TimePFN: Effective Multivariate Time Series Forecasting with Synthetic Data

Ege Onur Taga, Muhammed Emrullah Ildiz, Samet Oymak

University of Michigan, Ann Arbor
egetaga@umich.edu, eildiz@umich.edu, oymak@umich.edu

Abstract

The diversity of time series applications and scarcity of domain-specific data highlight the need for time-series models with strong few-shot learning capabilities. In this work, we propose a novel training scheme and a transformer-based architecture, collectively referred to as *TimePFN*, for multivariate time-series (MTS) forecasting. *TimePFN* is based on the concept of Prior-data Fitted Networks (PFN), which aims to approximate Bayesian inference. Our approach consists of (1) generating synthetic MTS data through diverse Gaussian process kernels and the linear coregionalization method, and (2) a novel MTS architecture capable of utilizing both temporal and cross-channel dependencies across all input patches. We evaluate *TimePFN* on several benchmark datasets and demonstrate that it outperforms the existing state-of-the-art models for MTS forecasting in both zero-shot and few-shot settings. Notably, fine-tuning *TimePFN* with as few as 500 data points nearly matches full dataset training error, and even 50 data points yield competitive results. We also find that *TimePFN* exhibits strong univariate forecasting performance, attesting to its generalization ability. Overall, this work unlocks the power of synthetic data priors for MTS forecasting and facilitates strong zero- and few-shot forecasting performance.

Code — https://github.com/egetaga/TimePFN

1 Introduction

Natural language processing has achieved remarkable success driven by advances in neural architectures and data pipelines. These advances underlie modern language and vision-language models that exhibit remarkable zero-shot and few-shot learning capabilities. Inspired by these, researchers have started exploring whether such methods and ideas could be extended to time series forecasting. For instance, a notable line of work (Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022; Zhang and Yan 2023) examine the use of transformer architecture (Vaswani et al. 2017) in time-series forecasting. More recently, there is also a push toward building foundation models for time series tasks (Ansari et al. 2024). However, the heterogeneous nature of time series data brings additional complications. As shown by (Zeng et al. 2023), even simple linear models are shown to outperform most existing transformer-based models in univariate and multivariate time-series forecasting. This could be attributed to the heterogeneous nature of time-series data and relatively naive tokenization methods, underscoring the need for richer datasets as well as more effective architectures that can capture both temporal and cross-channel dependencies.

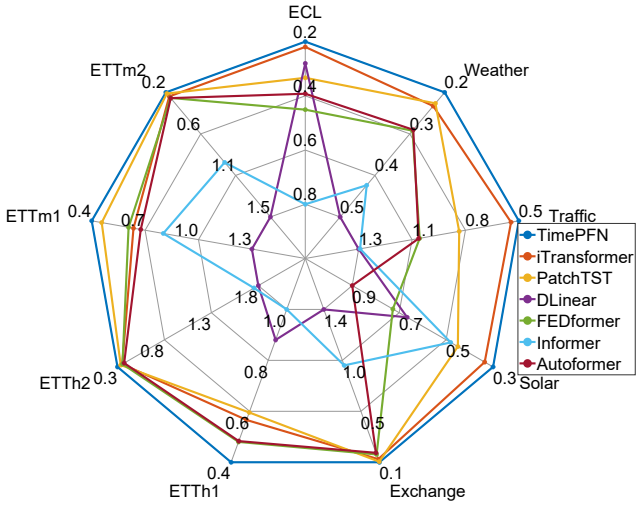

Figure 1: Average forecasting performance (MSE) of TimePFN. MSE values are averaged over all data budgets.

In language models, the discrete nature of the problem makes the tokenization fairly straightforward, which is in contrast to the continuous time series data. Additionally, the scalar value of a time series datapoint have no clear meaning, unlike words, where vector embeddings can capture semantic similarity. To address these problems, PatchTST (Nie et al. 2023) proposed using patching with overlapping strides and demonstrated its benefit for univariate forecasting. While PatchTST treats multivariate forecasting as multiple univariate problems, iTransformer (Liu et al. 2023) proposes representing each channel as a single token, resulting in an architecture that intuitively augments simple linear layers with a transformer architecture.

In this work, we approach MTS forecasting from a data-

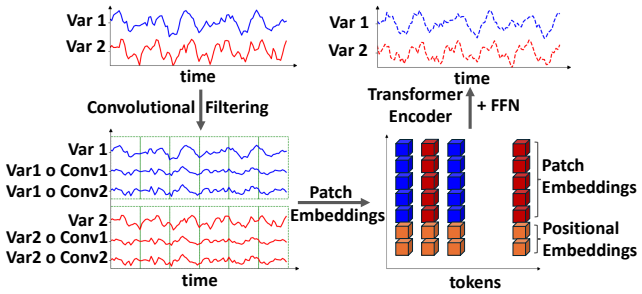

Figure 2: Illustration of the architecture of TimePFN. Variates are filtered with 1D convolutions, to be patched with overlapping strides, following (Nie et al. 2023). They are then fed into transformer encoder with channel mixing, with the final forecast coming from the final feedforward network.

centric perspective. While various architectural considerations have been incorporated into the forecasting process, we argue that the data aspect is relatively underappreciated. Existing transformer-based MTS approaches focus on the classical learning setup where a model is trained and tested on the same task. Although this often results in satisfactory performance for large datasets, it is likely to underperform in real-world applications where the training set is small or test set is out-of-distribution. This is especially so for modern sequence/transformer models that involve complex architectures and naturally require a substantial amount of data to operate at optimal performance.

