Representing Time: Towards Pragmatic Multivariate Time Series Modeling

Cristian Ignacio Challú

May 2024
CMU-ML-24-104

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Artur Dubrawski, Chair
Roni Rosenfeld
Barnabas Poczos
Ying Nian Wu (UCLA)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This work was partially supported by the National Institutes of Health (awards R01GM117622, R01NR013912, R01HL144692, R01HL141916,1R01EB029751, R01DK131586, and R01NR013912), U.S. Department of Defense (awards W81XWH19C0101, W911NF22F0014, and W519TC23F0045), DARPA (award FA87501720130), U.S. Department of Energy (awards DEAC5207NA27344, DENA0003525, and DENA0003525), U.s. Department of Transportation, Allegheny County Health Department, two industrial sponsors, and internal research funds of Carnegie Mellon University.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: time series, forecasting, anomaly detection, imputation, multivariate, representation learning, machine learning, deep learning, long-horizon forecasting, transfer learning

Dedicado a Daphne y Emma

Abstract

Time series models are specialized in learning temporal dependencies among observations and interactions between multiple features in a data stream. During the last decade, the unprecedented success of Deep Learning (DL) models in Computer Vision and Natural Language Processing has steadily permeated to time series tasks. From Recurrent Neural Networks to Transformers, new advancements in architectural design improved capabilities and performance. Despite this success, I identify several challenges to adopting current state-of-the-art (SoTA) methods, including handling distribution shifts and missing data, computational complexity, and interpretability.

The success of DL models is usually attributed to their ability to discover helpful data representations automatically. Multivariate time series models involve high-dimensional objects with numerous time series and temporal observations. However, they often exhibit strong temporal dependencies and inter-feature relations. In this thesis, I propose to design DL architectures and algorithms for forecasting and anomaly detection tasks that leverage these dependencies to induce efficient learning of representations that satisfy desirable properties that can (i) improve the models' performance, (ii) improve robustness by favoring domain adaptation, and (iii) reduce over-parameterization to improve scalability. The completed work is organized in three parts, presenting seven novel model types and algorithms that achieve state of the art performance in various tasks while addressing key adoption challenges.

In the first part, I explore the dynamic latent space principle and design latent temporal representations to make robust anomaly detection and forecasting models. In the second part, I present a novel scalable and interpretable model for multi-step forecasting based on a novel non-linear frequency decomposition with connections to Wavelet theory. It also features two extensions on using multivariate exogenous covariates for high-impact domains, including energy and healthcare. Finally, in the third part, I present a large-scale study on enabling conditions, on both model design and data characteristics, for transferability of pre-trained models on time series tasks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

Time series data are abundant in many fields, including social, applied, and natural sciences. *Time series models* are specialized in learning the temporal dependencies among observations and interactions between multiple features. In this thesis, I focus on *forecasting* and *anomaly detection*, two of the most useful applications of time series modeling. Forecasting is essential to optimize the production and distribution of goods, plan electricity markets, build financial portfolios, and predict healthcare patient outcomes, among other examples. Anomaly detection has been gaining attention with the Internet of Things (IoT), with applications for detecting fraud, faults in manufacturing processes, monitoring sensor readings, and predictive maintenance.

During the last decade, the unprecedented success of deep learning models (DL) on Computer Vision (CV) and Natural Language Processing (NLP) has steadily permeated to time series tasks. From Recurrent Neural Networks (RNN) to Transformers, new advancements in architectural design improved capabilities and performance. The high expressivity of DL models allows them to learn non-linear temporal dynamics spanning thousands of observations and relations between hundreds of time series. Cross-learning allows a single *global model* to forecast or monitor multiple time series in a dataset, simplifying pipelines. The success has been evident in academia and industry, and DL models are now predominant in large-scale industrial forecasting applications and monitoring systems (Benidis et al., 2020a).

Despite this recent success, I identify several challenges to adopting current state-of-the-art (SoTA) DL methods, especially in high-stakes applications. Most of the latest research has focused on improving the accuracy on curated datasets, which rarely resemble real-world data. For instance, two understudied recurrent challenges are *distribution shifts* and *missing data*. Distribution shifts are changes in the data-generating process over time and can considerably degrade accuracy (Du et al., 2021). Missing values form a generic data quality issue; some causes include faulty sensors and misplaced collected data. Additionally, many applications require models operating in resource-constrained environments due to limited hardware capacity or high-volume data. In this thesis, I propose to develop novel interpretable DL methods that push

the SoTA performance and address those key challenges to their adoption.

One of the most widely accepted explanations behind the success of deep learning models is their ability to automatically discover helpful data representations, referred to as *representation learning* (Goodfellow et al., 2016). Multivariate time series models involve high-dimensional objects with numerous time series and temporal observations. However, they exhibit strong temporal dependencies and relations between features, which have been thoroughly studied (Box et al., 2015b; Hyndman and Athanasopoulos, 2018a). In my work, I propose to design DL architectures that leverage these dependencies to induce efficient learning of *good* representations, as defined in (Bengio et al., 2013), that satisfy desirable properties that can (i) improve the models' performance, (ii) improve robustness by favoring domain adaptation, and (iii) reduce overparameterization to improve scalability.

1.2 Thesis Statement

This thesis focuses on improving the performance, scalability, and capabilities of multivariate deep learning time series models for forecasting and anomaly detection to improve their adoption and potential benefits. In particular, it is based on designing efficient architectures and latent representations leveraging time series dynamics. My dissertation is centered around the following statement:

> *Deep multivariate time series algorithms can be designed to yield compact informed representations while delivering superior performance, computational efficiency, and robustness.*

1.3 Overview and Summary of Contributions

I provide evidence that supports my thesis statement in the following three parts, containing six different case studies of novel forecasting and anomaly detection architectures with informed data representation structures that improve the capabilities of deep learning models.

Part I: Dynamic Latent Space for Multivariate Time Series

Chapter 2: Multivariate Online Anomaly Detection

Multivariate time series anomaly detection (MAD) is gaining relevance with the Internet of Things (IoT), with myriad crucial applications such as fraud or threat detection, monitoring sensor readings, and predictive maintenance. Despite its multiple applications and potential benefits, designing useful and more accurate MAD methods remains challenging. DL methods, in particular, struggle to detect *contextual anomalies* without overfitting the training data. Second, real sensor data streams are usually extremely noisy, with missing values, corrupted data, and variable features.

This chapter first presents DGHL, a novel Generative model for online multivariate time series anomaly detection that tackles the limitations of current approaches in high-stakes settings. It employs an adaptation of the Alternating Back-Propagation algorithm to dynamically infer the optimal latent vector to reconstruct the current behavior of the time series, equivalent to maximizing the posterior distribution of the target variable conditional on the available signal. A novel hierarchical structure of latent factors allows the model to reconstruct long signals efficiently, improving its capabilities to detect contextual anomalies. Extensive empirical analysis on four benchmark datasets demonstrates that DGHL achieves SoTA performance in detecting anomalies, increasing relative performance on datasets with higher missing data.

Second, this chapter presents SAAT, a framework for automatic optimal threshold selection for anomaly detection. Thresholds are crucial in pipelines since they determine the frontier of anomaly scores to label an observation as *nominal* or *anomalous*. An optimal threshold can be as important in performance as the model itself. Despite this, existing algorithms to find thresholds are simple and not tailored to the latest DL methods. We showcase the superior performance of SAAT on three benchmark datasets and seven models over existing alternatives. Finally, SAAT can incorporate domain knowledge by selecting the synthetic anomaly generator that best mimics true anomalies in the task.

Chapter 3: Robust Multivariate Forecasting and Imputation

We identify two widespread challenges for adopting deep learning methods to real applications, which have been largely understudied: *distribution shifts* and *missing data*. *Distribution shifts* are changes in the time series behavior over time and can considerably degrade the accuracy of forecasting models (Kuznetsov and Mohri, 2014; Du et al., 2021; Xu et al., 2022b; Ivanovic et al., 2022). *Missing values* occur when no data value is stored for some dataset entries. Some common causes include faulty sensors and misplaced collected data. These challenges limit forecasting methods and their potential benefits in many applications.

This chapter presents *Temporal Inference Networks* (TIN), a new paradigm for time series forecasting: a Generator Network directly takes latent factors as inputs, which are dynamically inferred by matching the model's output on past observations by minimizing a reconstruction error. This formulation is equivalent to the target variable's maximum a posteriori estimation (MAP) conditional on past observations. A novel *Temporal Dynamic Module* imposes temporal dynamics on the latent factors based on function templates. The empirical comparison shows the proposed approach outperforms current SoTA models with improvements over 50% on settings with up to 80% missing data while simultaneously imputing missing past observations with better accuracy than alternatives.

Part II: Scalable and Interpretable Multi-step Forecasting

Chapter 4: Long-Horizon Multi-Step Forecasting

The recent advancements in neural forecasting methods have steadily improved the performance and capabilities of forecasting systems. Machine learning methods won recent large-scale

competitions, such as the M4 (Makridakis et al., 2020), improving over classic statistical methods. However, their impressive accuracy is usually accompanied by a drastic increase in computational cost. This increase is exacerbated in long-horizon forecasting, where accurately predicting hundreds of timestamps requires modeling longer temporal dependencies. Current architectures scale poorly with the input and output dimensionality. Designing accurate and efficient methods is essential for many applications, particularly in domains where computational resources are constrained.

This chapter presents NHITS, a novel global model with *hierarchical interpolation* and *multi-rate sampling* techniques to decompose the forecast in frequency bands. The novel decomposition extends on the classic Fourier decomposition, as it is not constrained only to period signals in each frequency. The NHITS achieves remarkable performance in six benchmark datasets commonly used in the long-horizon forecasting literature, outperforming even the latest specialized transformers architectures. Finally, thanks to the hierarchical interpolation technique, the NHITS is 1.26x faster and requires only 54% of the parameters of the related NBEATS model, and an order of magnitude faster than Transformers baselines.

Chapter 5: Multivariate Forecasting

This chapter presents two extensions of the NHITS and NBEATS models on multivariate forecasting with *exogenous variables* on real applications. Exogenous variables are temporal or static features that provide additional information to forecast the *target time series*. In many settings, these additional covariates are crucial to achieving accurate forecasts. They can provide necessary information on the generating process of the target variable, which is not available in the autoregressive values alone.

The first extension, NBEATSx, incorporates a convolutional encoder to learn a decomposable basis of the exogenous covariates. The method is tested on the electricity price forecasting (EPF) task, a classic forecasting setting with exogenous covariates. In the second application, we propose a pharmacokinetic prior encoder to incorporate sparse treatment variables efficiently. The encoder pre-processed the sparse treatment variables to model their cumulative effect and passed them as inputs to the NHITS model. The capabilities and superior performance of the proposed method are compared to current alternatives in the glucose forecasting task.

