Time-Aware Prior Fitted Networks for Zero-Shot Forecasting with Exogenous Variables

Anonymous authors
Paper under double-blind review

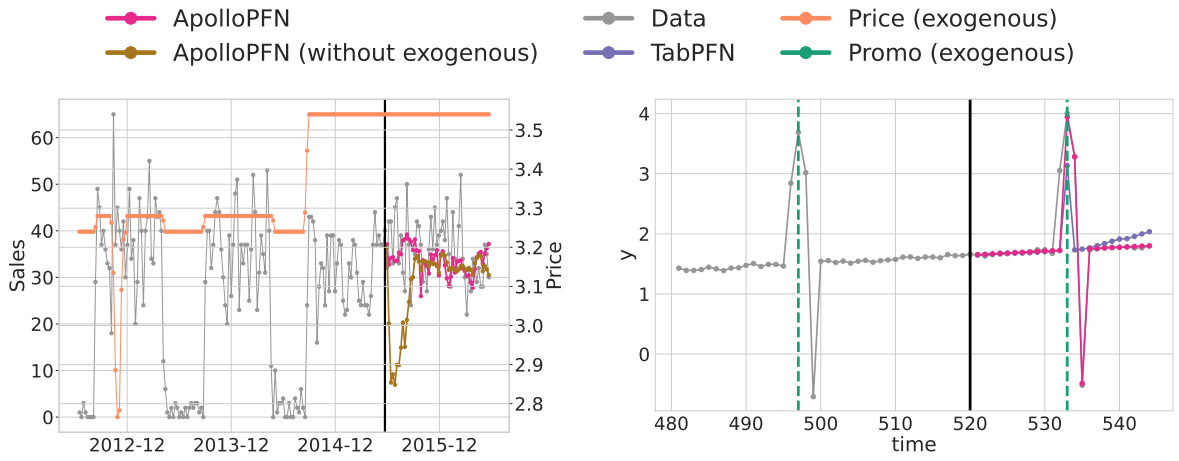
Abstract

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 and phase changes in the target series. Most current time series foundation models (e.g., `Chronos`, `Sundial`, `TimesFM`, `TimeMoE`, `TimeLLM`, and `LagLlama`) ignore exogenous covariates and make forecasts solely from the numerical time series history, thereby limiting their performance. In this paper, we develop `ApolloPFN`, a prior-data fitted network (PFN) that is time-aware (unlike prior PFNs) and that natively incorporates exogenous covariates (unlike prior univariate forecasters). Our design introduces two major advances: (i) a synthetic data generation procedure tailored to resolve the failure modes that arise when tabular (non-temporal) PFNs are applied to time series; and (ii) time-aware architectural modifications that embed inductive biases needed to exploit the time series context. We demonstrate that `ApolloPFN` achieves state-of-the-art results across most benchmarks, such as M5, electric price forecasting, UCI Air Quality and Solar Energy datasets, that contain exogenous information.

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. This is illustrated in Figure 1. 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. Consequently, zero-shot capability is an important requirement for a TSFM serving as a substrate for downstream systems and products (Bommasani et al., 2021). Therefore, a practical modern TSFM should natively incorporate accompanying exogenous covariates whenever they are available.

There are a few foundation-like models that support exogenous covariates in a zero-shot forecasting setting, most notably `TabPFN-TS` (Hoo et al., 2025) and `Moirai` (Woo et al., 2024). These methods represent two related but distinct directions for foundation models in time series forecasting. `Moirai` follows the paradigm of large-scale pretrained forecasting models trained on diverse time series corpora, whereas `TabPFN-TS` builds on the prior-data fitted network (PFN) framework, where predictions are generated by conditioning directly on the observed context of the test input. Because `Moirai` is trained on many public benchmarks, evaluating fully non-overlapping train/test exposure can be challenging in practice, especially for widely used benchmark



(a) M5 weekly sales

(b) time series with promotional activity

Figure 1: (a) **Not using exogenous information can lead to catastrophic forecasting errors.** We compare the predictions of `ApolloPFN` with and without exogenous information for the weekly sales of a real product from the M5 benchmark. Ignoring the price information leads the forecaster to predict a decreased demand via context parroting (brown), whereas the tracking of price dynamics helps the model focus on most up-to-date dynamics (pink). (b) **Prior-data fitted networks such as `TabPFN-TS` fail to capture ordered patterns.** We compare the prediction of `TabPFN-TS` and `ApolloPFN` for a synthetic time series that has a recurrent pattern of a ramp-up period before a promotion, a spike on the promotion, a ramp-down period, and then a subsequent decrease in demand. The exogenous promotion information is encoded as a binary indicator. Training data is to the left of the black line, and forecasts are to the right.

suites such as `GiftEval` (Aksu et al., 2024). Nevertheless, both paradigms have demonstrated strong zero-shot forecasting capability under different settings.

However, `TabPFN-TS` is not a time series architecture *per se*; instead, it augments a tabular PFN with a small set of time series features. As a result, it inherits inductive biases that are well suited to i.i.d. tabular prediction but are not always aligned with the sequential structure of time series data (Yu et al., 2025b;a). In particular, its architecture exhibits substantial invariance to observation order, which can limit its ability to fully capture temporal structure and recency effects. These limitations can manifest in weaker handling of order-dependent dynamics, reduced robustness across unseen frequencies, less reliable trend extrapolation, and weaker uncertainty calibration in some forecasting settings. Our work is motivated by the goal of incorporating stronger temporal inductive biases into the PFN framework while preserving its zero-shot and covariate-conditioning strengths.

In this paper, we show how to leverage exogenous variables effectively for zero-shot forecasting. Our main contributions are the following:

- We provide a detailed characterization of the shortcomings of existing PFNs such as `TabPFN-TS` *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 autocorrelations, making it challenging to predict ordered patterns accurately. This is illustrated in Figure 1 (b). Based on these findings, we argue that existing PFNs are *not* suitable as *time series* FMs (Section 2.)

