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

Anonymous authors
Paper under double-blind review

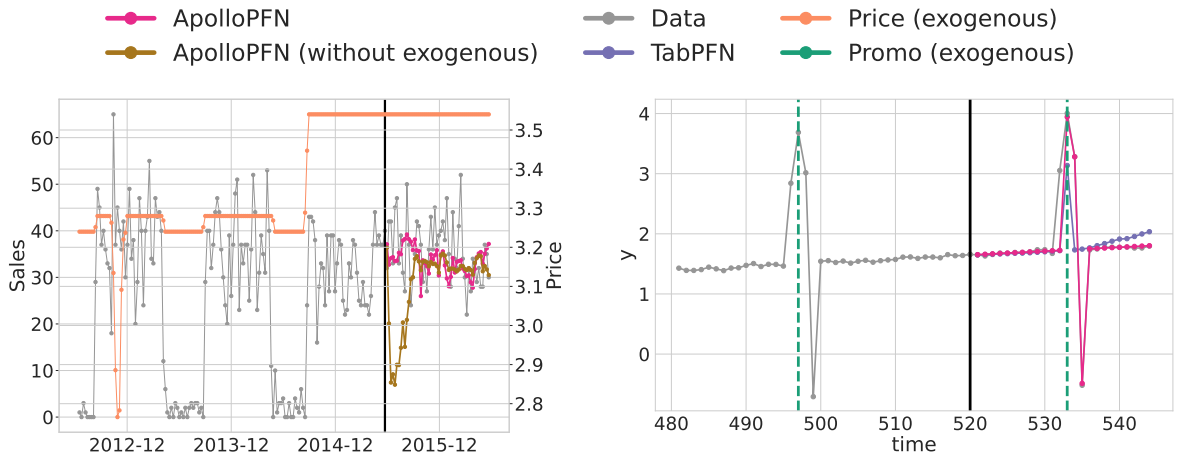
Abstract

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 benchmarks, such as M5 and electric price forecasting, 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 accept exogenous covariates in a zero-shot setting: in particular, `TabPFN-TS` (Hoo et al., 2025) and `Moirai` (Woo et al., 2024). Assessing `Moirai`'s true zero-shot capability is complicated, as it was exposed to almost all public time series benchmarks (including large-scale suites such as `GiftEval` (Aksu et al., 2024)) during training. Therefore, finding benchmarks with non-overlapping training and testing observations is difficult. Even so, it often ranks below `TabPFN-TS`, even against the benchmarks on which it was trained. 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 FM. Therefore, it lacks core temporal inductive biases (Yu et al., 2025b;a). As we discovered and describe below, a central problem is that the



(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.

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`. Such failure modes include disregarding order-dependent patterns, the inability to work well across unseen frequencies, weak trend extrapolation, insufficient emphasis on recent context, and poorly calibrated confidence intervals.

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 3.)

- We introduce `ApolloPFN`, a model that circumvents the limitations of `TabPFN-TS` through a novel data generation procedure and novel architectural choices (Section 4). 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 3) coupled with time-dependent root nodes (see Section 4.1). Second, we incorporate inductive biases into our architecture that reflect the importance of order in time series data (Section 4.2).

- 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 covariates*, as well as several benchmarks that do not have exogenous covariates, demonstrating the broad effectiveness of `ApolloPFN` (Section 5). We present several ablations on real and synthetic data to bolster our architectural and data generation choices (Section 5.4).