Part III: Towards Time Series Foundation Models

Chapter 6: Transferability of Neural Forecasting Methods

Transfer learning is a technique that involves applying knowledge acquired from one task to solve separate related tasks more accurately and efficiently. Therefore, transfer Learning and representation learning are intrinsically related, as models exploit commonalities between tasks by learning shared useful representations that capture common underlying factors of the data. In practice, models are pre-trained on source large-scale tasks and then directly used to forecast on a new task, with or without re-training the parameters.

This chapter aims to advance the applications of transfer learning in time series forecasting tasks to extend its enormous benefits. First, we perform a large-scale evaluation using different model architectures, implementing a unified framework to evaluate transfer learning pipelines. The empirical study presents evidence that pre-trained architectures outperform the accuracy of widely adopted statistical forecasting tools and baselines, with computational speed improvements on the orders of magnitude. Second, we explore the conditions on the model design and the data that enable transferability between tasks. Notably, we note that the number of learnable parameters, the size of the source dataset, and fine-tuning enhance accuracy, but they have diminishing returns. Finally, training data diversity plays a critical role in the accuracy of the transferred model. In particular, we propose a novel distance metric between time series tasks that strongly correlates with pre-trained model accuracy.

1.4 Bibliographic Notes

All the case studies presented in this thesis have been published in the following conferences and journals.

Part I is based on:

- Cristian Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. "Deep generative model with hierarchical latent factors for time series anomaly detection". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. PMLR. 2022, pp. 1643–1654
- Cristian Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. "SpectraNet: Multivariate Forecasting and Imputation under Distribution Shifts and Missing Data". In: *ML4IoT Workshop at ICLR, Oral*. 2023
- Cristian Challu, Mononito Goswami, Peihong Jiang, and Laurent Callot. "Automatic Thresholding with Agnostic Synthetic Anomalies". In: *Work in Progress. To be submitted*. 2024

Part II is based on:

- Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. "Nhits: Neural hierarchical interpolation for time series forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 6. 2023, pp. 6989–6997
- Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafal Weron, and Artur Dubrawski. "Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx". In: *International Journal of Forecasting* 39.2 (2022), pp. 884–900
- Willa Potosnak, Cristian Challu, Kin G Olivares, and Artur Dubrawski. "Forecasting Response to Treatment with Deep Learning and Pharmacokinetic Priors". In: *Findings Track at ML4H* (2023)

Part III is based on:

- Kin G Olivares, Cristian Challu, Azul Garza, Max Mergenthaler Canseco, and Artur Dubrawski. "Transferability of Neural Forecast Methods". In: *IIF-SAS Award to be submitted*

Part I

Dynamic Latent Space for Multivariate Time Series

Most existing Deep Generative models for time series are based on the encoder-decoder paradigm. This part explores the application of using a latent space inference step, replacing encoders or other auxiliary networks, by minimizing the reconstruction error on past observations. The latent space of the Generator model can incorporate domain knowledge to encode time series patterns more efficiently. In addition to state-of-the-art performance, two main advantages of this novel framework are robustness to data quality issues including missing data or distribution shifts. The two chapters in this Part present novel approaches based on the dynamic latent space inference principle for time series anomaly detection and forecasting.

Chapter 2

Multivariate Online Anomaly Detection

2.1 Motivation

Recent advancements in Deep Learning, such as Recurrent Neural Networks (RNN), Temporal Convolution Networks (TCN), and Graph Networks (GN), have been successfully incorporated by recent models to outperform previous approaches such as out-of-limits, clustering-based, distance-based, and dimensionality reduction on a wide range of tasks. (Chalapathy and Chawla, 2019) (Ruff et al., 2021) present comprehensive reviews of current state-of-the-art methods for time series anomaly detection.

This chapter proposes DGHL, a novel Deep Generative model based on a top-down Convolution Network (ConvNet), which maps multivariate time series windows to a novel hierarchical latent space. The model is trained by maximizing the observed likelihood directly with the Alternating Back-Propagation algorithm (Han et al., 2017) and short-run MCMC (Nijkamp et al., 2021), so it does not rely on auxiliary networks such as encoders or discriminators as VAEs and GANs do. DGHL, therefore, comprehends a separate family of generative models, previously unexplored for time series anomaly detection. We perform experiments on several popular datasets and show the proposed model outperforms the recent state-of-the-art while reducing training times against previous reconstruction-based and generative models.

With the advent of IoT, settings with corrupted or missing data have increasing relevance. For example, faulty sensors can cause missing values, privacy issues on consumer electronics devices, or heterogeneous hardware can lead to variable features. We present the first extensive analysis on the robustness of current state-of-the-art models on datasets with missing inputs and variable features with novel occlusion experiments. DGHL achieved superior performance on this setting, maintaining state-of-the-art performance with up to 90% of missing data, without modification to the architecture or training procedure. We perform additional qualitative experiments of our model to assess desirable properties of lower-dimensional representations, such as continuity and extrapolation capabilities. Finally, we show how DGHL can be used as a forecasting model, demonstrating its versatility on various time series tasks.

The main **contributions** of this section are:

- **Short-run MCMC**. The first time series anomaly detection generative model based on short-run MCMC for estimating the posterior of latent variables and inferring latent vectors. In particular, the first application of the Alternating Back-Propagation algorithm for learning generative models for time series data.
- **Hierarchical latent factors**. A novel hierarchical latent space representation to generate windows of arbitrary length. We demonstrate with ablation studies how DGHL achieves state-of-the-art performance by leveraging this representation on four benchmark datasets.
- **Robustness to missing data**. The first experiments on robustness to missing inputs of state-of-the-art anomaly detection models and demonstrate DGHL achieves superior performance in this setting.

2.2 Related Work

2.2.1 Anomaly Detection

Multivariate Anomaly detection (MAD) is an unsupervised binary classification problem consisting of detecting anomalous events at the timestamp level on a stream of time series data. In the *Online* setting, we want to detect anomalous events as soon as possible, using only past information. Recent advancements in Deep Learning, such as Recurrent Neural Networks (RNN), Temporal Convolution Networks (TCN), and Graph Networks (GN), have been successfully incorporated by recent models to outperform previous approaches such as out-of-limits, clustering-based, distance-based, and dimensionality reduction on a wide range of tasks. (Chalapathy and Chawla, 2019; Ruff et al., 2021) present comprehensive reviews of current state-of-the-art methods for time series anomaly detection.

2.2.2 Reconstruction-based models

Reconstruction-based models learn representations for the time series by reconstructing the input based on latent variables. The reconstruction error or the likelihood is commonly used as anomaly scores. Among these models, variational auto-encoders (VAE) are the most popular. The LSTM-VAE, proposed in (Park et al., 2018), uses LSTM as encoders and decoders and models the reconstruction error with support vector regression (SVR) to have a dynamic threshold based on the latent space vector. OmniAnomaly (Su et al., 2019) improves the LSTM-VAE by adding normalizing planar flows to increase the expressivity and including a dynamic model for the latent space. Generative Adversarial Networks (GANs) were also adapted for anomaly detection as alternatives to VAE, with models such as AnoGAN (Schlegl et al., 2017), MAD-Li (Li et al., 2018), and MAD-GAN (Li et al., 2019a).

Most recent models propose directly detecting anomalies in the latent representation and embeddings. THOC (Shen et al., 2020) proposed to use one-class classifiers based on multiple hyperspheres on the representations on all intermediate layers of a dilated RNN. NCAD (Carmona et al., 2021) uses a TCN to map context windows and suspect windows into a neural representation

and detect anomalies in the suspect window on the latent space with a contextual hypersphere loss.

2.2.3 Generative models with Alternating Back-Propagation

Virtually all current models, including our proposed approach, rely on mapping the original time series input into embeddings or a lower-dimensional latent space. DGHL, however, is trained with the Alternating Back-Propagation (ABP) algorithm, presented in (Han et al., 2017), and short-run MCMC, presented in (Nijkamp et al., 2021). ABP maximizes the observed likelihood directly; it does not rely on variational inference approximations or auxiliary networks such as discriminators. Instead, our approach uses MCMC sampling methods to sample from the true posterior to approximate the likelihood gradient.

Several generative models that rely on MCMC sampling, particularly Langevin Dynamics, have shown state-of-the-art performance on computer vision (Pang et al., 2020) and NLP (Pang et al., 2021) tasks. To our knowledge, this algorithm has not been used for time series forecasting and time series anomaly detection. We present the ABP algorithm in more detail in subsection 2.4.3.

2.3 What is an anomaly?

The definition of an anomaly is a fundamental question of the field. Anomalies are almost always subjective since they depend on the task and the user doing the labeling. In this chapter, we do not tackle this question but rather define and test models on a setting applicable to most applications.

Models are trained on a *clean* dataset without anomalies. The model, therefore, learns the *reference* distribution of the data. The trained model is then deployed in a test dataset, resembling the production deployment of the model. The test set can contain both reference and anomalous observations, and the task is to output a binary label to classify them. Figure 2.1 illustrates the setting.

To evaluate models, we use several public benchmark datasets with the F_1 score using anomaly labels, determined by experts, of the test set. These datasets were published by different organizations, such as NASA, and included labels created by experts. Note that the labels on the test set are only used to compare between models, so in real applications, users do not need to label anomalies to train models.

2.4 Methodology

2.4.1 DGHL

We propose DGHL, a novel Deep Generative model based on a top-down Convolution Network (ConvNet), which maps multivariate time series windows to a novel hierarchical latent space.

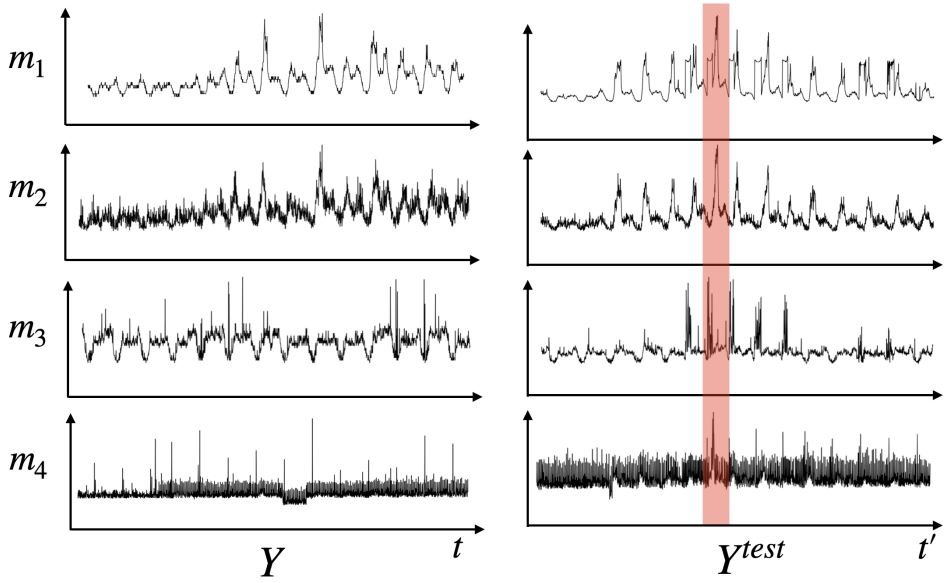


Figure 2.1: Evaluation setting. Models are trained on the train set without anomalies and are evaluated on the test set using the F1 score, using true labels provided by the benchmark dataset.

