Transformers Can Learn Posterior Predictive Distributions In-Context

STA 701S Graduate Research Seminar

Gyeonghun Kang

Department of Statistical Science, Duke University

Joint work with Changwoo J. Lee (University of Florida)
and Xiang Cheng (Duke University)

ICML 2026

November 9, 2026

1. Motivation

Prior-Data Fitted Networks (PFN)

PFNs approximate posterior predictive distributions using networks pretrained on simulated datasets.

Paper title	**Citations**
Transformers Can Do Bayesian Inference (ICLR 2022)	673
Accurate predictions on small data with a tabular foundation model (Nature 2025)	2,306
TabPFN (ICLR 2023, Technical Reports 2025, 2026)	1,848
TabICL (ICML 2025, 2026)	642

Google Scholar, 31 October 2026. TabPFN: original + 2.5 + 3 + 3.5; TabICL: v1 + v2.

TabPFN-3.5 leads TabArena and BeyondArena, with recommended limits of $n = 10^6$ observations and $p = 6{,}000$ features (Jäger et al., 2026).

Applications include causal inference (Balazadeh et al., 2025), Bayesian optimization (Müller et al., 2023), time-series forecasting (Hoo et al., 2025), survival analysis (Qi et al., 2026), and posterior sampling (Reuter et al., 2025).

PFN Amortizes Bayesian Prediction

1. Draw a model parameter ω from its prior, then generate a dataset $\mathcal{D}_n$ and a new pair (x, y) conditional on that same parameter.
2. Pretrain a network on these simulated prediction tasks (Müller et al., 2022):

$$\vartheta^\star = \arg\min_{\vartheta} \; \mathbb{E}_{\mathcal{D}_n, x, y}[-\log q_\vartheta(y \mid x, \mathcal{D}_n)] \,.$$

   The unrestricted population minimizer is the posterior predictive density

$$p_n(y \mid x, \mathcal{D}_n) = \int p(y \mid x, \omega) p(\omega \mid \mathcal{D}_n) \, d\omega.$$

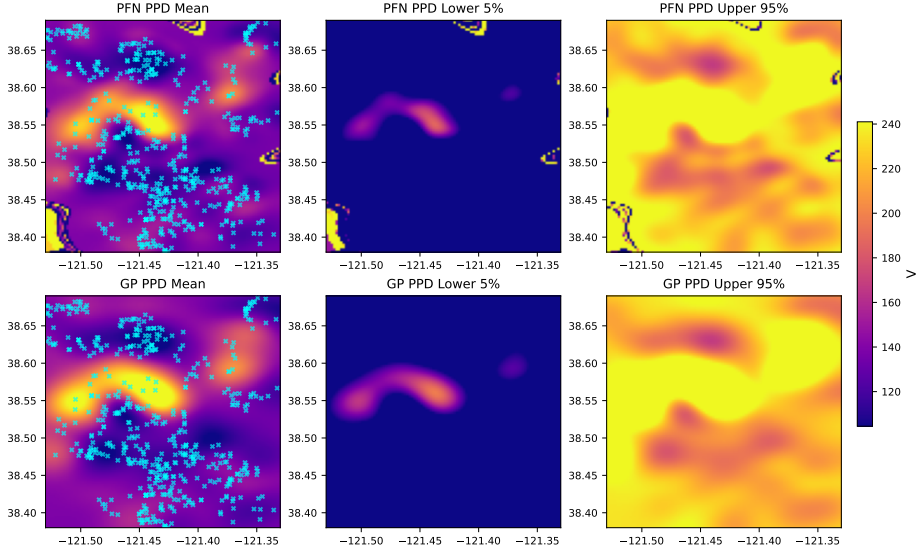
- After pretraining, given any observed dataset $\mathcal{D}_n$ and a new covariate value x, **a single network evaluation yields an approximate PPD**,

$$(x, \mathcal{D}_n) \mapsto q_\vartheta(\cdot \mid x, \mathcal{D}_n) \approx p_n(\cdot \mid x, \mathcal{D}_n),$$

  without repeating MCMC or variational inference for each dataset.

This follows amortized conditional density estimation in simulation-based inference, with the PPD as the target (Papamakarios and Murray, 2016; Radev et al., 2022; Zammit-Mangion et al., 2025).

Predictive Distributions in Practice



Sacramento house prices: predictive means and 5%/95% quantiles from a PFN (top) and an empirical-Bayes GP (bottom). Crosses mark context observations; color represents price per square foot.