Our approach *TimePFN* brings two key innovations: (1) Generating realistic and diverse large-scale multivariate time series data, where inter- and intra-channel dependencies are common, and (2) Developing an architecture capable of extracting time series features from this large-scale synthetic dataset. The architecture also allows for transfer learning to novel tasks with arbitrary number of channels. Overall, empowered by large amount of synthetic data (on the order of millions of samples), *TimePFN* facilitates state-of-the-art zero-shot and few-shot accuracy on benchmark datasets.

The strong zero-shot performance of our model, along with its superior performance in few-shot settings, supports the importance of the data-centric perspective. Evaluations demonstrate that our model, when fine-tuned on as few as 50 to 500 samples, is competitive with the performance of alternative methods trained on the entire dataset. More specifically, we make the following contributions:

- We present a new method to generate synthetic multivariate time series data using Gaussian processes with kernel compositions and a linear coregionalization model.
- We propose a variation of PatchTST (Nie et al. 2023) for multivariate forecasting. Unlike PatchTST, our architecture incorporates channel mixing and employs a convolutional embedding module for patch embeddings. This allows it to effectively extract cross-channel relations and generate more representative embeddings, as demonstrated by experiments.
- *TimePFN* is the first multivariate time-series PFN. Notably, *TimePFN* demonstrates strong zero-shot and few-shot performance and consistently outperforms comparable models/methods across various benchmarks.
- We find that *TimePFN* also exhibits strong univariate forecasting performance, although it is explicitly trained with synthetic multivariate data. This attests to the flexibility and generalization capability of our approach.

Algorithm 1: LMC-Synth

Input: Number of variates N, time-series length T, Weibull shape parameter β, Weibull scale parameter λ, (min, max) value of dirichlet concentration parameter (d_{min}, d_{max}), minimum number of latent functions m, maximum number of kernel composition in KernelSynth J

Output: Synthetic MTS $\boldsymbol{C}$ with N variates and length T

$L \sim max(min(Weibull(\beta, \alpha), N), m)$
$d \sim U(d_{min}, d_{max})$
for j $\in \{1 \dots L\}$ **do**
  $l_j(\boldsymbol{t}) \leftarrow KernelSynth(J, T)$
end for
for i $\in \{1 \dots N\}$ **do**
  $[\alpha_{i,1} \dots \alpha_{i,L}] \sim Dir(d)$
  $C_i(\boldsymbol{t}) \leftarrow \sum_{j=1}^{L} \alpha_{i,j} l_j(\boldsymbol{t})$
end for
return $\{C_i(\boldsymbol{t})\}_{i=1}^{N}$

2 Related Work

Transformers (Vaswani et al. 2017) have revolutionized NLP, significantly advancing zero-shot and few-shot capabilities in language and vision models. This has led researchers to explore the application of transformers to time-series forecasting, leading to a substantial body of work including but not limited to (Zhou et al. 2021; Wu et al. 2021; Zhou et al. 2022; Li et al. 2019; Nie et al. 2023; Liu et al. 2022; Zhang and Yan 2023). Informer by (Liu et al. 2023) introduces the ProbSparse attention mechanism, which alleviates the quadratic complexity of the naive transformer to log-linear complexity, to mitigate the scalability issues in long sequence time-series forecasting (LSTF). (Zhou et al. 2022) uses the sparsity of the time-series in the fourier domain to enhance the performance in LSTF. PatchTST (Nie et al. 2023) uses patching with overlapping strides as a tokenization mechanism to address issues associated with naive tokenization of time-series data. This approach yields patch-based tokens that are interpretable while maintaining channel independence, treating each channel as univariate but facilitating joint learning across all channels through the same set of shared weights. Our architecture deviates from PatchTST by incorporating convolutional layers before patching and using channel-mixing to capture interactions between tokens from different channels. The advantages of convolutions are highlighted in the speech literature by (Baevski et al. 2020; Hsu et al. 2021). On the other hand, iTransformer (Liu et al. 2023) treats each variate as a single token, showing the potential benefits of utilizing inter-

channel relationships.

In zero-shot forecasting, a line of work has emerged (Orozco and Roberts 2020; Oreshkin et al. 2021; Jin et al. 2022; Dooley et al. 2023; Ansari et al. 2024). More recently, (Ansari et al. 2024) has developed novel tokenization methods, employed quantization, and made time series data resemble language, enabling the training of LLM architectures for probabilistic univariate forecasting in a framework called Chronos. Chronos employs a data augmentation technique called KernelSynth, which generates synthetic time-series data using Gaussian processes to improve the generalization capability of the model. Meanwhile, another line of work, ForecastPFN (Dooley et al. 2023), is trained entirely on a synthetic dataset following the framework of Prior-data Fitted Networks (PFNs). Initially proposed by (Müller et al. 2022), PFNs are designed to approximate Bayesian inference. Another study (Verdenius, Zerio, and Wang 2024) integrates Joint-Embedding Predictive Architectures with PFNs for zero-shot forecasting. In addition to the mentioned models, Mamba4Cast (Bhethanabhotla et al. 2024) is also trained entirely on synthetic data using the Mamba architecture as its backbone (Gu and Dao 2024). While the mentioned literature addresses univariate settings, our work introduces the first multivariate Prior-data Fitted Network, to the best of our knowledge, featuring an architecture that enables strong zero-shot and few-shot performances on MTS forecasting.

3 Proposed Method

This work relies on two key aspects: a multivariate synthetic time series data generation mechanism that encapsulates inter- and intra-channel dependencies common across real time series data, and an architecture capable of generalization to real datasets when trained on such a dataset.

In the following section, we introduce the main concept behind our synthetic MTS data generation and training mechanism: Prior-data Fitted Networks (PFNs).