2 Background

`ApolloPFN` is based on a tabular FM, `TabPFN` (Hollmann et al., 2023; 2025), and so our notation refers tabular datasets in some contexts, and to time series in others. In the tabular context, the data is indexed by i as $\mathcal{D}_{\text{train}} = (\boldsymbol{x}_i, y_i)_{i=1}^{N_{\text{train}}}$, where we would make predictions for $(y_i)_{i=1}^{N_{\text{test}}}$ using $\mathcal{D}_{\text{train}}$ and the covariates $(\boldsymbol{x}_i)_{i=1}^{N_{\text{test}}}$. In contrast, when forecasting, we index our data by t, as $\mathcal{D}_{\text{train}} = (\boldsymbol{x}_t, y_t)_{t=1}^{T}$, where we therefore have T previous time steps as history, and we would make predictions for a horizon H $(y_t)_{t=T+1}^{T+H}$ using *all* of $\mathcal{D}_{\text{train}}$ as well as the future covariate information $(\boldsymbol{x}_t)_{t=T+1}^{T+H}$ (when available). Most neural forecasters in the literature solely provide predictions of the form $(y_T, \ldots, y_{T+H}) = f_\theta(y_1, \ldots, y_T)$, ignoring all $\boldsymbol{x}_t$. However, as seen in Figure 1 (a), such covariates provide crucial information to maintain accurate predictions. In this paper, we will provide a model that makes predictions of the type $(y_T, \ldots, y_{T+H}) = f_\theta(y_1, \ldots, y_T, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T+H})$ for varying T and F, where F is the covariate dimensionality $\boldsymbol{x}_t \in \mathbb{R}^F$.

2.1 Bayesian inference in prior-data fitted networks

Müller et al. (2022; 2025) introduced a novel paradigm to perform Bayesian inference through prior-data fitted networks (PFNs). First, a user defines an algorithm to sample datasets $\mathcal{D}_{\text{train}} = (\boldsymbol{x}_i, y_i)_{i=1}^{N_{\text{train}}}$, usually by sampling a vector or graph $\xi \sim p(\xi)$ and then sampling $(\boldsymbol{x}_i, y_i) \sim p(\boldsymbol{x}, y|\xi)$. By defining a neural network q_θ that minimizes the following loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{p(\boldsymbol{x},y)} \log q_\theta(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}}),$$

the neural network $q_\theta(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$ directly approximates the posterior predictive distribution (PPD) $p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$ (Müller et al., 2022). The key insight is that by having the neural network q_θ approximate the PPD, we circumvent the need to approximate a high-dimensional posterior $p(\xi|\mathcal{D}_{\text{train}})$ or to define a closed-form likelihood $p(y|\boldsymbol{x}, \xi)$, which is how the PPD is usually computed: $p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}}) = \int p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \xi) p(\xi|\mathcal{D}_{\text{train}}) d\xi$ (Murphy, 2012; Hoffman & Gelman, 2014; Wilson & Izmailov, 2020).