Two Questions About Posterior Predictive Approximation

Let $\mathcal{Q}_{\mathrm{TF}}$ denote the class of conditional predictive densities represented by transformers.

1. **Do there exist transformer weights for which q_ϑ approximates the PPD across datasets and covariate values?**

   $$\forall \varepsilon > 0, \quad \exists\, q_\vartheta \in \mathcal{Q}_{\mathrm{TF}} : \quad \mathbb{E}_{\mathcal{D}_n, x}[\mathrm{TV}\{q_\vartheta(\cdot \mid x, \mathcal{D}_n), p_n(\cdot \mid x, \mathcal{D}_n)\}] < \varepsilon.$$

   Here TV is total variation distance; the expectation averages over the prior predictive distribution of $(\mathcal{D}_n, x)$.

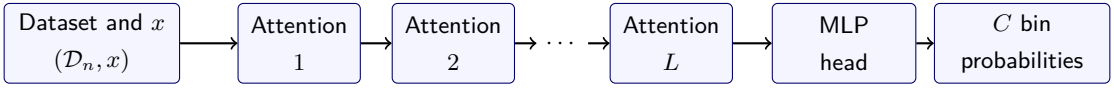
2. **How do its components perform this computation, and which architectural choices control predictive accuracy?**

We give an explicit construction for Gaussian process regression and evaluate its predictive accuracy through simulation experiments.

2. How Transformers Approximate the PPD

PFN Class of Conditional Predictive Densities

We study a simplified transformer with L attention blocks and a one-hidden-layer MLP output head.



- Arrange the labeled observations in $\mathcal{D}_n$ and the new covariate x as input tokens. Attention blocks pass information from the observations to the prediction token.
- An MLP head with C' hidden units maps the final prediction token to C logits $(\ell_1, \ldots, \ell_C)$.
- Softmax turns these logits into bin probabilities, giving a piecewise-constant predictive density:

$$\pi_c = \frac{e^{\ell_c}}{\sum_{a=1}^{C} e^{\ell_a}}, \qquad q_\vartheta(y \mid x, \mathcal{D}_n) = \frac{\pi_c}{\Delta_c} \quad \text{for } y \text{ in bin } c.$$

  Here Δ_c is the width of bin c.

Attention as Kernel Estimator

- For tokens $Z = [z_1, \ldots, z_{n+1}]$, learned maps K, Q, V form keys, queries, and values. Normalized attention at the last token is

$$[\mathrm{Attn}^{\mathrm{na}}(Z)]_{:,n+1} = \sum_{i=1}^{n} \frac{\kappa(Kz_i, Qz_{n+1})}{\sum_{a=1}^{n} \kappa(Kz_a, Qz_{n+1})} Vz_i.$$

  Masking excludes the query token from the values.

- With covariate similarities and response values, the prediction is

$$\widehat{y}_{n+1} = \sum_{i=1}^{n} \frac{\widetilde{\kappa}(x_i, x_{n+1})}{\sum_{a=1}^{n} \widetilde{\kappa}(x_a, x_{n+1})} y_i.$$

  For positive similarities $\widetilde{\kappa}$, this is a Nadaraya–Watson kernel smoother (Nadaraya, 1964; Watson, 1964; Tsai et al., 2019).

- Different key and query maps allow an asymmetric learned similarity; it need not be a positive semidefinite covariance kernel.

Gaussian Process and Kernel Ridge Regression

A GP prior gives an explicit posterior predictive distribution (Rasmussen and Williams, 2006):

$$\phi \sim \mathrm{GP}(0, \kappa), \qquad y_i = \phi(x_i) + \epsilon_i, \qquad \epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2).$$

Write $Z = (x, \mathcal{D}_n)$, $Y = (y_i)_{i=1}^n$, $G_{ij} = \kappa(x_i, x_j)$, and $k_x = (\kappa(x_i, x))_{i=1}^n$. The PPD is $p_n(\cdot \mid Z) = \mathcal{N}(\mu(Z), \tau(Z))$, with

$$\mu(Z) = k_x^\top (G + \sigma^2 I_n)^{-1} Y, \qquad \tau(Z) = \kappa(x, x) + \sigma^2 - k_x^\top (G + \sigma^2 I_n)^{-1} k_x.$$

- Both moments use the highlighted factor. A **Richardson iteration** approximates the required linear solves through repeated updates using kernel-weighted residuals.
- In our construction, unnormalized attention computes these residual sums, and the skip connection supplies ridge shrinkage.
- Running two recursions in parallel, with responses Y and kernel values k_x, computes the predictive mean and variance reduction.

