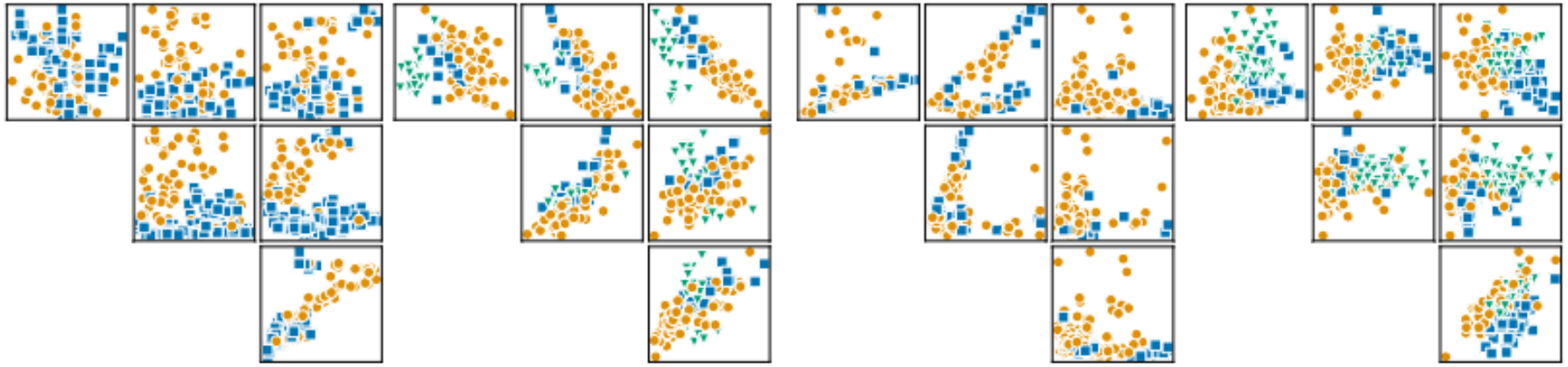
Slight change (More layer…)

Zero-padding for various # features

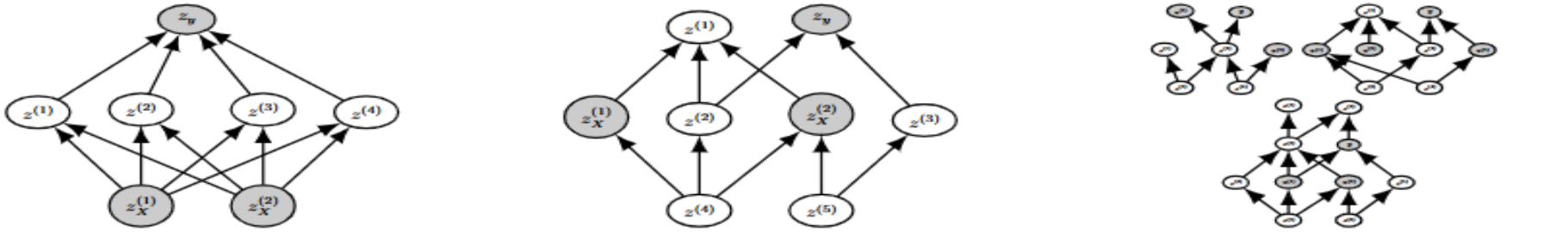
Ensemble
: 32 forward pass
: scaling, rotating feature & class label
:(actual performance is permutation variant)

TabPFN v_1

Prior : SCM(50%) + fully Bayesian BNN(50%)



