FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

TempoPFN-morph: generating data for a foundational forecasting model using time series morphing

Miguel Ângelo Aguiar e Nogueira

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Carlos Soares

Second Supervisor: Dr. Moisés Santos

July 31, 2026

TempoPFN-morph: generating data for a foundational forecasting model using time series morphing

Miguel Ângelo Aguiar e Nogueira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Carla Gonçalves

Examiner: Prof. Luís Miguel Matos

Supervisor: Prof. Carlos Soares

July 31, 2026

Resumo

Modelos Fundacionais para previsão de séries temporais são cada vez mais treinados com dados sintéticos produzidos por um conjunto de geradores, sendo a qualidade de tais modelos limitada pela riqueza desta prior sintética. Esta dissertação investiga se o *dataset morphing*, a interpolação entre séries temporais, produz uma melhor prior quando usado para construir os dados de treino, gerando séries que interpolam entre geradores distintos e, assim, cobrem as regiões de comportamento que se situam entre eles. Testamos o *morphing* de forma isolada, como substituição dos dados dos geradores e não como complemento, através de uma comparação controlada, duas instâncias do modelo TempoPFN, idênticas em arquitetura, otimização e orçamento de treino, uma treinada com os dados não modificados dos geradores e a outra com os dados resultantes do *morphing* de séries temporais entre geradores. Avaliado de forma *zero-shot* no *benchmark* GIFT-Eval, o modelo treinado com dados *morphed* foi substancialmente pior do que o modelo treinado com dados gerados em quase todos os *datasets* e em ambas as métricas, probabilística e de previsão pontual, sendo a degradação mais acentuada em horizontes mais longos. No âmbito deste estudo, utilizando apenas uma única configuração de *morphing* e um orçamento de treino reduzido, construir a prior a partir de séries *morphed* desta forma não o tornou melhor, mas sim pior.

Palavras-chave: Inteligência Artifical • Modelos Fundacionais • Dados Sintéticos • Morphing

Abstract

Forecasting foundation models are increasingly trained on synthetic data produced by a suite of generators, each contributing a different class of temporal behaviour, and the quality of such a model is bounded by the richness of this synthetic prior. This thesis investigates whether dataset morphing, the interpolation between time series, yields a better prior when used to build the training data, by generating series that interpolate between distinct generators and so cover the regions of behaviour that lie between them. We test morphing in isolation, as a replacement for the generator data rather than an addition to it, through a controlled comparison, two instances of the TempoPFN model, identical in architecture, optimisation, and training budget, one trained on the unmodified generator output and the other on a dataset of cross-generator morphed series of equivalent size. Evaluated zero-shot on the GIFT-Eval benchmark, the morphed model was substantially worse than the clean model across almost all datasets and on both probabilistic and point-forecast metrics, with the degradation most pronounced at longer horizons. Within the scope of the study, a single morphing configuration and a reduced training budget, building the prior from morphed series in this way did not make it better, but worse.

Palavras-chave: Artificial Intelligence • Foundation Models • Synthetic Data • Morphing

The United Nations Sustainable Development Goals

The United Nations Sustainable Development Goalss (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals (https://sdgs.un.org/goals) to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation contributes most directly to SDG 9 (Industry, Innovation and Infrastructure). Forecasting foundation models promise to make accurate time-series prediction available in settings where task-specific models cannot be trained, but their reliability depends entirely on the synthetic data used to pre-train them. By rigorously testing whether a candidate data-generation strategy, dataset morphing, improves or degrades this training prior, this work contributes to the methodological foundations on which such models are built. This is applied machine-learning research that strengthens the scientific and technological basis of an emerging class of AI infrastructure, and its negative result is itself a contribution: it narrows the space of viable approaches and directs future research away from an intervention that, at this scale, does not help.

The work also relates, more modestly, to SDG 7 (Affordable and Clean Energy). Several of the real-world benchmarks used to evaluate the models in this dissertation, such as electricity demand and solar generation, are drawn from the energy domain, where accurate short- and long-term forecasting supports grid balancing and the integration of renewable sources. This contribution is indirect: the dissertation evaluates general-purpose forecasting ability on these datasets as part of a broader benchmark, rather than developing or deploying a model for energy applications, and any benefit to energy forecasting is a plausible downstream implication rather than an outcome measured in this work.

The specific Sustainable Development Goals mentioned have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation.

SDG 7 Ensure access to affordable, reliable, sustainable and modern energy for all.

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.5	A controlled empirical study of a candidate synthetic-data strategy for forecasting foundation models, clarifying which approaches to prior construction do and do not improve zero-shot performance.	CRPS and MASE on the GIFT-Eval benchmark, compared across training data configurations.
	9.b	Builds on and extends an existing research line (dataset morphing) developed within the host research group, applying it to a new setting and reporting the resulting methodological knowledge openly.	Reproducible experimental pipeline and reported findings, including negative results.
7	7.3	Forecasting foundation models evaluated in part on energy-domain benchmarks (electricity, solar) may, if improved by future work, support more efficient demand forecasting and renewable integration.	Forecasting accuracy (CRPS, MASE) on electricity- and solar-generation datasets within the benchmark.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Professor Carlos Soares and Professor Moisés Santos, for their guidance, insight, and support throughout this work. Their expertise shaped the direction of this thesis, and their feedback was invaluable at every stage. I am especially grateful for their patience and encouragement, and for the time they dedicated to helping me see this project through.

I would also like to thank Omar Mohammed without whom the experiments in this work would not have been possible. This work was supported by computational resources from FCT I.P.

To my family and friends, thank you for being there throughout this journey. Your support, your patience, and your belief in me meant more than I can express, and your encouragement carried me through the difficult moments. I am deeply grateful for everything you have done for me.

Finally, I would like to thank Faculdade de Engenharia da Universide do Porto, the faculty members and colleagues I had the privilege of meeting there, for the years of learning and growth, and for the conversations and shared knowledge that enriched both this work and my time as a student.

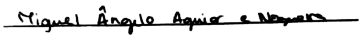
Miguel Nogueira

Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.

I also declare that I have used Artificial Intelligence tools to aid in the writing of this dissertation, mostly for clarity, grammar and phrasing. All the results, analysis and conclusions were of my authorship and I'm fully responsible for the content of this dissertation.

Miguel Ângelo Aguiar e Nogueira



Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Problem . . . 2
- 1.4 Research Questions . . . 2

2 State of the art **3**
- 2.1 Prior-Data Fitted Network (PFN)-based learning methods . . . 3
  - 2.1.1 Prior-Data Fitted Networks (PFNs) . . . 3
- 2.2 Time Series Forecasting . . . 4
  - 2.2.1 Mathematical Definition . . . 4
  - 2.2.2 Evaluation . . . 5
  - 2.2.3 Traditional Algortithms . . . 7
- 2.3 The TempoPFN model . . . 8
  - 2.3.1 Input representation . . . 8
  - 2.3.2 Recurrence . . . 8
  - 2.3.3 State Weaving . . . 9
- 2.4 Synthetic Data Generation . . . 9
  - 2.4.1 Why train on synthetic data? . . . 9
  - 2.4.2 TempoPFN's generators . . . 10
  - 2.4.3 Dataset Morphing . . . 14

3 Methodology **16**
- 3.1 Algorithm . . . 16
  - 3.1.1 Standardisation . . . 17
  - 3.1.2 Morphing . . . 17
  - 3.1.3 Restandardization . . . 18
  - 3.1.4 Magnitude Reintroduction . . . 18
- 3.2 Cross-Generator Morphing . . . 18

4 Empirical Study **20**
- 4.1 Experimental Setup . . . 20
  - 4.1.1 Controlled Experiment . . . 20
  - 4.1.2 Model and Training Configuration . . . 20
  - 4.1.3 Training Dataset . . . 20
  - 4.1.4 Evaluation Protocol . . . 22
- 4.2 Results . . . 23
- 4.3 Discussion . . . 23

4.3.1 Morphing creates harder to model time series . . . 25
4.3.2 Granularity . . . 25
4.3.3 Time Horizon . . . 26
4.3.4 Fairness in the training dataset . . . 26
4.3.5 Limitations . . . 27

5 Conclusion **28**

References **29**

A Literature Review **31**
A.1 Methodology . . . 31
A.2 Queries . . . 31
A.2.1 Query Formulation . . . 31
A.2.2 Query Justification . . . 32
A.2.3 Search Query Results . . . 32
A.3 Filtering and Selection Process . . . 32

B My approach (schedule) **33**
B.1 Methodology . . . 33
B.1.1 Activity Cards . . . 33
B.1.2 Task Dependencies . . . 36

List of Figures

2.1 Time series generated by KernelSynth . . . 10
2.2 Time series generated by GP . . . 10
2.3 Time series generated by CauKer . . . 11
2.4 Time series generated by ForecastPFN . . . 11
2.5 Time series generated by Sine wave . . . 12
2.6 Time series generated by Sawtooth . . . 12
2.7 Time series generated by Step . . . 13
2.8 Time series generated by Anomaly . . . 13
2.9 Time series generated by Spike . . . 13
2.10 Time series generated by Audio Stochastic Rhythm . . . 13
2.11 Time series generated by Audio Financial Volatility . . . 13
2.12 Time series generated by Audio Network Topology . . . 13
2.13 Time series generated by Audio Multi Scale Fractal . . . 14
2.14 Time series generated by OU process . . . 14

3.1 Simplified illustration of an uncovered gap . . . 17

4.1 Granularity illustration. Each x represents a morphing of the source and target. Granularity indicates how many to output. . . . 22
4.2 Change in Continuous Ranked Probability Score (CRPS) under morphing for each of the 97 GIFT-Eval evaluations, computed as the morphed model CRPS minus the generated model CRPS and ordered from largest improvement to largest degradation. Green bars (negative) indicate datasets where morphing reduced error, red bars (positive) indicate datasets where it increased error. The predominance of red bars, and their concentration among the longer-horizon evaluations, illustrates the consistent and horizon-dependent degradation under morphing. . . . 24
4.3 Per-dataset comparison of the generated and morphed arms across the 97 GIFT-Eval evaluations, for CRPS (left) and Mean Absolute Scaled Error (MASE) (right). Each point is one dataset–term evaluation, plotting the morphed model error against the generated model errorm, the dashed line marks equal performance. Points above the line indicate higher error for the morphed model. The large majority of points lie above the diagonal on both metrics, showing that morphing degraded performance across most datasets. . . . 25