Attention Approximates GP Predictive Moments

Each attention layer implements one Richardson update.

Theorem 3.1 (Exponential approximation of predictive moments)

For a constant step size $0 < \eta < 1/\{\lambda_1(G) + \sigma^2\}$, there exist transformer weights such that

$$\left\| \mathrm{TF}_{\vartheta,L}(Z^{(0)}) - (\mu(Z), \tau(Z))^\top \right\|_\infty \lesssim \underbrace{e^{-(1-\rho)L}}_{\text{exponential convergence in depth}}, \qquad 0 < \rho < 1.$$

Here $\rho = 1 - \eta\{\lambda_n(G) + \sigma^2\}$; $\lambda_1(G)$ and $\lambda_n(G)$ are the largest and smallest Gram eigenvalues.

- The same attention construction approximates the **predictive mean and variance in parallel**.
- More layers provide more solver iterations, reducing the approximation error exponentially.

From Predictive Moments to a Density

- Attention approximates (μ, τ). A shallow MLP approximates the Gaussian natural parameters $(\mu/\tau, -1/(2\tau))$.
- Evaluate the log density at bin midpoints ξ_c, then apply softmax:

$$\ell_c \approx \frac{\mu}{\tau}\xi_c - \frac{\xi_c^2}{2\tau}, \qquad \pi_c = \frac{e^{\ell_c}}{\sum_{a=1}^{C} e^{\ell_a}}, \qquad q_\vartheta(y) = \frac{\pi_c}{\Delta} \quad \text{on bin } c.$$

Theorem 4.1 (Transformer PPD approximation)

Under the regularity conditions, suitable transformer weights give

$$\mathrm{TV}(p_n, q_\vartheta) \lesssim e^{-(1-\rho)L} + \delta_{\mathrm{MLP}} + \frac{1}{C} + \varepsilon_{\mathrm{tail}}.$$

The four terms are **attention error**, **MLP error**, **binning error**, and **tail truncation**.

More layers and a wider MLP head improve the moment-to-density approximation; finer bins improve density resolution.

Depth and Bin Resolution Improve PPD Approximation

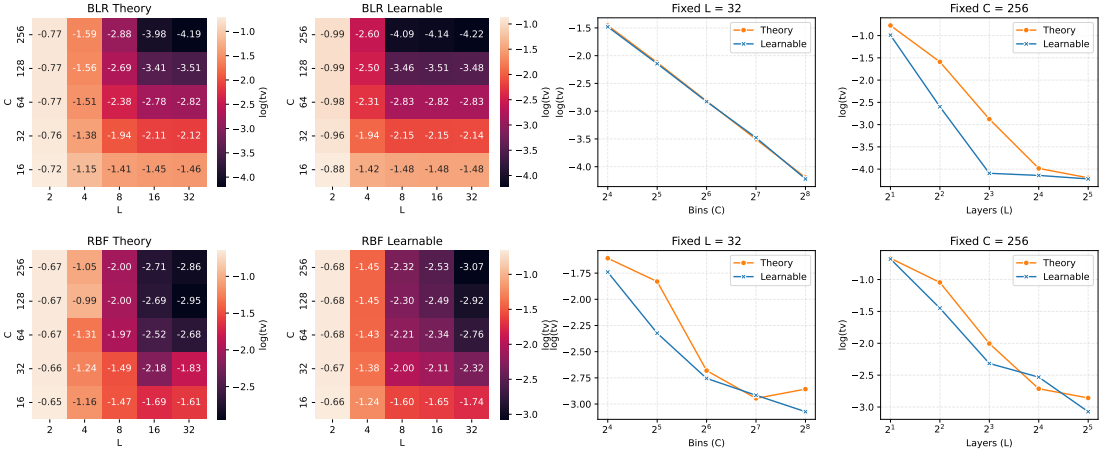


Figure 2, complete: BLR ($d = 5$, $n \in [128, 512]$) above; RBF GP ($d = 2$, $n \in [64, 128]$) below. All panels show $\log \mathbb{E}[\mathrm{TV}(p_{(a,b]}, q_\vartheta)]$; depth and bin resolution improve accuracy.

