THE ARROW OF TIME: WHAT TABULAR FOUNDATION MODELS MISS IN TIME SERIES FORECASTING

Andres Potapczynski[1,2]*, Ravi Kiran Selvam[1]*, Tatiana Konstantinova[1], Malcolm Wolff[1], Kin G. Olivares[1], Ruijun Ma[1], Michael W. Mahoney[1], Andrew G Wilson[1,2], Boris N. Oreshkin[1], Dmitry Efimov[1]
[1]Amazon Science, [2]New York University

ABSTRACT

In many time series forecasting settings, the target time series is accompanied by exogenous covariates, such as promotions and prices in retail demand; temperature in energy load; calendar and holiday indicators for traffic or sales; and grid load or fuel costs in electricity pricing. Ignoring these exogenous signals can substantially degrade forecasting accuracy, particularly when they drive spikes, discontinuities, or regime changes in the target series. Most current time series foundation models ignore exogenous covariates and make forecasts solely from the numerical time series history. Recent attempts to apply tabular foundation models to time series forecasting show promise but exhibit fundamental failure modes due to the absence of temporal inductive biases. In this paper, we provide a detailed characterization of these failure modes and propose targeted modifications to address them with empirical results.

Track: Research

1 INTRODUCTION

In many high-impact forecasting scenarios, leveraging *exogenous information*, i.e., inputs beyond the raw numerical target time series values, is essential. For example, in electricity price forecasting and consumer demand forecasting, information about planned prices and promotions, merchandising changes, holidays and local events, weather forecasts, and competitor pricing, are naturally encoded categorically and can shift demand sharply. Ignoring this information often induces large, systematic errors.

Despite the clear value of exogenous information, most existing time series foundation models (TSFMs), including `Chronos` (Ansari et al., 2024), `Sundial` (Liu et al., 2025), `TimesFM` (Das et al., 2023), `TimeMoE` (Shi et al., 2025), `TimeLLM` (Jin et al., 2024), and `LagLlama` (Rasul et al., 2023) either cannot incorporate exogenous covariates directly or require task-specific fine-tuning (Arango et al., 2025; Wang et al., 2024; Potapczynski et al., 2024). Such fine-tuning is often undesirable: it adds runtime overhead, complicates inference pipelines, increases deployment costs, and weakens the anonymity and isolation of downstream customer data. Therefore, a practical modern TSFM should natively incorporate accompanying exogenous covariates whenever they are available in a zero-shot manner.

There are a few foundation-like models that accept exogenous covariates in a zero-shot setting: in particular, `TabPFN-TS` (Hoo et al., 2025), `Moirai` (Woo et al., 2024) and more recently `Chronos-2` (Ansari et al., 2025). Crucially, though, `TabPFN-TS` is *not* a time series model *per se*—instead, it simply appends a handful of time series features to a tabular foundation model. Therefore, it lacks core temporal inductive biases. As we discovered and describe below, a central problem is that the architecture of `TabPFN-TS` is invariant to the order of the data. Order invariance is a reasonable inductive bias in the tabular i.i.d. case, but it is *not* a reasonable inductive bias for the time series context, where the "arrow of time" defines an important structure. In practice, this bias leads to characteristic failure modes when forecasting with `TabPFN-TS`.

*Equal contribution

Our main contributions are the following:

- We provide a detailed characterization of the shortcomings of existing tabular PFNs such as `TabPFN-TS` adapted *for time series forecasting*. In particular, we show that `TabPFN-TS` has intrinsic limitations due to the i.i.d. assumption that informs how the synthetic training data is generated, as well as due to its architectural specification. For these reasons, it fails to understand temporal auto-correlations, making it challenging to predict ordered patterns accurately. Based on these findings, we argue that existing tabular PFNs are *not* suitable as *time series* FMs as it is. (Section 2.)
- We propose modifications that can circumvent these limitations through time-aware data generation procedures and architectural choices that reflect the importance of order in time series data.
- We present preliminary empirical results demonstrating that architectural modifications, specifically the incorporation of positional encodings and adjustments to the attention mechanism, yield improved performance in time series forecasting tasks relative to directly adapting tabular foundation models.