B.1 Gantt Chart of implementation timeline . . . 36

List of Tables

4.1 Model and training configuration, identical across both arms. 21
4.2 Aggregate comparison of the generated and morphed arms across the 97 GIFT-Eval evaluations. Lower is better for both metrics. 23

A.1 Search Query Results 32

List of Acronyms

AI Artificial Intelligence

SDG Sustainable Development Goals

PFN Prior-Data Fitted Network

PFNs Prior-Data Fitted Networks

GPT Generative Pre-trained Transformer

GP Gaussian Process

GPs Gaussian Processes

PPD Posterior Predictive Distribution

KL Kullback-Leibler

i.i.d independent and identically distributed

MAE Mean Absolute Error

RMSE Root Mean Squared Error

MAPE Mean Absolute Percentage Error

sMAPE Symmetric Mean Absolute Percentage Error

MASE Mean Absolute Scaled Error

CV Cross Validation

ARIMA Autoregressive Integrated Moving Average

GARCH Generalized Autoregressive Conditional Heteroskedasticity

SMOTE Synthetic Minortiy Over-sampling technique

DSP Digital Signal Processing

SDE Stochastic Differential Equation

OU Ornstein–Uhlenbeck

LSTM Long Short-Term Memory

DeepAR Deep Autoregressive

CRPS Continuous Ranked Probability Score

SDG Sustainable Development Goals

Chapter 1

Introduction

1.1 Context

The current landscape of Artificial Intelligence (AI) is increasingly defined by the rise of foundation models. Architectures such as Generative Pre-trained Transformer (GPT) for natural language processing and DALL-E for computer vision have demonstrated that pre-training on massive, diverse datasets allows a single model to generalize across a vast array of tasks without the need for traditional retraining.

While this success was initially concentrated in natural language processing and computer vision, it has recently expanded to structured data via Prior-Data Fitted Networks (PFNs). Unlike traditional models that require gradient-based optimization for every new dataset, PFNs are pre-trained offline to approximate Bayesian inference, allowing for instantaneous predictions on unseen data. In the domain of time series forecasting, this approach has led to the development of several emerging architectures, such as timePFN [14], forecastPFN [8], and tempoPFN [10]. These models demonstrate that diverse neural architectures can effectively learn temporal patterns from synthetic priors to outperform or match classical state-of-the-art methods in both zero-shot and few-shot settings [14].

1.2 Motivation

The primary motivation for this research comes from the realization that for any pre-trained forecasting model, the "scarcity of publicly available time series datasets, both in quantity and quality, is arguably more critical than the modeling framework"[2]. Because PFNs rely entirely on the synthetic data they observe during the pre-training phase, the diversity and realism of these priors determine the model's real world utility. Recent literature emphasizes that the heterogeneous nature of time series data requires far richer datasets than what is currently provided by standard generators [14].

Consequently, the goal of this project is to investigate and analyze new ways of synthetic data generation that can more accurately represent the complexity of real-world time series, moving beyond the limitations of existing models.

1.3 Problem

Despite the initial success of time series PFNs, there remains a significant gap between synthetic training environments and real-world deployment [2]. Current models typically rely on static generation techniques, such as Gaussian Process (GP) kernels, which often fail to capture the structural breaks and complex dependencies prevalent in real-world tasks.

Consequently, these models are likely to underperform in real-world applications where the test data is out-of-distribution relative to the simple synthetic priors used during training [14]. A core limitation in current research is the absence of mechanisms for gradual variation of data characteristics, which prevents models from learning the fluid transitions between different types of temporal behaviors [13]. This research addresses this gap by testing whether a dynamic approach, specifically data morphing, can cover the diversity of real-world time series data more effectively than the static, heuristic-based generation methods currently used in the literature.

1.4 Research Questions

- **RQ1**: How can the tsMorph framework [13] be adapted to generate a continuous and diverse stream of synthetic priors for the pre-training of a Prior-Data Fitted Network (PFN)?
- **RQ2**: Does a PFN model trained using morphed data only achieve better, worse or similar forecasting accuracy on real-world benchmarks compared to models trained on more standard, static synthetic priors like those used in [14] or [10]?
- **RQ3**: To what extent does the diversity introduced by morphed data improve the model's robustness to edge cases and structural breaks compared to traditional generation methods?

Chapter 2

State of the art

2.1 PFN-based learning methods

2.1.1 PFNs

PFNs represent a novel approach to machine learning that uses transformer architectures to perform approximate Bayesian inference through in-context learning [11]. Rather than fitting model parameters to a specific dataset, PFNs are pre-trained on synthetic data, then deployed without further training to make predictions on new datasets in a single forward pass.

2.1.1.1 Bayesian Framework

In Bayesian inference, given training data $D_n = \{(X_i, Y_i)\}_{i=1}^n$, we predict labels for new features by computing the Posterior Predictive Distribution (PPD):

$$\pi(y \mid x, D_n) = \int p(y \mid x, P)\, d\Pi(P \mid D_n) \tag{2.1}$$

where Π is a prior over possible data-generating distributions [12]. This PPD represents our uncertainty about the label y at feature location x, averaging over all plausible models weighted by their posterior probability given the observed data.

2.1.1.2 How PFNs Work

Training Phase

As shown in [11] and [12], instead of computing the PPD analytically, PFNs learn to approximate the entire family of posteriors offline. The network q_θ is trained to maximize the expected log-likelihood over the prior:

$$\theta^* = \arg\max_\theta \mathbf{E}_{N \sim \Pi_N} \mathbf{E}_{D_N \sim \Pi} \left[\log q_\theta(Y \mid X, D_N) \right] \tag{2.2}$$

The key insight is that minimizing this objective is equivalent to minimizing the expected Kullback-Leibler (KL) divergence between the network's predictions and the true PPD π:

$$\mathbf{E}_{D,x}\left[\mathrm{KL}(\pi(\cdot \mid x, D) \,\|\, q_\theta(\cdot \mid x, D))\right] \tag{2.3}$$

Inference Phase

Given a new dataset D_{train} and a test point x_{test}, the PFN approximates the PPD in a single forward pass:

$$q_{\theta^*}(y_{test} \mid x_{test}, D_{train}) \approx \pi(y_{test} \mid x_{test}, D_{train}) \tag{2.4}$$

2.1.1.3 Why PFNs Can Learn In-Context

A central question addressed in [12] is how a frozen network improves its predictions as the inference dataset size n increases. This is explained through two statistical mechanisms:

- **Vanishing Variance:** The Transformer's multi-head attention mechanism distributes weight across training samples. If each sample receives weight $a_j \approx 1/n$, the individual influence of any single data point diminishes as $n \to \infty$. This ensures that the predictor's variance decays at a rate of $\mathbf{O}(n^{-1/2})$, leading to increasingly stable predictions.
- **Implicit Model Selection (Bias Reduction):** Multiple attention heads extract different patterns from the data. At inference, the network adapts which heads dominate based on the specific structure of D_{train}. While this allows for complex pattern matching, standard Transformers may lack "locality", the ability to focus only on samples in the immediate neighborhood of x_{test}, meaning the bias might plateau rather than vanish entirely for extremely large datasets.

Finally, the distribution Π_N over training set sizes acts as a complexity regularizer. By training on a range of N, the network learns a family of predictors that automatically scale their complexity: they remain smooth and close to the prior for small n, but capture high-frequency features as more data points are provided in the context window.

2.2 Time Series Forecasting

2.2.1 Mathematical Definition

We consider the multivariate time series forecasting problem.

A time series can be represented as an ordered sequence of observations indexed by time:

$$\{x_t\}_{t \in T} = x_1, x_2, x_3, \ldots, x_n \tag{2.5}$$

where x_t in R^d is the value of the series at time t, $T = \{t_1, t_2, \ldots, t_n\}$ is an ordered set of time points, n is the number of observations, and d is the number of features (dimensions) in the multivariate series.

To forecast a future horizon of length H, we consider an input window of the past L observations:

$$X_{t-L+1:t} = (x_{t-L+1}, x_{t-L+2}, \ldots, x_t) \in R^{L \times d} \tag{2.6}$$

The objective is to predict:

$$Y_{t+1:t+H} = (\hat{x}_{t+1}, \ldots, \hat{x}_{t+H}) \in R^{H \times d} \tag{2.7}$$

The forecasting function is defined as:

$$f_\theta : R^{L \times d} \to R^{H \times d} \tag{2.8}$$

where θ are the model parameters.

The optimization objective is to minimize an expected loss function:

$$\theta^\star = \arg\min_\theta \; E\Big[\mathscr{L}\big(Y_{t+1:t+H}, f_\theta(X_{t-L+1:t})\big)\Big] \tag{2.9}$$

2.2.2 Evaluation

Evaluating time series forecasting models is particularly challenging because observations are not independent and identically distributed (i.i.d) [4]. The temporal ordering and dependencies among observations violate the independence assumptions of many standard evaluation methods used in traditional machine learning.

2.2.2.1 Statistics for Evaluation Metrics

A forecast error e_{T+h} is the difference between the observed value and its forecast 2.10:

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T} \tag{2.10}$$

where y_{T+h} is the $(T+h)$-th observation and $\hat{y}_{T+h|T}$ is the forecast based on data up to time T [9].

Scale-Dependent Metrics

These are expressed in the same units as the data. They are ideal for comparing different models on a single time series [9].

- **Mean Absolute Error (MAE):**

$$MAE = \text{mean}(|e_t|) \tag{2.11}$$

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{\text{mean}(e_t^2)} \tag{2.12}$$

Note: RMSE penalizes larger errors more heavily than MAE.

Percentage-Based Metrics

These are used to compare performance across different datasets, but they become infinite or undefined if $y_t = 0$ [9].