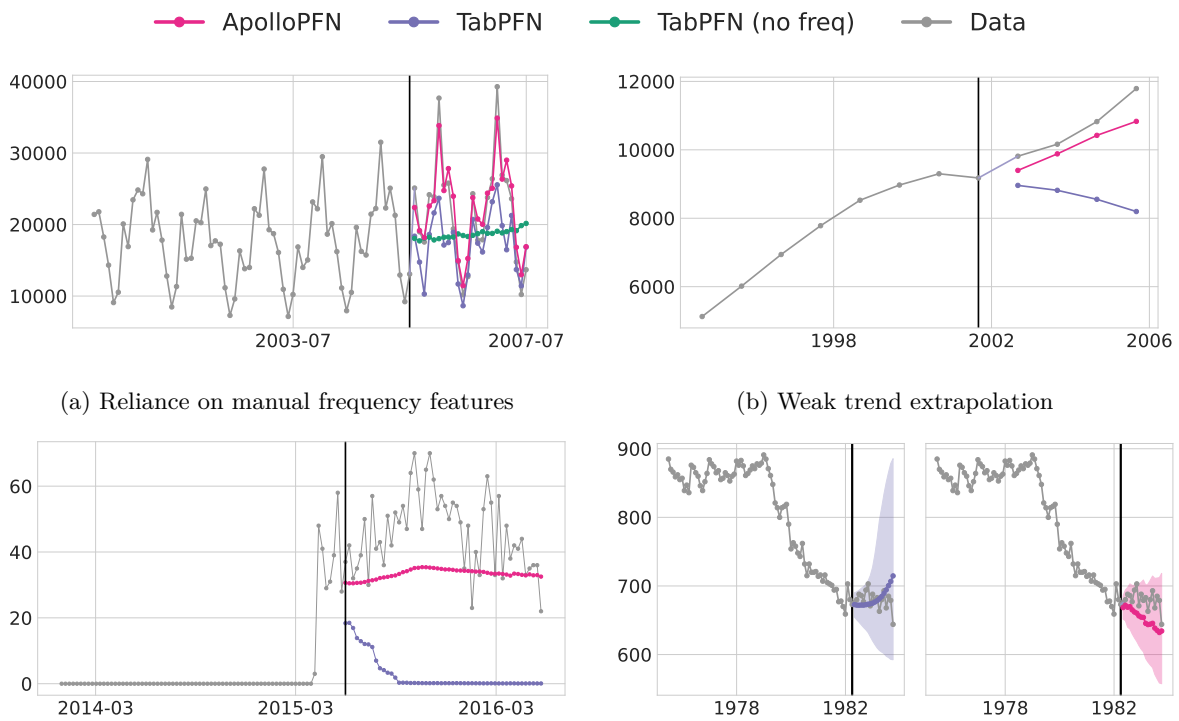
The data creation in `TabPFN` (Hollmann et al., 2023; 2025) is illustrative of how a user can generate implicit priors through sampling. `TabPFN` uses structured causal models (SCMs), which are directed acyclical graphs (DAGs) where the nodes z_i are defined by the relationship with their parent nodes $\text{PA}(i)$, as $z_i = f_i(z_{\text{PA}(i)}) + \epsilon_i$, where f_i is some function and ϵ_i is measurement noise. To generate SCMs, at a high level, Hollmann et al. (2025) samples DAGs from the random growing networks with preferential attachment process (Krapivsky & Redner, 2023) and then defines f_i as either MLPs (with distinct activations), categorical functions, or decision trees (with distinct depths). To generate N observations, we pass random noise to the root nodes and propagate the values through the graph in topological order. We then pick some F nodes and set them as the features $\boldsymbol{x}_i \in \mathbb{R}^F$, and set a node as $y_i \in \mathbb{R}$ for each $i = 1, \ldots, N$.

2.2 Sample-feature separable Transformer

The architecture in `TabPFN` (Hollmann et al., 2025) closely resembles the transformer architecture from Radford et al. (2019). Given a tensor $\boldsymbol{Z} \in \mathbb{R}^{N \times F \times D}$, where N is the number of observations (both train and test, $N = N_{\text{train}} + N_{\text{test}}$), F the number of features, and D the embedding dimension, we have that the main blocks of the `TabPFN` architecture work as follows:

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

for $\ell = 1, \ldots, L$ layers. Appendix A explains how we embed the input data $(y_i)_{i=1}^{N_{\text{train}}}$ and $(\boldsymbol{x}_i)_{i=1}^{N}$ into $\boldsymbol{Z}$. The first and second operations are variants of the classical attention mechanism (Vaswani et al., 2017), $\texttt{LN}(\cdot)$ stands for layer normalization (Ba et al., 2016), and $\texttt{MLP}(\cdot)$ is a MLP applied to the embedding dimension. The key distinction of this approach is the separability of feature and sample processing implemented via two attention mechanisms acting independently on feature and sample dimensions. `AttnFeat` assumes the



(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.

F axis is the variable part of the mechanism, the D axis is the embedding, and the remaining axes are batch axes. In contrast, `AttnSamp` assumes that the N axis is the variable part of the mechanism, the D axis is the embedding and the remaining axes are also treated as batch axes. The double feature and sample attention thus allows for a variable number of observations N and a variable number of features F. Moreover, as no positional encodings are used for `AttnSamp`, the mechanism is permutation invariant in time, which is sensible for i.i.d. data (but perhaps less so for time series). Additionally, the attention matrix $\boldsymbol{A}_{f,:,:} \in \mathbb{R}^{N \times N}$ explicitly masks interactions between predicted test points, i.e., $\boldsymbol{A}_{f,i,j} = 0$ if both i and j belong to test indices.