2 Failure Modes of TabPFN-TS

`TabPFN-TS` (Hoo et al., 2025) introduces a series of manually engineered time series features into the tabular FM `TabPFN-v2` (Hollmann et al., 2025) in order to make time series forecasts. Although `TabPFN-TS` achieves competitive performance on several time series forecasting benchmarks, it exhibits fundamental failure modes due to the absence of time series specific inductive biases, raising concerns about the deployment of such models in industry-critical applications. Here, we describe several failure modes; see Figure 1.

Dependency on manually engineered frequency features. `TabPFN-TS` relies on a running index feature as well as frequency features that are taken from the timestamp of the data (such as day-of-week, day-of-month, month-of-year, etc.) or estimated frequencies obtained through a FFT decomposition of the time series (Hoo et al., 2025). That is, $x_{t,j} = \sin(2\pi \frac{\tau(t)}{P_j})$ or $x_{t,j} = \cos(2\pi \frac{\tau(t)}{P_j})$ where, for example, in the case of day-of-week $\tau(t) \in \{1, \cdots, 7\}$ and $P_j = 7$, and so forth. As seen in Figure 1(a), if the frequencies are not used, then `TabPFN-TS` only estimates the mean of the previous observations. `TabPFN-TS` makes accurate predictions when the relevant frequencies are explicitly included in the data, but otherwise it struggles to capture patterns that do not align with regular calendar structures.

Weak trend extrapolation. As already noted by Hoo et al. (2025), `TabPFN-TS` demonstrates a limited ability to extrapolate time series trends. This is seen in Figure 1(b). This phenomenon most likely results from the model's inability to consider the order of the data when estimating the trend.

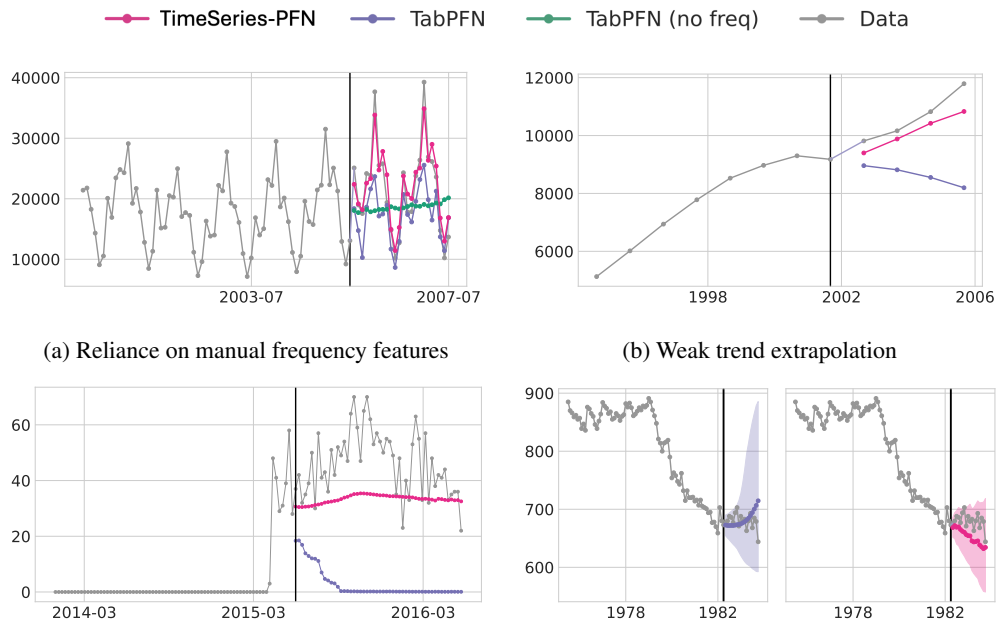
Lack of a recency bias. `TabPFN-TS` treats all historical time points equally when making predictions. Many applications operate in environments with constant distribution shifts, e.g., the underlying data changes over time due to factors like promotions, policy changes, or macroeconomic conditions. Accurately predicting under these distribution shifts is critical for a reliable deployment of time series models. Figure 1(c) shows that `TabPFN-TS` struggles to capture a sudden uptick in demand, failing to forecast based on the most recent observations.