3. Generalization Across Sample Sizes

Generalizing Across Sample Sizes

PFNs are pretrained on datasets of varying sizes $n \in [n_{\min}, n_{\max}]$:

$$\vartheta = \arg\min_{\vartheta} -\mathbb{E}_n \mathbb{E}_{y, Z^{(0)}} \log q_\vartheta(y \mid Z^{(0)}).$$

- n-**generalization:** retain reliable point prediction and uncertainty quantification when the same pretrained network is evaluated beyond this range.

$$q_\vartheta(\cdot \mid x, \mathcal{D}_n) \approx p_n(\cdot \mid x, \mathcal{D}_n), \qquad n > n_{\max}, \quad \vartheta \text{ fixed.}$$

- Our attention construction makes the mechanism explicit: one shared ϑ must approximately solve

$$(G + \sigma^2 I_n)u = Gv, \qquad v \in \{Y, k_x\},$$

  for different n within a fixed iteration budget L.

- The fitted solver is therefore tuned to the spectra of G encountered during pretraining. Outside that range, its updates may become unstable or converge too slowly within L layers.

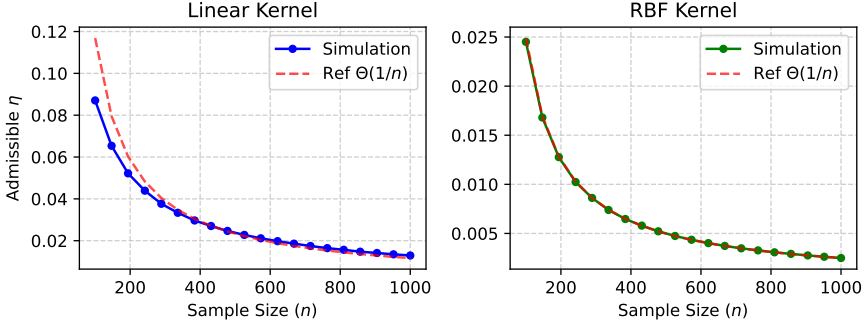
Larger Datasets Need Smaller Steps

The step size must balance stability and fast convergence within L layers.

Theorem 5.1

For $x_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d/d)$ and fixed d, linear and RBF kernels have a stable step-size limit of order $1/n$:

$$0 < \eta < \frac{2}{\lambda_1(G) + \sigma^2}, \qquad \lambda_1(G) = \Theta(n).$$



Stable step-size ranges shrink with n ($d = 16$, $\sigma^2 = 0.2$; 100 trials).

Preconditioning Stabilizes the Iteration

- A standard remedy is diagonal preconditioning: rescale the equations by the inverse row-sum matrix D^{-1} (Cutajar et al., 2016):

$$D^{-1}(G + \sigma^2 I_n)u = D^{-1}Gv, \qquad D = \mathrm{diag}(G\mathbf{1}_n).$$

  This changes the scale of the updates while preserving the solution of the original system.

- For the RBF kernel, $D^{-1}G$ is row stochastic. The largest eigenvalue of the preconditioned system is bounded independently of n:

$$\lambda_1\big(D^{-1}(G + \sigma^2 I_n)\big) \in [1, 1 + \sigma^2].$$

- A sufficient step-size range can therefore be chosen independently of sample size:

$$0 < \eta < \frac{2}{1 + \sigma^2}.$$

  The same step size can stabilize the system as the dataset grows.

Normalized Attention Implements Preconditioning

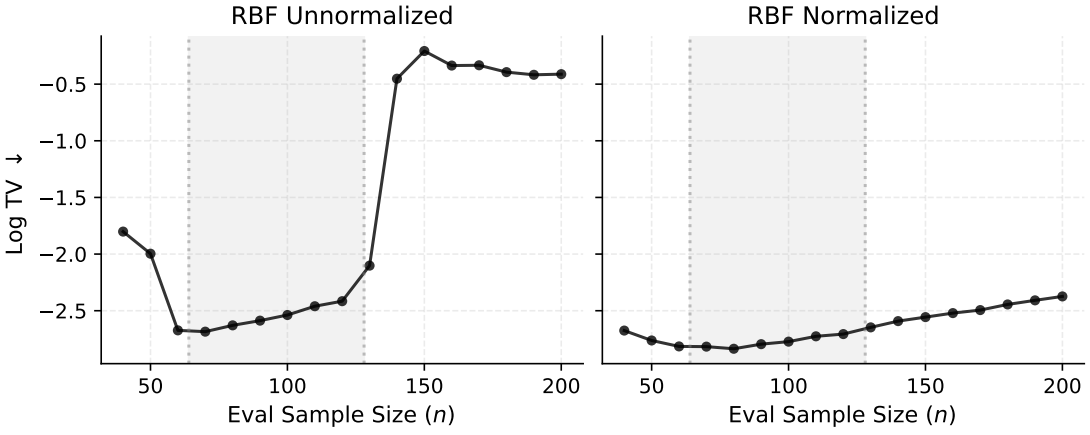
- Normalized attention divides each kernel weight by the total weight of the observations. In the solver, this rescales the update:

$$\text{effective step at } x = \frac{\eta}{s_x}, \qquad s_x = \sum_{i=1}^{n} \kappa(x_i, x).$$

- More kernel weight makes the effective step smaller. This lets the update adapt to the dataset while keeping the learned η fixed.

- In our construction, normalized attention and the corresponding scaling of the skip connection implement the preconditioned Richardson solver, denoted $\mathrm{TF}^{\mathrm{pr}}_{\vartheta,L}$.

Normalization Helps Beyond the Pretraining Range



Shading: pretraining sizes $n \in [64, 128]$. Lower log TV is better. RBF GP, $d = 8$, $L = 32$, $C = 256$.

- Without normalization, error rises sharply beyond the pretraining range.
- With normalization, error stays low on datasets larger than those used in pretraining.

Stable Updates Can Still Converge Slowly

- With the optimal constant step size, Richardson iteration has convergence factor

$$\rho^* = 1 - \frac{2}{\mathrm{cond}(G + \sigma^2 I_n) + 1} \in (0, 1).$$

  Here cond is the condition number. Larger values bring ρ^* closer to one, slowing convergence.

- For RBF kernels, conditioning worsens as n grows (Rasmussen and Williams, 2006; Burt et al., 2019). Ridge regularization lifts small eigenvalues and preconditioning bounds the largest, but the condition number still grows.

Theorem 5.2

For $x_i \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d/d)$, fixed d, and an RBF kernel,

$$\mathrm{cond}\{D^{-1}(G + \sigma^2 I_n)\} = \Theta(n) \qquad \text{almost surely for large } n.$$

Even after normalization, larger datasets require more iterations to recover predictive moments accurately.

More Layers Give the Solver Time to Converge

- As n grows, convergence slows, so fixed depth leaves more approximation error.
- More layers supply additional solver updates; broader pretraining exposes the model to more varied spectra.

Theorem 5.3

Under the Gaussian-design and RBF assumptions, suppose our preconditioned construction uses the optimal constant Richardson step size. To achieve

$$\left\|\mathrm{TF}^{\mathrm{pr}}_{\vartheta,L}(Z^{(0)}) - (\mu(Z^{(0)}), \tau(Z^{(0)}))^\top\right\|_\infty < \epsilon,$$

it suffices that

$$L \gtrsim n \log(1/\epsilon) \qquad \text{almost surely for large } n.$$

This is a sufficient depth for our explicitly constructed transformer, with fixed geometry and a single step size. Trained transformers are more flexible, so this is not a universal depth requirement for practical PFNs.

Testing Depth and Pretraining Range

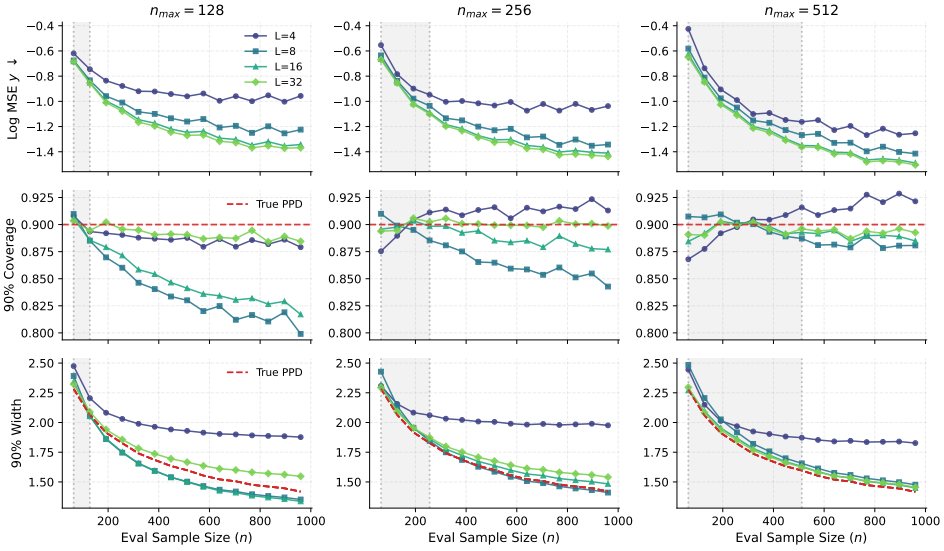
We test whether these mechanisms also appear when the network weights are learned.