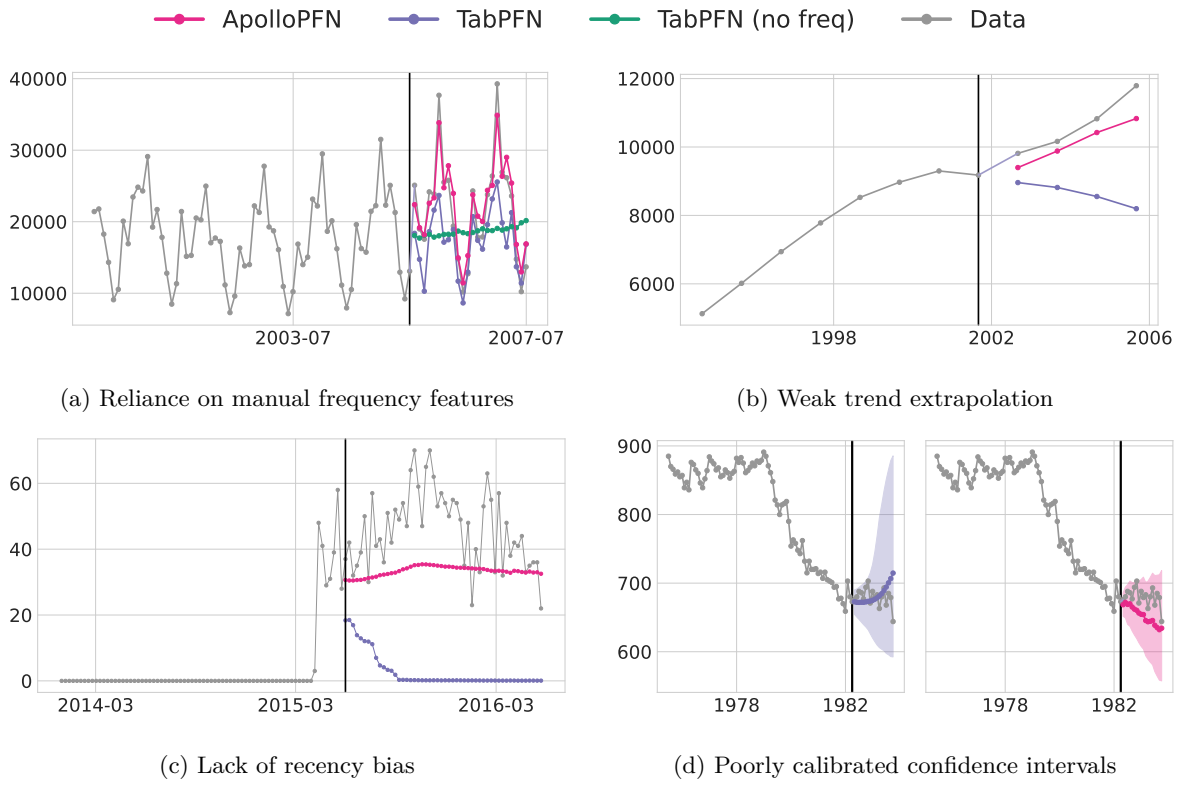
- We introduce `ApolloPFN`, a model that circumvents the limitations of `TabPFN-TS` through a novel data generation procedure and novel architectural choices (Section 3). This improvement consists of two complementary components. First, we introduce a synthetic data generation procedure for time series that relies on a new graph generation algorithm (which accelerates learning, as seen in Figure 4) coupled with time-dependent root nodes (see Section 3.2). Second, we incorporate inductive biases into our architecture that reflect the importance of order in time series data (Section 3.3).

- We extensively compare `ApolloPFN` against state-of-the-art (SOTA) baselines, including `TabPFN-TS` and `Moirai`, in several datasets spanning more than 90K time series *that have accompanying exogenous co-*

variates, as well as several benchmarks that do not have exogenous covariates, demonstrating the broad effectiveness of `ApolloPFN` (Section 4). We present several ablations on real and synthetic data to bolster our architectural and data generation choices (Section 4.4).

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 and Figure 2.



(a) Reliance on manual frequency features

(b) Weak trend extrapolation

(c) Lack of recency bias

(d) Poorly calibrated confidence intervals

Figure 2: **Failure modes of `TabPFN-TS` for time series data that `ApolloPFN` 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.

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) and energy consumption (where usage steadily builds up toward peak hours and then declines overnight). These types of patterns are not purely cyclical, but instead they reflect structured temporal dependencies that unfold over multiple horizons. An example of such a pattern is shown in Figure 1(b), which shows that `TabPFN-TS` cannot capture in-context a sequence of events, as it lacks appropriate temporal inductive biases. Instead, the model resolves to outputting a smaller spike in the promotional event.

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 2(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 2(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 2(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 2(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.

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.

3 `ApolloPFN`

`ApolloPFN` is a prior-data fitted network (PFN) designed for time series forecasting with exogenous covariates. Given a time series with historical observations $\mathcal{D}_{\text{train}} = (\boldsymbol{x}_t, y_t)_{t=1}^{T}$, where $y_t \in \mathbb{R}$ is the target and $\boldsymbol{x}_t \in \mathbb{R}^F$ are F-dimensional exogenous covariates observed at each time step. When future exogenous information are available, they are represented as $(\boldsymbol{x}_t)_{t=T+1}^{T+H}$ for horizon H. `ApolloPFN` produces probabilistic forecasts for a horizon H:

$$(y_{T+1}, \ldots, y_{T+H}) = f_\theta(y_1, \ldots, y_T,\ \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T+H}),$$

where both the context length T and the covariate dimensionality F may vary across instances. Crucially, `ApolloPFN` can anticipate covariate-driven events such as promotions, price changes, or weather shifts by conditioning on future exogenous information whereas most of the neural forecasters in the literature ignore them.

Unlike existing PFN-based forecasters that retrofit tabular models with hand-crafted time features, `ApolloPFN` integrates temporal inductive biases into both its training data distribution and its architecture. Below, we describe the three pillars of `ApolloPFN`: (i) its probabilistic inference formulation, (ii) its time-series-aware synthetic data generation, and (iii) its architectural design.

3.1 Prior-Data Fitted Inference for Time Series

`ApolloPFN` builds on the PFN paradigm introduced by Müller et al. (2022; 2025), which provides a mechanism for amortized Bayesian inference. The core idea is to train a neural network q_θ on synthetically generated datasets so that, at inference time, it directly approximates the posterior predictive distribution (PPD) without requiring explicit posterior computation or likelihood specification.

Concretely, we define a generative process that samples time series datasets $\mathcal{D} = (\boldsymbol{x}_t, y_t)_{t=1}^{T+H}$ by first sampling a latent structure $\xi \sim p(\xi)$ (in our case, a structured causal model over a directed acyclic graph) and then generating observations $(\boldsymbol{x}_t, y_t) \sim p(\boldsymbol{x}, y \mid \xi)$ with temporal dependencies. The network q_θ is trained to minimize the loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{p(\boldsymbol{x},y)} \log q_\theta(y_{\text{test}} \mid \boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$$
$$\text{i.e., } \mathcal{L}(\theta) = -\mathbb{E}_{p(\boldsymbol{x},y)} \log q_\theta\big(y_{T+1:T+H} \;\big|\; (\boldsymbol{x}_t)_{t=T+1}^{T+H}, (\boldsymbol{x}_t, y_t)_{t=1}^{T}\big)$$

so that q_θ directly approximates the PPD $p(y_{T+1:T+H} \mid (\boldsymbol{x}_t)_{t=T+1}^{T+H}, (\boldsymbol{x}_t, y_t)_{t=1}^{T})$ (Müller et al., 2022). This circumvents the need to approximate a high-dimensional posterior $p(\xi \mid \mathcal{D}_{\text{train}})$ or to define a closed-form likelihood $p(y \mid \boldsymbol{x}, \xi)$, which is how the PPD is usually computed: $p(y_{\text{test}} \mid \boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}}) = \int p(y_{\text{test}} \mid \boldsymbol{x}_{\text{test}}, \xi)\, p(\xi \mid \mathcal{D}_{\text{train}})\, d\xi$ (Murphy, 2012; Hoffman & Gelman, 2014; Wilson & Izmailov, 2020).

The key advantage of this formulation for time series is that the network learns to perform in-context inference: given a new time series at inference time, it conditions on the observed history and covariates to produce calibrated probabilistic forecasts in a single forward pass, without any gradient updates or fine-tuning.