Generic confidence intervals. `TabPFN-TS` produces confidence intervals that emphasize the entire historical context rather than weighting observations according to their consistency with the prevailing trend in the time series. Figure 1(d) clearly shows this phenomenon where the confidence interval simply reflects values obtained in the distant past. This failure undermines trust and complicates decision-making in industry critical time series applications.

Inability to learn ordered patterns. Ordered patterns such as ordered seasonal patterns that span across multiple time steps are very common in industry applications such as demand forecasting (where a product has a gradual increase in demand until its promotion date and sharply drops after the promotion). These types of patterns are not purely cyclical, but instead they reflect structured temporal dependencies that unfold over multiple horizons.

The underlying reason why `TabPFN-TS` exhibits these failure modes is that it was trained and designed for i.i.d. data. Although the model incorporates certain time-series specific features, these

are imposed *post hoc*, and the core architecture does not capture the temporal dependencies intrinsic to the data.



(a) Reliance on manual frequency features

(b) Weak trend extrapolation

(c) Lack of recency bias

(d) Poorly calibrated confidence intervals

Figure 1: **Failure modes of `TabPFN-TS` for time series data that `TimeSeries-PFN` addresses.** We provide illustrative examples of each failure case with different real time series: we use a time series in Tourism Monthly for (a), in Tourism Yearly for (b), in M5 Weekly for (c), and in M1 Monthly for (d). In the plots, the training data is to the left of the black line, and the forecasts are to the right. *(a)* When `TabPFN-TS` is not given frequency features, it predicts an average of prior history (green line). In contrast, `TabPFN-TS` might capture some time patterns when frequency features are available but miss others outside the frequency range (e.g., it does not capture the largest spikes). *(b)* `TabPFN-TS` has problems extrapolating trends especially in short context cases. *(c)* The predictions of `TabPFN-TS` erroneously revert back to zero, as that is the most common value in the context. *(d)* The range of the 90% confidence intervals in `TabPFN-TS` substantially increases to capture previously seen values rather than to reflect the uncertainty over the trend of the time series.

3 Proposed Modifications for Time Series

We now present few of the data-level and architectural interventions that can enable PFN models to explicitly leverage ordering and temporal dependencies inherent in time series data.

3.1 Time-Aware Synthetic Data Generation

The baseline tabular data generation samples DAGs from random growing networks (RGN) with preferential attachment, then defines SCMs where each node is determined as a function of parent nodes. To induce time series structure, sampling root node values $(v_{t,r})_{t=1}^{T}$ through a stochastic process helps introduce temporal dependency. Specifically, root nodes can become combinations of sine and cosine functions with randomly sampled frequencies $(\phi_1^{(r)}, \phi_2^{(r)})$ and amplitudes $(\alpha_1^{(r)}, \alpha_2^{(r)})$. That is, $v_{t,r} = \alpha_1^{(r)} \sin(\phi_1^{(r)} t) + \alpha_2^{(r)} \cos(\phi_2^{(r)} t)$, for all $t = 1, \ldots, T$. As a result, this can now generate datasets $\mathcal{D}_\mathcal{G} = (\boldsymbol{x}_t, y_t)_{t=1}^{T}$, where y_t are correlated, in contrast to sampling root nodes as $v_{i,r}$ independently for each i.