- **More layers:** vary $L \in \{4, 8, 16, 32\}$ to give the network more solver updates.
- **Broader pretraining:** use $n \in [64, n_{\max}]$, with $n_{\max} \in \{128, 256, 512\}$, to expose the network to systems from larger datasets.
- Hold each fitted network fixed and evaluate on new datasets with $n' \in [64, 960]$ observations.

All models use normalized attention and skip terms. The data come from RBF GP regression with $d = 16$ and $C = 256$ predictive bins.

We assess **mean prediction** through MSE and **uncertainty** through 90% interval coverage and interval width. This checks whether improved point predictions also come with useful predictive distributions.

Depth and Pretraining Range Improve Generalization



Depth improves mean prediction; interval calibration also depends on the pretraining range. Assess width together with coverage, using the GP reference (red). Columns vary $n_{\max}$; colors vary L.

Q&A

References I

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=cp5PvcI6w8_`.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637 (8045):319–326, 2025.

Benjamin Jäger, Nick Erickson, Léo Grinsztajn, et al. TabPFN-3.5: Technical report, 2026. URL `https://arxiv.org/abs/2609.17895`.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 50817–50847, 2025. URL `https://proceedings.mlr.press/v267/qu25d.html`.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 102477–102555, 2026. URL `https://proceedings.mlr.press/v306/qu26a.html`.

References II

Vahid Balazadeh, Hamidreza Kamkari, Valentin Thomas, Benson Li, Junwei Ma, Jesse C Cresswell, and Rahul G Krishnan. Causalpfn: Amortized causal effect estimation via in-context learning. *Advances in Neural Information Processing Systems*, 38, 2025.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25444–25470. PMLR, 23–29 Jul 2023.

Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From tables to time: Extending TabPFN-v2 to time series forecasting, 2025. URL `https://arxiv.org/abs/2501.02945`.

Shi-ang Qi, Vahid Balazadeh, Michael Cooper, Russell Greiner, and Rahul G. Krishnan. SurvivalPFN: Amortizing survival prediction via in-context Bayesian inference, 2026. URL `https://arxiv.org/abs/2605.15488`.

Arik Reuter, Tim G. J. Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full Bayesian inference in context? In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 51531–51582, 2025. URL `https://proceedings.mlr.press/v267/reuter25a.html`.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, volume 29, 2016. URL `https://proceedings.neurips.cc/paper_files/paper/2016/file/6aca97005c68f1206823815f66102863-Paper.pdf`.

References III

Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33 (4):1452–1466, 2022. doi: 10.1109/TNNLS.2020.3042395. URL https://arxiv.org/abs/2003.06281.

Andrew Zammit-Mangion, Matthew Sainsbury-Dale, and Raphaël Huser. Neural methods for amortized inference. *Annual Review of Statistics and Its Application*, 12:311–335, 2025. doi: 10.1146/annurev-statistics-112723-034123. URL https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-112723-034123.

E. A. Nadaraya. On estimating regression. *Theory of Probability and Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020. URL https://epubs.siam.org/doi/10.1137/1109020.

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 26(4):359–372, 1964. URL https://www.jstor.org/stable/25049340.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4344–4353. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1443. URL https://aclanthology.org/D19-1443/.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006. URL https://gaussianprocess.org/gpml/.

References IV

Kurt Cutajar, Michael Osborne, John Cunningham, and Maurizio Filippone. Preconditioning kernel matrices. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2529–2538, New York, New York, USA, 20–22 Jun 2016. PMLR.

David Burt, Carl Edward Rasmussen, and Mark Van Der Wilk. Rates of convergence for sparse variational Gaussian process regression. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 862–871. PMLR, 09–15 Jun 2019.