3 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.

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.

4 `ApolloPFN`

We now present the architectural and data-level interventions that enable the development of `ApolloPFN`, a PFN model that explicitly leverages the ordering and temporal dependencies inherent in time series data.

4.1 Synthetic Data Generation

As a baseline, we re-implement the `TabPFN` training procedure of Hollmann et al. (2025) by generating synthetic tabular data as follows. To generate a tabular dataset $\mathcal{D}_{\mathcal{G}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ with $\boldsymbol{x}_i \in \mathbb{R}^F$ and $y_i \in \mathbb{R}$, we first sample the number of observations $N \sim p(N)$ and the number of features $F \sim p(F)$. We begin by sampling a directed acyclic graph (DAG) $\mathcal{G} \sim p(\mathcal{G})$ using a Random Growing Network (RGN) with preferential attachment (Krapivsky & Redner, 2023) (see Algorithm 2 in Appendix B). The sampled graph $\mathcal{G}$ defines the parent set $\mathrm{PA}(j)$ for each node j. We then specify a structural causal model (SCM) (Pearl, 2009) of the form $V_j = f_j\big(V_{\mathrm{PA}(j)}\big) + \epsilon_j$, where f_j is chosen to be either a multilayer perceptron (MLP), a categorical encoding, or a decision tree, and ϵ_j denotes i.i.d. noise. Under this construction, each node j in $\mathcal{G}$ corresponds to a feature, with dependencies governed by the SCM. Once we have the graph of size $|\mathcal{G}|$, we randomly select $F + 1 \leq |\mathcal{G}|$ features (excluding root nodes) and set $x_{i,j} = v_{i,\pi(j)}$ and $y_i = v_{i,\pi(F+1)}$ where $\pi(\cdot)$ represents the random selection of features and the target variable.

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 $G = (V, E)$

There are two key modifications to this simple baseline synthetic data generation procedure that we introduce for `ApolloPFN`.

Single Root Node Growing Network (SRNGN). 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 6 (Bottom) in Appendix B.1 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 in Appendix B.1) are characterized by multiple root nodes and short path lengths, as illustrated in Figure 6 (Top) in Appendix B.1. Our empirical results imply that generating data using SRNGN accelerates training (see Figure 3).

Time Series Root Node Excitation. To induce time series structure in the generatied synthetic data, we sample the values of root nodes $(v_{t,r})_{t=1}^{T}$ through a stochastic process, thereby introducing a time dependency. In particular, we make the root nodes a combination between a sine and cosine function 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, we 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. After we define the temporal root nodes, we then propagate the values in the graph to obtain the rest of the features, as in Hollmann et al. (2025). We still follow the input normalization procedure from `TabPFN`. That is, we z-score the data $(y_t)_{t=1}^{T}$ before passing it to the model, and then we invert the z-scoring when outputting the predictions $(y_t)_{t=T+1}^{T+H}$.

4.2 Architectural modifications

There are two key architectural modifications that we introduce for `ApolloPFN`.

Positional Encodings. Once we have a data generation procedure that has an appropriate time dependency, it then makes sense to introduce an inductive bias to the attention mechanism that reflects these temporal relationships. A natural choice is to incorporate `RoPE` embeddings (Su et al., 2023) to the attention mechanism in $\texttt{AttnSampl}^{(\ell)}(\cdot)$. In doing so, `RoPE` would then make the keys and query interactions obey

sCRPS	DE(24)	NP(24)	FR(24)	BE(24)	PJM(24)	DE(48)	NP(48)	FR(48)	BE(48)	PJM(48)
ApolloPFN $^{(0x)}$	0.040	**0.038**	**0.040**	**0.042**	**0.040**	**0.056**	**0.053**	0.069	**0.058**	**0.057**
TabPFN-TS$^{(0x)}$	**0.033**	0.048	0.067	0.048	0.047	0.065	0.055	**0.068**	0.073	0.069
Moirai-Large$^{(\dagger x)}$	0.078	0.082	0.079	0.082	0.078	0.120	0.124	0.121	0.123	0.121
Chronos-Large$^{(0)}$	0.119	0.110	0.139	0.117	0.107	0.088	0.106	0.105	0.089	0.094
Sundial-Base$^{(0)}$	0.152	0.147	0.151	0.150	0.149	0.097	0.099	0.096	0.095	0.097

Table 1: **Electricity Price Forecasting.** ApolloPFN beats other neural forecasters that leverage exogenous information. sCRPS results on electric price forecasting across different datasets and prediction horizons (24, 48). 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.