3.2 ARCHITECTURAL MODIFICATIONS

The tabular foundational model architecture inherently lacks positional awareness, as tabular entries are fundamentally unordered. However, when adapting such models for time series data, several key modifications can be implemented to capture the inherent temporal inductive biases, including,

Positional Encodings. With time-dependent data generation, it makes sense to introduce inductive bias reflecting temporal relationships. So, incorporating position encodings like `RoPE` embeddings (Su et al., 2023) to the attention mechanism in $\texttt{AttnSamp}^{(\ell)}(\cdot)$ can make key-query interactions obey $\boldsymbol{q}_{t+h}^{\intercal}\boldsymbol{R}_h\boldsymbol{k}_t$, where $\boldsymbol{R}_h$ is a weight matrix such that $\boldsymbol{q}_{t+h}^{\intercal}\boldsymbol{R}_h\boldsymbol{k}_t \to 0$ as $h \to \infty$. In other words, the keys and queries of nearby observations are weighted more. Similar to Vaswani et al. (2017), employing another notion of absolute positional encodings of the form $\boldsymbol{\Omega} \in \mathbb{R}^{T\times D}$ can also be helpful.

$$\Omega_{t,2d+1} = \sin\left(2\pi t\frac{2^{2d+1}}{2^{12}}\right) \quad \text{and} \quad \Omega_{t,2d} = \cos\left(2\pi t\frac{2^{2d}}{2^{12}}\right),$$

Full Attention. In `TabPFN`, test observations do not attend to each other, only to train observations. For time series, future *exogenous* information should inform current predictions so enforcing $p(y_{T+h}|(\boldsymbol{x}t)t = T + 1^{T+H}, (\boldsymbol{x}t, yt)_{t=1}^{T})$ for all $h = 1, \ldots, H$ can help to capture ordered patterns. This can be achieved by allowing all points to attend to each other in $\texttt{AttnSamp}^{(\ell)}(\cdot)$.

4 EMPIRICAL EVALUATION

Building upon the aforementioned modifications, we adapt the PFN architecture to train directly on exogenous time series data. We evaluate our proposed model, `TimeSeries-PFN`, on the M5 competition dataset (Makridakis et al., 2022) across multiple aggregations and frequencies. M5 competition dataset contains exogenous information such as price and promotional events to inform the predictions. The M5 dataset contains units sold daily for a given SKU (product) with identifying attributes such as brand, store and state. At the SKU and store level, M5 contains over 30K time series. We create multiple versions of the M5 dataset by aggregating across time (to weekly and monthly grains) and across geographies (to state and store grains).

As presented in Table 1, preliminary results indicate that `TimeSeries-PFN` demonstrates substantial performance improvements over both tabular foundational models such as `TabPFN-TS` and univariate foundational models across most dataset-frequency combinations.

We use the following well-established metric to report accuracy.

RMSSE is a point prediction metric used in the original M5 competition (Makridakis et al., 2022), which is defined as:

$$\text{RMSSE}(y, \hat{y}) = \frac{\frac{1}{H}\sum_{t=T+1}^{T+H}(y_t - \hat{y}_t)^2}{\frac{1}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1})^2}.$$

The motivation for RMSSE is two-fold. First, it compares the predictions against a naive baseline, giving us a sense of how easy or hard it is to make predictions for a specific time series. Second, it focuses on a square error thereby penalizing models that do not capture spikes induced by exogenous variables (e.g., promotion-induced sales spikes).

Training setup. We train our models for 400K steps using a batch size of 64 with a learning rate of 1e-4, no weight decay, 20K linear warm-up steps, and we used a cosine annealing schedule that terminates with a learning rate of 1e-6. We vary the number of samples and number of features available to the model for each batch. The number of samples ranges from 34 to 512 and the number of features from 2 to 64 and we predict for a horizon of up to 128. Our model has a total parameter count of 11M (same as the `TabPFN-TS` model).

5 CONCLUSION

This work identifies critical failure modes when applying tabular prior-data fitted networks to time series forecasting. We demonstrate that `TabPFN-TS`, despite incorporating time-series features,