3.2 Synthetic Data Generation

Algorithm 1 Single Root Node Random Growing Network (SRNGN)

Require: V: total number of nodes, ρ additional attachment probability
Initialize graph G with nodes $n = 0$, $n = 1$ and edge $(1, 0)$
Initialize in-degree $k_j = 0$ for all $j \neq 0$, $k_0 = 1$
for $n = 2, \ldots, V-1$ **do**
  Compute attachment probabilities for all nodes $i < n$
  $\Pi_i = \frac{k_i+1}{\sum_{j=0}^{n-1}(k_j+1)}$
  Select target node t with probability Π_t
  Select an additional source node uniformly at random from $s \in \{0, \ldots, n-1\} \setminus \{t\}$
  Add new node n, source node connects to new node, add edge (s, n)
  Update: $k_n \leftarrow 1$
  Sample $u \sim U(0, 1)$
  if $u < \rho$ **then**
    Target connects to new node, add edge (t, n)
    Update: $k_n \leftarrow k_n + 1$
  end if
end for
Eliminate cycles in G (if any)
return DAG $G = (V, E)$

To train `ApolloPFN`, we generate synthetic time series datasets via structured causal models (SCMs) defined over directed acyclic graphs (DAGs). Each SCM specifies a generative process: given a DAG $\mathcal{G}$ with nodes $\{V_j\}$, the value at each node is determined by its parents as $V_j = f_j(V_{\text{PA}(j)}) + \epsilon_j$, where f_j is a randomly sampled function (MLP with activation functions) and ϵ_j is measurement noise.

To generate a time series dataset $\mathcal{D}_{\mathcal{G}} = (\boldsymbol{x}_t, y_t)_{t=1}^{N}$ with $\boldsymbol{x}_t \in \mathbb{R}^F$, we first sample the total number of time steps $N \sim p(N)$ where $N = T + H$. Here, T and H represent the number of historical and future time steps respectively. Next, we sample the number of covariates $F \sim p(F)$. We then sample a DAG $\mathcal{G} \sim p(\mathcal{G})$ using SRNGN method (Algorithm 1) and define the SCM over it. Once we have the graph of size $|\mathcal{G}|$, we randomly select $F + 1 \leq |\mathcal{G}|$ non-root nodes and assign them as covariates $x_{t,j} = v_{t,\pi(j)}$ and target $y_t = v_{t,\pi(F+1)}$, where $\pi(\cdot)$ represents the random selection of covariates and target variable.

There are two key modifications that we make in our data generation procedure that distinguish `ApolloPFN` from prior tabular PFN approaches which are explained in detail below.

Single Root Node Growing Network (SRNGN). Prior work (Hollmann et al., 2025) generates DAGs using Random Growing Networks (RGN) with preferential attachment (Krapivsky & Redner, 2023) (see Algorithm 2 in Appendix B), which produce graphs with multiple root nodes and short path lengths. This is suboptimal for time series generation because many features end up disconnected (no path connects them), making them uninformative about the target.

We introduce SRNGN, a graph generation algorithm that produces graphs with a single root node and longer causal paths. SRNGN reverses the mechanics of RGN: we always incorporate new nodes by having at least one prior node connect to them, and additionally connect them to a popular node with probability ρ. The SRNGN construction is summarized in Algorithm 1; Figure 7 (Bottom) in Appendix B.1 illustrates the resulting graph structure. In contrast, graphs sampled via RGN (Algorithm 2 in Appendix B.1) are characterized by multiple root nodes and short path lengths (Figure 7 (Top)). Our empirical results show that SRNGN accelerates training convergence (see Figure 4).

Time Series Root Node Excitation. The critical distinction from tabular PFNs is how we generate observations indexed by time. Rather than sampling root node values independently for each observation (as in the i.i.d. tabular setting), we sample root node values $(v_{t,r})_{t=1}^{T}$ through a stochastic process that introduces temporal correlation. Specifically, we define root node values as a combination of sine and cosine functions with randomly sampled frequencies $(\phi_1^{(r)}, \phi_2^{(r)})$ and amplitudes $(\alpha_1^{(r)}, \alpha_2^{(r)})$: $v_{t,r} = \alpha_1^{(r)} \sin(\phi_1^{(r)} t) + \alpha_2^{(r)} \cos(\phi_2^{(r)} t)$, for all $t = 1, \ldots, T$. These temporally correlated root signals then propagate through the SCM graph in topological order, producing target values y_t and covariates $\boldsymbol{x}_t$ that exhibit realistic temporal dependencies, including autocorrelations, quasi-periodic patterns, and covariate-driven regime changes.

We z-score the target series $(y_t)_{t=1}^{T}$ before passing it to the model, and invert the z-scoring when outputting the predictions $(y_t)_{t=T+1}^{T+H}$.

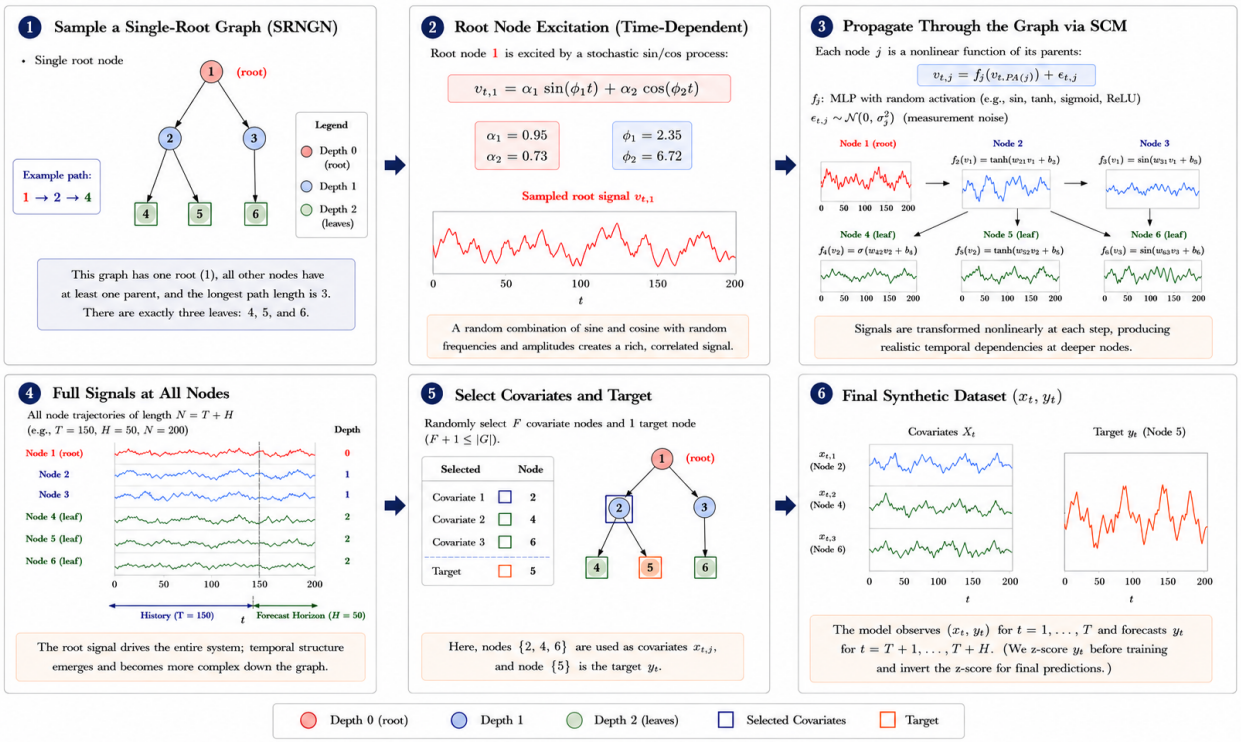


Figure 3: **Overall synthetic data generation workflow of ApolloPFN.** A stochastic root signal is propagated through a nonlinear structural causal graph to generate multivariate node trajectories, from which covariates and forecasting targets are selected to construct the final dataset (X_t, y_t)