(a) Synthetic datasets (b) Actual datasets

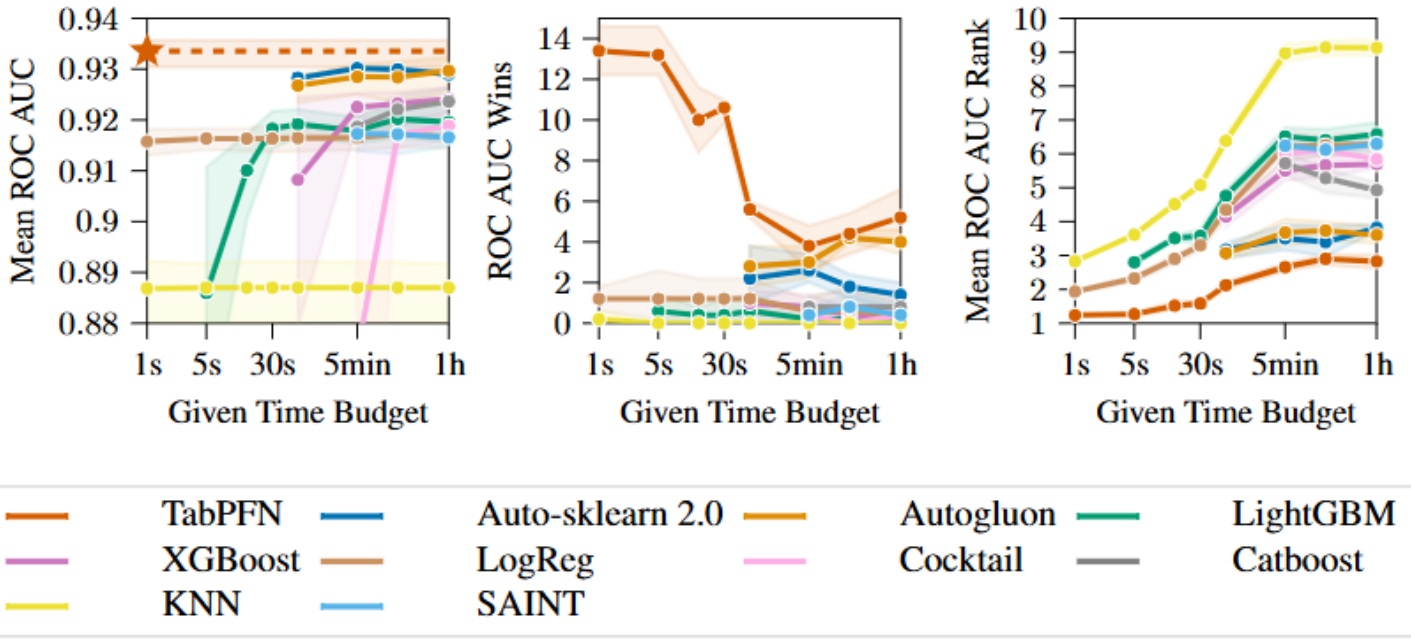


(a) A BNN (b) An SCM (c) SCMs sampled from the prior

Figure 2: Overview of graphs generating data in our prior. Inputs x are mapped to the output y through unobserved nodes z. Plots based on Müller et al. (2022).