The model is trained by maximizing the observed likelihood directly with the Alternating Back-Propagation algorithm (Han et al., 2017) and short-run MCMC (Nijkamp et al., 2021).

2.4.2 Hierarchical Latent Factors

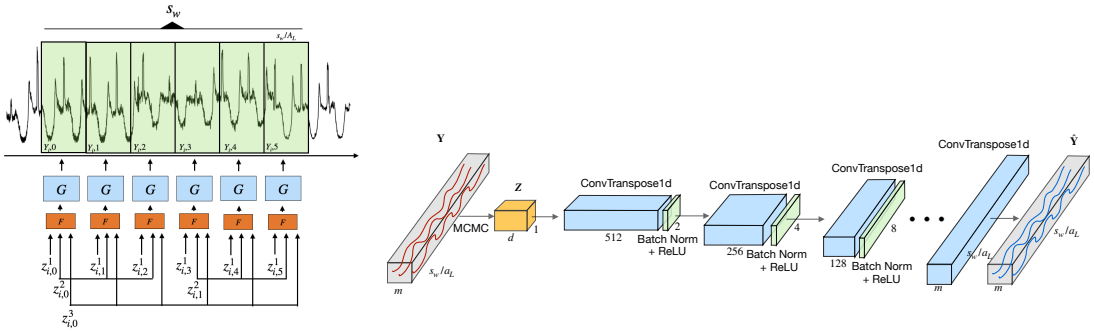
Let $\boldsymbol{Y} \in \mathbb{R}^{m \times s_w}$ be a window of size s_w of a multivariate time series with m features. The window $\boldsymbol{Y}$ is further divided in sub-windows of equal length $\boldsymbol{Y_j} \in \mathbb{R}^{m \times \frac{s_w}{a_L}}, j = 0, ..., a_L$. The structure of the hierarchy is specified by $\boldsymbol{a} = [a_1, ..., a_L]$, where L is the number of levels, and a_l determines the number of consecutive sub-windows with shared latent vector on level l, with $a_l \mid a_L$. Figure 2.2 presents an example of how $\boldsymbol{a}$ determines the hierarchy. Our model for each sub-window $\boldsymbol{Y_j}$ of $\boldsymbol{Y}$ is given by,

$$\begin{aligned} \boldsymbol{s_j} &= F_{\boldsymbol{\alpha}}(\boldsymbol{z}^1_{\lfloor \frac{j}{a_1} \rfloor}, ..., \boldsymbol{z}^L_{\lfloor \frac{j}{a_L} \rfloor}) \\ \boldsymbol{Y_j} &= G_{\boldsymbol{\beta}}(\boldsymbol{s_j}) + \boldsymbol{e_j} \end{aligned} \tag{2.1}$$

where F_α is the *State model*, G_β is the *Generator model*, $\boldsymbol{\theta} = [\boldsymbol{\alpha}, \boldsymbol{\beta}]$ are the parameters, $\boldsymbol{s_j} \in \mathbb{R}^d$ is the state vector, $\boldsymbol{e_j} \sim N(0, \boldsymbol{I}_D)$, and

$$\boldsymbol{Z} = \{\boldsymbol{z}^l_{\lfloor \frac{j}{A_l} \rfloor} \in \mathbb{R}^{d_l}\}_{l,j} \tag{2.2}$$

is the hierarchical latent factor space for window $\boldsymbol{Y}$. For the *State model*, we used a concatenation layer. For the *Generator model* we used a top-down Convolution Network (ConvNet),



(a) Hierarchial latent factors **(b)** Architecture

Figure 2.2: Panel (a): Example of hierarchical latent factor space for $\boldsymbol{a} = [1, 3, 6]$. On each level l, the latent vectors of a_l consecutive sub-windows are tied. For instance, the latent vector on the highest layer, L, is shared by all sub-windows of $\boldsymbol{Y}$. Panel (b): DGHL architecture

which maps an input state vector to a multivariate time series window. The *State model* for each sub-window has L latent vector inputs. On each level, l, the latent vectors of a_l consecutive sub-windows are tied. For instance, the latent vector on the highest layer, L, is shared by all sub-windows of $\boldsymbol{Y}$. Figure 1 shows an example of a hierarchical latent space with $\boldsymbol{a} = [1, 3, 6]$.

The key principle of the hierarchical latent space is to leverage dynamics on the time series, such as seasonalities, to encode the information on the latent space more efficiently with lower-dimensional vectors. The hierarchical latent space allows the generating of realistic time series of arbitrary length while preserving their long-term dynamics. The hierarchical structure can be incorporated as hyper-parameters to be tuned or pre-defined based on domain knowledge. For instance, hierarchies can correspond to the multiple known seasonalities on the time series.

The hierarchical latent space $\boldsymbol{Z}$ is jointly inferred using Langevin Dynamics. The relative size of the lowest level state vector and the upper levels controls the flexibility of the model. Larger lower hierarchy level vectors make the model more flexible, making it robust to normal changes or randomness in long-term dependencies of the time series and, therefore, reducing false positives by reducing the reconstruction error. Larger tied vectors will make the model stricter and better for detecting contextual anomalies. The model presented in (Han et al., 2017) can be seen as a single-level hierarchical latent space model, with $\boldsymbol{a} = [1]$, in the current framework.

Previous work, such as OmniAnomaly, incorporates transition models to learn dynamics in the latent space. We believe our proposed hierarchical latent factors structure has several advantages over transition models. First, the computational cost and training time are lower for the proposed model since it does not rely on sequential computation and, therefore, on back-propagation through time for training parameters. Second, transition models implicitly assume the dynamics are constant over time, a non-realistic assumption in many settings. Our solution allows the model to share information across windows to model long-term dynamics without relying on a parametric model which assumes constant dynamics.

2.4.3 Training with Alternating Back-Propagation

The parameters $\boldsymbol{\theta}$ of DGHL are learned with the Alternating Back-Propagation algorithm. First, the training multivariate time series $\boldsymbol{Y} \in \mathbb{R}^{m \times T}$ with m features and T timestamps is divided in consecutive windows of size s_w and step size s in a rolling-window fashion. Let $\{\mathbf{Y}^{(i)}, i = 1, ..., n\}$ be the training set of time series windows. Alternating Back-Propagation algorithm learns parameters $\boldsymbol{\theta}$ by maximizing the observed log-likelihood directly, given by,

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}) = \sum_{i=1}^{n} \log \int p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) d\mathbf{Z}^{(i)} \tag{2.3}$$

where $\mathbf{Z}^{(i)}$ is the latent vector for window i specified in equation 2. The observed likelihood $L(\boldsymbol{\theta})$ is analytically intractable. However, the gradients $L'(\boldsymbol{\theta})$ for a particular observation can be simplified to,

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}) &= \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)})} \frac{\partial}{\partial \boldsymbol{\theta}} \int p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) d\mathbf{Z} \\ &= \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{Z}^{(i)}|\mathbf{Y}^{(i)})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) \right] \end{aligned} \tag{2.4}$$

where $p_{\boldsymbol{\theta}}(\mathbf{Z}^{(i)}|\mathbf{Y}^{(i)}) = p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)}, \mathbf{Z}^{(i)}) / p_{\boldsymbol{\theta}}(\mathbf{Y}^{(i)})$ is the posterior. The expectation in the previous equation can be approximated with the Monte Carlo average by taking samples using MCMC. In particular, Alternating Back-Propagation takes a single sample of the posterior using Langevin Dynamics (Neal et al., 2011), a Hamiltonian Monte Carlo algorithm, which iterates,

$$\begin{aligned} \mathbf{Z}_{t+1}^{(i)} &= \mathbf{Z}_t^{(i)} + \frac{s}{\sigma_z} \frac{\partial}{\partial \mathbf{Z}^{(i)}} \log p_{\boldsymbol{\theta}}(\mathbf{Z}_t^{(i)}|\mathbf{Y}^{(i)}) + \sqrt{2s}\epsilon_t \\ &= \mathbf{Z}_t^{(i)} + \sqrt{2s}\epsilon_t + \\ &\frac{s}{\sigma_z} \left[(\mathbf{Y}^{(i)} - f(\mathbf{Z}_t^{(i)}, \boldsymbol{\theta})) \frac{\partial}{\partial \mathbf{Z}^{(i)}} f(\mathbf{Z}_t^{(i)}, \boldsymbol{\theta}) - \mathbf{Y}_t^{(i)} \right] \end{aligned} \tag{2.5}$$

where $\boldsymbol{\epsilon}_t \sim N(0, \boldsymbol{I}_D)$, t is the time step of the dynamics, s is the step size, and σ_z controls the relative size of the injected noise. This iteration is an explain-away process where latent factors are chosen such that the current residual on the reconstruction, $\mathbf{Y}^{(i)} - f(\mathbf{Z}_t^{(i)}, \boldsymbol{\theta})$, is minimized. With large values of σ_z, the posterior will be close to the prior, while small σ_z allows for a richer posterior. The iterative process is truncated to a predefined number of iterations, and the rejection step is not considered. As explained in (Neal et al., 2011), for an observation $\mathbf{Y}^{(i)}$, the resulting vector is a sample from an approximated posterior, $p_{\boldsymbol{\theta}}(\mathbf{Z}^{(i)}|\mathbf{Y}^{(i)})$. The Monte Carlo approximation of the gradient then becomes,

$$\begin{aligned} L'(\boldsymbol{\theta}) &\approx \frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{Z}^{(i)}, \mathbf{Y}^{(i)}) \\ &= \frac{1}{\sigma^2} (\mathbf{Y}^{(i)} - f(\mathbf{Z}^{(i)}, \boldsymbol{\theta})) \frac{\partial}{\partial \boldsymbol{\theta}} f(\mathbf{Z}^{(i)}, \boldsymbol{\theta}) \end{aligned} \tag{2.6}$$

2.4.4 Online Anomaly Detection

By learning how to generate time series windows based on the training data $\boldsymbol{Y}$, DGHL implicitly learns *reference* (non-anomalous) temporal dynamics and correlations between the multiple time series. In this subsection, we explain the proposed approach to reconstruct windows on unseen test data $\boldsymbol{Y}^{test}$ to detect anomalies.