$\boldsymbol{q}_{t+h}^{\intercal}\boldsymbol{R}_h\boldsymbol{k}_t$, 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 highly.

We note that RoPE solely incorporates a *relative* notion of distance between the observations. To incorporate an *absolute* notion of distance, we use a similar construction to Vaswani et al. (2017), and we define absolute positional encodings of the form $\boldsymbol{\Omega} \in \mathbb{R}^{T\times D}$

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

which we then add to $\boldsymbol{Z}_f \leftarrow \boldsymbol{Z}_f + \boldsymbol{\Omega}$ for all $f = 1, \ldots, F$ (see Equation 1).

Full Attention. Given that TabPFN (Hollmann et al., 2023; 2025) was trained on i.i.d. data, a key modification in the attention mechanism of $\texttt{AttnSampl}^{(\ell)}(\cdot)$ is that test observations do not attend to each other, but instead they only attend to the train observations. Therefore, when making M predictions for $(\boldsymbol{x}_j)_{j=1}^{N_{\text{test}}}$, we use the PPD of the form $p(y_j|\boldsymbol{x}_j, (\boldsymbol{x}_i, y_i)_{i=1}^{N})$ for each $j = 1, \ldots, N_{\text{test}}$ independently of each other. However, in the case of time series, if we are to make H predictions we require that all future *exogenous* information (if present) informs the current predictions. In other words, we expect that $p(y_{T+h}|(\boldsymbol{x}_t)_{t=T+1}^{T+H}, (\boldsymbol{x}_t, y_t)_{t=1}^{T})$ for all $h = 1, \ldots, H$. To achieve this, we simply allow all points to attend to each other on $\texttt{AttnSampl}^{(\ell)}(\cdot)$.

5 Empirical evaluation

In this section, we comprehensively compare ApolloPFN in several forecasting scenarios and against different forecasting models. Tables 1 and Table 2 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 3), 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.

In the setting with exogenous variables, we evaluate ApolloPFN on the electricity price forecasting (Lago et al., 2021) and the M5 competition (Makridakis et al., 2022) benchmarks 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.

Level	RMSSE	M5(D-B)	M5(W-B)	M5(M-B)	M5(D-S)	M5(W-S)	M5(M-S)
State	ApolloPFN $^{(0x)}$	**0.580**	1.652	**2.191**	0.973	**1.561**	**2.588**
	TabPFN-TS$^{(0x)}$	0.608	1.253	2.580	1.006	1.666	2.636
	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	2.405	**0.933**	1.649	2.841
Store	ApolloPFN $^{(0x)}$	0.675	1.829	**2.208**	0.990	**1.449**	**2.049**
	TabPFN-TS$^{(0x)}$	**0.651**	1.729	2.278	1.024	1.572	2.119
	Moirai-Large$^{(\dagger x)}$	0.900	2.004	3.053	0.984	1.539	2.334
	Chronos-Large$^{(0)}$	0.709	**1.715**	2.272	0.998	1.601	2.250
	Sundial-Base$^{(0)}$	0.733	2.108	2.536	**0.922**	1.452	2.202

Table 2: **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 underlined.

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).

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

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

Table 3: **`ApolloPFN` performance on the classical univariate benchmarks.** Lower values are better. Best results for each dataset are **bold**, and second best are underlined.

a square error thereby penalizing models that do not capture spikes induced by exogenous variables (e.g., promotion-induced sales spikes).

5.1 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.