- **Mean Absolute Percentage Error (MAPE):**

$$MAPE = \text{mean}(|100e_t/y_t|) \tag{2.13}$$

- **Symmetric Mean Absolute Percentage Error (sMAPE):**
  Introduced to address the asymmetry of MAPE, though it can still be unstable [9]

$$sMAPE = \text{mean}\left(\frac{200|y_t - \hat{y}_t|}{y_t + \hat{y}_t}\right) \tag{2.14}$$

Scaled Metrics

- **Mean Absolute Scaled Error (MASE):**

$$q_j = \frac{e_j}{\frac{1}{T-1}\sum_{t=2}^{T}|y_t - y_{t-1}|} \tag{2.15}$$

$$MASE = \text{mean}(|q_j|) \tag{2.16}$$

A $MASE < 1$ indicates the model is better than the average naive forecast.

2.2.2.2 Evaluation Procedures

The resampling strategy determines how data is partitioned to simulate real-world forecasting scenarios.

Holdout (Out-of-Sample Evaluation)

The most common approach in forecasting is the holdout procedure, also known as out-of-sample evaluation [4]. The time series is split chronologically into two contiguous parts:

- **Training set:** The initial portion of the series (e.g., the first 70%) used for model estimation.
- **Test set:** The final portion (e.g., the last 30%) representing future observations.

This approach respects temporal ordering and mimics real-world deployment where models predict unseen future values.

- **Repeated Holdout:** An extension that repeats the holdout process multiple times using randomized testing periods. For each iteration, a point is chosen randomly within the series; a fixed percentage of observations preceding it are used for training, while the subsequent observations form the test set.

Prequential Evaluation (Time Series Cross-Validation)

Prequential evaluation follows a sequence where an observation (or block) is first used to test the model and subsequently used to update it [4].

Growing (Expanding) Window is the most common implementation in practice:

1. The series is split into K contiguous blocks.
2. **Iteration 1:** Train on block 1, test on block 2.
3. **Iteration 2:** Train on blocks 1 and 2, test on block 3.
4. **Iteration k:** Train on blocks 1 through k, test on block $k+1$.

Cross-Validation Adaptations for Model Selection

Standard K-fold Cross Validation (CV) is often biased for time series because it ignores the temporal order of data. They mention specialized adaptations designed to mitigate this issue [4]:

- **Blocked CV:** This method partitions the data into K contiguous blocks without shuffling. It is frequently recommended for stationary series because it preserves the local temporal structure of the data better than random shuffling.
- **hv-Blocked CV:** This approach enhances Blocked CV by introducing a gap (of size p) between the training and test sets. By removing observations that are highly correlated with the test set, this gap ensures that the evaluation is performed on data that is more genuinely "independent" from the training set.

2.2.3 Traditional Algortithms

While modern architectures rely on deep learning and massive datasets, the foundations of time-series forecasting are built on statistical methods like Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH). ARIMA focuses on modeling the linear relationships within a series by combining its own past values and past forecast errors, making it highly effective for stable, univariate data with clear trends. GARCH extends this by modeling "volatility clustering," specifically addressing the tendency of variance in financial data to change over time. Because these models are mathematically rigorous and require very little data to produce a forecast, they remain a standard baseline in many industries where interpretability is more important than predictive power.

However, these traditional methods face significant limitations compared to the newer architectures discussed bellow. First, they are primarily linear and struggle to capture the complex, non-linear patterns, such as irregular shocks or shifting seasonalities—that recent models handle with ease. Second, traditional models like ARIMA are generally univariate and lack a natural mechanism to incorporate high-dimensional cross-channel correlations. Finally, statistical models must

be "fit" to every new dataset from scratch, whereas foundation models offer zero-shot capabilities. These modern models use their pre-trained "knowledge" to forecast immediately, providing a level of generalization across diverse domains that traditional statistical methods cannot achieve.

2.3 The TempoPFN model

Whereas the classical methods above fit each series individually, TempoPFN [10] is a univariate, zero-shot forecasting model. Zero-shot means the model is trained once and can then forecast time series it has never seen during training, producing predictions directly from a new series' history without any per-series fitting or fine-tuning, in contrast to classical models, which must be re-trained for every new dataset. TempoPFN achieves this by being pre-trained on synthetic data and forecasting unseen series in context. It uses the PFN framework (sec.2.1.1), using the synthetic temporal prior described later in this chapter.

2.3.1 Input representation

The model receives a single token sequence that concatenates the observed history with the future positions to be predicted. Each time step is represented by two kinds of information: a value embedding and a time-feature embedding. For historical steps, the observed value is passed through a linear projection and added to an embedding of the step's calendar and index features. Missing values are not imputed but instead replaced by a dedicated learnable embedding, allowing the model to treat absence as a distinct signal rather than an arbitrary number. Future steps, whose values are unknown, are represented by their time-feature embedding alone. Because history and future occupy the same sequence, information can flow between them, which is what allows the model to produce a coherent forecast for all horizon positions at once.

2.3.2 Recurrence

The core of the model is a stack of encoder layers based on the GatedDeltaProduct recurrence. Each layer maintains a hidden state that is updated as the sequence is processed, following the general form of a linear recurrence,

$$H_t = A_t H_{t-1} + B_t \tag{2.17}$$

where A_t governs how past information is carried forward and B_t injects the current input. What distinguishes this architecture from simpler linear recurrent models, such as diagonal state-space models, is the structure of the transition A_t. Rather than scaling each hidden dimension independently, GatedDeltaProduct constructs it as a product of near-orthogonal rank-one updates,

$$A_t = g_t \prod_{j=1}^{n_h} \left(I - \beta_{t,j} k_{t,j} k_{t,j}^\top \right) \tag{2.18}$$

where g_t is a forget gate and each factor is a reflection determined by a key vector $k_{t,j}$ and a step size $\beta_{t,j}$ predicted from the input. Because each factor preserves the geometry of the state rather than simply shrinking it, the recurrence can mix information across hidden dimensions and retain trend and level information over long contexts.

The practical advantage of this design is that the recurrence, despite being sequential in form, can be evaluated in parallel across the sequence length using a prefix-scan formulation. This avoids the step-by-step processing that limits the scalability of conventional non-linear RNNs, as well as the quadratic cost in sequence length incurred by attention-based models.

2.3.3 State Weaving

In autoregressive language modelling, a recurrence must be strictly causal, a prediction at one position may depend only on earlier positions. Forecasting across a fixed horizon does not have this constraint, since the entire future window is predicted jointly. TempoPFN does this through state weaving, the final hidden state produced by one layer is passed forward to initialise the next layer's recurrence, rather than each layer beginning from a blank state. This lets deeper layers build on the aggregated temporal representation of earlier ones and allows information to propagate in both directions across the sequence, so that predictions for the forecast window draw on the full context rather than a strictly left-to-right view.

2.4 Synthetic Data Generation

2.4.1 Why train on synthetic data?

Modern forecasting foundation models are data-hungry: to forecast well across unseen domains, a model must be exposed during training to a correspondingly wide range of temporal behaviours. Relying on real-world data to provide this coverage raises several problems. Real time-series datasets are finite, unevenly distributed across domains, and often subject to privacy or licensing restrictions. Synthetic data solves these issues, it can be generated in arbitrary quantity and contains no private information. Moreover, synthetic generation offers control over the distribution. Rather than hoping a collected dataset happens to contain enough trends, enough seasonality, enough abrupt regime changes, one can deliberately generate examples of each.

The idea of synthesising training data to improve coverage of the data distribution is well established. An early example is Synthetic Minortiy Over-sampling technique (SMOTE) [5] , proposed to address class imbalance. Rather than simply duplicating scarce minority-class examples, SMOTE generates new ones by interpolation: for a minority sample, it selects one of its nearest neighbours, and creates a synthetic example at a random point along the line segment joining the two. The motivation is that the region between observed samples is itself plausibly valid, and populating it gives the learner broader, less fragmented coverage of the minority class rather than a few isolated points. More broadly, data augmentation in computer vision (cropping, rotating, colour-shifting images) and in time series (jittering, warping, mixing) rests on the same premise,

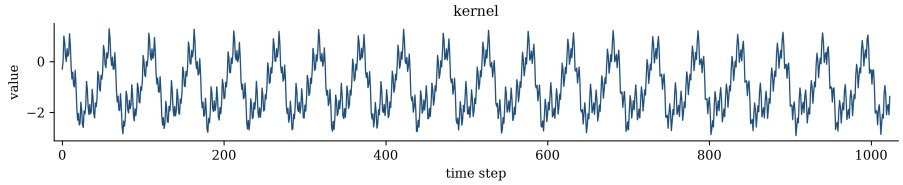


Figure 2.1: Time series generated by KernelSynth

that systematically generated variations of the data expand the effective training distribution and improve generalisation.

2.4.2 TempoPFN's generators

TempoPFN's [10] training prior is produced by a set of synthetic generators, each contributing a different class of temporal behaviour. The design follows a principle of structural coverage, rather than collecting many similar series, the generators are chosen so that, between them, they span the fundamental dynamical properties a real-world series might exhibit, smooth trends, stochastic volatility, asymmetric periodicity, and abrupt discontinuities [10]. We describe each generator below. Even though the original pipeline additionally applies several augmentations to these base series, we use only the base generators and do not apply this augmentation stage.

2.4.2.1 Gaussian Process Generators

Three generators draw on Gaussian processes to produce smooth, correlated trajectories. KernelSynth, introduced in [2], samples from a GP whose covariance is a composite of randomly combined base kernels yielding smooth but varied series (Figure 2.1). The broader Gaussian Process generator extends this with a larger kernel bank and occasional injected periodic spikes, producing both stationary and non-stationary patterns that mix smooth and abrupt dynamics (Figure 2.2). CauKer [15] embeds Gaussian Processes (GPs) within a structural causal model, each node is a GP, and a random directed acyclic graph applies nonlinear transformations between nodes, introducing causal dependencies and non-Gaussian structure. Together these supply the smooth, latent-trend behaviour that dominates many real series (Figure 2.3).