In Online Anomaly Detection, we consider the test set $\boldsymbol{Y}^{test} \in \mathbb{R}^{m \times T_{test}}$ to be a stream of m time series. The goal is to detect anomalies (the evaluation is equivalent to a supervised setting with two classes) as soon as possible. As with the training set, $\boldsymbol{Y}^{test}$ is first divided in consecutive windows with the same parameters s_w and s. We propose to reconstruct and compute anomaly scores one window at a time.

Let $\boldsymbol{Y_{t^*}}$ be the current window of interest. The latent space $\boldsymbol{Z_{t^*}}$ is jointly inferred to reconstruct the target window, namely $\hat{\boldsymbol{Y}}_{\boldsymbol{t^*}}$. The anomaly score for a particular timestamp t in the window is computed as the Mean Square Error (MSE) considering all m time series, given by

$$s_t = \frac{1}{m} \sum_{i=1}^{m} (y_{i,t} - \hat{y}_{i,t})^2 \tag{2.7}$$

The size of the window s_w and step size s control how early anomalies can be detected. With a smaller s, anomaly scores for newer values in the stream are computed sooner. When $s < s_w$, consecutive windows have overlapping timestamps. In this case, scores are updated by considering the average reconstruction. In datasets with multiple entities (for instance, machines in SMD), we scale the scores by the accumulated standard deviation of scores before window t^*.

One main difference with the inference step during training is removing the Gaussian noise, ϵ_t, of the Langevin Dynamics update. The inferred factors then correspond to the maximum a posteriori mode, which in turn minimizes the reconstruction error conditional on the learned models F and G. This novel strategy makes DGHL unique among reconstruction-based models: it avoids overfitting during training by sampling from the posterior with Langevin Dynamics and minimizes the reconstruction error to reduce false positives by MAP estimation.

Many previous models rely on complex and unusual specific scores, but DGHL uses the simple MSE. The anomaly scores of our approach are interpretable since they can be disaggregated by the m features. Users can rank the contribution to the anomaly score of each feature to gather insights of the anomaly.

2.4.5 Online Anomaly Detection with missing data

The first step of the ABP algorithm is to infer latent vectors with Langevin Dynamics. This is an explain-away process where latent factors are chosen such that the current residual on the reconstruction, $\mathbf{Y} - f(\mathbf{Z}_t, \boldsymbol{\theta})$, is minimized. The model can intrinsically deal with missing data by inferring $\mathbf{Z}$, computing the residuals only on the observed signal $\mathbf{Y}_{obs}$. The inferred vectors then correspond to samples from the posterior distribution conditional on the available signal, $p_{\boldsymbol{\theta}}(\mathbf{Z}|\mathbf{Y}_{obs})$. Since no explicit learnable parameters map inputs to the latent space, the model is

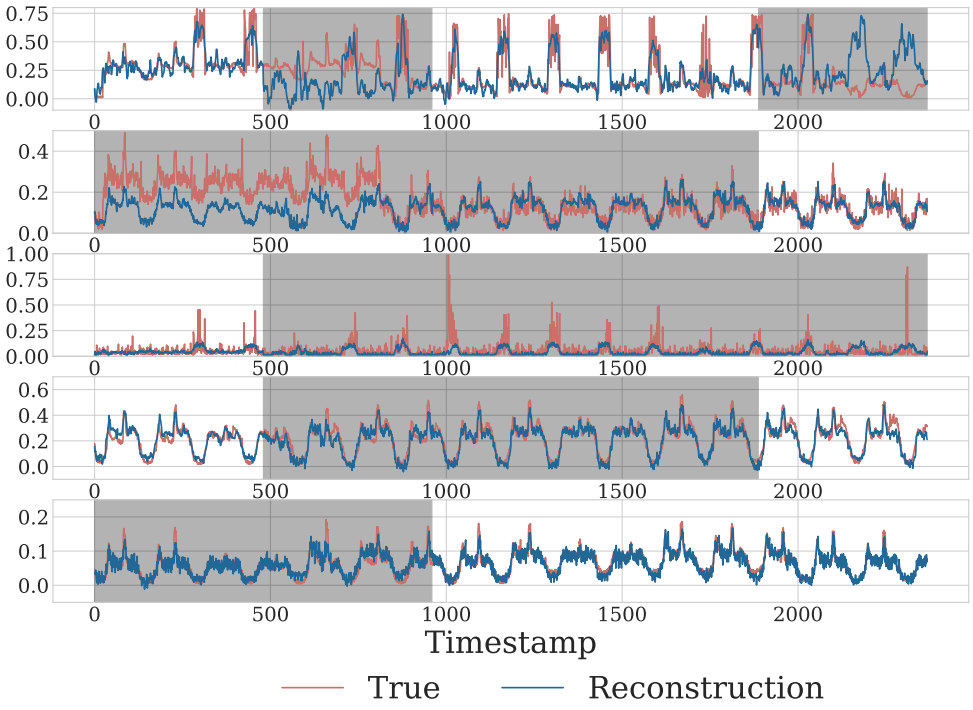


Figure 2.3: Occlusion experiment on machine-1-1 of the SMD. Red lines correspond to the actual values, and blue lines present the reconstructed time series with DGHL. Gray areas correspond to the occluded information during training.

more robust to missing values and outliers (masked as missing data). Generative models trained with ABP algorithm outperformed VAEs and GANs on experiments with missing information on computer vision and NLP tasks (Han et al., 2017).

Figure 2.3 shows an example of occluded data, for a subset of features of one machine of the SMD dataset. Occluded segments are marked in gray. First, DGHL is able to precisely reconstruct the observed data (white region), even when most features are missing. This is most relevant for the anomaly detection task since only the observed features are used to compute the anomaly score. Second, the model is able to recover missing data with great precision, which can be helpful in complete pipelines with downstream applications.

2.5 Experiments

In this section, we compare DGHL to current state-of-the-art (SoTA) models and simple one-line approaches such as *Mean deviation* and *Nearest Neighbors*. Table 2.1 presents the main results. Our methods consistently achieve the Top-2 F_1 scores, with overall performance superior to SoTA in four benchmark datasets from various domains. Moreover, our approach achieved the highest performance among all reconstruction-based and generative models on all datasets.

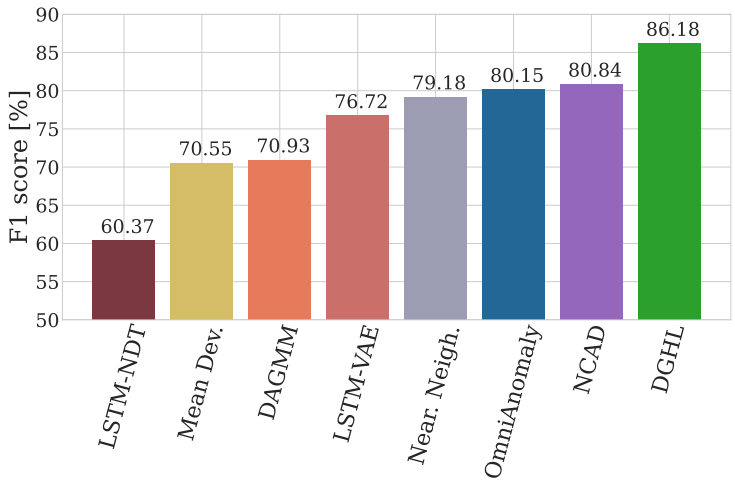


Figure 2.4: F_1 scores on SMD dataset using a single threshold across all machines.

2.5.1 Datasets

Server Machine Dataset (SMD) — Introduced in (Su et al., 2019), SMD is a multivariate time series dataset with 38 features for 28 server machines, monitored during 5 weeks. The time series includes common activity metrics in servers, such as CPU load, network, and memory usage, among others. Both training and testing sets contain around 50k timestamps each, with 5 % of anomalous cases. We trained separate models for each machine as suggested by the authors but with the same hyperparameters.

Soil Moisture Active Passive satellite (SMAP) and Mars Science Laboratory rover (MSL)— Published by NASA in (Hundman et al., 2018), they contain real telemetric data of the SMAP satellite and MSL rover. SMAP includes 55 multivariate time series datasets, each containing one anonymized channel and 24 variables encoding information sent to the satellite. MSL includes 27 datasets, each with one telemetry channel and 54 additional variables. Again, we trained separate models for each telemetry channel, considering additional variables as exogenous, i.e., only the anomaly score of the telemetry channel was used for detecting anomalies.

Secure Water Treatment (SWaT) — Is a public dataset with information of a water treatment testbed meant for cyber-security and anomaly detection research. It contains network traffic and data from 51 sensors for 11 days, 7 days of normal operation (train set), and 4 days with cyber attacks (test set).

2.5.2 Evaluation

We evaluate the performance of DGHL and benchmark models on the four datasets with the F_1-score, considering the anomaly detection problem as a binary classification task where the positive class corresponds to anomalies. Anomalies often occur continuously over a period of

Table 2.1: F_1 scores on benchmark datasets (the larger the better). The benchmark models performance was taken from (Shen et al., 2020), (Carmona et al., 2021) and (Zhao et al., 2020). The first place is marked in bold and the second place is marked in bold and italic. DGHL corresponds to the full model described in previous section, without Hierarchical factors corresponds to the simpler model with fully independent latent vectors for each window.

Model	**SMAP**	**MSL**	**SWaT**
Mean deviation (one-line)	57.61	68.91	85.71
Nearest Neighbors	75.10	90.01	86.72
AnoGAN	74.59	86.39	86.64
DeepSVDD	71.71	88.12	82.82
DAGMM	82.04	86.08	85.37
LSTM-VAE	75.73	73.79	86.39
MAD-GAN	81.31	87.47	86.89
MSCRED	85.97	77.45	86.84
OmniAnomaly	85.35	90.14	86.67
MTAD-GAT	90.13	90.84	-
THOC	***95.18***	93.67	**88.09**
NCAD	94.45	**95.60**	-
DGHL	**96.38 ± 0.72**	***94.08 ± 0.35***	***87.47 ± 0.22***
DGHL, without Hierarchial factors	94.87 ± 0.71	91.26 ± 0.71	87.08 ± 0.12
DGHL, with encoder, no Hier. factors	78.41 ± 0.92	87.10 ± 0.54	86.39 ± 0.32

time, creating anomalous segments. (Xu et al., 2018) proposed an adjustment approach where the predicted output is re-labeled as an anomaly for the whole continuous anomalous segment if the model correctly identifies the anomaly in at least one timestamp. We use this adjustment technique for SMAP, MSL and SMD datasets to make results comparable with existing literature. Moreover, we followed the common practice of comparing the performance using the best F_1-score, by choosing the best threshold on the test set. For SMAP, MSL, and SMD, we use a single threshold through the entire dataset (not different thresholds for each machine or channel).