Prior-data Fitted Networks for MTS Forecasting

Let $\mathcal{D} := \{t, X_t\}_{t=1}^{T}$ represent an N-channel multivariate time series data spanning a time horizon T, where $X_t := [x_{t,1}, \ldots, x_{t,N}]$. Each $x_{t,i}$ is potentially causally dependent on previous time steps and on one another. Given the data $\{t, X_t\}_{t=1}^{\widetilde{T}}$ where $\widetilde{T} < T$, the task is to forecast $X_{\widetilde{T}+1}, \ldots, X_T$. We tackle this problem using a Bayesian framework. Assuming a hypothesis space Ω with a prior distribution $p(\omega)$, each hypothesis $\omega \in \Omega$ models a multivariate time series (MTS) generating process, i.e., $X_t = \omega(t)$. For example, Ω could represent the space of hypotheses for vector autoregression (VAR) models, where a particular instance $\omega \in \Omega$ corresponds to a specific VAR process, such as VAR(2), and data $\mathcal{D}$ can be generated via this process. Now, given a data $\mathcal{D} := \{t, X_t\}_{t=1}^{\widetilde{T}}$ where $\widetilde{T} < T$, the posterior predictive distribution (PPD) of $\boldsymbol{x} \in \mathbb{R}^N$ at time T is $p(\cdot \mid T, \mathcal{D})$. By Bayes' theorem,

$$p(\boldsymbol{x} \mid T, \mathcal{D}) \propto \int_{\Omega} p(\boldsymbol{x} \mid T, \omega) p(D \mid \omega) p(\omega) d\omega \qquad (1)$$

As shown by (Müller et al. 2022; Hollmann et al. 2023; Dooley et al. 2023), the posterior predictive distribution (PPD) is approximated using prior fitting networks (PFNs) as follows: We iteratively sample a hypothesis ω from the hypothesis space Ω according to the probability $p(\omega)$. Next, we generate a prior dataset $\mathcal{D}$ from this hypothesis, denoted as $\mathcal{D} \sim p(\mathcal{D} \mid \omega)$. We then optimize the parameters of the PFN on these generated datasets using standard methods. The time series dataset is divided into input and output parts, where $\mathcal{D}_{input} := \{t, X_t\}_{t=1}^{\widetilde{T}}$ and $\mathcal{D}_{output} := \{t, X_t\}_{t=\widetilde{T}+1}^{T}$. Subsequently, we train the PFN to forecast $\mathcal{D}_{output}$ from $\mathcal{D}_{input}$ using standard time-series transformer training techniques, aiming to minimize the mean-squared error loss as our optimization objective, following the setting of (Dooley et al. 2023).

In our work, we define the hypothesis space Ω as consisting of single-input, multi-output Gaussian processes represented by the linear model of coregionalization (LMC) (Journel and Huijbregts 2003). Our choice is driven by the representational power of Gaussian processes and their ability to generate a diverse range of time series through the LMC framework.

Synthetic MTS Data Generation

In synthetic MTS (multivariate time series) data generation, our goal is twofold. First, we strive to create variates that are realistic, exhibiting periodic patterns, trends, and other common features found in real-world data. Second, we aim for these variates to be correlated with one another, which better represents MTS data characteristics. Fortunately, our first goal is addressed by a method called KernelSynth. Chronos (Ansari et al. 2024) uses KernelSynth to enrich its training corpus by randomly composing kernels using binary operators (such as addition and multiplication) to generate diverse, univariate synthetic time-series data. This method is essentially the inverse of the kernel composition approach described in (Duvenaud et al. 2013), where kernel compositions are used for structure discovery in nonparametric regression. In contrast, KernelSynth focuses on generating realizations from these kernels. For example, combining a linear kernel with a periodic kernel results in a pattern that exhibits both a linear trend and sinusoidal seasonality. Similarly, multiplying a squared-exponential kernel with a periodic kernel creates locally periodic patterns (Duvenaud et al. 2013). Chronos aggregates kernels of various types—such as Linear, Periodic, Squared-Exponential, Rational, and Quadratic—and with different parameters (such as daily, weekly, and monthly periodic kernels) in a kernel bank, composing them as described above to define the Gaussian processes.

However, generating a MTS time-series data is yet to be addressed. To address the second goal, generating variates that are correlated in a realistic manner, we use a generative Gaussian modelling, called linear model of coregionalization (LMC), which is developed initially in the field of geostatistics (Journel and Huijbregts 2003). For ease of understanding, we adopt the time-series notation we used above. In LMC, the outputs are obtained as linear combinations of

independent latent random functions. In other words, given $\boldsymbol{t} \in \mathbb{R}^T$, the outputs in each channel $\{C_i(\boldsymbol{t})\}_{i=1}^N$ is the linear combination of L latent functions

$$C_i(\boldsymbol{t}) = \sum_{j=1}^{L} \alpha_{i,j} l_j(\boldsymbol{t}) \tag{2}$$

Observe that, since latent functions are independent with zero-mean the resulting output covariance is a well-defined PSD function with zero-mean (Álvarez, Rosasco, and Lawrence 2012). In our synthetic data generation algorithm, to avoid scaling issues, we restrict ourselves to convex combinations. Thus, for each i, we have $\alpha_{i,1} + \dots \alpha_{i,L} = 1$ with $\alpha_{i,j} \geq 0$, meaning that the outputs lie in the convex hull of latent functions l_j's. We generate the latent functions based on KernelSynth's algorithm due to it's extensive descriptive value. Note that the LMC formulation encapsulates the cases where the correlations between different variates are small or nonexistent. Specifically, the case where each variate is independent from the rest corresponds to $L = N$ with $C_i(\boldsymbol{t}) = l_i(\boldsymbol{t})$. Such a modelling is important, as some MTS data have strong correlation between different variates, whereas others have small or non-existent correlation.