3.3 Temporal-Feature Factorized Transformer Architecture

ApolloPFN employs a temporal-feature factorized transformer that processes an embedding tensor $\boldsymbol{Z} \in \mathbb{R}^{N \times F \times D}$, where $N = T + H$ is the total number of observations across both history and future horizon time steps, F is the number of features (covariates plus target), and D is the embedding dimension. Each layer $\ell = 1, \ldots, L$ applies three operations:

$$\begin{aligned} \boldsymbol{Z} &\leftarrow \mathtt{LN}_1^{(\ell)}(\boldsymbol{Z} + \mathtt{AttnFeat}^{(\ell)}(\boldsymbol{Z})) \\ \boldsymbol{Z} &\leftarrow \mathtt{LN}_2^{(\ell)}(\boldsymbol{Z} + \mathtt{AttnSamp}^{(\ell)}(\boldsymbol{Z})) \\ \boldsymbol{Z} &\leftarrow \mathtt{LN}_3^{(\ell)}(\boldsymbol{Z} + \mathtt{MLP}^{(\ell)}(\boldsymbol{Z})), \end{aligned} \tag{1}$$

where $\mathtt{LN}(\cdot)$ denotes layer normalization (Ba et al., 2016) and $\mathtt{MLP}(\cdot)$ is applied along the embedding dimension. The architecture separates feature interactions from temporal interactions via two distinct attention mechanisms: AttnFeat attends across the F feature axis (learning cross-covariate relationships), while AttnSamp attends across the temporal axis of N time steps (learning temporal dependencies). This separability allows ApolloPFN to handle a variable number of time steps and a variable number of covariates. Appendix A provides additional details on the input embedding procedure and how it relates to Tabular PFNs such as TabPFN.

There are two key architectural modifications that distinguish ApolloPFN from Tabular PFNs and enable it to capture temporal structure.

Temporal Positional Encodings. Tabular PFNs such as TabPFN (Hollmann et al., 2025) use no positional encodings in AttnSamp, making the mechanism permutation invariant across observations. This is appropriate for i.i.d. data but fundamentally limits the model's ability to capture temporal order. ApolloPFN introduces two complementary positional encoding schemes.

First, we incorporate RoPE embeddings (Su et al., 2023) into $\mathtt{AttnSamp}^{(\ell)}(\cdot)$, which encode *relative* temporal distances. RoPE modifies the key-query interactions so that $\boldsymbol{q}_{t+h}^{\intercal} \boldsymbol{R}_h \boldsymbol{k}_t \to 0$ as $h \to \infty$, where $\boldsymbol{R}_h$ is a rotation matrix parameterized by the temporal lag h. This biases the model toward prioritizing temporally proximal observations, naturally inducing a recency effect.

Second, to incorporate an *absolute* notion of temporal position, we define sinusoidal positional encodings $\boldsymbol{\Omega} \in \mathbb{R}^{N \times D}$:

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

which we add to the embedding as $\boldsymbol{Z}_f \leftarrow \boldsymbol{Z}_f + \boldsymbol{\Omega}$ for all $f = 1, \ldots, F$ (see Equation 1). The combination of relative (RoPE) and absolute positional encodings enables ApolloPFN to learn both local temporal dynamics and global position-dependent patterns.

Full Attention for Horizon-Aware Forecasting. In tabular PFNs, test observations do not attend to each other in AttnSamp; each prediction is made independently as $p(y_j \mid \boldsymbol{x}_j, (\boldsymbol{x}_i, y_i)_{i=1}^N)$. This is appropriate for i.i.d. prediction but inappropriate for time series forecasting, where future exogenous covariates carry information relevant to all horizon steps.

In ApolloPFN, we allow all time steps (both history and horizon) to attend to each other in $\mathtt{AttnSamp}^{(\ell)}(\cdot)$. This means that when forecasting, the model computes $p(y_{T+h} \mid (\boldsymbol{x}_t)_{t=T+1}^{T+H}, (\boldsymbol{x}_t, y_t)_{t=1}^{T})$ for all $h = 1, \ldots, H$, where each horizon prediction is informed by the full set of future exogenous covariates. This is essential for scenarios where, e.g., a promotion scheduled at $t = T + 3$ should influence the demand forecast at $t = T + 1$ (such as anticipation or ramp up effects) or where multiple correlated events in the horizon interact.

4 Empirical evaluation

In this section, we comprehensively compare ApolloPFN in several forecasting scenarios and against different forecasting models. Tables 1, 2 and 3 summarize our main results. Overall, ApolloPFN performs extremely

well on challenging time series benchmarks *that have exogenous information*. Furthermore, `ApolloPFN` also has strong zero-short performance on classical benchmarks which do not contain exogenous information (Table 4), even against much larger models like `Moirai-Large` and `Chronos-Large`. These models have $30 - 70\times$ more parameters than `ApolloPFN`, which only has 11M parameters. Since `ApolloPFN` employs standard quadratic attention in its transformer architecture, the maximum context length is limited by computational and memory constraints. Accordingly, we fix the maximum context length to 512 time steps across all benchmarks in our experiments.

In the setting with exogenous variables, we evaluate `ApolloPFN` on the M5 competition (Makridakis et al., 2022) benchmarks, the electricity price forecasting (Lago et al., 2021) dataset and additional domain-specific benchmarks such as UCI Air Quality De Vito et al. (2008) dataset and Solar Energy Emami (2025) dataset which are described below.

The electricity price forecasting dataset consists of hourly measurements of electric prices (Lago et al., 2021) for five major markets in Europe, namely Nord Pool (NP), PJM (COMED zone), France (FR), Belgium (BE), and Germany (DE). These datasets contain exogenous variables such as system load and power generation measurements. We provide a detailed description of the time spans and exogenous features for each market in Appendix C.

M5 competition dataset (Makridakis et al., 2022) 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).

Domain-specific datasets. We further evaluate on two additional datasets from distinct domains: the UCI Air Quality dataset De Vito et al. (2008), which provides daily CO concentration measurements, and the Solar Energy dataset Emami (2025), which includes hourly energy generation with weather-related exogenous variables. For the air quality dataset, we consider 37 non-overlapping 14-day forecasting horizons at a daily granularity. For the solar dataset, we evaluate 51 non-overlapping 24-hour forecasting horizons at an hourly granularity.

In the univariate setting, we evaluate `ApolloPFN` on M-series suite of benchmarks described below.

M-series suite of benchmarks constitutes a comprehensive evaluation suite for the univariate case (no exogenous variables). The evaluation is across varying prediction lengths, different frequencies (hourly, daily, weekly, quarterly, yearly), and distinct sources of data, resulting in widely different time series behaviors. It is worth mentioning that these M-series competitions, M1 (Makridakis & Hibon, 1979), M2 (Makridakis et al., 1993), M3 (Makridakis & Hibon, 2000), and M4 (Makridakis et al., 2020), have been a consistent benchmark to evaluate forecasting models throughout the years.

We use the following set of well-established metrics to report accuracy.

sCRPS captures the accuracy of probabilistic predictions. This is a scale-free metric defined on quantiles $\alpha_1 < \cdots < \alpha_Q$, $\alpha_j \in (0, 1)$:

$$\mathrm{sCRPS}(y, \hat{y}) = \frac{\sum_{t=T+1}^{T+H} \frac{2}{Q} \sum_{j=1}^{Q} \alpha_j \left(y_t - \hat{y}_t^{\alpha_j}\right)_+ + (1 - \alpha_j) \left(y_t - \hat{y}_t^{\alpha_j}\right)_-}{\sum_{t=T+1}^{T+H} |y_t|},$$

where $(\cdot)_+$ is the positive part and $(\cdot)_-$ the negative part functions. Additionally, $\hat{y}_t^{\alpha_j}$ represents the α_j-th quantile prediction for time step t.