TabPFN v_1

Competitive performance



Just need for a second to beat 1h trained model

TabPFN v_2

Article | Open access | Published: 08 January 2025

Accurate predictions on small data with a tabular foundation model

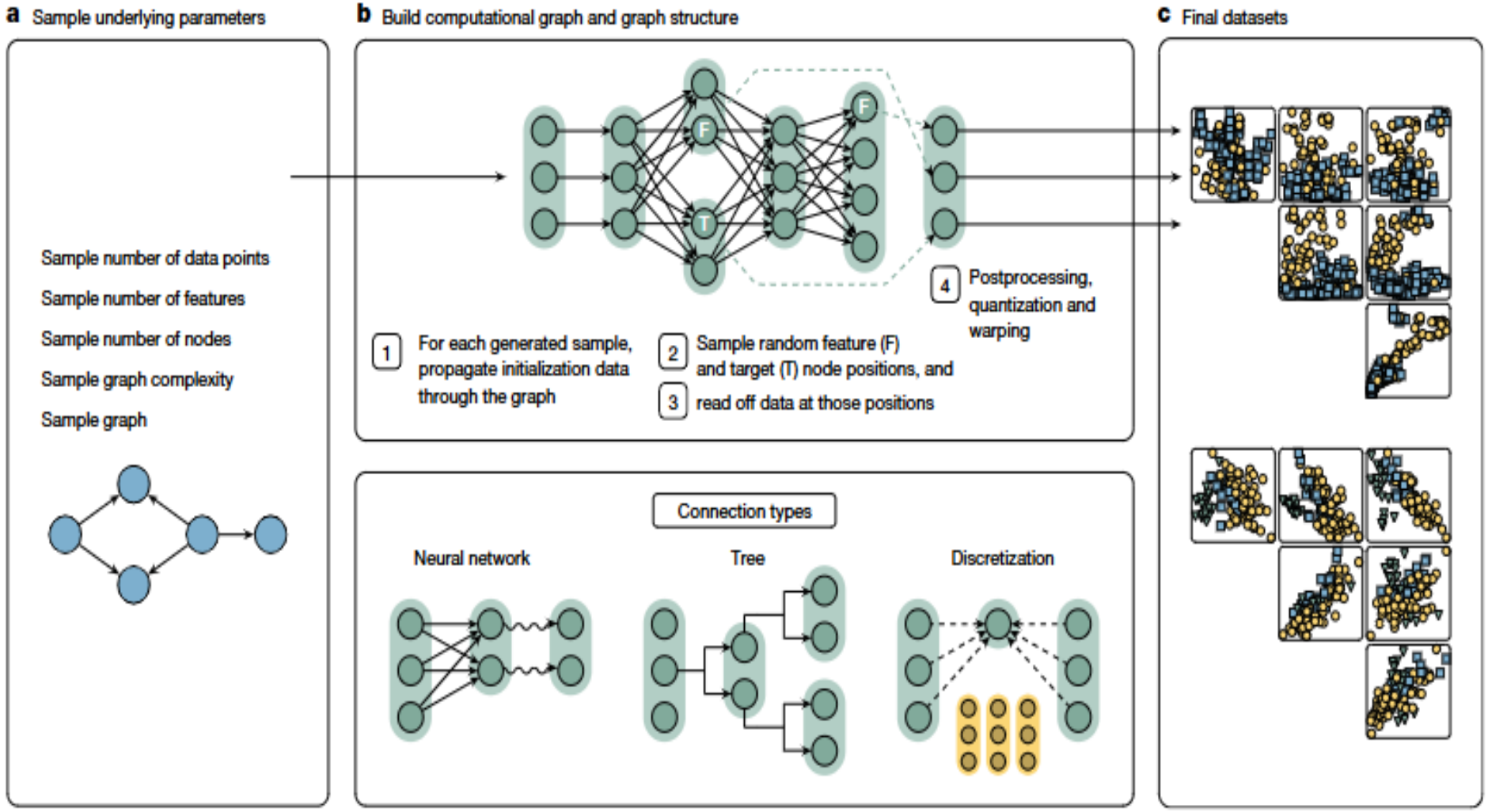
Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister & Frank Hutter

Nature **637**, 319–326 (2025) | Cite this article

496k Accesses | **691** Citations | **521** Altmetric | Metrics

TabPFN v_2

Toward scalable prior



Core changes : cell-wise encoding

Encode

: feature normalize $x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$

: linear($[x_{i,j}, y_i, \text{is_missing}_{i,j}]$)

-> row-wise characteristic

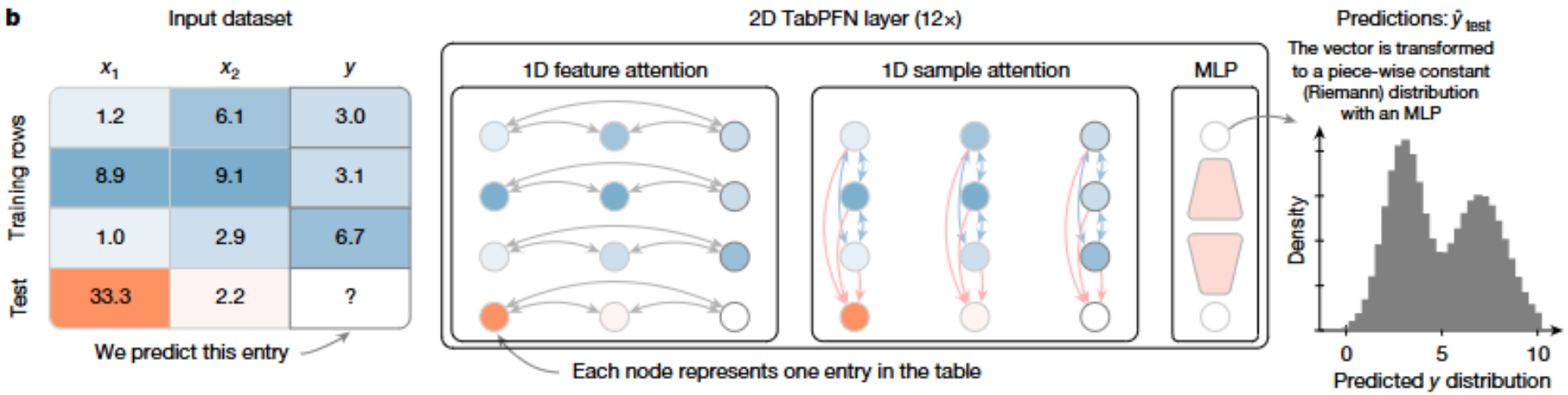
: random feature embedding $z_{i,j} = e_{i,j} + v_j$