5.2 Zero-shot performance with exogenous variables

Tables 1 and 2 compare the `ApolloPFN` model against foundational forecasters that leverage exogenous information such as `TabPFN-TS` and `Moirai-Large`, as well as univariate foundational forecasters such as `Chronos-Large` and `Sundial-Base`, against electricity forecasting and M5 aggregations benchmarks. In the electricity forecasting benchmark, `ApolloPFN` achieves on average 12% improvement over the next best model (`TabPFN-TS`), and it achieves SOTA across most datasets. In the M5 aggregations benchmark, `ApolloPFN` achieves SOTA performance on most aggregation levels, and it remains highly competitive with much larger TSFMs.

5.3 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 3 for a summary. Notably, `ApolloPFN` performs 10% better than `TabPFN-TS` on average, and it achieves SOTA across the different benchmarks.

5.4 Ablation Studies

Data Generation. As seen in Figure 3, our use of SRNGN dramatically increases the speed at which the model starts to make accurate predictions. For Figure 3, we trained two different `ApolloPFN` models, one with RGN and one with SRNGN, leaving the rest of the hyperparameters fixed. We then evaluated the

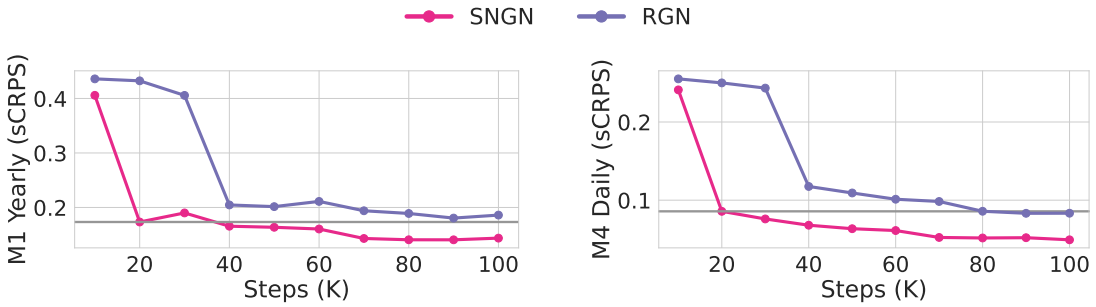


Figure 3: **SRNGN graph generation algorithm used by `ApolloPFN` accelerates learning.** We compare the test benchmark performance of our `ApolloPFN` model trained with the random growing network (RGN) algorithm and our Single Node Growing Network (SRNGN) algorithm at different training steps. With SRNGN, we achieve better performance at 20K iterations than at 80K with RGN.

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.

Architecture Modifications. Our new time series synthetic data generation process coupled with the architectural changes presented in Section 4.2 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 4, which presents the relative accuracy improvement of architectural variants with respect to `TabPFN-TS` baseline. Figure 4 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.

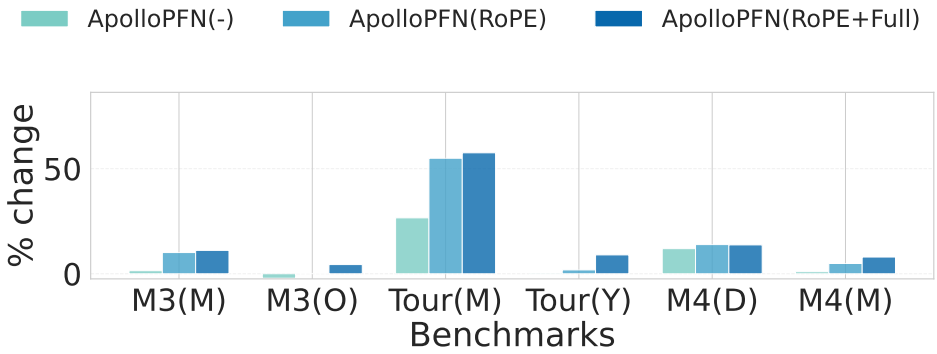


Figure 4: **Interventions in `ApolloPFN` to improve performance on time series data.** Ablation on the use of `RoPE` and full attention. We compare the effect of progressively adding `RoPE` and full attention in several benchmarks against the baseline of `TabPFN-TS`.