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

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

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).

4.1 Training Setup

Data Generation. The sampling procedure for our SCMs is the following. We selected the number of nodes uniformly from a minimum of 20 to a maximum of 150. Each node then contains a state of dimensionality 6 which we propagate through the graph. Moreover, when using a MLP edge we select our activation from the following options: tanh, sine, abs, identity, log, sigmoid, smooth relu, modulo and step wise (or indicator). The entries weights of the layers in the MLPs are sampled from $\mathcal{N}(0,1)$. The sample frequencies ϕ are sampled from $\log \phi \sim \mathcal{U}(1,10)$ and the amplitudes $\alpha \sim \mathcal{N}(0,1)$.

Training. 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.

4.2 Zero-shot performance with exogenous variables

Tables 1, 2, and 3 compare `ApolloPFN` against foundation models that leverage exogenous information, such as `TabPFN-TS` and `Moirai-Large`, as well as univariate foundation models, including `Chronos-Large` and `Sundial-Base`, across electricity forecasting, M5 aggregations, and additional domain-specific benchmarks. On the electricity and domain-specific benchmarks, `ApolloPFN` achieves state-of-the-art performance on most datasets, with an average improvement of approximately 5.6% in sCRPS on the domain-specific tasks, demonstrating strong generalization across diverse domains.

On the M5 aggregation benchmark, `ApolloPFN` attains state-of-the-art results at most aggregation levels, yielding an average improvement of 5.7% in RMSSE over the next best model (`TabPFN-TS`), while remaining competitive with substantially larger time-series foundation models.

Level	Model	M5(D-B)	M5(W-B)	M5(M-B)	M5(D-S)	M5(W-S)	M5(M-S)
State	AutoARIMAX	0.900	1.374	<u>2.218</u>	<u>0.840</u>	1.555	3.006
	Prophet	<u>0.525</u>	1.685	3.223	0.859	1.945	4.593
	Sundial-Base$^{(0)}$	0.572	2.052	2.398	**0.825**	1.614	3.042
	Chronos-Large$^{(0)}$	0.565	**1.223**	2.572	0.894	1.853	2.989
	`Moirai-Large`$^{(\dagger x)}$	0.794	1.533	2.336	0.879	1.681	<u>2.780</u>
	TabPFN-TS$^{(0x)}$	0.527	<u>1.251</u>	2.599	0.883	<u>1.522</u>	2.864
	`ApolloPFN` $^{(0x)}$(ours)	**0.500**	1.275	**2.088**	0.875	**1.497**	**2.721**
Store	AutoARIMAX	<u>0.845</u>	1.775	<u>2.117</u>	**0.811**	<u>1.452</u>	2.463
	Prophet	1.056	2.070	3.205	0.822	1.738	3.507
	Sundial-Base$^{(0)}$	1.094	2.124	2.542	<u>0.815</u>	1.453	2.422
	Chronos-Large$^{(0)}$	0.887	1.704	2.259	0.880	1.628	2.476
	`Moirai-Large`$^{(\dagger x)}$	1.046	1.734	2.241	0.867	1.544	<u>2.258</u>
	TabPFN-TS$^{(0x)}$	0.872	<u>1.667</u>	2.247	0.898	1.457	2.355
	`ApolloPFN` $^{(0x)}$(ours)	**0.834**	**1.663**	**2.052**	0.874	**1.429**	**2.245**

Table 1: **M5 Competition.** RMSSE results on M5 at state and store levels across different temporal aggregations. Lower is better. (D/W/M) denote Daily, Weekly, Monthly frequencies; (B/S) denote Category and SKU-level data. Best results are **bold**, second-best are <u>underlined</u>.

sCRPS	DE(24)	NP(24)	FR(24)	BE(24)	PJM(24)	DE(48)	NP(48)	FR(48)	BE(48)	PJM(48)
AutoARIMAX	0.229	0.081	0.196	0.218	0.219	0.268	0.092	0.243	0.259	0.316
Prophet	0.064	0.032	0.073	0.080	0.129	0.069	0.040	0.075	0.099	0.088
Sundial-Base$^{(0)}$	0.063	0.022	0.054	0.053	0.121	0.081	0.032	0.053	0.074	0.057
`Chronos-Large`$^{(0)}$	0.059	0.022	0.048	0.049	0.108	0.077	0.030	**0.047**	0.070	0.055
`Moirai-Large`$^{(\dagger x)}$	0.102	0.035	0.081	0.077	0.140	0.128	0.049	0.094	0.105	0.101
TabPFN-TS$^{(0x)}$	0.043	**0.018**	0.043	0.054	**0.086**	**0.048**	**0.023**	0.048	**0.069**	**0.049**
`ApolloPFN` $^{(0x)}$(ours)	**0.042**	0.019	**0.040**	**0.048**	0.091	0.050	0.026	0.056	0.070	0.050

Table 2: **Electricity Price Forecasting.** `ApolloPFN` achieves the best overall performance among models leveraging exogenous information. sCRPS results across datasets and prediction horizons (24, 48). Lower values are better. The notation $^{(0x)}$ denotes zero-shot forecasters with exogenous variables; $^{(\dagger x)}$ denotes models trained with exposure to the datasets; and $^{(0)}$ denotes zero-shot univariate models without exogenous inputs. Best results are **bold**, second-best are underlined.

sCRPS	Air Quality	Solar Weather
AutoARIMAX	0.160	0.712
Prophet	0.121	0.295
Sundial-Base$^{(0)}$	0.118	0.142
`Chronos-Large`$^{(0)}$	0.121	0.157
`Moirai-Large`$^{(\dagger x)}$	0.121	0.346
TabPFN-TS$^{(0x)}$	0.110	0.135
`ApolloPFN` $^{(0x)}$(ours)	**0.106**	**0.126**

Table 3: **Multivariate datasets from other domains.** `ApolloPFN` also beats other neural forecasters for series from other domains as well. sCRPS results for Air Quality dataset and Solar with Weather dataset. Lower values are better. 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 are **bold**, and second best are underlined.

4.3 Zero-shot performance on classical univariate benchmarks

Given the limited availability of large-scale publicly accessible time series datasets, most neural forecasting models in the literature use all or a substantial portion of the M-competition data for training. Consequently, this practice complicates a fair and unbiased comparison of zero-shot model performance on these benchmarks. For completeness, however, we compare `ApolloPFN` against several of the best performing univariate TSFMs. See Table 4 for a summary. Notably, `ApolloPFN` performs 10% better than `TabPFN-TS` on average, and it achieves SOTA across the different benchmarks.

sCRPS	M1(M)	M1(Y)	M3(M)	M3(O)	M4(D)	M4(M)	M4(Y)	Tou(M)	Tou(Y)
Sundial-Base	0.157	0.183	0.121	0.047	0.026	0.116	0.160	0.126	0.174
`Chronos-Large`	0.173	**0.119**	0.113	0.036	0.028	0.108	**0.106**	0.155	**0.103**
`Moirai-Large`	**0.135**	0.210	**0.093**	0.035	0.033	0.117	0.187	0.275	0.275
TabPFN-TS	0.169	0.123	0.106	0.035	0.027	0.096	0.115	0.203	0.146
`ApolloPFN` (ours)	0.152	0.142	0.094	**0.034**	**0.023**	**0.092**	0.113	**0.084**	0.137

Table 4: **`ApolloPFN` performance on the classical univariate benchmarks.** Lower values are better. Best results for each dataset are **bold**, and second best are underlined.