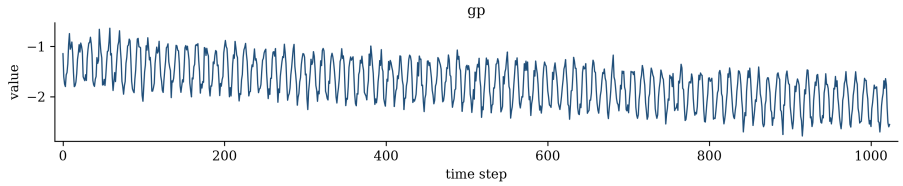


Figure 2.2: Time series generated by GP

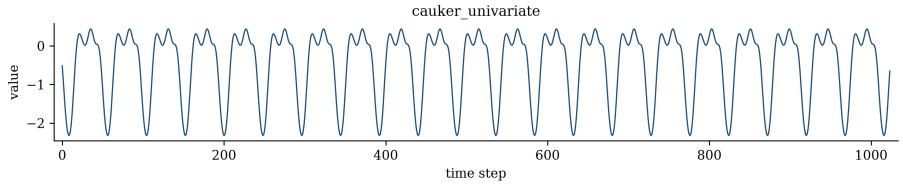


Figure 2.3: Time series generated by CauKer

2.4.2.2 ForecastPFN

Introduced in [8], this generator composes series from explicit trend and seasonality components, perturbed with Weibull-distributed noise and augmentations such as time-warping and spike injection. It contributes clean, structured trend–seasonality patterns (Figure 2.4).

2.4.2.3 Sine wave

The sine wave generator produces oscillatory series by summing one to three sinusoids whose amplitudes and frequencies are themselves slowly modulated, then adding a trend and noise (Figure 2.5).

2.4.2.4 Sawtooth

Where sinusoids are time-symmetric, the sawtooth generator targets temporal asymmetry, a gradual ramp followed by a sharp drop (or the reverse), with small added trend and seasonal terms (Figure 2.6).

2.4.2.5 Step

The step generator builds piecewise-constant series by concatenating subseries with configurable changepoints, step sizes, and drift, optionally smoothed at the transitions and overlaid with noise and seasonality (Figure 2.7).

2.4.2.6 Anomaly and Spike

Two related generators place sharp events on otherwise flat baselines. Anomaly produces a baseline with periodic spike anomalies of varying magnitude regime and timing (Figure 2.8). Spikes

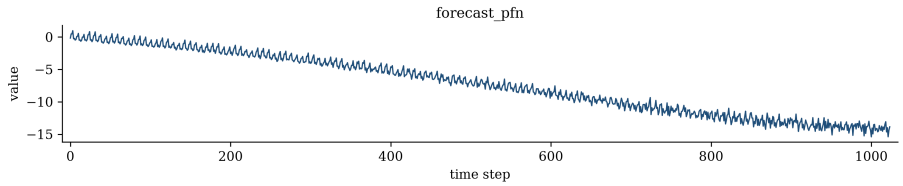


Figure 2.4: Time series generated by ForecastPFN

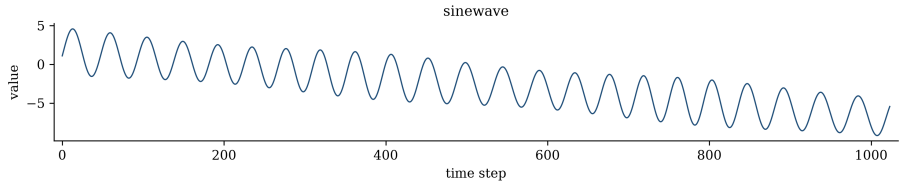


Figure 2.5: Time series generated by Sine wave

makes the event itself the signal, by placing spikes on flat baselines with configurable shapes (Figure 2.9).

2.4.2.7 Audio Inspired Generators

A family of four generators is built using procedural audio-synthesis techniques (oscillators and modulators rendered with a Digital Signal Processing (DSP) library, then resampled to the series length), producing highly complex, layered dynamics. Stochastic Rhythm creates polyrhythmic, event-driven patterns from layered triggers (Figure 2.10), Financial Volatility mimics market dynamics with a trend, volatility clustering (Figure 2.11), and sudden shocks, Network Topology simulates traffic with baseline flow, bursts, congestion dips, and sharp spikes (Figure 2.12) and Multi-Scale Fractal produces self-similar structure across time scales via band-pass-filtered noise (Figure 2.13).

2.4.2.8 Stochastic Differential Equations: Ornstein–Uhlenbeck

This generator models continuous-time stochastic dynamics through a regime-switching Ornstein–Uhlenbeck (OU) process. The state evolves according to the Stochastic Differential Equation (SDE) eq.2.19

$$dy_t = \theta(t, r_t)\big(\mu(t, r_t) - y_t\big)\,dt + \sigma(t, r_t)\,dW_t \tag{2.19}$$

where θ is the mean-reversion speed, μ the time-varying mean,σ the volatility and W_t a Brownian motion. A latent regime r_t switches between parameter sets, allowing abrupt changes in dynamics, while the mean and volatility drift smoothly through trend and seasonal terms. This produces mean-reverting series with state-dependent, clustered volatility — behaviour that simple additive noise cannot capture (Figure 2.14).

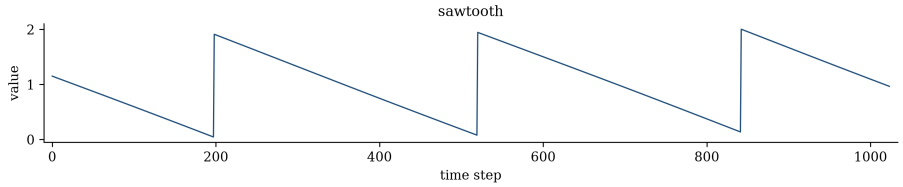


Figure 2.6: Time series generated by Sawtooth

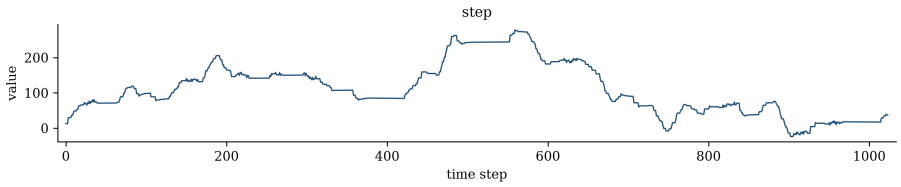


Figure 2.7: Time series generated by Step

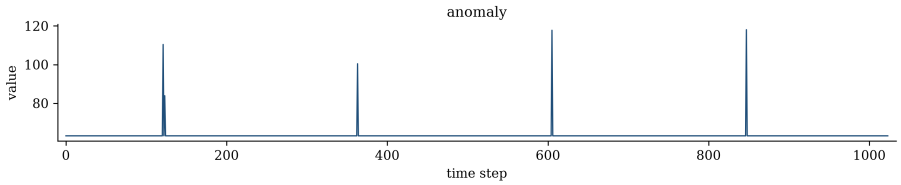


Figure 2.8: Time series generated by Anomaly

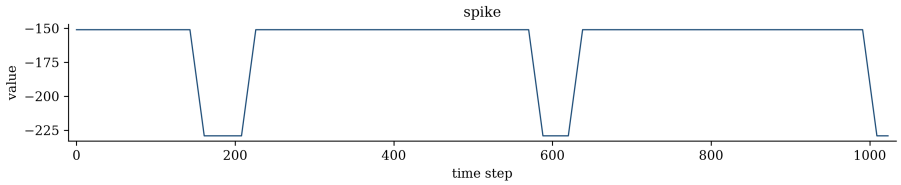


Figure 2.9: Time series generated by Spike

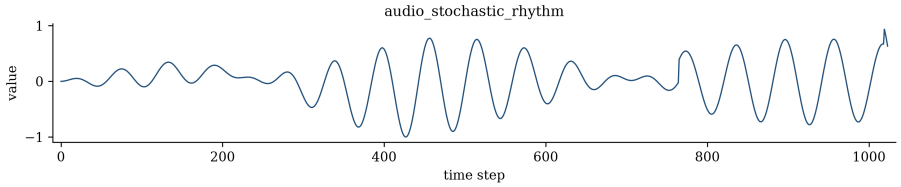


Figure 2.10: Time series generated by Audio Stochastic Rhythm

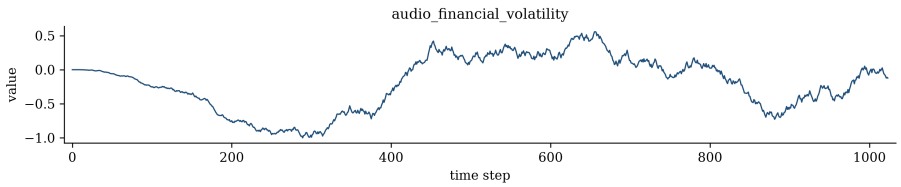


Figure 2.11: Time series generated by Audio Financial Volatility

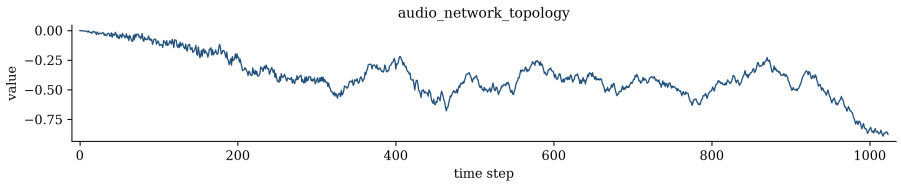


Figure 2.12: Time series generated by Audio Network Topology

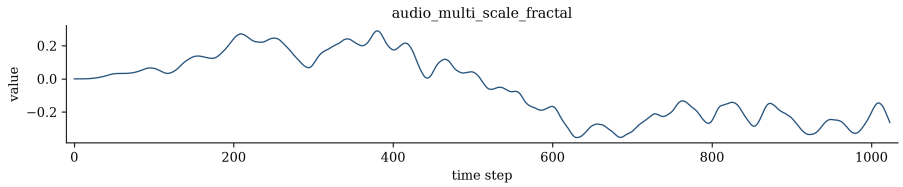


Figure 2.13: Time series generated by Audio Multi Scale Fractal

2.4.3 Dataset Morphing

The synthetic generators described above each produce series of a single characteristic type. A complementary question is what lies between them, given two time series with different dynamics, can one construct a controlled sequence of intermediate series that transitions smoothly from one to the other? This is the idea of dataset morphing.