In our algorithm, LMC-Synth, we sample the number of latent functions from a Weibull distribution and $[\alpha_{i,1} \dots \alpha_{i,L}]$ from a Dirichlet distribution. To avoid highly skewed cases, we impose upper and lower bounds on the possible number of latent functions. Since the uncorrelated setting of $L = N$ with $C_i(\boldsymbol{t}) = l_i(\boldsymbol{t})$ is crucial for modeling MTS problems with low correlation among variates, we also generate data under this setting. Incorporating this extra setting is shown to yield the strongest performance in zero-shot settings.

Architecture for TimePFN

In designing the architecture, our main principle was to create a system capable of extracting time-series features useful for MTS forecasting. Through this, we aimed for the architecture to achieve better generalization when applied to real-world datasets. The primary advantage of the PFN framework in our case is that, since synthesizing large-scale synthetic MTS data is feasible with LMC-Synth, we are no longer constrained by data scarcity. Previous MTS models were compelled to balance model complexity with their datasets due to limited data, often restricting the use of certain components or their quantity (such as the number of transformer layers) to avoid overfitting. However, with access to large-scale MTS data, we can expand our architecture and freely incorporate additional components that we believe will improve forecasting performance on new datasets. In light of this, we proceed to explain our architecture and design choices.

The *TimePFN* model resembles PatchTST (Nie et al. 2023) in several aspects when processing MTS data, but it differs significantly in two areas: our convolutional filtering of the variates prior to patching and channel-mixing.

Convolutional Filtering. Before patching, consider an MTS dataset $X = [x_1 \dots x_N]$ in $\mathbb{R}^{L \times N}$, where L is the length and N is the number of variates. We apply learnable 1D convolution operations to each variate, with convolutional weights shared across all variates. After convolutions, we apply 1D magnitude max pooling to each newly generated variate, followed by a new set of 1D convolutions. In *TimePFN*, each $x_i \in \mathbb{R}^L$ is transformed into $\bar{x}_i \in \mathbb{R}^{(C+1) \times L}$, where C rows come from 1D convolutional operations and magnitude max pooling, whereas one row is the original x_i. We keep the original x_i to not loose any information, analogous to skip connections used in NLP (He et al. 2015). In practice, we used $C = 9$. Filtering with convolutions is a valuable tool in time-series analysis. Many operations, such as differencing to de-trend data, can be effectively represented by convolutions. We utilized this approach to extract common time-series features across various datasets, thereby improving the generalization capability of our model.

Patch Embeddings. Given $\bar{x}_i \in \mathbb{R}^{(C+1) \times L}$, we extract overlapping patches of size P with a stride of S, following the settings described in (Nie et al. 2023). Each patch thus has dimensions $\mathbb{R}^{(C+1) \times P}$, and a total of $\left\lfloor \frac{L-P}{S} + 2 \right\rfloor$ patches are extracted from a single variate. In total, we get $N \times \left\lfloor \frac{L-P}{S} + 2 \right\rfloor$ patches. Each patch is then flattened and fed into a 2-layer feedforward neural network to be mapped to embedding dimension D. We add 2D sinusoidal positional encodings (Wang and Liu 2019) to the embeddings to correctly capture channel-wise and temporal information. In practice, we used $(P = 16, S = 8)$, similar to (Nie et al. 2023).

Channel-mixing. Unlike PatchTST (Nie et al. 2023), where the tokens from each channel are fed independently into a transformer encoder, we input all tokens into the transformer encoder after applying the positional encodings described above. Consequently, tokens from different variates can attend to each other.

Transformer Encoder. We employ a naive multihead transformer encoder, incorporating layer normalizations (Ba, Kiros, and Hinton 2016) and skip connections (He et al. 2015) to improve training stability. After feeding the tokens into the multilayer encoder, we rearrange them into their respective channels and apply a channel-wise flattening operation. This is followed by a two-layer feedforward network that processes the flattened variate representations using shared weights (single FFN is applied to all variates).

Normalization. We normalize each variate x_i to have zero mean and unit standard deviation prior to any other process described above, as recommended by (Kim et al. 2021), to alleviate the impact of distribution shifts between our synthetic dataset and test examples (Liu et al. 2023; Nie et al. 2023). Before forecasting, we revert the time series to its original scale by de-normalizing.

Architectural Details. Due to our architectural specifications, *TimePFN* has fixed input sequence and forecasting lengths. However, it can accept an arbitrary number of variates. Thus, although we trained *TimePFN* with a synthetic dataset of a fixed channel size ($C = 160$), it can forecast with both fewer and more channels than those used in its training data. When forecasting with a number of channels $\bar{C} \leq C$, we directly input the data to *TimePFN*. To mitigate