4.4 Ablation Studies

Data Generation. As seen in Figure 4, our use of SRNGN dramatically increases the speed at which the model starts to make accurate predictions. For Figure 4, we trained two different `ApolloPFN` models, one with RGN and one with SRNGN, leaving the rest of the hyperparameters fixed. We then evaluated the performance of the model checkpoints every 10K iterations on different benchmarks. We consistently see the model trained with SRNGN achieves a better performance faster than the model trained with RGN.

Figure 4: **SRNGN graph generation algorithm used by `ApolloPFN` accelerates learning.** We compare the test benchmark performance of our `ApolloPFN` model trained with the random growing network (RGN) algorithm and our Single Node Growing Network (SRNGN) algorithm at different training steps. With SRNGN, we achieve better performance at 20K iterations than at 80K with RGN.

Architecture Modifications. Our new time series synthetic data generation process coupled with the architectural changes presented in Section 3.3 enable `ApolloPFN` to resolve the failure modes of `TabPFN-TS` (see Figure 2 for concrete examples). In addition, we perform architectural ablation. We fix the control by training vanilla `TabPFN-TS` (Hollmann et al., 2025) with our time-dependent data (`ApolloPFN (-)`), then we add positional encodings (`ApolloPFN (RoPE)`), and, finally, we allow the attention mechanism to learn interactions between future exogenous samples (`ApolloPFN (RoPE+Full)`). Results appear in Figure 5, which presents the relative accuracy improvement of architectural variants with respect to `TabPFN-TS` baseline. Figure 5 shows a clear and consistent trend across test benchmarks: `ApolloPFN` achieves its strongest performance only after all proposed modifications are introduced. In particular, the most substantial improvement arises from incorporating positional encodings. `RoPE` is the primary driver of this effect, as it biases the model toward prioritizing temporally proximal observations when forming predictions. At the same time, for more complex behaviors, such as learning ordered patterns, the desired performance emerges only when all modifications are combined.

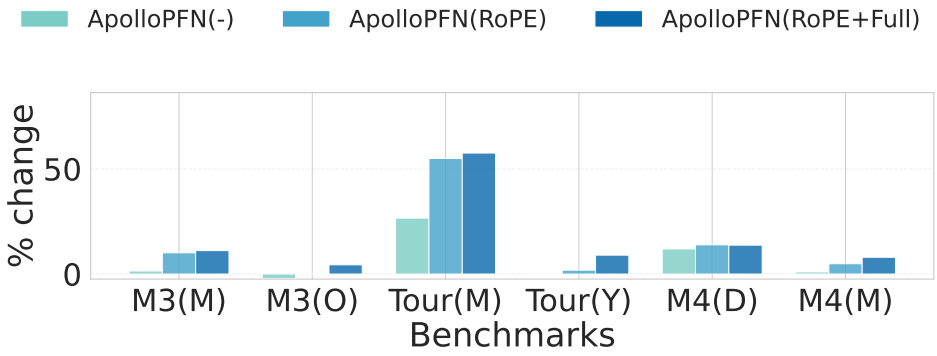


Figure 5: **Interventions in `ApolloPFN` to improve performance on time series data.** Ablation on the use of `RoPE` and full attention. We compare the effect of progressively adding `RoPE` and full attention in several benchmarks against the baseline of `TabPFN-TS`.

5 Conclusion

This work introduces `ApolloPFN`, a time-series specific PFN that extends the PFN paradigm to forecasting settings with temporal structure and exogenous covariates. By jointly rethinking the training distribution, architectural design, and positional bias, `ApolloPFN` moves PFNs beyond i.i.d. tabular data, toward a genuinely temporal inference model. Empirically, `ApolloPFN` achieves SOTA zero-shot performance across a diverse set of forecasting benchmarks, particularly in regimes where exogenous covariates drive sharp changes, discontinuities, or heterogeneous responses across series. More broadly, our results suggest that prior-data fitted inference is a viable and powerful alternative to fine-tuning for TSFMs. By encoding temporal inductive biases directly into both the architecture and the synthetic data prior, `ApolloPFN` demonstrates how zero-shot forecasting can be made practical, scalable, and deployment-friendly. We view this work as a step toward true TSFMs that are able to "reason" over time, structure, and context in a unified way, opening the door to richer forms of probabilistic forecasting and decision-making under uncertainty.

Despite its strong empirical performance, `ApolloPFN` has several limitations that are important to acknowledge. First, the model relies on standard quadratic attention, which constrains its applicability to very long time series. This can be limiting in high-frequency settings. Second, as an in-context learning model, `ApolloPFN` may struggle in regimes with very few observations, where there is insufficient context to infer series-specific temporal structure or exogenous effects. Finally, the model's capabilities are inherently constrained by the synthetic data distribution used during training. Because the PFN paradigm learns to approximate inference under a specific prior, behaviors or dependencies that are not represented in the training data cannot be reliably recovered at test time. We expect that leveraging insights from recent work (Yu et al., 2025b;a) may help to address this issue.

These considerations naturally point to several specific directions for future work. One important avenue is the development of theoretical understanding of how the scale, diversity, and structure of the synthetic training data influence model performance and generalization. Another promising direction is the introduction of hierarchical mechanisms that allow the model to pool information across related time series, while still retaining the ability to model series-specific responses. Finally, from a systems perspective, it would be valuable to study the extent to which `ApolloPFN` can be quantized or otherwise compressed, enabling training and deployment with reduced GPU memory requirements.

References

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation. *arXiv 2410.10393*, 2024.

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.

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv 1607.06450*, 2016.

Rishi Bommasani et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

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

Saverio De Vito, Ettore Massera, Marco Piga, Luca Martinotto, and Girolamo Di Francia. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical*, 129(2):750–757, 2008. doi: 10.1016/j.snb.2007.09.060.

Saman Emami. Renewable energy and weather conditions. Kaggle, 2025. URL `https://www.kaggle.com/datasets/samanemami/renewable-energy-and-weather-conditions`.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research 15 1593-1623*, 2014.

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.

Rob J Hyndman, Anne B Koehler, J Keith Ord, and Ralph D Snyder. Forecasting with Exponential Smoothing: the State Space Approach. *Springer*, 2008.

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.

Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafal Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy, Volume 293*, 2021.

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 and Michele Hibon. Accuracy of Forecasting: An Empirical Investigation. *Journal of the Royal Statistical Society*, 1979.

Spyros Makridakis and Michele Hibon. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 2000.

Spyros Makridakis, Chris Chatfield, Michele Hibon, Michael Lawrence, Terence Mills, Keith Ord, and LeRoy F. Simmons. The M2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 1993.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 2020.

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

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers Can Do Bayesian Inference. *International Conference on Learning Representations (ICLR)*, 2022.

Samuel Müller, Arik Reuter, Noah Hollmann, David Rügamer, and Frank Hutter. Position: The Future of Bayesian Prediction Is Prior-Fitted. *International Conference on Machine Learning (ICML)*, 2025.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

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.

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.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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.

Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Yuyang Wang. Understanding the implicit biases of design choices for time series foundation models. *arXiv preprint arXiv:2510.19236*, 2025a.

Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Yuyang Wang. Understanding transformers for time series: Rank structure, flow-of-ranks, and compressibility. *arXiv preprint arXiv:2510.03358*, 2025b.