Dataset morphing was introduced by [6] as a method for understanding algorithm behaviour. Rather than evaluating a model on a fixed collection of datasets, morphing constructs a sequence of semi-synthetic datasets that gradually transform a source dataset into a target dataset, by repeatedly applying a transformation τ. Observing how an algorithm's performance changes along this sequence reveals how its behaviour depends on the changing characteristics of the data, information that a few isolated benchmarks cannot provide.

This idea was adapted to time series with tsMorph. Given a source series $Y^{(source)}$ and a target series $Y^{(target)}$ of equal length, tsMorph generates intermediate series by linear interpolation [13]:

$$\tau(i) = \alpha_i Y^{(target)} + (1-\alpha_i) Y^{(source)}, \qquad \alpha_i = \frac{i}{n-1} \tag{2.20}$$

where i indexes the n steps of the morphing process and the coefficient α moves from 0 to 1. At α = 0 the result is the source series, at α= 1 it is the target and intermediate values produce series whose characteristics lie progressively further from the source and closer to the target. The transformation is simple and computationally cheap, and because it is model-agnostic it can be applied ahead of any forecasting algorithm. In tsMorph this construction is used as a diagnostic tool, by morphing between series and tracking how forecasting error and time-series meta-features coevolve, the authors characterise the conditions under which algorithms such as Long Short-Term Memory (LSTM) and Deep Autoregressive (DeepAR) perform well or poorly. The same morphing primitive has since been extended beyond forecasting. tsMIST [3] applies it to time-series

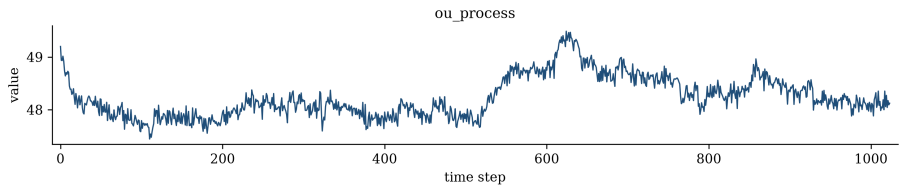


Figure 2.14: Time series generated by OU process

classification, interpolating between samples of different classes to measure how much deformation is required before a classifier changes its prediction, thereby quantifying the robustness of its decision boundaries. Across this line of work, a consistent theme emerges: morphing is used as an instrument for understanding or auditing models, to probe where they succeed, fail, or become unstable as the data is gradually deformed.

Chapter 3

Methodology

The previous chapters introduced a foundation model whose performance depends on the richness of its synthetic prior, and dataset morphing as a way of generating intermediate series between a source and a target. So our idea is as follows, can we use morphing to create a prior that is better, worse, or no different for zero-shot forecasting.

Each generator produces series of a characteristic type, occupying its own region of the space of temporal behaviours, it is expected that series from the same generator are more similar to each other than to series from another generator leaving then some uncovered region in between, as illustrated schematically in Figure 3.1.

Morphing populates this intermediate region.

Interpolating between series is not in itself new. The pipeline already includes a mixing augmentation, TS-Mixup [7], which convex-combines randomly sampled series. Our goal is to study a controlled and isolated form of it, rather than mixing random series as one ingredient among many, we intend to use the morphed datasets on their own. The result is a controlled comparison of two priors, one of pure generator data and one of morphed data, identical in every other respect, evaluated on the same benchmark.

3.1 Algorithm

When the two series are drawn from the same dataset this is usually well-behaved. The series share a comparable scale, so both contribute meaningfully to the blend across the full range of α. In our setting, however, the source and target come from different generators, which can differ in magnitude by orders of magnitude, a step-function series may range over thousands while an audio-derived series oscillates around unit amplitude. Under raw interpolation, the larger-magnitude series dominates. For most values of α, the morph is essentially a scaled copy of the larger series, and the shape of the smaller one is invisible. The interpolation then no longer traverses a meaningful path between the two dynamics.

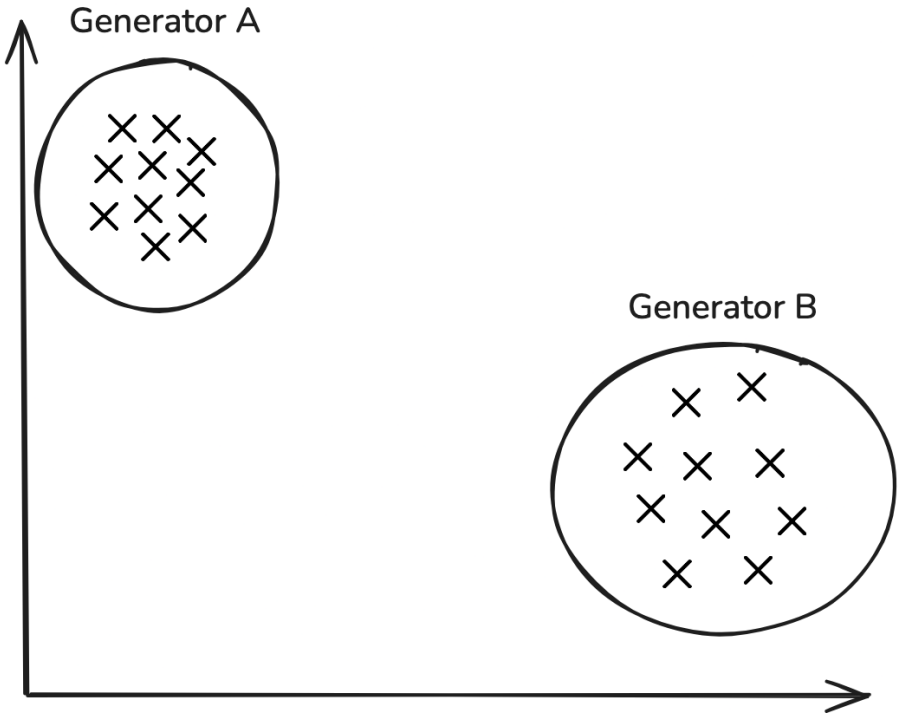


Figure 3.1: Simplified illustration of an uncovered gap

To address this, we separate a series into its shape and its magnitude, and interpolate the two independently. Given a source series S and target series T and a mixing coefficient $\alpha \in [0,1]$, the morphed series is constructed in four steps.

3.1.1 Standardisation

Both series are first standardised to zero mean and unit variance as seen in equation,

$$\hat{S} = \frac{S - \mu_S}{\sigma_S}, \qquad \hat{T} = \frac{T - \mu_T}{\sigma_T} \tag{3.1}$$

where μ and σ denote the mean and standard deviation of each series. This removes the scale and offset, leaving only the shape of each series, its pattern of variation, independent of how large its values are.

3.1.2 Morphing

The standardised shapes are combined by convex interpolation as shown in equation,

$$\hat{B} = (1 - \alpha)\,\hat{S} + \alpha\,\hat{T} \tag{3.2}$$

Because both shapes now have the same scale, each contributes to the blend in proportion to α alone.

3.1.3 Restandardization

Morphing two unit-variance shapes does not in general yield a unit-variance result. We therefore restandardise the blend back to zero mean and unit variance in equation,

$$\hat{B}' = \frac{\hat{B} - \mu_{\hat{B}}}{\sigma_{\hat{B}}} \tag{3.3}$$

so that the morphed time series is on the same standardised footing as its constituents before magnitude is reintroduced.

3.1.4 Magnitude Reintroduction

Finally, the standardised morphed time series is rescaled to a target magnitude that is itself interpolated between the two original series. We define the target mean and standard deviation as convex combinations of the source and target statistics,

$$\mu^* = (1-\alpha)\,\mu_S + \alpha\,\mu_T, \qquad \sigma^* = (1-\alpha)\,\sigma_S + \alpha\,\sigma_T, \tag{3.4}$$

$$M = \sigma^*\,\hat{B}' + \mu^*. \tag{3.5}$$

The result is a series whose shape interpolates between the two generators' patterns and whose scale interpolates between their magnitudes, with both transitions governed by the single coefficient α.

Unlike raw interpolation, neither series can dominate the other purely because of its scale. A series ranging over thousands and one oscillating around one contribute equally to the shape at α = 0.5.

A guard handles the degenerate case of a constant (zero-variance) series, for which standardisation is undefined, such series are passed through without standardisation to avoid division by zero. We use direct pointwise interpolation rather than alignment-based morphing (for example, DTW-barycentre averaging).

3.2 Cross-Generator Morphing

The preceding section defined how a single pair of series is morphed. We now describe how this operation is applied to construct the morphed training data. The morphing operation is applied exclusively between series drawn from different generators, never between two series of the same generator. This restriction follows directly from the idea introduce earlier, the aim of morphing is to populate the space between the generator clusters illustrated in Figure 3.1 , the intermediate regions that no single generator produces.

For each pair of generators, we do not produce a single time series at one fixed α coefficient, but instead we generate multiple by varying α between 0 and 1, the morphed series traces a path

from the dynamics of the first generator to those of the second, sampling the intermediate region at several points along the way rather than only at its midpoint. This produces a graded spectrum of series per pair, some predominantly resembling the source, some the target, and some balanced between them, rather than a single hybrid, giving a denser and more continuous coverage of the space between the two generators.

Chapter 4

Empirical Study

4.1 Experimental Setup

4.1.1 Controlled Experiment

The goal of our experiment is a controlled comparison between two models that are identical in every respect except their training data. The first model, which we refer to as the generated model, is trained on the unmodified output of the synthetic generators. The second, the morphed model, is trained on morphed data only. Both models share the same architecture, the same optimiser and learning-rate schedule, the same training budget, and the same evaluation protocol, the only variable that differs between them is the training data. Any difference in performance can therefore be attributed to the morphing of the training data alone, rather than to architectural or optimisation differences.

4.1.2 Model and Training Configuration

Both models are trained with the configuration presented in Table 4.1.

Where the original model is trained for 3 million iterations, resource constraints limit each of our models to 100000 iteration steps. Both arms are trained for exactly the same number of steps, so the comparison between them remains fair. However, neither model is trained to the extent of the original, and thus the results are not comparable to theirs.

4.1.3 Training Dataset

As mentioned before, there are 14 different generators. We targeted a total of 1 million time series and distribute these uniformly across the generators,

$$\frac{1{,}000{,}000}{14} \approx 71{,}429 \tag{4.1}$$

time series per generator.