	Dataset	ECL		Weather		Traffic		Solar		Exchange		ETTh1		ETTh2		ETTm1		ETTm2	
	Models	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
z.s.	TimePFN	**0.315**	**0.383**	**0.209**	**0.255**	**1.108**	**0.613**	0.941	**0.730**	0.105	0.229	**0.453**	**0.439**	**0.328**	**0.362**	**0.637**	**0.512**	**0.212**	**0.291**
	Naive	1.587	0.945	0.259	0.254	2.714	1.077	1.539	0.815	**0.081**	**0.196**	1.294	0.713	0.431	0.421	1.213	0.664	0.266	0.327
	SeasonalN.	1.618	0.964	0.268	0.263	2.774	1.097	1.599	0.844	0.086	0.204	1.325	0.727	0.445	0.431	1.227	0.673	0.274	0.334
	Mean	0.845	0.761	0.215	0.271	1.410	0.804	**0.910**	0.734	0.139	0.269	0.700	0.558	0.352	0.387	0.693	0.547	0.229	0.307
Budget =50	TimePFN	**0.235**	**0.322**	**0.190**	**0.235**	**0.746**	**0.468**	**0.429**	**0.450**	**0.096**	**0.218**	**0.438**	**0.429**	**0.324**	**0.359**	**0.419**	**0.418**	**0.195**	**0.276**
	iTransformer	0.278	0.360	0.237	0.278	0.801	0.499	0.513	0.479	0.145	0.275	0.838	0.617	0.410	0.422	0.884	0.608	0.268	0.337
	PatchTST	0.667	0.646	0.221	0.269	1.295	0.746	0.810	0.669	0.127	0.255	0.778	0.587	0.372	0.401	0.656	0.528	0.231	0.310
	DLinear	0.406	0.463	0.742	0.612	1.888	0.937	0.956	0.813	3.432	1.349	1.404	0.881	3.928	1.383	1.332	0.846	3.484	1.290
	FEDFormer	0.908	0.758	0.306	0.381	1.587	0.874	0.972	0.757	0.165	0.300	0.676	0.570	0.424	0.468	0.745	0.589	0.291	0.387
	Informer	1.226	0.896	0.464	0.511	1.714	0.901	0.887	0.783	1.470	1.007	1.172	0.819	2.045	1.093	1.003	0.745	1.590	0.995
	Autoformer	0.729	0.675	0.322	0.401	1.600	0.883	1.065	0.808	0.213	0.351	0.607	0.560	0.492	0.506	0.763	0.592	0.316	0.407
Budget =500	TimePFN	**0.190**	**0.283**	**0.178**	**0.222**	**0.487**	**0.335**	**0.269**	**0.305**	0.083	0.203	**0.401**	**0.412**	**0.311**	**0.352**	**0.360**	**0.386**	**0.185**	**0.268**
	iTransformer	0.200	0.284	0.211	0.248	0.514	0.354	0.307	0.334	0.113	0.239	0.489	0.470	0.361	0.394	0.569	0.494	0.231	0.310
	PatchTST	0.236	0.320	0.210	0.246	0.740	0.455	0.321	0.353	**0.081**	**0.198**	0.596	0.515	0.358	0.392	0.369	0.386	0.190	0.275
	DLinear	0.235	0.328	0.335	0.394	1.312	0.727	0.622	0.656	0.655	0.551	0.749	0.609	1.098	0.712	0.817	0.621	0.870	0.626
	FEDformer	0.317	0.407	0.265	0.341	0.888	0.548	0.821	0.706	0.157	0.288	0.444	0.452	0.358	0.401	0.674	0.542	0.238	0.322
	Informer	0.869	0.760	0.320	0.393	1.411	0.774	0.318	0.385	0.699	0.694	0.913	0.713	1.311	0.940	0.704	0.595	1.121	0.803
	Autoformer	0.303	0.396	0.237	0.312	0.896	0.549	0.950	0.787	0.158	0.290	0.456	0.456	0.339	0.384	0.672	0.534	0.223	0.308
Budget = All	TimePFN	**0.138**	**0.137**	**0.166**	**0.208**	**0.392**	**0.260**	0.203	0.219	0.100	0.223	0.402	0.417	**0.293**	**0.343**	0.392	0.402	0.180	0.262
	iTransformer	0.147	0.239	0.175	0.215	0.393	0.268	0.201	0.233	0.086	0.206	**0.387**	0.405	0.300	0.349	0.342	0.376	0.185	0.272
	PatchTST	0.185	0.267	0.177	0.218	0.517	0.334	0.222	0.267	**0.080**	**0.196**	0.392	**0.404**	**0.293**	**0.343**	**0.318**	**0.357**	**0.177**	**0.260**
	DLinear	0.195	0.278	0.341	0.412	0.690	0.432	0.286	0.375	0.101	0.237	0.400	0.412	0.357	0.406	0.344	0.371	0.195	0.293
	FEDformer	0.196	0.310	0.227	0.313	0.573	0.357	0.242	0.342	0.148	0.280	0.380	0.417	0.340	0.386	0.363	0.408	0.191	0.286
	Informer	0.327	0.413	0.455	0.481	0.735	0.409	**0.190**	**0.216**	0.921	0.774	0.930	0.763	2.928	1.349	0.623	0.559	0.396	0.474
	Autoformer	0.214	0.327	0.273	0.344	0.605	0.376	0.455	0.480	0.141	0.271	0.440	0.446	0.364	0.408	0.520	0.490	0.233	0.311
# of Variates		321		21		862		137		8		7		7		7		7	

Table 1: MTS forecasting results of TimePFN and comparable architectures with best results in bold. Input and forecast lengths are set to be 96. SeasonalN. stands for Seasonal Naive. *TimePFN* demonstrates remarkable performance in budget-limited settings, as well as with the full dataset, particularly in scenarios involving a large number of variates.

the effects of distribution shifts when forecasting with more channels at test time, we process the data by splitting it into non-overlapping channels of size at most C. If the test data has $\bar{C}$ channels, we divide it into $\left\lfloor \frac{\bar{C}}{C} \right\rfloor + 1$ segments, input them separately, and then stack them afterwards.

4 Experiments

In all MTS evaluations, our primary objective is to forecast a horizon of 96 time steps using an MTS input of 96 time steps. We trained a single *TimePFN* model on a large-scale, multivariate synthetic dataset generated by LMC-Synth and conducted all experiments using this model. We generated 15,000 synthetic datasets with a length of 1024 and a channel size of 160 from LMC-Synth, further augmenting with datasets having independent variates as in the case $C_i(\boldsymbol{t}) = l_i(\boldsymbol{t})$. The independent data comprises approximately 25% of the purely correlated data. During training, we extracted time-series input and output pairs using a sliding window of size 192 (96 for input, 96 for output), resulting in approximately 1.5 million synthetic data points. We trained the model to forecast the MTS output based on the given input using MSE loss with our 160 channel synthetic dataset. Training a single *TimePFN* of 8 transformer layers takes around 10 hours on L40S GPU.