A TabPFN Architecture

Here, we provide additional details that were not covered in Section 3.1. 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 $\boldsymbol{Z} \in \mathbb{R}^{N \times F \times D}$, as used in Equation 1. 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 $i = 1, \ldots, N_{\text{train}}$, and then by setting the rest of the N_{test} positions $i = N_{\text{train}} + 1, \ldots, N$ 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 $(i = 1, \ldots, N_{\text{train}})$ and $\tilde{Y}_{i,2} = -2$ if not $(i = N_{\text{train}} + 1, \ldots, N)$. 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 $\boldsymbol{Y} = \tilde{\boldsymbol{Y}} \boldsymbol{W_Y}$ where $\boldsymbol{W_Y} \in \mathbb{R}^{2 \times D}$.

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 $f = 1, \ldots, F'/2$ features, we then compute $\boldsymbol{X} = \tilde{\boldsymbol{X}} \boldsymbol{W_X} \in \mathbb{R}^{N \times F-1 \times D}$, where $\boldsymbol{W_X} \in \mathbb{R}^{2 \times D}$ 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 $i = 1, \ldots, N$. 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 6 and discussed Equation 1 in Section 3.1.

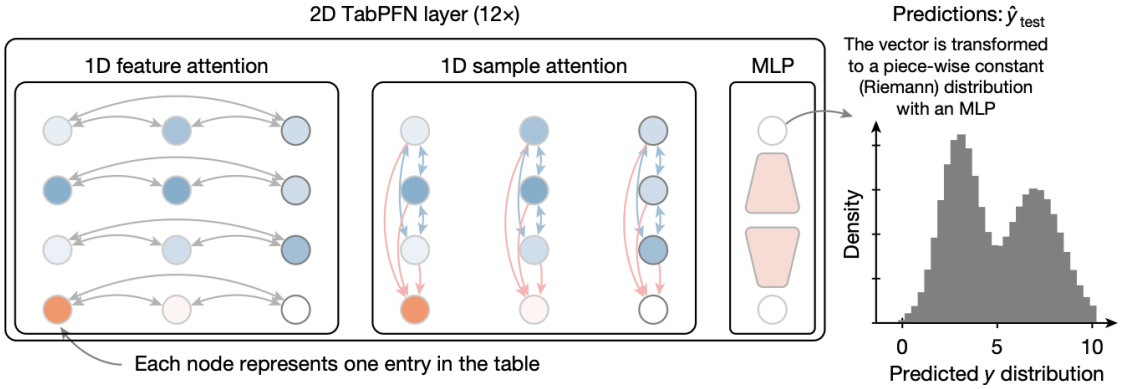


Figure 6: **How TabPFN combines attention across features and samples.** Taken from Hollmann et al. (2025), the figure illustrates the main components of the `TabPFN` architecture, discussed in Equation 1, plus the translation of the embedding into a Riemann distribution approximation of the PPD $p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$.

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 $Z_{N_{\text{train}}:,-1,:} \boldsymbol{W_Z} \in \mathbb{R}^{N_{\text{test}} \times Q}$, where $\boldsymbol{W_Z} \in \mathbb{R}^{D \times Q}$ and Q is the number of quantiles needed to compute the PPD.

Algorithm 2 Random Growing Network (RGN), with Redirection and Preferential Attachment

Require: V: total number of nodes, ρ redirection probability
Initialize graph G with nodes $n = 0$, $n = 1$ and edge $(1, 0)$
Initialize in-degree $k_j = 0$ for all $j \neq 0$, $k_0 = 1$
for $n = 2, \ldots, V - 1$ **do**
  Compute attachment probabilities for all nodes $i < n$
  $\Pi_i = \frac{k_i+1}{\sum_{j=0}^{n-1}(k_j+1)}$
  Select target node t with probability Π_t
  Sample $u \sim U(0, 1)$
  if $u < \rho$ **then**
    Connect with target, add edge (n, t)
    Update: $k_t \leftarrow k_t + 1$
  else
    Connect with target's only descendant, add edge (n, d)
    Update: $k_d \leftarrow k_d + 1$
  end if
end for
return DAG $G = (V, E)$

B Data Generation

B.1 Vanilla Graph Generation

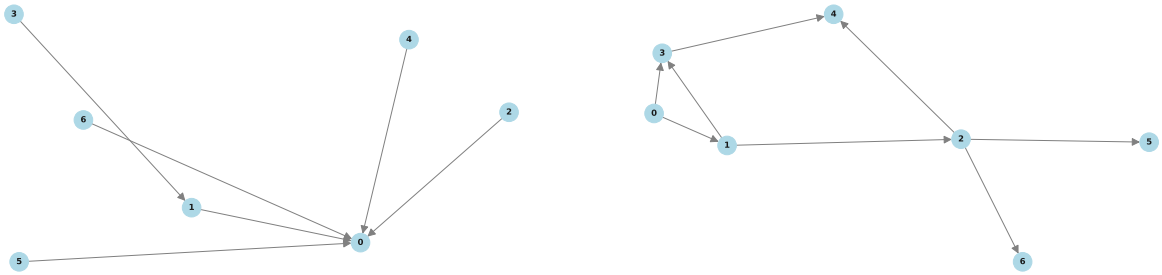
As explained in Section 3.1 and Section 3.2, we need to randomly generate DAGs to define diverse SCMs for our synthetic data procedure. The initial procedure to construct a graph (Hollmann et al., 2023) was through a MLP, where each node is connected to all other nodes in the next layer, and the depth 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). This is represented in Algorithm 2.

As illustrated in Figure 7 (Top), a characteristic of Algorithm 2 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.

B.2 Proposed Graph Generation

Our proposed SRNGN is a graph generation algorithm that generates graphs with a single root node and various paths that connect new nodes. To generate our graphs to train `ApolloPFN`, we essentially reverse the mechanics of RGN with preferential attachment (Krapivsky & Redner, 2023). That is, we always incorporate nodes in a graph by having at least one prior node connect to it, and, also, we make it connect to a popular node with probability ρ. The SRNGN graph construction algorithm is summarized in Algorithm 1; and Figure 7 (Bottom) illustrates that this results in graphs that are connected via some path and that have only one single root node by construction. In contrast, graphs sampled via RGN (Algorithm 2) are characterized by multiple root nodes and short path lengths, as illustrated in Figure 7 (Top). Our empirical results imply



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

(b) Single-root growing random network inducing longer causal paths.

Figure 7: Example graphs from distinct graph generation algorithms.

that generating data using SRNGN accelerates training (see Figure 4). Similar to Hollmann et al. (2025), we sample the number of total nodes as $\log V \sim \mathcal{U}[a, b]$ and $\rho \sim \text{TrancatedGamma}(\alpha, \beta)$.

C Dataset Details

All the dataset that we used are publicly available and can be found either the `GiftEval` (Aksu et al., 2024) repository or the `LOTSA` (Woo et al., 2024) huggingface repository. Table 5 summarizes the dataset and citations for reference.

Dataset	Source
M1	Makridakis & Hibon (1979)
M3	Makridakis & Hibon (2000)
M4	Makridakis et al. (2020)
Tourism	Hyndman et al. (2008)
M5	Makridakis et al. (2022)
Electric Price	Lago et al. (2021)
UCI Air Quality	De Vito et al. (2008)
Solar Energy	Emami (2025)

Table 5: Data sources used for benchmarking.

In terms of Electric Prices (Lago et al., 2021), we have the following.