To compare our results with previous work, we follow the train, validation, and test split described in (Shen et al., 2020) for SMAP, MSL, and SWaT. For SMD we use the train and test splits described in (Su et al., 2019). All architecture hyper-parameters of the *Generator* model, optimization hyper-parameters, and all hyper-parameters of the Langevin Dynamics were kept constant across the four datasets.

We compare DGHL to current state-of-the art models, such as THOC(Shen et al., 2020), NCAD(Carmona et al., 2021), and MTAD-GAT (Zhao et al., 2020); and previous widely used models such as AnoGAN(Schlegl et al., 2017), DeepSVDD(Ruff et al., 2018), DAGMM(Zong et al., 2018), OmniAnomaly, (Su et al., 2019), MAD-GAN(Li et al., 2019a) and LSTM-VAE(Park et al., 2018). We also include simple one-line and non deep learning approaches such as *Mean deviation* and *Nearest Neighbors*. *Mean deviation* uses the average deviation to the mean of each feature as an anomaly score. The latter uses the average distance to the k nearest windows of the training

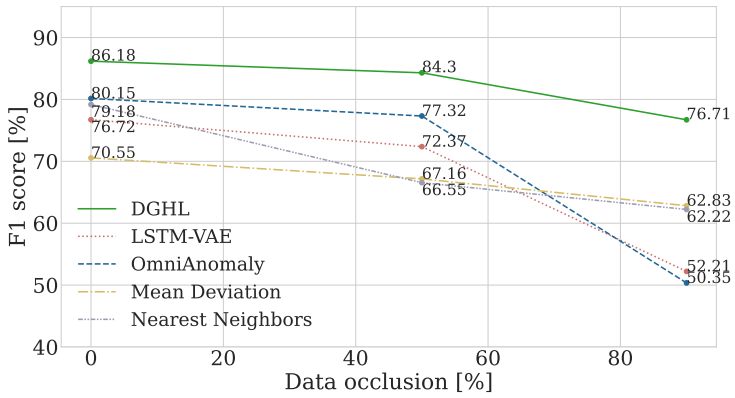


Figure 2.5: F_1 scores for DGHL and LSTM-VAEbenchmark for occlusion experiments on SMD for three levels of occlusion probability p, 0, 0.5, 0.9 and $r = 5$.

set as the anomaly score. Finally, we include two additional versions of DGHL removing the key contributions of our work. First, we remove the hierarchical factors ($\mathbf{a} = [1]$), and second, we replace the Langevin Dynamics algorithm for inferring latent factors with a convolutional encoder.

Table 2.1 shows the F_1 scores for DGHL, and the benchmark models for SMAP, MSL, and SWaT datasets. Our methods consistently achieve the Top-2 F_1 scores, with overall performance superior to state-of-the-art such as MTAD-GAT (Zhao et al., 2020), THOCand NCAD. Moreover, our approach achieved the highest performance between all reconstruction-based and generative models on all datasets. Figure 2 shows the F_1 scores for SMD dataset. DGHL achieved a score of 86.18 ± 0.66, outperforming all benchmark models. Our model without hierarchical factors had a score of 80.84 ± 0.40, and with encoder had a score of 76.59 ± 0.78.

DGHL significantly outperformed simple baselines in all datasets. The one-line solution ranked worst consistently. Nearest Neighbors, however, achieved a better performance than several complex models in all datasets with a fraction of the computational cost, demonstrating how simple models need to be considered to understand the benefits of recent models. DGHL outperforms other pure reconstruction-based models because inferring latent vectors for computing anomaly scores provides several advantages. First, it provides additional flexibility and generalization capabilities to prevent false positives, which is instrumental in noisy or non-constant temporal dynamics datasets. Second, it helps to reduce the lasting impact of anomalies on the reconstruction error over time, reducing false positives once anomalies end.

DGHL took an average of 2 minutes to train for each entity (e.g., one machine of SMD or one channel of SMAP) consistently across datasets. For instance, the training time was around 60 minutes for SMD and MSL, and 100 minutes for SMAP. This is comparable to other state-of-the-art models' self-reported training times, such as NCAD, and is faster than RNN-based models. For instance, OmniAnomaly took an average of 20 minutes to train each model for each machine on the SMD dataset. The inference time varies depending on the length of the test set. The average time to infer 3000 timestamps (average downsampled SMD test set), with $s_w = 32$, was lower than 5 seconds.

2.5.3 Online Anomaly Detection with missing data

All current benchmark datasets in the time series anomaly detection literature assume *perfect* data. However, this is not usually the case in real scenarios, with issues such as missing values, corrupted data, and variable features. This section presents the first experiments to assess the robustness of current state-of-the-art models to common data issues such as missing values. In particular, we adapt the popular occlusion experiments from computer vision literature for training models with incomplete data.

We define the occlusion experiments with two parameters. First, the original time series $\boldsymbol{Y} \in \mathbb{R}^{m \times T}$, is divided in r segments of equal length, $\boldsymbol{Y_i} \in \mathbb{R}^{m \times \frac{T}{r}}$. Second, each feature m in each segment is occluded for model training or inference with probability p.

We assess the robustness of models to incomplete training data with occluding experiments on the SMD dataset for different levels of r and p and using F_1 scores to evaluate performance. Figure 2.5 shows the F_1 score for DGHL, LSTM-VAE, and OmniAnomaly. DGHL achieves the highest scores consistently, with increasing relative performance on higher data occlusion probability. Moreover, DGHL maintained high F_1 scores even with up to 90% of missing information, without any changes to the hyperparameters, architecture, or training procedure.

2.6 Anomaly Threshold Selection

2.6.1 Motivation

The evaluation in the last section was based on prior work that compares performance with the *optimal* F_1. This metric uses the true labels of the test set to define the best threshold that maximizes the F_1 score. While this methodology is useful for comparing models, it lacks practical applications as the true anomaly labels are unavailable during inference.

Few algorithms exist for threshold selection, as most research endeavors have focused on developing models. Regarding performance, the threshold definition can be as important as the underlying model. Moreover, automatic thresholding can benefit large systems where defining manual thresholds for each instance is not feasible, and a common pre-defined threshold is too restrictive.

In this section, we evaluate existing methods and propose a novel Synthetic Anomalies Automatic Thresholding algorithm (SAAT).

2.6.2 Methodology

Let $Y_{train} \in \mathbb{R}^{m \times T}$ be a stream of non-anomalous m time series of length T, F an anomaly detection model trained on Y_{train} which for each timestamp t produces an anomaly score s_t. Let $Y_{test} \in \mathbb{R}^{m \times T'}$ be the test set, a stream of m labeled time series observed after the train set. The model F labels each timestamp t as an anomaly if $s_t > h$, i.e., the score is greater than a

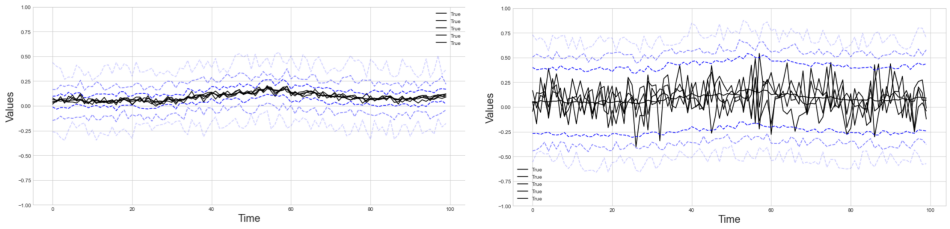


Figure 2.6: Intuition of estimating normal variance on common windows (states) on a time series. Each plot shows a window (state), and black lines show multiple instances (neighbors). Some states have higher *normal* variance, so injected anomalies should be larger.

threshold h. The goal of automatic thresholding is to define the threshold $h(F, Y_{train})$, such that it maximizes $T(Y_{test}, F, h)$, where T is a performance metric (such as F1-score).

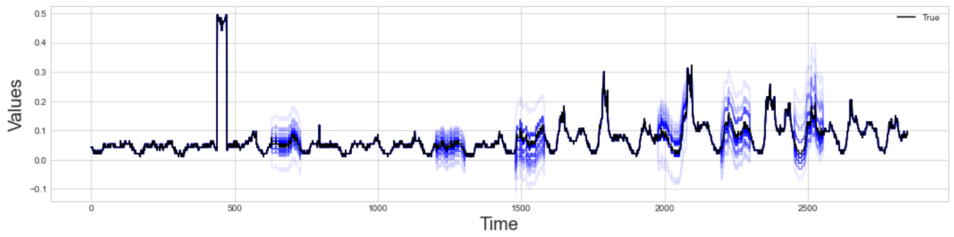
Algorithm 2.1 presents the SAAT methodology, maximizing the F_1 score on the augmented $\hat{Y}$ data with injected synthetic anomalies. First, SAAT randomly samples n disjoint windows of size s_w. The intuition is that these windows represent different states of the stream. Next, for each window, it finds similar windows (other instances of the same state) based on a distance $d(Y_i, Y_j)$ between two windows. In the experiments, we set d as the Euclidean distance. Next, it generates synthetic anomalies based on the Generator $G(\Phi, \hat{d}_n)$ that parameterize anomalies based on a distribution Φ and the average distance $\hat{d}_n$ on the true training data. The goal of using $\hat{d}_n$ is for the generator to produce true out-of-distribution anomalies outside the normal behavior of the time series, with the intuition presented in Figure 2.6. Finally, injected anomalies are labeled, and optimal h^* is computed such that it maximizes the F_1 score. Figure 2.7 presents an example of injected scale anomalies and how the optimal threshold is defined.

Algorithm 2.1 SAAT

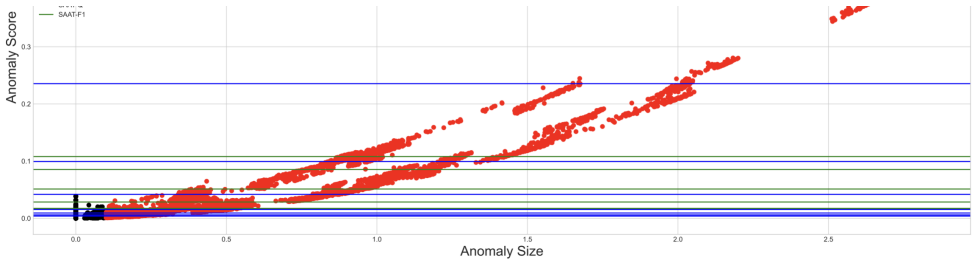
Input: train data$\mathbf{Y}$, model $F_{\boldsymbol{\theta}}$, distance $d(Y_i, Y_j)$, distribution Φ, anomaly generator $G(\Phi, d)$, parameters n s_w.
Output: h^*.
Randomly select n disjoint windows from Y, and find similar windows for each with motif discovery algorithm based on $d(Y_i, Y_j)$.
$\hat{d}_n \leftarrow$ Average d for each n group of windows.
$\hat{Y} \leftarrow$ Inject anomalies from $G(\Phi, \hat{d}_n)$ on Y.
$h^* \leftarrow \arg\max F_1(F, \hat{Y})$

2.6.3 Synthetic Anomalies and Domain Knowledge

How to generate synthetic anomalies is an interesting and important problem in the field. SAAT consists of a general framework, which is agnostic to any particular type of injected anomalies.