In the few-shot evaluations, we fine-tuned *TimePFN* using the specified data budget. We did not perform any hyperparameter tuning on *TimePFN*, and the same set of hyperparameters was used in all few-shot settings. Details about the model hyperparameters are provided in the appendix.

Benchmark Datasets. We evaluated *TimePFN* on nine widely-used, real-world benchmark datasets for MTS forecasting. These datasets include ETTh1, ETTh2, ETTm1, ETTm2 (collectively referred to as ETT, representing Electricity Transformer Temperature), Weather, Solar Energy, ECL (Electricity Consuming Load), Exchange, and Traffic. The Solar Energy dataset was introduced by (Lai et al. 2018), while the others were introduced by (Wu et al. 2021). We provide the specifications of these datasets in the appendix section *Datasets*.

Baselines. Since no MTS PFN is available, we compared *TimePFN* with state-of-the-art transformer-based MTS forecasting models, including FEDformer (Zhou et al. 2022), Autoformer (Wu et al. 2021), Informer (Zhou et al. 2021), PatchTST (Nie et al. 2023), and iTransformer (Liu et al. 2023). We evaluated these models across the entire dataset and at various data budgets, including 50, 100, 500, and 1000 data points. For instance, at a data budget of 500, the model is trained using 500 MTS input and output pairs. Additionally, we included DLinear (Zeng et al. 2023), a linear model, as part of our baseline. Given its lower complexity, we consider it a strong baseline for smaller data budgets.

For smaller data budgets and our zero-shot evaluations, we incorporated three algorithmic baselines as suggested by

Dataset	TimePFN-36		TimePFN-96		ForecastPFN		Chronos-s		SeasonalN.		Naive		Mean	
Models	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ECL	**0.752**	**0.703**	0.509	0.549	1.416	0.958	1.152	0.792	1.559	0.995	1.211	0.829	0.963	0.805
Weather$\times 10^2$	0.042	1.381	0.046	1.477	0.084	1.999	0.036	1.136	0.045	1.352	**0.035**	**1.123**	0.069	1.893
Traffic	**1.503**	**1.032**	0.414	0.503	4.521	1.742	3.103	1.364	4.301	1.771	3.330	1.463	2.125	1.256
Exchange	0.027	0.125	0.034	0.139	0.057	0.180	0.049	0.113	0.028	0.128	**0.022**	**0.107**	0.040	0.156
ETTh1	**0.029**	0.130	0.030	0.133	0.102	0.237	0.061	0.155	0.039	0.151	0.031	**0.128**	0.040	0.154
ETTh2	**0.126**	**0.273**	0.086	0.224	0.434	0.517	0.207	0.321	0.279	0.408	0.215	0.336	0.168	0.321

Table 2: Zero-shot results of TimePFN on univariate time-series forecasting with input length = 36. TimePFN-96 has input length of 96. The errors are averaged over forecasting lengths of $\{6, 8, 14, 18, 24, 36, 48\}$. Chronos-s stands for Chronos-small. The best results are in bold (excluding TimePFN-96).

(Dooley et al. 2023) and (Ansari et al. 2024): Mean, Naive, and Seasonal Naive. These baselines are applied independently to each variate. The Mean baseline forecasts by repeating the mean value of the input variate. The Naive approach forecasts by repeating the last value of the input variate. In the Seasonal Naive method, we assume a periodicity of seven.

Although *TimePFN* is specifically trained for multivariate time-series forecasting, we also evaluated its performance on univariate forecasting (C=1) to demonstrate its robust generalization capabilities. In this context, we compared it with ForecastPFN (Dooley et al. 2023) and Chronos (Ansari et al. 2024), two state-of-the-art univariate zero-shot forecasters. ForecastPFN utilizes an input sequence length of 36, while *TimePFN* operates with a sequence length of 96. To accommodate this discrepancy, we padded the additional 60 time steps with the mean value of the input sequence when running *TimePFN*. These models are evaluated over forecast lengths of 6, 8, 14, 18, 24, 36, and 48. We used the smaller version of Chronos. The full results are detailed in the appendix, while in the main text, we report the averaged MSE and MAE values across these forecast lengths. Furthermore, to showcase the complete forecasting performance of *TimePFN*, we also conducted runs with a non-padded sequence length of 96. We evaluated all results ourselves.

Experimental Setting. When comparing *TimePFN* with the aforementioned baselines, we use the hyperparameters reported in their official codebases. We re-run the experiments with limited budgets and by utilizing the entire training dataset. (Liu et al. 2023) presents the forecasting results for the mentioned transformer-based MTS architectures using the full training dataset. We re-run all the experiments and selected the best results from both our run and their report to ensure that the performance of other architectures is not underreported when we use the entire training set. Our unaltered results are included in the appendix.

In *TimePFN*, we use a single model with fixed hyperparameters that is trained only once on our large-scale multivariate synthetic dataset. In few-shot evaluations, we finetune *TimePFN* with a given data budget, maintaining the same hyperparameters across different datasets. In all evaluations except for univariate cases, we report the forecasting errors for the next 96 time steps, given a multivariate time series (MTS) of sequence length 96. Our implementation details and further experimental settings such as hyperparameters are reported in the Appendix: *Implementation Details*.