- The Nord pool (NP) market which is one of the largest European power markets containing hourly measurements from 2023-01-01 to 2018-12-24. The NP dataset comes with exogenous variables measuring the grid load and wind power.
- Next, we have the zonal prices for the COMED area of Pennsylvania, New Jersey and Maryland (PJM), containing hourly measurements from 2023-01-01 to 2018-12-14. The PJM dataset comes with exogenous measurements of the system load and zonal load.
- Next, we have the French electricity market (FR) containing hourly measurements from 2011-01-09 to 2016-12-31. The FR dataset contains exogenous measurements of system load and power generation.
- Next, we have the Belgian electricity market (BE) containing hourly measurements from 2011-01-09 to 2016-12-31. The BE dataset contains exogenous measurements of system load and power generation.
- Finally, we have the German electricity market (DE) containing hourly measurements from 2012-01-09 to 2017-12-31. The DE dataset contains exogenous measurements of zonal load and both solar and wind generation measurements.

D Additional Analysis

D.1 Robustness Analysis to Noisy and Irrelevant Exogenous Features

We evaluate the robustness of ApolloPFN under noisy and partially irrelevant exogenous inputs through controlled perturbations that vary noise type, structure, and informativeness. We consider four settings: (1) SNR-based noise injection, where Gaussian noise is added to covariates with magnitude proportional to their empirical standard deviation to control the signal-to-noise ratio; (2) partial informativeness, where covariates are replaced by a convex combination of the original signal and independent noise, governed by a mixing coefficient α, simulating weakly predictive features; (3) irrelevant Gaussian features, where additional i.i.d. random covariates are appended; and (4) structured irrelevant features, including autoregressive (AR(1)) processes and random Fourier signals, which introduce temporally structured but task-irrelevant dependencies.

Across all settings, we follow a consistent protocol: either 50% of the original covariates are corrupted to simulate noisy or partially informative features, or 50% additional covariates are injected as irrelevant inputs.

Table 6 reports the results. ApolloPFN remains stable with minimal degradation under both unstructured (Gaussian) and structured (AR(1), Fourier) irrelevant feature injections, with only minimal deviation from the unperturbed setting. In contrast, performance degrades more noticeably under partial informativeness and low-SNR regimes, where the effective signal content of covariates is reduced. This degradation is gradual rather than abrupt, indicating that the model avoids overfitting to spurious signals even when informative structure is weakened. Overall, these experiments demonstrate that ApolloPFN is robust not only to random noise but also to structured, temporally correlated distractions that better reflect real-world exogenous covariates.

Perturbation	**Level**	**M5(D-B)**	**M5(W-B)**	**M5(M-B)**	**M5(D-S)**	**M5(W-S)**	**M5(M-S)**
None	State	0.500	1.275	2.088	0.875	1.497	2.721
SNR (50%, medium)	State	0.512	1.630	2.227	0.878	1.544	2.819
SNR (50%, high)	State	0.576	1.726	2.436	0.888	1.564	2.867
Partial (50%)	State	0.657	1.795	2.455	0.896	1.566	2.868
Gaussian extra (50%)	State	0.503	1.317	2.183	0.873	1.494	2.739
AR(1) extra (50%)	State	0.511	1.312	2.006	0.874	1.507	2.777
Fourier extra (50%)	State	0.503	1.368	2.079	0.873	1.498	2.724

Table 6: Robustness of ApolloPFN to noisy and irrelevant exogenous features under controlled perturbations across various M5-state aggregation benchmarks

D.2 Coverage Analysis of ApolloPFN and TabPFN-TS models

We evaluate prediction interval calibration of ApolloPFN against TabPFN-TS on EPF, Solar Energy, and UCI Air Quality benchmarks. Calibration is assessed using empirical coverage at nominal levels of 20%, 40%, 60%, and 80%.

Table 7 reports the average coverage across datasets and forecasting horizons. ApolloPFN consistently achieves coverage closer to nominal levels on EPF benchmarks, particularly at mid-range intervals (40 to 60%), where calibration differences are most pronounced. In contrast, TabPFN-TS exhibits more frequent under-coverage, especially at lower and intermediate prediction intervals. ApolloPFN also demonstrates more stable behavior across datasets and horizons, with fewer large deviations from the target coverage.

To quantify calibration error, Table 8 reports the absolute deviation from nominal coverage. ApolloPFN achieves lower mean deviation across all interval levels, indicating improved and more consistent calibration. The gains are most evident in the 40% and 60% intervals, while performance at 80% remains competitive across both methods. On Solar and Air Quality datasets, both methods exhibit larger deviations at lower intervals, reflecting increased uncertainty in these settings, but ApolloPFN remains competitive overall.

Dataset	20% PI		40% PI		60% PI		80% PI	
	ApolloPFN	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**
BE(24)	0.214	0.148	0.434	0.325	0.622	0.521	0.813	0.747
BE(48)	0.229	0.209	0.438	0.412	0.620	0.616	0.793	0.799
DE(24)	0.212	0.177	0.422	0.375	0.608	0.563	0.813	0.778
DE(48)	0.206	0.201	0.398	0.387	0.608	0.583	0.816	0.782
FR(24)	0.242	0.181	0.480	0.359	0.672	0.542	0.833	0.753
FR(48)	0.224	0.211	0.427	0.401	0.614	0.599	0.785	0.791
NP(24)	0.181	0.163	0.389	0.353	0.578	0.544	0.773	0.727
NP(48)	0.237	0.204	0.428	0.390	0.620	0.575	0.830	0.768
PJM(24)	0.208	0.162	0.406	0.326	0.598	0.529	0.783	0.735
PJM(48)	0.204	0.177	0.432	0.349	0.630	0.560	0.828	0.777
Solar	0.462	0.519	0.650	0.630	0.779	0.737	0.898	0.844
Air Quality	0.224	0.198	0.418	0.433	0.648	0.644	0.785	0.804

Table 7: Average empirical coverage at different prediction interval (PI) levels for ApolloPFN and TabPFN-TS across EPF, Solar Energy, and UCI Air Quality datasets. Values closer to the nominal targets (0.2, 0.4, 0.6, 0.8) indicate better calibration.

Dataset	20% PI		40% PI		60% PI		80% PI	
	ApolloPFN	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**	**ApolloPFN**	**TabPFN-TS**
BE(24)	0.014	0.053	0.034	0.075	0.022	0.079	0.013	0.053
BE(48)	0.029	0.009	0.038	0.012	0.020	0.016	0.008	0.001
DE(24)	0.012	0.023	0.022	0.025	0.008	0.037	0.013	0.022
DE(48)	0.006	0.001	0.002	0.013	0.008	0.017	0.016	0.018
FR(24)	0.042	0.019	0.080	0.041	0.072	0.058	0.033	0.048
FR(48)	0.024	0.011	0.027	0.001	0.014	0.001	0.015	0.009
NP(24)	0.019	0.038	0.011	0.048	0.023	0.056	0.027	0.073
NP(48)	0.037	0.004	0.028	0.010	0.020	0.025	0.030	0.032
PJM(24)	0.008	0.038	0.006	0.074	0.003	0.071	0.018	0.065
PJM(48)	0.004	0.023	0.032	0.051	0.030	0.040	0.028	0.023
Solar	0.262	0.319	0.250	0.230	0.179	0.137	0.098	0.044
Air Quality	0.024	0.002	0.018	0.033	0.048	0.044	0.015	0.004
Mean	**0.040**	0.045	**0.046**	0.051	**0.037**	0.048	**0.026**	0.033

Table 8: Absolute deviation from nominal coverage, computed as $|\text{empirical coverage} - \text{nominal coverage}|$ for prediction intervals at 20%, 40%, 60%, and 80% for ApolloPFN and TabPFN-TS across EPF, Solar Energy and UCI Air Quality datasets. Lower values indicate better calibration, i.e., closer alignment between predicted and target coverage levels.