Level	RMSSE	M5(D-B)	M5(W-B)	M5(M-B)	M5(D-S)	M5(W-S)	M5(M-S)
State	TimeSeries-PFN $^{(0x)}$	**0.580**	1.652	**2.191**	<u>0.973</u>	**1.561**	**2.588**
	TabPFN-TS$^{(0x)}$	<u>0.608</u>	<u>1.253</u>	2.580	1.006	1.666	<u>2.636</u>
	Moirai-Large$^{(\dagger x)}$	0.844	1.669	3.546	0.992	1.710	2.882
	Chronos-Large$^{(0)}$	0.655	**1.237**	2.484	1.007	1.847	2.788
	Sundial-Base$^{(0)}$	0.720	2.010	<u>2.405</u>	**0.933**	<u>1.649</u>	2.841
Store	TimeSeries-PFN $^{(0x)}$	<u>0.675</u>	1.829	**2.208**	0.990	**1.449**	**2.049**
	TabPFN-TS$^{(0x)}$	**0.651**	<u>1.729</u>	2.278	1.024	1.572	<u>2.119</u>
	Moirai-Large$^{(\dagger x)}$	0.900	2.004	3.053	<u>0.984</u>	1.539	2.334
	Chronos-Large$^{(0)}$	0.709	**1.715**	<u>2.272</u>	0.998	1.601	2.250
	Sundial-Base$^{(0)}$	0.733	2.108	2.536	**0.922**	<u>1.452</u>	2.202

Table 1: **M5 Competition.** RMSSE results on M5 at a state and store level for different data aggregations. Lower values are better. We have brand level data (B) on the left and SKU level data (S) on the right for the following frequencies: Daily (D), Weekly (W), and Monthly (M). The notation $^{(0x)}$ denotes zero-shot forecasters that leverage exogenous information; $^{(\dagger x)}$ denotes forecasters that leverage exogenous information but were exposed to the data during training; and $^{(0)}$ denotes zero-shot univariate forecasters that do not use exogenous information. Best results for each dataset-level are **bold**, and second best are <u>underlined</u>.

exhibits fundamental limitations including inability to learn ordered patterns, dependency on manual frequency features, weak trend extrapolation, lack of recency bias, and poorly calibrated confidence intervals. To address these shortcomings, we propose modifications at both the data generation and architectural levels: graph generation algorithms with time-dependent root node excitation through stochastic processes, and architectural enhancements including rotary positional encodings (RoPE), absolute positional encodings, and full attention mechanisms enabling future exogenous information to inform predictions.

Our analysis suggests that prior-data fitted networks (PFNs) represent a viable alternative to fine-tuning for time series foundational models, provided that temporal inductive biases are encoded directly into both architecture and synthetic data priors. These modifications advance PFNs beyond i.i.d. tabular data toward genuinely temporal inference models capable of leveraging exogenous covariates in zero-shot settings. While preliminary, these findings underscore the promising potential of this research direction and highlight the necessity of further exploration.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the Language of Time Series. *arXiv:2403.07815*, 2024.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. 2025. URL https://arxiv.org/abs/2510.15821.

Sebastian Pineda Arango, Pedro Mercado, Shubham Kapoor, Abdul Fatir Ansari, Lorenzo Stella, Huibin Shen, Hugo Senetaire, Caner Turkmen, Oleksandr Shchur, Danielle C. Maddix, Michael Bohlke-Schneider, Yuyang Wang, and Syama Sundar Rangapuram. ChronosX: Adapting Pre-

trained Time Series Models with Exogenous Variables. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv 2310.10688*, 2023.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. *International Conference on Learning Representations (ICLR)*, 2023.

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

Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models . *arXiv:2501.02945*, 2025.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. *International Conference on Learning Representations (ICLR)*, 2024.

P. L. Krapivsky and S. Redner. The Magic of Networks Grown by Redirection. *arXiv 2305.10628*, 2023.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A Family of Highly Capable Time Series Foundation Models. *arXiv 2502.00816*, 2025.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 2022.

Andres Potapczynski, Kin G. Olivares, Malcolm Wolff, Andrew Gordon Wilson, Dmitry Efimov, and Vincent Quenneville-Belair. Effectively Leveraging Exogenous Information across Neural Forecasters. *NeurIPS TSALM 2024*, 2024.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. *arXiv:2310.08278*, 2023.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts. *International Conference on Learning Representations (ICLR)*, 2025.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv 2104.09864*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified Training of Universal Time Series Forecasting Transformers. *International Conference on Machine Learning (ICML)*, 2024.