(a) Synthetic anomalies



(b) Anomaly scores from vs synthetic anomaly size

Figure 2.7: Optimal threshold is selected to separate true negative (in black) and true positives (in red) based on the anomaly scores produced by model F.

For the following experiments, we consider the 9 different types of synthetic anomalies from (Goswami et al., 2023).

One key advantage of SAAT is the capability to incorporate domain knowledge from experts through the choice of synthetic anomalies generator. As mentioned in section 2.3, anomalies are identified by experts in each application domain. SAAT allows users to define the synthetic anomalies to match true expected anomalies in production to improve anomaly detection performance by defining optimal thresholds that minimize errors.

2.6.4 Experiments

We repeat the anomaly detection experimental setting from section 2.5 for SMD, SMAP, and MSL datasets, but varying the threshold method instead of only using the optimal F_1 score. We compare the performance of SAAT against two popular alternatives in the literature: Peak-over-threshold (POT), proposed in (Siffer et al., 2017), and modeling anomaly scores as a Gaussian Distribution (`Gaussian`). In terms of models, we consider the simple baselines from the previous section, the `LSTM-VAE` and `LSTM-NDT`, and two new models: Random Cut Forest (RCF) and the recent `AnoTrans` (Xu et al., 2022a).

Table 2.2 presents the main results. First, SAAT improves over alternative methods consistently across models and datasets. The improvements are larger for more complex deep learning and machine-learning models. The top performing models, regardless of the threshold algorithm, are DGHL and `AnoTrans`.

Table 2.2: F_1 scores on benchmark dataset (larger is better). First place, excluding `Optimal`, is marked in bold.

		Gaussian	POT	SAAT	Optimal
SMD	Mean deviation	62	64	**69**	84
	Nearest Neighbors	58	52	**72**	89
	LSTM-NDT	57	59	**66**	70
	LSTM-VAE	50	55	**63**	89
	OmniAnomaly	38	40	**74**	88
	RCF	57	51	**70**	71
	AnoTrans	48	44	**81**	90
	DGHL	59	60	**84**	92
SMAP	Mean deviation	**72**	62	68	79
	Nearest Neighbors	71	39	**74**	84
	LSTM-NDT	56	52	**71**	95
	LSTM-VAE	**74**	60	74	83
	RCF	58	41	**61**	65
	AnoTrans	89	85	**91**	97
	DGHL	88	86	**89**	97
MSL	Mean deviation	52	**55**	50	86
	Nearest Neighbors	48	50	**66**	93
	LSTM-NDT	52	48	**67**	93
	LSTM-VAE	53	49	**68**	91
	RCF	59	47	**64**	92
	AnoTrans	62	64	**78**	97
	DGHL	61	67	**76**	96

2.7 Discussion and Conclusion

DGHL outperforms SoTA baselines and simple approaches by the current experimental standards of the literature. Although DGHL relies on MCMC for posterior sampling, it remains computationally efficient thanks to a lower number of training iterations needed. The ablations studies presented in Table 2.1 demonstrate the complementary gains of our two main contributions, namely, a novel hierarchical latent representation and training the *Generator* with the alternating back-propagation algorithm.

(Wu and Keogh, 2021) strongly argue that some of the benchmark datasets used in our experiments have mislabeled observations and are therefore worthless to compare to. While we agree this adds noise to the evaluation metrics, we believe the consistent improvement of our model (and current SoTA deep learning models) demonstrates their superior performance on this task over simpler approaches. We also observed the SMD dataset does not have a considerable amount of mislabeled observations that could significantly alter the results [1]. We decided to use

[1]Even though we do not know the ground truth, most labeled anomalies seem to relate to some form of a rare

these benchmark datasets for comparison purposes with existing methods.

Most of the anomalies in the benchmark datasets used in this paper can be easily identified by humans (e.g. large spikes), as noted in (Wu and Keogh, 2021). Accurately detecting such anomalies is relevant for many applications, particularly large-scale automatized systems, for which the current benchmarks provide representative estimates of the relative performance of models. Identifying contextual anomalies, which can be hard to detect even for humans, is also highly relevant, but current benchmarks do not provide insights on the performance of models on such tasks. Creating relevant benchmark datasets with contextual anomalies is a pressing necessity.

Table 2.2 on the results of threshold selection shows there is still a considerable gap between all methods and the optimal threshold, showing the need for further developing these methods. It also raises concerns about using the optimal F_1 as an evaluation metric for comparing models, as it does not have practical applications directly.

In summary, we present DGHL, a state-of-the-art Deep Generative model for multivariate time series anomaly detection. The proposed model maps time series windows to a novel hierarchical latent space representation, which leverages the time series dynamics to encode information more efficiently. Our model has several advantages over existing methods: i. shorter training times, ii. demonstrated superior performance on several benchmark datasets, and iii. better robustness to missing values and variable features. Second, we present SAAT, an algorithm for automatic optimal threshold selection based on injecting synthetic anomalies and optimizing the desired performance metric.

pattern, and we did not observe clear anomalies (e.g. large spikes) labeled as normal observations.

Chapter 3

Robust Multivariate Forecasting and Imputation

3.1 Motivation

Multivariate time series forecasting is an essential task in various domains. Forecasts are a key input to optimize the production and distribution of goods (Bose et al., 2017), predict patient outcomes in healthcare (Chen et al., 2015), plan electricity production (Olivares et al., 2022b), build financial portfolios (Emerson et al., 2019), among other examples (Challu et al., 2021; Challu et al., 2022b). Due to its high potential benefits, researchers have dedicated many efforts to improving the forecasting models' capabilities, with breakthroughs in architectures leading to increased performance and scalability (Benidis et al., 2022).

While neural forecasting models differ in architecture design and how they model temporal and inter-feature relations, they share some key characteristics: (i) they explicitly or implicitly rely on encoders that map historical values as inputs to embeddings or latent representations; (ii) their architecture contains explicit operations (usually multiplication or addition) between learnable parameters and inputs; and (iii) once the model is trained, the forecast (or in the case of probabilistic models, the distribution parameters) are fixed, as the inputs and parameters fully determine it.

We identify and empirically document the limitations of state-of-the-art (SoTA) methods following these principles in forecasting settings with missing data, a problem that has been largely understudied. Missing values are a generalized data problem with common causes including faulty sensors and human error (Tashiro et al., 2021; Yi et al., 2016), and are predominant in high-stake domains such as healthcare and finance where settings with up to 80% missing data are common (Silva et al., 2012). Missing data can severely degrade performance for most applications and downstream tasks (Figure 1.b). Designing robust methods that can intrinsically handle missing values or developing imputation models to accurately recover missing data can, therefore, provide enormous benefits.

To this end, we propose *Temporal Inference Networks* (TIN), a novel family of time series

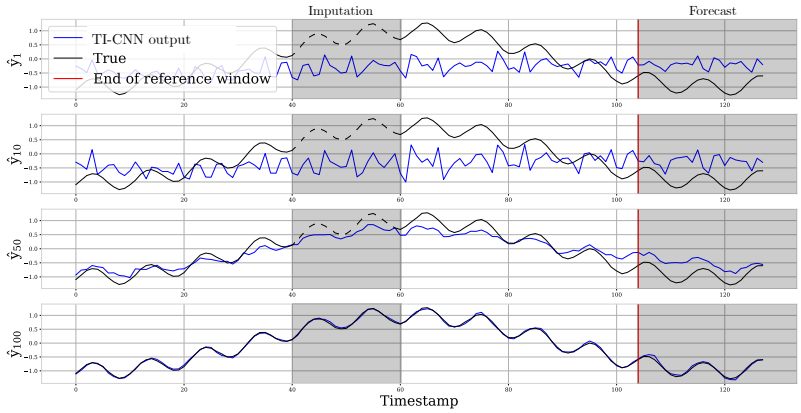


Figure 3.1: TIN-CNN's output evolution during latent factors inference, where $\hat{y}_i$ is the output after i inference steps. The model generates the complete window, simultaneously imputing missing data and forecasting future values, using the available information on the reference window (white region).

forecasting and imputation models that challenges the established principles of most SoTA models. A TIN network consists of three components: the *Latent Factors inference* procedure, a *Temporal Dynamic Module* (TDM), and the Generator network. The Generator network generates complete time series windows from latent factors, interpolates missing data, and forecasts future values. Latent factors are inferred by matching the Generator's output on available past observations, minimizing a reconstruction loss using gradient descent methods. The TDM module endows the latent factors with temporal dynamics and imposes strict temporal dependencies along the window. By combining the three modules, a TIN network does not contain operations between learnable parameters and inputs and can adjust the imputation and forecasts to match the latest behavior of the data.

To the best of our knowledge, TIN is the first approach to achieve SoTA performance in settings with full data and missing values while simultaneously performing forecasting and imputation tasks. Forecasting and imputation have been studied separately, and virtually all work proposing new methods has specialized in one task. Our method can greatly simplify production systems by unifying both into a single efficient and parsimonious model. The contributions are summarized below:

- **Latent Factors inference**: methodology to infer latent factors of a Generator network that replaces parametric encoders, theoretically motivated as a MAP estimation of the posterior distribution of latent factors.
- **Temporal Dynamic Module:** representation of a multivariate time series window on a shared latent space with temporal dynamics.
- **TIN-CNN**: multivariate forecasting and imputation model with a Convolution Neural Network (CNN) Generator that combines the previous techniques to simultaneously achieve SoTA performance on forecasting and imputation tasks in several benchmark datasets.

3.2 Related Work

3.2.1 Multivariate Forecasting

The earliest multivariate forecasting methods are the Vector Autoregression (VAR) statistic models from the 1980's (Sims, 1980; Granger, 1969), used mainly by econometricians to forecast macroeconomic indicators. In recent years, several multivariate neural forecasting approaches have been proposed using a wide range of architectures. The success of Transformers (Vaswani et al., 2017) in sequential data, such as natural language processing (NLP) and audio processing, inspired many multivariate models with attention mechanisms. For example, the `Informer` (Zhou et al., 2020) introduces a Prob-sparse self-attention to reduce the quadratic complexity of vanilla Transformers; the `Autoformer` (Wu et al., 2021) proposed a decomposition architecture in trend and seasonal components and the Auto-correlation mechanism. Other approaches incorporated Graph Neural Networks (GNN) to model complex relations between a large number of time series. Some examples include the `GraphWaveNet`(Wu et al., 2019) and `StemGNN`(Cao et al., 2020) models.