-> Column-wise charactristic

=> Cell-wise encoding

TabPFN v_2

2-way attention



Feature attention

Sample attention (masking)

Further Engineering

Cache

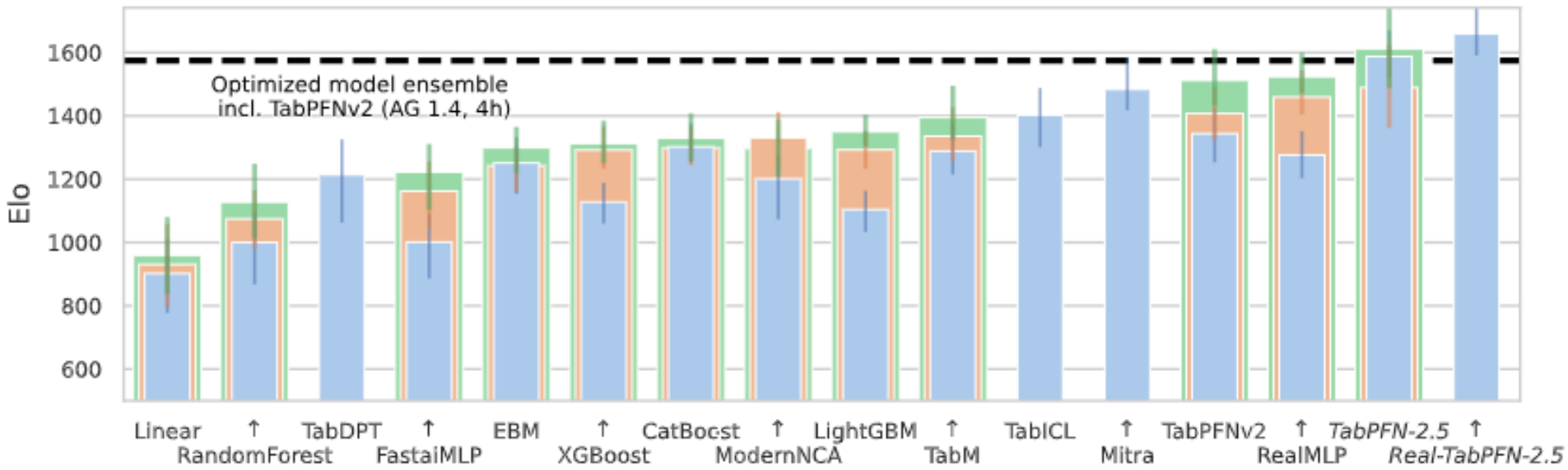
Data grouping (homogenous)

Feature subsampling

…

TabPFN v_2

Outperform GBDT



TabPFN v_2

Further Scaling

Model	Description	Limits	Publication
TabPFN-3-Plus	All in TabPFN-3, plus thinking mode and native text feature support. Available through the API and enterprise deployments.	1M rows × 200 features (trade-off: 100K × 2K, 1K × 20K) 160 classes	—
TabPFN-3	Latest foundation model. Up to 1M rows in-context with a row/feature trade-off.	1M rows × 200 features (trade-off: 100K × 2K, 1K × 20K) 160 classes	—
TabPFN-2.6	Good general-purpose choice for datasets up to 100K rows. Previous default model.	100K rows × 2K features 10 classes	—
TabPFNv2	Published in Nature. First TabPFN to handle real-world messy data — mixed types, missing values, categorical features. Best suited for smaller datasets.	10K rows × 500 features 10 classes	Nature (2025)

Takeaway

In context learning(ICL)

Prior Fitted Network(PFN)
: trained with only synthetic but realistic data
: inductive bias

Tabular data adaptation
: 2way attention / cell wise encoding