A TABPFN ARCHITECTURE

Here, we provide additional details about the TabPFN architecture. Assume that we have the following N_{train} observations for our target $(y_i)_{i=1}^{N_{\text{train}}}$, and $N = N_{\text{train}} + N_{\text{test}}$ observations for covariate information $(\boldsymbol{x}_i)_{i=1}^{N}$, where each $\boldsymbol{x}_i \in \mathbb{R}^{F'}$, and that we want to make N_{test} predictions for the target $(y_i)_{i=1}^{N_{\text{test}}}$. The goal of the preprocessing step is to transform the information of $(\boldsymbol{x}_i)_{i=1}^{N}$ and $(y_i)_{i=1}^{N_{\text{train}}}$ into an embedding $\boldsymbol{Z} \in \mathbb{R}^{N\times F\times D}$. In terms of the target, we first create a tensor $\tilde{\boldsymbol{Y}} \in \mathbb{R}^{N\times 2}$ by first z-scoring all the train targets, $\tilde{Y}_{i,1} = (y_i - \mu_{\text{train}})/\sigma_{\text{train}}$, where $\mu_{\text{train}} = \frac{1}{N_{\text{train}}}\sum_{i=1}^{N_{\text{train}}} y_i$ and $\sigma^2_{\text{train}} = \frac{1}{N_{\text{train}}-1}\sum_{i=1}^{N_{\text{train}}}(y_i-\mu_{\text{train}})^2$, for the positions of $i = 1,\dots,N_{\text{train}}$, and then by setting the rest of the N_{test} positions $i = N_{\text{train}}+1,\dots,N$ as $\tilde{Y}_{i,1} = \mu_{\text{train}}$. Then, the other columns of $\tilde{\boldsymbol{Y}}$ are filled with $\tilde{Y}_{i,2} = 0$ if the entry is observed ($i = 1,\dots,N_{\text{train}}$) and $\tilde{Y}_{i,2} = -2$ if not ($i = N_{\text{train}}+1,\dots,N$). After that, we create $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$ by embedding $\tilde{\boldsymbol{Y}}$ with a linear layer on a D dimensional space as $\boldsymbol{Y} = \tilde{\boldsymbol{Y}}\boldsymbol{W}_{\boldsymbol{Y}}$ where $\boldsymbol{W}_{\boldsymbol{Y}} \in \mathbb{R}^{2\times D}$.

An analogous procedure is done for each of the features in $\boldsymbol{x}_i \in \mathbb{R}^{F'}$ after first grouping them into pairs, as discussed in Hollmann et al. (2025). The grouping can done easily with a reshape, as follows. If we have $\tilde{\boldsymbol{X}}'_i = \boldsymbol{x}_i$, then $\tilde{\boldsymbol{X}} = \text{Reshape}(\tilde{\boldsymbol{X}}', (N, F'/2, 2))$ would have the desired effect (assuming that F' is divisible by 2, else we 0-pad the feature dimension). After z-scoring each of the $f = 1,\dots,F'/2$ features, we then compute $\boldsymbol{X} = \tilde{\boldsymbol{X}}\boldsymbol{W}_{\boldsymbol{X}} \in \mathbb{R}^{N\times F-1\times D}$, where $\boldsymbol{W}_{\boldsymbol{X}} \in \mathbb{R}^{2\times D}$ and $F = F'/2 + 1$. After the embedding $\boldsymbol{X}$ is constructed, we then add a fixed random positional encoding $\boldsymbol{\Omega} \in \mathbb{R}^{F-1\times D}$ to each feature shared across all N samples. In other words, we do $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \boldsymbol{\Omega}$ for all $i = 1,\dots,N$. Finally, we set $\boldsymbol{Z} = [\boldsymbol{X}, \boldsymbol{Y}] \in \mathbb{R}^{N\times F\times D}$, which is the embedding passed to the architecture seen in Figure 2.