This collection of generator-produced series constitutes the training data for the first model.

Table 4.1: Model and training configuration, identical across both arms.

Category	**Value**
Model	
Total parameters	∼42M
Embedding dimension	512
Encoder layers	10
Number of heads	4
Short convolution kernel size	32
State weaving	True
Quantile levels	0.1, 0.2, . . . , 0.9
Training	
Training steps	100000
Batch size (per GPU)	40
Gradient accumulation steps	1
Effective batch size	160
Peak learning rate	2×10^{-4}
LR scheduler	Cosine
Min LR ratio	0.01
Warmup ratio	0.003
Optimization	
Optimiser	AdamW
β_1, β_2	0.9, 0.98
Weight decay	0.01
Adam ε	1×10^{-6}
Gradient clipping	100
Hardware	
GPUs	4 × NVIDIA A100 (40 GB)
Precision	bfloat16

The morphed dataset is also built from these series. A single morphing operation between a source and a target series can yield more than one output, controlled by a parameter we call the granularity, which is better understood with Figure 4.1, which essentially shows that granularity is the number of series produced by the morphing, some close to the source, some in the middle and some close to target.

In this project, we use a constant granularity of 5. Morphing the entire dataset of 1M time series would yield a dataset of 5M time series, which could be deemed unfair as one model has a lot more avaiable data than the other.

For that reason, we first calculate the number of possible cross-generator combinations which is 91 (For example, GP-Kernel, GP-spike, ...). Then with equation,

$$\frac{1{,}000{,}000}{91 \times 5} \approx 2{,}198 \tag{4.2}$$

per combination of generators. With this number in mind, all we have to do is take, as an

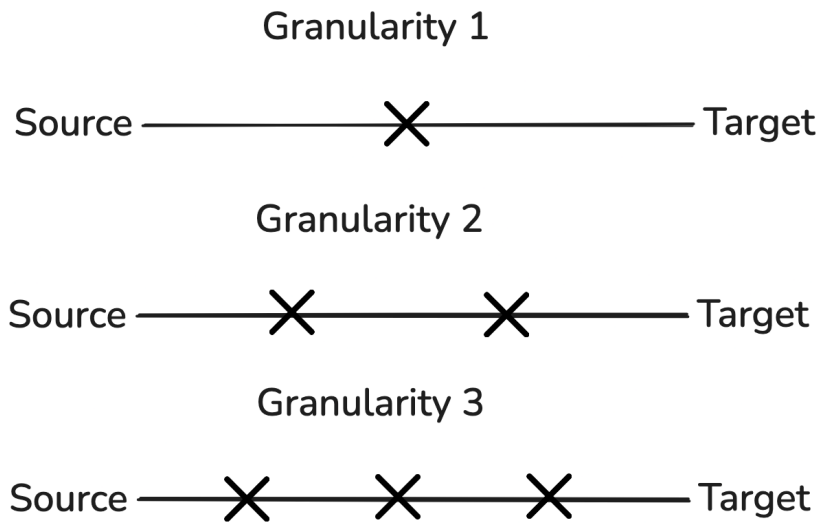


Figure 4.1: Granularity illustration. Each x represents a morphing of the source and target. Granularity indicates how many to output.

example, 2198 series of the GP generator from the 71429 we had already generated and 2198 of the kernel generator and morph them pair-wise. We repeat this process for all combinations of generators.

When morphing, we do not use the same time series, for instance when morphing GP-kernel and GP-spike, we did not use the same series from the GP for both, in fact all of them were different. This holds for all combinations.

4.1.4 Evaluation Protocol

Both models are evaluated zero-shot on the GIFT-Eval benchmark [1], a suite of real-world forecasting datasets spanning a range of domains, sampling frequencies, and forecast horizons. Neither model is exposed to any of these datasets during training.

We evaluate on 55 datasets across the short, medium, and long horizon terms, giving 97 total tests. The two models are evaluated under identical settings, the same datasets, the same horizons and the same number of evaluation windows per series.

Performance is reported with two complementary metrics. The mean weighted quantile loss, a discrete approximation to the Continuous Ranked Probability Score (CRPS), measures the quality of the full predictive distribution by averaging the pinball loss across the predicted quantile levels. The MASE measures point-forecast accuracy, scaling the absolute error by that of a naive baseline so that values are comparable across series of different magnitude. Together, the two capture the probabilistic and point-forecast aspects of performance respectively.

We report both metrics as raw values rather than normalising them against a seasonal-naive baseline. Our central comparison is between the two models, their raw metrics are therefore directly comparable, and normalisation would change both models identically without affecting

the comparison between them. Normalisation is required only when comparing against externally published figures, which is not the aim of this project.

4.2 Results

We compare the two arms across the 97 dataset–term evaluations of the GIFT-Eval benchmark, reporting CRPS and MASE for each. Aggregate results are given in Table 4.2.

Across both metrics, the clean arm outperforms the morphed arm by a substantial margin. On CRPS, the generated model achieves a mean of 0.278 and a median of 0.186, against 0.909 and 0.522 for the morphed model. The point-forecast metric shows the same pattern, a mean MASE of 2.01 and median of 1.23 for the clean arm, against 5.34 and 3.05 for the morphed arm. The gap is consistent rather than driven by a few outliers, the generated model achieves the lower CRPS on 94 of the 97 evaluations and the lower MASE on 92 of 97.

Table 4.2: Aggregate comparison of the generated and morphed arms across the 97 GIFT-Eval evaluations. Lower is better for both metrics.

	CRPS		**MASE**	
	Generated	Morphed	Generated	Morphed
Mean	**0.278**	0.909	**2.007**	5.342
Median	**0.186**	0.522	**1.232**	3.049
Wins (of 97)	**94**	3	**92**	5

The per-dataset comparison is shown in Figure 4.3, plotting each evaluation's morphed model error against its generated model error. The large majority of points lie above the diagonal on both metrics, indicating higher error for the morphed model, the few points on or below the line correspond to the small number of evaluations where morphing did not degrade performance. Figure 4.2 shows the same comparison as the per-dataset change in CRPS, ordered from largest improvement to largest degradation. Morphing improved CRPS on only three evaluations, and in each case the margin was small; the degradation was more pronounced on the longer-horizon evaluations.

4.3 Discussion

The experiment was designed to answer whether morphing between generators produces a better training prior for zero-shot forecasting. And with the particular setup analysed here it does not. The morphed model was worse than the generated model on the large majority of evaluations and by a substantial margin on both CRPS and MASE. Replacing generator data with morphed series degraded the model's forecasting across almost all of the benchmark.

Several factors have contributed to this result which we analyse now.

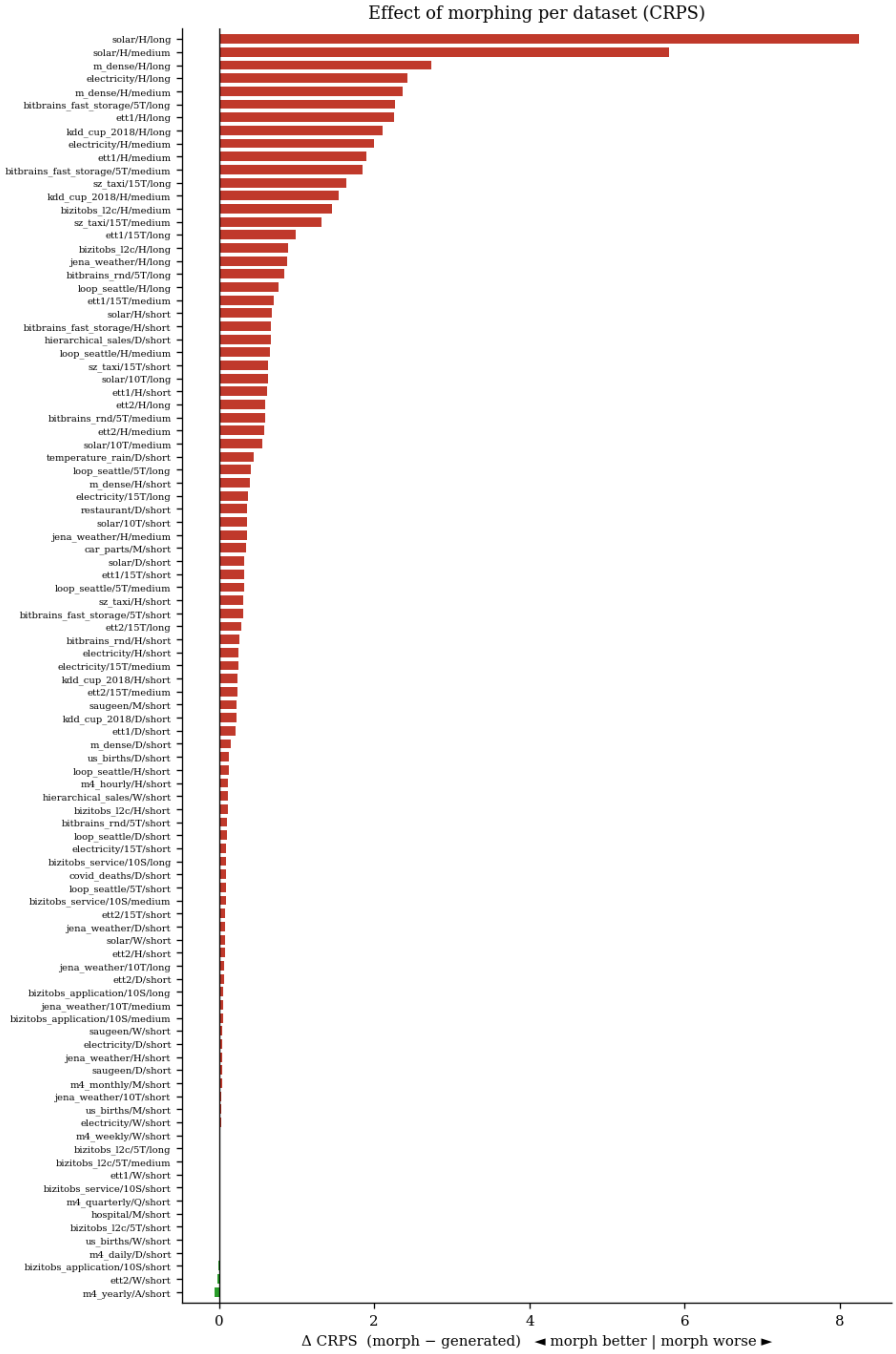


Figure 4.2: Change in CRPS under morphing for each of the 97 GIFT-Eval evaluations, computed as the morphed model CRPS minus the generated model CRPS and ordered from largest improvement to largest degradation. Green bars (negative) indicate datasets where morphing reduced error, red bars (positive) indicate datasets where it increased error. The predominance of red bars, and their concentration among the longer-horizon evaluations, illustrates the consistent and horizon-dependent degradation under morphing.