Main Results

In MTS forecasting, we compared *TimePFN* with various baselines in zero-shot settings, as well as with different data budgets, and by utilizing the entire dataset. Table 1 presents our results for zero-shot settings, data budgets of 50 and 500, and scenarios using the entire dataset. Our comprehensive results, which also include data budgets of 100 and 1000, can be found in the Appendix under the section *Extended Results*. With a data budget of 50, *TimePFN* outperforms all transformer-based architectures and DLinear. However, with a data budget of 500, it surpasses all baselines except for PatchTST (Nie et al. 2023) in the exchange dataset, closely competing with it. When utilizing the entire dataset, *TimePFN* achieves the best results in four datasets, equaling the performance of PatchTST. Given that we fine-tuned *TimePFN* with fixed hyperparameters across all datasets, and selected the best results from the baselines and the findings reported in (Liu et al. 2023), the performance of *TimePFN* is noteworthy. We observe that *TimePFN* excels in datasets with a greater number of variates and a more multivariate nature, while PatchTST primarily excels in ETT datasets and Exchange. This outcome is anticipated, as *TimePFN* is designed to incorporate channel mixing, whereas PatchTST is designed with channel independence. Indeed, the lower forecasting performance of PatchTST on Traffic dataset supports this hypothesis.

In zero-shot settings, *TimePFN* outperforms all zero-shot baselines except on the Solar-Energy and Exchange datasets, with Solar-Energy being in close proximity. In fact, the Exchange dataset is highly non-trivial, as simply using the last value as the forecast outperforms all baselines, except for PatchTST in the entire budget case. We observed that the Solar-Energy data exhibits sudden spikes e.g. as a function of sun rising or going down. Our model, based on its training data from the LMC-Synth prior, fails to anticipate such sudden spikes. However, these spiky behavior is well within the capabilities of changepoint kernels in Gaussian processes, suggesting a clear path for future improvements.

Univariate Time-Series Forecasting

Although *TimePFN* was specifically trained for MTS forecasting using a synthetic dataset with a channel size of 160,

	Dataset	ECL		Weather		Traffic		Solar		Exchange		ETTh1		ETTh2		ETTm1		ETTm2	
	Models	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	TimePFN	**0.315**	**0.383**	**0.209**	**0.255**	**1.108**	**0.613**	**0.941**	**0.730**	**0.105**	**0.229**	**0.453**	**0.439**	0.328	**0.362**	**0.637**	**0.512**	**0.212**	**0.291**
z.s.	TimePFN-w/o Conv	0.653	0.637	0.221	0.271	1.287	0.757	1.197	0.829	0.111	0.237	0.608	0.517	0.338	0.374	0.771	0.565	0.224	0.307
	PatchTST-PFN	0.470	0.522	0.212	0.262	1.172	0.702	1.014	0.787	0.108	0.231	0.554	0.501	**0.322**	0.366	0.746	0.560	0.215	0.301

Table 3: In TimePFN-w/o-Convolution, we eliminate the convolutional operator that is normally applied to the initial input variates. In PatchTST-PFN, we train a PatchTST model to evaluate the significance of channel-mixing and the appropriateness of our architecture for PFNs. Both the sequence length and the forecasting length are set to 96.

	Dataset	ECL		Weather		Traffic		Solar		Exchange		ETTh1		ETTh2		ETTm1		ETTm2	
	Models	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
z.s.	TimePFN	**0.315**	**0.383**	**0.209**	**0.255**	**1.108**	**0.613**	**0.941**	**0.730**	**0.105**	**0.229**	**0.453**	**0.439**	0.328	**0.362**	**0.637**	**0.512**	**0.212**	**0.291**
	TimePFN-Ind	0.350	0.416	0.214	0.260	1.180	0.651	1.197	0.829	0.113	0.238	0.468	0.447	**0.326**	0.363	0.761	0.542	0.215	0.295

Table 4: TimePFN-ind is the model trained using only independent variates, while the other model is our standard one, which we used throughout the experiments. Both models have a sequence and forecasting length of 96.

we also tested it in a zero-shot scenario for univariate time-series forecasting where $C = 1$. Moreover, we used the sequence length of 36 that ForecastPFN (Dooley et al. 2023) was specifically trained on. To accommodate this sequence length, we padded the remaining $96 - 36 = 60$ sequence lengths with the mean value of the input time-series to mitigate any scaling issues, and named this model configuration *TimePFN*-36. To demonstrate the full performance of our model, we included results for *TimePFN* without padding using a sequence length of 96, referred to as *TimePFN*-96 in Table 2. All other results were reported with a sequence length of 36.

As demonstrated in Table 2, *TimePFN* outperforms models that were specifically trained for univariate time series forecasting, which attests to its robust generalization and zero-shot performance. Our extensive evaluations, which detail the errors for different sequence lengths, can be found in the appendix under the section *Extended Results*.

Ablation Study

Training a single *TimePFN* model requires approximately 10 hours on a single L40S GPU, which limited our capacity for ablation studies. Nevertheless, we conducted two types of key ablations: the first type focused on the architecture, while the second type focused on the synthetic data generation.

Architectural Ablation. In the first part, we first aim to understand the impact of our 1D convolutional operation applied to time-series variates before any patching. To do this, we remove the operation and train a *TimePFN*-convolutionless model, then report the zero-shot results in Table 3. We observe that without the convolutional operation, the zero-shot performance significantly decreases. Additionally, since our architecture differs from that of (Nie et al. 2023) particularly in terms of channel mixing, we trained the PatchTST architecture to assess the impact of channel mixing on zero-shot forecasting performance. As seen in Table 3, both of our ablation experiments supports our model design principles and underscores the usefulness of *TimePFN*'s architecture for synthetic time series learning.