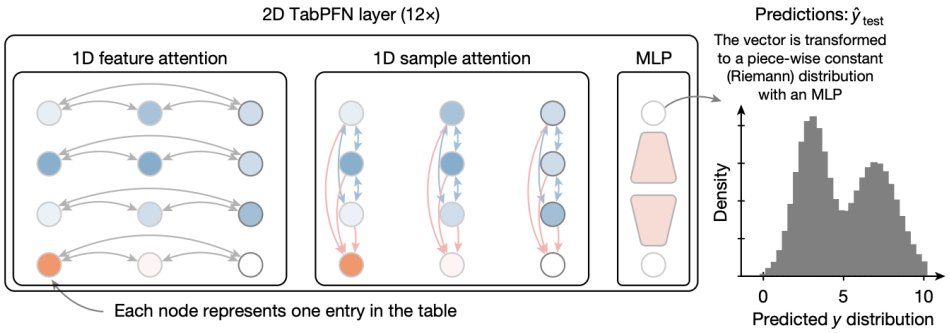


Figure 2: **How TabPFN combines attention across features and samples.** Taken from Hollmann et al. (2025), the figure illustrates the main components of the `TabPFN` architecture, plus the translation of the embedding into a Riemann distribution approximation of the PPD $p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$.

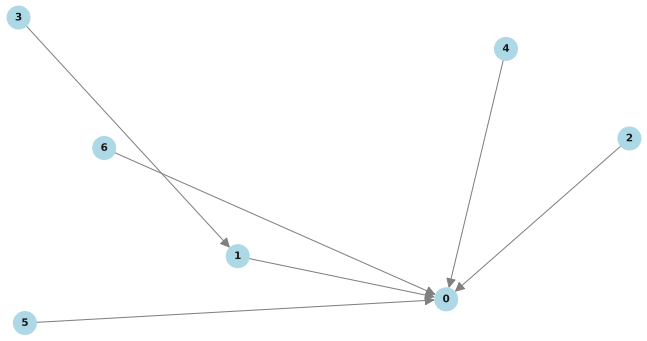
The transformation of $\boldsymbol{Z} \in \mathbb{R}^{N\times F\times D}$ into the Riemman approximation of the PPD is done with another linear layer as $Z_{N_{\text{train}}:,-1,:}\boldsymbol{W}_{\boldsymbol{Z}} \in \mathbb{R}^{N_{\text{test}}\times Q}$, where $\boldsymbol{W}_{\boldsymbol{Z}} \in \mathbb{R}^{D\times Q}$ and Q is the number of quantiles needed to compute the PPD.

B DATA GENERATION

As explained in Section 3.1, we need to randomly generate DAGs to define diverse SCMs for our synthetic data procedure. The initial procedure to construct a graph (Hollmann et al., 2023) was through a MLP, where each node is connected to all other nodes in the next layer, and the depth of the MLP is the depth of the graph which culminates with 1 node at the end which would be the target. To illustrate, if we have a 3-layered MLP with a width of 10, then we would have a graph with $21 = 10 + 10 + 1$ nodes and $110 = 10 \times 10 + 10 \times 1$ edges (assuming that the MLP is fully connected). A step to reduce the density of the graph is to drop some edges uniformly at random or by blocks (Hollmann et al., 2023).

In Hollmann et al. (2025), the authors adopted a "more realistic" DAG generation by using a classical algorithm in the study of random networks called the random growing network with redirection (Krapivsky & Redner, 2023).

As illustrated in Figure 3 (Top), a main characteristic is that it generates graphs with many root nodes (as each added root node in might never get an incoming edge) and, if the redirection probability ρ is high, then several of the root nodes might point to the first node. When selecting which features to use from a graph, the root nodes are always excluded (Hollmann et al., 2025), and so having a graph that has many root nodes is not necessarily optimal. Furthermore, if the graph happens to concentrate in a few nodes, then many of the features would not be related (that is, there would not be a path that connects them), making many of the features in the dataset not informative about the target.



(a) Growing random network with redirection and preferential attachment (Krapivsky & Redner, 2023).

Figure 3: Example graph from graph generation algorithms.