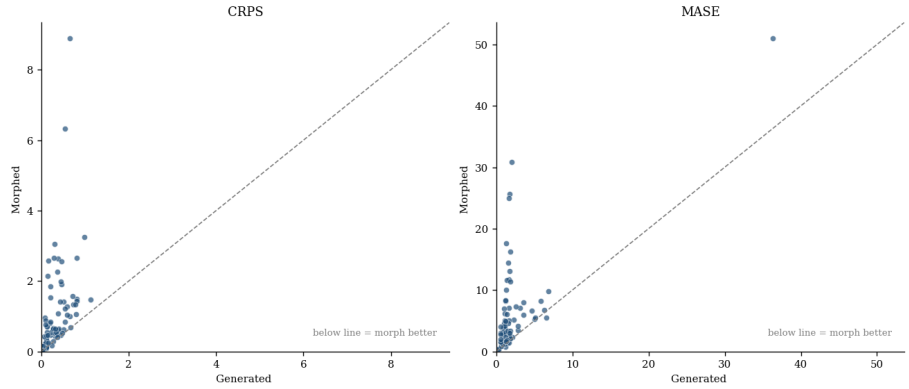


Figure 4.3: Per-dataset comparison of the generated and morphed arms across the 97 GIFT-Eval evaluations, for CRPS (left) and MASE (right). Each point is one dataset–term evaluation, plotting the morphed model error against the generated model errorm, the dashed line marks equal performance. Points above the line indicate higher error for the morphed model. The large majority of points lie above the diagonal on both metrics, showing that morphing degraded performance across most datasets.

4.3.1 Morphing creates harder to model time series

First, morphed series are by construction more varied and less structured than the series of any single generator. A pure Gaussian-process series has a clear, learnable structure, a series that mrophs a Gaussian process with a spike generator is a hybrid that belongs to neither family and exhibits a less coherent pattern. It is plausible that such hybrids are simply harder to fit, so that a model trained on them learns a less accurate forecaster. This interpretation is consistent with the origins of morphing, in prior work, morphing was introduced precisely as a tool for surfacing hard cases, the regions of the input space where models struggle. Used as training data rather than as a diagnostic, morphing may therefore supply exactly the kind of difficult, ambiguous examples that hinder learning rather than aid it.

4.3.2 Granularity

A further limitation concerns the fixed granularity of five. Two distinct issues arise from it. First, the five series produced from a single morph are points along the interpolation between one source and one target. When the source and target happen to be similar, which can occur even across generators, if the two sampled series are not very different in shape, the five morphed outputs are themselves nearly identical to one another. In such cases the granularity yields five near-duplicate series rather than five distinct ones, contributing little additional diversity while still consuming five of the corpus's fixed one-million slots.

Second, the granularity reduces the number of distinct original series underlying the morphed dataset. Because each morph expands one source–target pair into five outputs, fewer underlying pairs are needed to reach the one million target. Concretely, with 2198 pairs per generator combination, 4.2 and each generator appearing in 13 combinations, only 2,198×13 = 28,5742 distinct

series of each generator are drawn into the morphed dataset, each then repeated, in varied form, five times over. The generated model, by contrast, trains on the full 71429 distinct series it generates per generator. The morphed model therefore sees a less diverse set of underlying series, padded out by the granularity, which may itself contribute to its weaker performance independently of any effect of morphing as such.

The redundancy from fixed granularity could be addressed by making the granularity a function of the dissimilarity between the source and target series. A pair of very different series should have higher granularity, since intermediate points are distinct and worth sampling, whereas a pair of similar series should yield few morphs, or none, to avoid producing near-duplicates. This way the 1M time series budget could have been allocated to more meaningful data points.

4.3.3 Time Horizon

The degradation was not uniform across forecast horizons, the largest increases in error fell almost entirely on the medium- and long-term evaluations, while short-horizon forecasts were affected least (Figure 4.2). This is consistent with the harder-data interpretation. Longer horizons demand that the model extrapolate coherent structure further into the future, which depends on having learned clean, stable temporal patterns, if the model could not learn these patterns because of the less coherent morphed data, the error would be expected to grow with horizon length, exactly as observed.

4.3.4 Fairness in the training dataset

As noted earlier, morphing the 1M generator output at granularity five would have produced 5M morphed series. To keep the comparison fair in dataset size, we instead restricted the morphed dataset to 1M series. However, equal size does not guarantee equal fairness. As discussed, the morphed dataset is built from fewer distinct underlying series, each expanded 5 times, so although the two datasets contain the same number of series, the morphed one is less diverse in its underlying data. Dataset size alone could be a misleading measure of parity.

This is reinforced by considering how much data each model sees during training. With 100000 iterations at an effective batch size of 160, each model is exposed to $100000 \times 160 =$ 1.6M series over the course of training, a figure fixed by the training schedule, not by the size of the dataset. A model trained on a 1M dataset simply revisits each series more often than one trained on a larger dataset, however because of the morphing technique applied, even though all series are visually different they may contain the same properties, because there are multiple combinations of the source and the target, so we don't think having a larger dataset would necessarily be an advantage.

So perhaps another interesting experiment would have been to test the generated data against the morphing of all that data.

4.3.5 Limitations

There are some limitations that should be mentioned. For instance the original tempoPFN model was trained on a dataset of 10M time series and 3M iteration steps, due to compute constraints, we did only 1M time series as mentioned previously and 100k iteration steps. Perhaps, the morphed model could have learned the supposed harder patterns if it were given more iteration steps and a bigger, more diverse dataset.

Finally, our results cannot be compared directly to the figures reported in the original work. The published GIFT-Eval results are normalised against a seasonal-naive baseline, so that each dataset's score is expressed relative to that baseline, whereas we report raw, un-normalised metrics. The two sets of numbers are therefore on different scales and are not comparable. This does not affect our central comparison, both our models are evaluated identically, so their raw metrics are directly comparable to each other, since that was the goal of our project and not comparison against other published models.

Chapter 5

Conclusion

This project investigated whether dataset morphing, applied between the synthetic generators of a forecasting foundation model, yields a better training prior for zero-shot forecasting. Starting with the tsMorph tool, we repurposed it from its original role as a diagnostic tool into a strategy for generating training data.

We then evaluated the idea through a controlled comparison, two instances of the TempoPFN model, identical in architecture, optimisation, and training budget, differing only in their training data, one trained on the unmodified generator output, the other on a dataset of cross-generator morphed series of equivalent size. The model trained on morphed data was substantially worse than the model trained on generator data alone, across almost all of the GIFT-Eval benchmark and on both probabilistic and point-forecast metrics, with the degradation most pronounced at longer forecast horizons. With the experimental setup used in this project, replacing generator data with cross-generator morphed series did not produce a better prior.

This suggests that the intermediate regions between generators, populated in this way, are harder for the model to learn from than the generators themselves, which is consistent with the diagnostic origins of morphing, a technique conceived to surface difficult cases rather than to ease learning. The finding should be read within its limits. Both models were trained at a small fraction of the original schedule, on a fraction of the dataset and only one morphing configuration was studied. Whether morphing might help under a longer training budget or with a granularity adapted to the similarity between series, remains open.

References

[1] Taha Aksu et al. "GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation". In: *arxiv preprint arxiv:2410.10393* (2024).

[2] Abdul Fatir Ansari et al. "Chronos: Learning the Language of Time Series". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856. URL: https://openreview.net/forum?id=gerNCVqqtR.

[3] Antónia Brito et al. "tsMIST: Model Sensitivity Analysis with Time Series Morphing". In: *Discovery Science*. Cham: Springer Nature Switzerland, 2025, pp. 270–285.

[4] Vítor Cerqueira, Luís Torgo, and Carlos Soares. "Model Selection for Time Series Forecasting: An Empirical Analysis of Multiple Estimators". In: *Neural Processing Letters* 55.8 (2023), pp. 10073–10091. DOI: 10.1007/s11063-023-11239-8. URL: https://doi.org/10.1007/s11063-023-11239-8.

[5] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. DOI: 10.1613/jair.953.

[6] André Correia, Carlos Soares, and Alípio Jorge. "Dataset Morphing to Analyze the Performance of Collaborative Filtering". In: *Discovery Science (DS 2019)*. Vol. 11828. Lecture Notes in Computer Science. Springer, 2019, pp. 29–39. DOI: 10.1007/978-3-030-33778-0_3.

[7] Luke Nicholas Darlow et al. "TSMix: Time Series Data Augmentation by Mixing Sources". In: *Proceedings of the 3rd Workshop on Machine Learning and Systems (EuroMLSys '23)*. ACM, 2023, pp. 109–114. DOI: 10.1145/3578356.3592584.

[8] Samuel Dooley et al. "ForecastPFN: Synthetically-trained zero-shot forecasting". In: *Advances in Neural Information Processing Systems*. 2023.

[9] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. 3rd. Melbourne, Australia: OTexts, 2021. URL: https://otexts.com/fpp3/ (visited on 01/31/2026).

[10] Vladyslav Moroshan et al. *TempoPFN: Synthetic Pre-training of Linear RNNs for Zero-shot Time Series Forecasting*. arXiv:2510.25502 [cs]. Oct. 2025. DOI: 10.48550/arXiv.2510.25502. URL: http://arxiv.org/abs/2510.25502 (visited on 11/17/2025).

[11] Samuel Müller et al. "Transformers Can Do Bayesian Inference". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=KSugKcbNf9.