Synthetic Dataset Ablation. To understand whether the synthetic data generation algorithm LMC-Synth gives any benefits over just using the variates generated by KernelSynth (Ansari et al. 2024) independently in each channel, we trained *TimePFN* with using data where each channel is generated independently. This case, as we described previously, corresponds to the case where $C_i(\boldsymbol{t}) = l_i(\boldsymbol{t})$ with number of channels equaling to number of latent functions. We see in Table 4 that using generative coregionalization provides clear benefits.

5 Conclusion

In this work, we demonstrate that with large-scale synthetic training and a suitable architecture for extracting useful time series features, fine-tuning with as few as 50 to 500 examples are sufficient to achieve competitive performance in multivariate time series forecasting. To this end, we present a novel method for generating large-scale synthetic MTS data with realistic intra- and inter-channel dependencies, called LMC-Synth, utilizing Gaussian processes and linear coregionalization model. Simultaneously, we developed an architecture capable of transfer learning, utilizing 1D convolutions applied to time series variates and channel-mixed patching. *TimePFN* exhibits strong zero-shot performance, and although it is explicitly trained for MTS forecasting, it also excels in zero-shot univariate forecasting, demonstrating the flexibility and generality of our framework. To the best of our knowledge, *TimePFN* is the first multivariate time-series PFN. For future work, we aim to improve our synthetic data-generation mechanism to better model sudden changes and multi-scale challenges that are prevalent in many time-series tasks. Additionally, integrating time series PFNs with tabular models presents an intriguing avenue for research. Moreover, we plan to extend our efforts into developing foundation models for multivariate time series.

Acknowledgements

This work was supported in part by the NSF grants CCF-2046816, CCF-2403075, the Office of Naval Research grant N000142412289, and gifts by Open Philanthropy and Google Research.

References

Álvarez, M.; Rosasco, L.; and Lawrence, N. 2012. *Kernels for Vector-Valued Functions: A Review*. Foundations and Trends® in Machine Learning Series. Now Publishers. ISBN 9781601985583.

Ansari, A. F.; Stella, L.; Turkmen, C.; Zhang, X.; Mercado, P.; Shen, H.; Shchur, O.; Rangapuram, S. S.; Pineda Arango, S.; Kapoor, S.; Zschiegner, J.; Maddix, D. C.; Wang, H.; Mahoney, M. W.; Torkkola, K.; Gordon Wilson, A.; Bohlke-Schneider, M.; and Wang, Y. 2024. Chronos: Learning the Language of Time Series. *arXiv preprint arXiv:2403.07815*.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer Normalization. arXiv:1607.06450.

Baevski, A.; Zhou, Y.; Mohamed, A.; and Auli, M. 2020. wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 12449–12460. Curran Associates, Inc.

Bhethanabhotla, S. K.; Swelam, O.; Siems, J.; Salinas, D.; and Hutter, F. 2024. Mamba4Cast: Efficient Zero-Shot Time Series Forecasting with State Space Models. arXiv:2410.09385.

Dooley, S.; Khurana, G. S.; Mohapatra, C.; Naidu, S. V.; and White, C. 2023. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*.

Duvenaud, D.; Lloyd, J.; Grosse, R.; Tenenbaum, J.; and Zoubin, G. 2013. Structure Discovery in Nonparametric Regression through Compositional Kernel Search. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 1166–1174. Atlanta, Georgia, USA: PMLR.

Gu, A.; and Dao, T. 2024. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. arXiv:2312.00752.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Hollmann, N.; Müller, S.; Eggensperger, K.; and Hutter, F. 2023. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. In *The Eleventh International Conference on Learning Representations*.

Hsu, W.-N.; Bolte, B.; Tsai, Y.-H. H.; Lakhotia, K.; Salakhutdinov, R.; and Mohamed, A. 2021. HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29: 3451–3460.

Jin, X.; Park, Y.; Maddix, D.; Wang, H.; and Wang, Y. 2022. Domain Adaptation for Time Series Forecasting via Attention Sharing. In Chaudhuri, K.; Jegelka, S.; Song, L.; Szepesvari, C.; Niu, G.; and Sabato, S., eds., *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, 10280–10297. PMLR.

Journel, A.; and Huijbregts, C. 2003. *Mining Geostatistics*. Blackburn Press. ISBN 9781930665910.

Kim, T.; Kim, J.; Tae, Y.; Park, C.; Choi, J.-H.; and Choo, J. 2021. Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift. In *International Conference on Learning Representations*.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, 95–104. New York, NY, USA: Association for Computing Machinery. ISBN 9781450356572.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Liu, S.; Yu, H.; Liao, C.; Li, J.; Lin, W.; Liu, A. X.; and Dustdar, S. 2022. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In *International Conference on Learning Representations*.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2023. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *arXiv preprint arXiv:2310.06625*.

Müller, S.; Hollmann, N.; Arango, S. P.; Grabocka, J.; and Hutter, F. 2022. Transformers Can Do Bayesian Inference. In *International Conference on Learning Representations*.

Nie, Y.; H. Nguyen, N.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Oreshkin, B. N.; Carpov, D.; Chapados, N.; and Bengio, Y. 2021. Meta-Learning Framework with Applications to Zero-Shot Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10): 9242–9250.

Orozco, B. P.; and Roberts, S. J. 2020. Zero-shot and few-shot time series forecasting with ordinal regression recurrent neural networks. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020*, 503–508.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Verdenius, S.; Zerio, A.; and Wang, R. L. 2024. LaT-PFN: A Joint Embedding Predictive Architecture for In-context Time-series Forecasting. *arXiv preprint arXiv:2405.10093*.

Wang, Z.; and Liu, J.-C. 2019. Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training. arXiv:1908.11415.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition Transformers with Auto-Correlation

for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting. In *International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, 11106–11115. AAAI Press.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*.