3.2.2 Time series Imputation

The standard practice to handle missing data is filling the missing information, a process called *imputation*. Simple interpolation alternatives include replacing missing values with zeros, the mean, the most recent value (naive), and linear interpolation. Most recent deep learning approaches consist of Generative Adversarial Networks (GANs) and RNN-based architectures. Some notable examples are E2gan (Luo et al., 2019), Brits (Cao et al., 2018), and NAOMI (Liu et al., 2019). More recent approaches include the CDSI (Tashiro et al., 2021) model, a score-based diffusion auto-regressive architecture that produces a distribution for the imputed values.

3.2.3 Alternating back-propagation

The method we propose to infer latent vectors is inspired by the Alternating back-propagation algorithm (ABP) for Generative models (Han et al., 2017). The key idea of this algorithm is to sample latent vectors from the posterior distribution with MCMC methods and train a Generative model by maximizing the observed likelihood directly. Generative models trained with ABP do not need an encoder, such as Variational Autoencoders (VAE), or Discriminator networks, such as GANs. Some recent work extended the original architecture with energy-based models for Computer Vision (Pang et al., 2020) and LSTM networks for text generation (Pang et al., 2021).

3.3 Notation and Problem definition

We introduce a notation that we believe is lighter than the standard notation while being intuitive and formally correct. Let $\mathbf{Y} \in \mathbb{R}^{M \times T}$ be a multivariate time series with M features and

T timestamps. Let $\mathbf{Y}_{a:b} \in \mathbb{R}^{M\times(b-a)}$ be the observed values for the interval $[a, b)$, that is, $\mathbf{Y}_{0:t}$ is the set of t observations of $\mathbf{Y}$ from timestamp 0 to timestamp $t-1$ while $\mathbf{Y}_{t:t+H}$ is the set of H observations of $\mathbf{Y}$ from timestamp t to timestamp $t+H-1$. Let $y_{m,t} \in \mathbb{R}$ be the value of feature m at timestamp t. Finally, let $\mathbf{M} \in \{0,1\}^{M\times T}$ be the observation mask indicating data availability.

The multivariate point forecasting task comprises predicting the future values of a multivariate time series sequence based on past observations. The task of a forecasting model at timestamp t is to predict the future H values, denoted by $\hat{\mathbf{Y}}_{t:t+H}$, based on the previous history $\mathbf{Y}_{0:t}$. The imputation task, at a timestamp t, consists of recovering historic missing values in $\mathbf{Y}_{0:t}$ from available values.

We evaluate the performance for both tasks with two common metrics, mean squared error (MSE) and mean absolute error (MAE), given by equation 3.1 (Hyndman and Athanasopoulos, 2018a).

$$\text{MSE} = \frac{1}{MH}\sum_{h=0}^{H-1}\sum_{m=1}^{M}(y_{m,t+h} - \hat{y}_{m,t+h})^2 \quad \text{MAE} \quad = \frac{1}{MH}\sum_{h=0}^{H-1}\sum_{m=1}^{M}|y_{m,t+h} - \hat{y}_{m,t+h}| \tag{3.1}$$

3.4 Methodology

Temporal Inference Networks are composed of three major components: the *Latent Factors Inference* procedure, a *Temporal Dynamic Module* (TDM), and the Generator network, $G_{\boldsymbol{\theta}}$. Details and motivations of each component are presented in the following subsections. The core idea of TIN is to generate multivariate time series windows of size $s_w = L + H$ from a latent vector $\mathbf{z} \in \mathbb{R}^d$. The window comprises the reference window of size L, and the forecast window of size H. The model *infers* the optimal latent vector by minimizing the reconstruction error on the available values of the reference window, $\mathbf{Y}_{t-L:t}$. By generating the complete window, TIN can impute missing data in the reference window and forecast future values, $\hat{\mathbf{Y}}_{t:t+H}$. The overall architecture, with a Convolution Network as a Generator, is illustrated in Figure 3.2. The main steps are given by:

$$\hat{\mathbf{Y}}_{t-L:t+H} = [\hat{\mathbf{Y}}_{t-L:t}, \hat{\mathbf{Y}}_{t:t+H}] = G_{\boldsymbol{\theta}}(\mathbf{E}) \tag{3.2}$$

$$\mathbf{E} = \text{TDM}(\mathbf{z}^*) \tag{3.3}$$

where $\mathbf{z}^*$ is the optimal *inferred* latent vector, given by:

$$\mathbf{z}^* = argmin_{\mathbf{z}} \quad L(\mathbf{Y}_{t-L:t}, \hat{\mathbf{Y}}_{t-L:t}(\mathbf{z})) \tag{3.4}$$

3.4.1 Latent factors inference

The proposed latent factor inference procedure is based on the Alternating back-propagation algorithm (ABP) (Han et al., 2017) for training generative models in computer vision (CV). A

single generator architecture is trained by maximizing the observed likelihood directly. To achieve this, ABP samples latent vectors from the posterior distribution $P(\mathbf{Z}|\mathbf{Y})$ using MCMC methods such as Langevin dynamics. Generative models trained with ABP demonstrated superior performance in recovering missing segments of images and videos over Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN).

Let $\mathbf{y} \in \mathbb{R}^D$ be a D-dimension data vector (such as an image or time series window), TDM the *Temporal Dynamic Module*, G the Generator network with parameters $\boldsymbol{\theta}$,

$$\mathbf{y} = G_{\boldsymbol{\theta}}(\text{TDM}(\mathbf{z})) + \boldsymbol{\epsilon} \tag{3.5}$$

where $\mathbf{z} \sim N(0, \boldsymbol{I}_d)$, $\boldsymbol{\epsilon} \sim N(0, \sigma^2 \boldsymbol{I}_D)$, $\mathbf{z} \in \mathbb{R}^d$ are latent factors and $d < D$. Let $\{\mathbf{y}^{(i)}, i = 1, ..., n\}$ be n training observations. Given that TDM is a fixed set of operations and does not contain learnable parameters, it can be excluded from the next equations for simplicity. In Alternating Back-Propagation, parameters $\boldsymbol{\theta}$ are trained by maximizing the observed log-likelihood:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log p_{\boldsymbol{\theta}}(\mathbf{y}^{(i)}) = \sum_{i=1}^{n} \log \int p_{\boldsymbol{\theta}}(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}) d\mathbf{z}^{(i)} \tag{3.6}$$

The gradients for observation i are given by,

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{y}^{(i)}) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}|\mathbf{y}^{(i)})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{y}^{(i)}, \mathbf{z}^{(i)}) \right] \tag{3.7}$$

where $p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}|\mathbf{y}^{(i)})$ is the posterior distribution. Next, the expectation is approximated with Monte Carlo by taking a single sample $\mathbf{z}^*$ from the posterior distribution with approximate Langevin Dynamics, by iterating

$$\mathbf{z}_{j+1}^{(i)} = \mathbf{z}_j^{(i)} + \frac{s}{\sigma_z} \frac{\partial}{\partial \mathbf{z}^{(i)}} \log p_{\boldsymbol{\theta}}(\mathbf{z}_t^{(i)}|\mathbf{y}^{(i)}) + \sqrt{2s}\boldsymbol{\epsilon}_j \tag{3.8}$$

where $\boldsymbol{\epsilon}_j \sim N(0, \boldsymbol{I}_D)$, s is the step size, $\boldsymbol{z}_0^{(i)} \sim N(0, \boldsymbol{I}_D)$, and σ_z controls the annealing or tempering.

We reformulate the posterior sampling as a minimization problem presented in equation 3.4, which aims to minimize the mean square error (MSE) between the reconstruction and ground truth $\mathbf{y}^{(i)}$. This formulation is equivalent to the maximum a posteriori estimation (MAP) of $p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}|\mathbf{y}^{(i)})$ assuming Gaussian distributions. This methodology also allows for more robust gradient-based methods than equation 3.8. In particular, we use ADAM (Kingma and Ba, 2014) optimizer. Given that this optimization problem is non-convex, converging to the global minima is not guaranteed. We perform a bagging procedure (Breiman, 1996) to produce more stable reconstruction by inferring multiple latent vectors with different initializations $\boldsymbol{z}_0^{(i)}$, and taking the median as the final forecast. Figure 3.1 demonstrates how TIN-CNN's output evolves during the inference of $\mathbf{z}$. Section 3.4.4 presents how TIN-CNN is trained by alternating the inference and parameter learning steps.

Temporal Inference Networks, relying on latent factor inference, have two key properties: (i) the Generator model does not contain any operation between parameters and historical values (see Figure 3.2), and (ii) the forecast is not fixed after the model is trained, the model will iterate different forecasts adapting it to match the latest dynamics on the reference window. These properties improve the model's robustness to missing data, as the latent factors are not a direct function of the incomplete inputs. Additionally, the reconstruction loss is masked only to consider available data points, and therefore latent factors correspond to the MAP estimation of the posterior conditional on the observed values, $p_{\boldsymbol{\theta}}(\mathbf{z}^{(i)}|\mathbf{y}_o^{(i)})$. When the reference window has missing values, the latent factors are inferred following:

$$\mathbf{z}^* = argmin_{\mathbf{z}} \quad L(\mathbf{M}_{t-L:t} \circ \mathbf{Y}_{t-L:t}, \mathbf{M}_{t-L:t} \circ \hat{\mathbf{Y}}_{t-L:t}(\mathbf{z}))$$

where $\circ$ is the element-wise matrix multiplication. As demonstrated in the experimental section results in Tables 3.1 and 3.3, and in congruity with previous studies in CV and NLP such as (Han et al., 2017; Pang et al., 2020), inferring latent vectors provides superior robustness to missing data.

3.4.2 Temporal Dynamic Module

The previous procedure allows the model to encode the characteristics of a time series window in latent factors $\mathbf{z}$. Time series data is characterized by exhibiting temporal dependencies between consecutive observations. The *Temporal Dynamic Module* (TDM) is proposed to endow the learned representation with temporal dynamics in the form of an embedding $\mathbf{E} \in \mathbb{R}^{d \times d_t}$, where d is the number of elements, and d_t is the temporal length. This temporal embedding imposes strict temporal dependencies along the reference window and with the forecasting window, allowing the model to better impute and forecast unobserved regions of the sequence. The embedding $\mathbf{E}$ is passed as input to the Generator to produce the final output.