[12] Thomas Nagler. "Statistical Foundations of Prior-Data Fitted Networks". en. In: *Proceedings of the 40th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2023, pp. 25660–25676. URL: https://proceedings.mlr.press/v202/nagler23a.html (visited on 11/19/2025).

[13] Moisés Santos, André de Carvalho, and Carlos Soares. "Enhancing Algorithm Performance Understanding through tsMorph: Generating Semi-Synthetic Time Series for Robust Forecasting Evaluation". In: *arXiv preprint arXiv:2312.01344* (2024).

[14] Ege Onur Taga, Muhammed Emrullah Ildiz, and Samet Oymak. "TimePFN: Effective Multivariate Time Series Forecasting with Synthetic Data". English. In: *Proc. AAAI Conf. Artif. Intell.* Vol. 39. Issue: 19 Journal Abbreviation: Proc. AAAI Conf. Artif. Intell. Association for the Advancement of Artificial Intelligence, 2025, pp. 20761–20769. ISBN: 978-1-57735-897-8. DOI: 10.1609/aaai.v39i19.34288.

[15] Shifeng Xie et al. "CauKer: Classification Time Series Foundation Models Can Be Pretrained on Synthetic Data Only". In: *ICML Workshop on Foundation Models for Structured Data (FMSD)*. 2025.

Appendix A

Literature Review

In this chapter, the methodology for identifying and selecting relevant research literature is presented. The purpose of this process is to compile a collection of state-of-the-art research papers that will later be analyzed in order to gain a comprehensive understanding of existing work related to PFNs and foundation models for time series forecasting. The chapter begins by outlining the search strategy, including the databases consulted and the search queries used. This is followed by a description of the inclusion and exclusion criteria applied to ensure that only the most relevant, high quality, and peer-reviewed publications were considered.

A.1 Methodology

To provide a comprehensive review, a systematic literature search was conducted in major digital libraries and academic search engines. The primary sources consulted were Scopus and the ACM Digital Library, both recognized for their peer-reviewed publications in computer science, and to complement these sources, Google Scholar was also used to capture additional studies and recent papers not yet indexed in formal digital libraries.

A.2 Queries

A.2.1 Query Formulation

- **Q1**: "Prior-data Fitted Networks" OR "PFN"
- **Q2**: ("Prior-data Fitted Networks" OR "PFN") AND "time series" AND ("forecasting" OR "classification")
- **Q3**: "foundation model" AND "time series" AND ("forecasting" OR "classification")

A.2.2 Query Justification

- **Q1**: this query is designed to identify literature specifically focused on the base model itself. It ensures a comprehensive understanding of the foundational model, its architecture, and its capabilities before exploring applications.
- **Q2**: this query narrows the focus to works that are directly relevant to the research topic, applying PFNs to time series forecasting. It helps identify existing methods, applications, and potential gaps in the literature that this project can address.
- **Q3**: this query targets state-of-the-art approaches beyond PFNs. By looking at foundation models applied to time series tasks, it provides a broader context for comparison and benchmarking against leading methods.

A.2.3 Search Query Results

Table A.1: Search Query Results

	Scopus	ACM Digital Library	Google Scholar
Q1	568	134	1140
Q2	64	49	40
Q3	1564	57	731

A.3 Filtering and Selection Process

For all searches conducted in Scopus, ACM Digital Library, and Google Scholar, filters were applied to include only papers published from 2023 onward, as the field is rapidly evolving and focusing on recent advances ensures that the review captures the latest developments. Only journal articles and conference proceedings were considered, and papers were limited to English language and the field of Computer Science.

To make the screening process feasible given the large number of initial results, only papers with at least five citations were selected for further review. Duplicate entries across databases were removed prior to screening, and papers were excluded based on relevance, primarily determined by their titles and, when necessary, by abstracts.

In cases where a paper demonstrated exceptional relevance, certain criteria, such as the minimum citation count or the requirement that the work be a journal article or conference paper, were relaxed to allow its inclusion. Additionally, for papers closely aligned with the research topic, reference lists were examined to identify further relevant studies.

After all filtering and screening steps, a total of 33 papers were retained for thorough analysis.

Appendix B

My approach (schedule)

This research adopts a data-centric methodology to advance the state of time series foundation models. While recent developments have focused on optimizing neural architectures, this project hypothesizes that the limiting factor in current PFN performance is the quality and diversity of the synthetic training data.

Consequently, the hypothesis of this work is that a Prior-data Fitted Network (PFN) trained on a morphed prior will achieve superior zero-shot forecasting accuracy on real-world data compared to identical models trained on standard synthetic generation methods.

B.1 Methodology

This research adopts a data-centric methodology to advance the state of time series foundation models. While recent developments have focused on optimizing neural architectures, this project hypothesizes that the limiting factor in current PFN performance is the quality and diversity of the synthetic training data. Consequently, the hypothesis of this work is that a Prior-data Fitted Network (PFN) trained on a morphed prior will achieve superior zero-shot forecasting accuracy on real-world data compared to identical models trained on standard synthetic generation methods.

B.1.1 Activity Cards

The following activity cards detail each research task, including timing, technical goals, and specific deliverables. All timelines are synchronized with the 11-week project schedule.

Activity Card 1: Task 1 - Adapt Dataset Morphing

Duration: 2 weeks (Weeks 1–3)

Goals: Transform tsMorph from a paired dataset evaluation tool into a scalable, high-volume synthetic data generator suitable for PFN training.

Description: This activity involves refactoring the existing tsMorph codebase to support continuous generation of morphed time series. Key technical challenges include optimizing the

morphing algorithm for efficiency and implementing batching mechanisms for parallel generation. The adapted framework must generate thousands of synthetic series per hour using Python and parallel processing frameworks.

Deliverables: Python module with batch API; documentation of morphing parameters; unit tests.

Milestones: Initial refactoring (Week 2); optimization and testing (Week 3); documentation and validation (Week 4).

Activity Card 2: Task 2 - Implement Training Pipeline

Duration: 3 weeks (Weeks 3–6)

Goals: Successfully integrate the morphed data generator into a PFN training pipeline and establish baseline comparison infrastructure.

Description: Building on the adapted morphing module, this activity creates the training infrastructure. This includes selecting a PFN architecture,such as the Transformer design used in Taga, Ildiz, and Oymak [14],implementing data loaders, and establishing hyperparameters. A baseline system will be implemented in parallel to ensure a fair comparison using GPU compute resources.

Deliverables: Training codebase supporting both morphed and baseline priors; configuration files; baseline models; checkpointing infrastructure.

Milestones: Architecture selection (Week 4); baseline training pipeline operational (Week 5).

Activity Card 3: Task 3 - Run Experiments

Duration: 1 week (Weeks 6–7)

Goals: Conduct systematic comparison of morphed prior against baseline synthetic priors across multiple benchmark datasets.

Description: This activity involves training model variants and evaluating zero-shot performance across benchmarks. Experiments will be monitored for convergence, controlling for factors such as training time and model capacity using experiment tracking tools.

Deliverables: Trained models for both morphed and baseline priors; comprehensive evaluation results; training logs and learning curves.

Milestones: Initial training runs (Week 5); full experiment complete (Week 6).

Activity Card 4: Task 4 - Analyse Results

Duration: 2 weeks (Weeks 7–9)

Goals: Identify performance patterns, understand failure modes, and extract insights about the relationship between data diversity and generalization.

Description: This phase focuses on deep analysis of experimental results beyond aggregate metrics. Techniques include error analysis on specific series. The analysis will distinguish between statistical noise and meaningful performance differences.

Deliverables: Detailed analysis report with visualizations; identification of dataset characteristics where morphing excels or fails; statistical significance tests.

Milestones: Initial visualization complete (Week 7); synthesis of findings (Week 8).

Activity Card 5: Task 5 - Refinement Loop

Duration: 1 week (Weeks 9–10)

Goals: Iteratively improve the morphing approach based on analytical insights and validate improvements through targeted experiments.

Description: Based on findings from Task 4, this activity implements targeted improvements to the morphing methodology. This may include adjusting morphing parameters to cover underperforming regions or hybridizing morphing with other synthetic techniques.

Deliverables: Improved morphing implementation; validation experiments; final model evaluation results.

Milestones: First refinement iteration complete (Week 10); final experiments and documentation (Week 11).

Activity Card 6: Task 6 - Synthesis and Documentation

Duration: 2 weeks (Weeks 10–11)

Goals: Integrate final experimental results into the research paper and ensure the technical narrative aligns with the observed outcomes.

Description: This phase focuses on formalizing the research findings. It involves updating the Results and Discussion sections with data from the Refinement Loop (Task 5), generating high-quality visualizations, and refining the Conclusion to reflect the project's technical contributions.

Deliverables: Finalized manuscript draft; comprehensive data visualizations; updated bibliography and technical appendices.

Milestones: Drafting of Results and Discussion sections complete (Week 10); final proofreading and submission-ready document (Week 11).

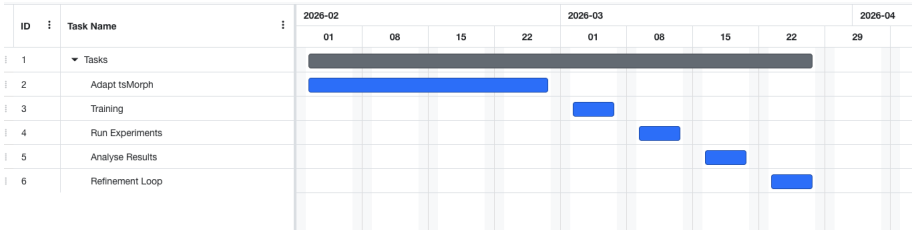


Figure B.1: Gantt Chart of implementation timeline

B.1.2 Task Dependencies

- **Task 1** (Weeks 1–3): No dependencies. This is the initial foundational task.
- **Task 2** (Weeks 3–6): Depends on Task 1. Requires the adapted morphing module.
- **Task 3** (Weeks 6–7): Depends on Task 2. Requires a functional training pipeline.
- **Task 4** (Weeks 7–9): Depends on Task 3. Requires experimental results to analyze.
- **Task 5** (Weeks 9–10): Depends on Task 4. Requires analytical insights to guide refinements.