6 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.

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.

Judea Pearl. Causality: Models, Reasoning and Inference. *Cambridge University Press*, 2009.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI*, 2019.

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.

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 2.1. 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 $\boldsymbol{Y} = \tilde{\boldsymbol{Y}} \boldsymbol{W}_{\boldsymbol{Y}}$ 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 $f = 1, \ldots, F'/2$ 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 $i = 1, \ldots, N$. 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 5 and discussed Equation 1 in Section 2.1.

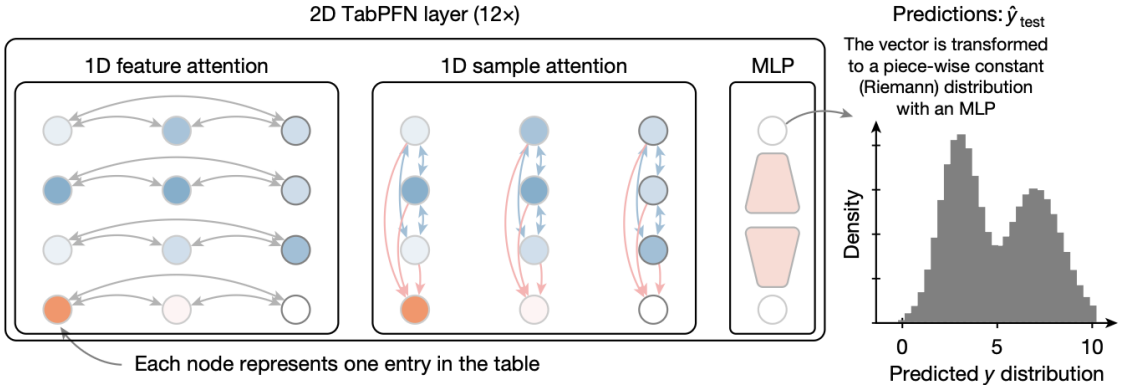


Figure 5: **How TabPFN combines attention across features and samples.** Taken from Hollmann et al. (2025), the figure illustrates the main components of the `TabPFN` architecture, discussed in Equation 1, plus the translation of the embedding into a Riemann distribution approximation of the PPD $p(y_{\text{test}}|\boldsymbol{x}_{\text{test}}, \mathcal{D}_{\text{train}})$.

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

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

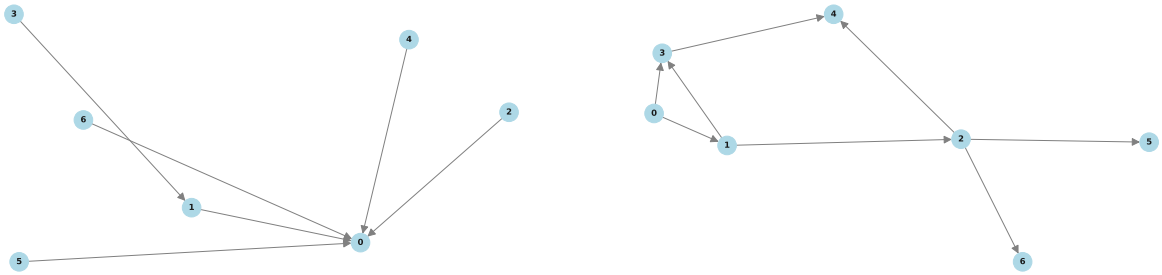
As explained in Section 2.1 and Section 4.1, we need to randomly generate DAGs to define diverse SCMs for our synthetic data procedure. The initial procedure to construct a graph (Hollmann et al., 2023) was through a MLP, where each node is connected to all other nodes in the next layer, and the depth 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). This is represented in Algorithm 2.

As illustrated in Figure 6 (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 6 (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 6 (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 6: Example graphs from distinct graph generation algorithms.

that generating data using SRNGN accelerates training (see Figure 3). 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 4 summarizes the dataset and citations for reference.

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.

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)

Table 4: Data sources used for benchmarking.