The TDM module does not contain learnable parameters. Instead, it enforces temporal dynamics with a predefined set of d template functions, forming a library $\mathbf{B} \in \mathbb{R}^{d \times d_t}$. The library includes patterns commonly found in time series: trends, represented by polynomial functions, and seasonalities, by harmonic functions. The final library is the row-wise concatenation of the three following matrices:

$$\begin{aligned}
\mathbf{B}_{i,t}^{trd} &= t^i \qquad\qquad \text{, for } i \in \{0, ..., p\}, t \in \{0, ..., d_t\} \\
\mathbf{B}_{i,t}^{cos} &= \cos(2\pi i t), \text{for } i \in \{1, ..., \frac{s_w}{2}\}, t \in \{0, ..., d_t\} \\
\mathbf{B}_{i,t}^{sin} &= \sin(2\pi i t), \text{for } i \in \{1, ..., \frac{s_w}{2}\}, t \in \{0, ..., d_t\}
\end{aligned} \tag{3.9}$$

where p is a hyperparameter controlling the polynomial basis's max degree. The final size of the basis d is equal to $s_w + p + 1$, and we set $d_t = \frac{s_w}{2}$. Finally, $\mathbf{E}$ is composed by relating one element of $\mathbf{z}$ with a template function. The i-th row of $\mathbf{E}$ is then given by,

$$\mathbf{E}_{\mathbf{i},:} = z_i^* \mathbf{B}_{i,:} \tag{3.10}$$

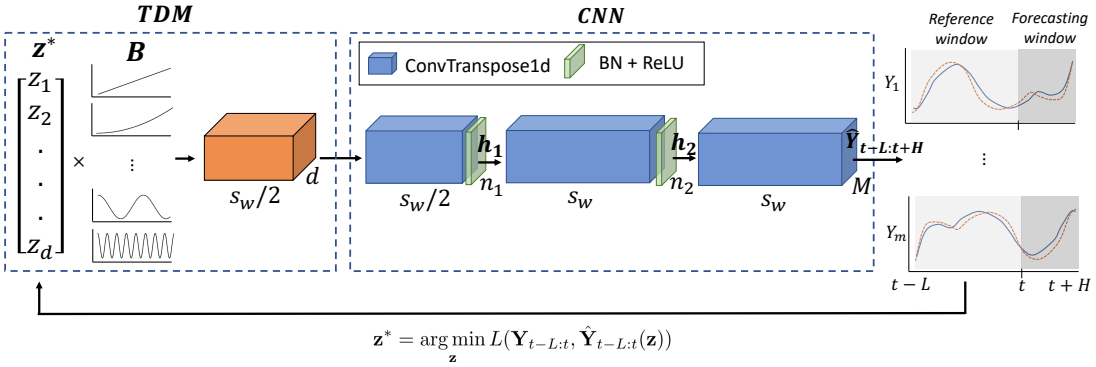


Figure 3.2: TIN-CNN architecture. Latent factors $\mathbf{z}$ are inferred with Gradient Descent methods, minimizing the reconstruction error on the reference window. The TDM encodes shared temporal dynamics into Fourier waves and polynomial functions. The Convolution Network generates the time series window by sequentially mixing the embedding components and refining the output.

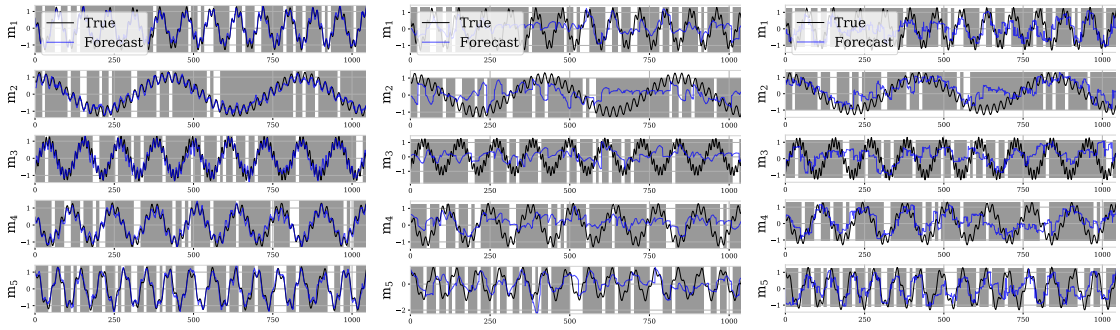
The temporal embedding $\mathbf{E}$ can also be motivated as a *latent space spectral decomposition* (LSSD) that encodes shared temporal dynamics, as the latent vector $\mathbf{z}^*$ selects the relevant trend and frequency bands for the current window. The principle of encoding temporal dynamics in template functions is inspired by previous work on signal processing to synthesize and generate audio (Engel et al., 2020; Shan et al., 2022). The NBEATS (Oreshkin et al., 2019) contains a similar *interpretable basis expansion* component, but it is used as a final layer to decompose the final forecast.

3.4.3 Generator Network

The final component of TIN is the Generator (Decoder) Network, which will produce the forecast and reconstruction of the reference window $\hat{\mathbf{Y}}_{t-L:t+H}$ from the temporal embedding $\mathbf{E}$. The Generator contains all the trainable parameters of the model, which are fixed after the model is trained. The general TIN framework is compatible with different architectures. In this work, we propose using a Top-Down Convolution Network (CNN), given its remarkable capability to generate highly complex objects by modeling interactions between channels and features. A diagram of the CNN is presented in Figure 3.2, and additional details and ablation studies are presented in Appendix B.2. We referred to the particular TIN network with CNN generator as TIN-CNN.

3.4.4 Training procedure

Each training iteration consists of two steps: the *inference step* and the *learning step*. During the *inference step*, the optimal latent vector $\mathbf{z}^*$ is inferred, solving the optimization problem given in equation 3.4 with the current parameters $\boldsymbol{\theta}$. During the *learning step*, the latent vector is used as



(a) TIN-CNN **(b)** NHITS **(c)** Informer

Figure 3.3: Forecasts for five feature of **Simulated7** with TIN-CNN, NHITS, and Informer, for 80% missing data (missing in grey). The forecast horizon is 24, produced in a rolling window strategy. Full forecasts for all baseline models are included in Appendix B.7.

the input to the Generator, and the parameters $\boldsymbol{\theta}$ are updated using ADAM optimizer (Kingma and Ba, 2014). For each iteration, we sample a small batch (with replacement) of windows $\mathbf{Y}_{t:t+s_w}$ from the training data, each starting at a random timestamp t. Each sampled observation is first normalized with the mean and standard deviation on the reference window to decouple the scale and patterns (the output is scaled back before evaluation). The full training procedure is presented in Appendix B.5.

The computational cost is one potential drawback of relying on the inference process for finding optimal latent factors. We tackle this in several ways. First, by using gradient descent methods to solve 3.4, we can use automatic differentiation libraries to compute gradients efficiently. Second, backpropagation is parallelizable between windows since the forecasts are independent. Finally, during training, we *persist* the optimal latent vector to future iterations. For a given window starting in t, the final latent vector is stored and used as a warm start when it is sampled again, considerably reducing the number of iterations. We discuss training times and memory complexity in section 3.5.

3.5 Experiments

We base our experimental setting, benchmarks, train/validation/test splits, and data processing on previous works on multivariate forecasting (Zhou et al., 2020; Wu et al., 2021; Challu et al., 2023b).

3.5.1 Datasets

Models are evaluated on synthetic data and five benchmark datasets commonly used in the forecasting literature, comprising various applications and domains. All datasets are normalized with the mean and standard deviation computed on the train set. We present summary statistics in Appendix B.2.

Simulated7 is a synthetic multivariate dataset. Each feature is generated as the sum of two cosines with different frequencies and small Gaussian noise; more details are presented in Appendix B.3. The **Influenza-like illness (ILI)** dataset contains the weekly proportion of patients with influenza-like symptoms in the US, reported by the Centers for Disease Control and Prevention, from 2002 to 2021. **Exchange** reports the daily exchange rates of eight currencies relative to the US dollar from 1990 to 2016. **Solar** dataset contains the hourly photo-voltaic production of 32 solar stations in Wisconsin during 2016. **Electricity Transformer Temperature (ETTm2)** dataset has eight sensor measurements of an electricity transformer in China from July 2016 to July 2018. **Weather** contains 21 meteorological conditions recorded at the German Max Planck Biogeochemistry Institute in 2021.

3.5.2 Training and Evaluation setup

Models are trained on the training set comprising the earliest history of each dataset. Our experimental setting extends the standard practice of using the default configuration for baselines. We select the optimal hyperparameters for all models (including baselines) on each dataset and occlusion percentage based on the performance, measured by MSE, on the validation set using a Bayesian optimization algorithm, HYPEROPT (Bergstra et al., 2011). The complete list of hyperparameters explored is included in Appendix B.4. We repeated the experiment with different random seeds for HYPEROPT five times and reported average performance. We report standard deviations in Appendix B.9. We ran the experiments on an AWS `g3.4xlarge` EC2 instance with NVIDIA Tesla M60 GPUs.

We compare our approach against univariate and multivariate SoTA based on different architectures: Transformers, feed-forward networks (MLP), Graph Neural Networks (GNN), and Recurrent Neural Networks (RNN). For Transformers, we include the PatchTST (Nie et al., 2022), FED`former`(Zhou et al., 2022b), Informer (Zhou et al., 2020), and `Autoformer`(Wu et al., 2021); for MLP, the univariate NHITS (Challu et al., 2023b) and N-BEATS (Oreshkin et al., 2019) models (combined in NHITS column); for GNN the `StemGNN`(Cao et al., 2020), lastly, we include a univariate RNN with dilations (Chang et al., 2017). Some baselines are only included in Appendix B.9 due to limited space.

One standard practice to handle missing data is first to recover missing values with an imputation model before training and predicting with the forecasting model. We include in the comparison two pipelines, CSDI+NHITS and CSDI+`PatchTST`, which first use a SoTA imputation diffusion-based model, CSDI(Tashiro et al., 2021), to recover missing values. These baselines comprise the current gold standard for handling missing data by combining SoTA approaches from both literatures.

3.5.3 Missing data setup

We present a new experimental setting for testing the models' robustness to missing data based on classic occlusion experiments in Computer Vision. We parameterize the experiments with the size of missing segments s and the occlusion probability p_o. First, the time series is divided

Table 3.1: Main forecasting accuracy results on benchmark datasets with different proportions of missing values (p_o), forecasting horizon of 24 timestamps, lower scores are better. Metrics are averaged over five runs, best model highlighted in bold. We report standard deviations in Appendix JB.9

		TIN-CNN(M)		CSDI+NHITS		CSDI+PatchTST		NHITS (U)		PatchTST (U)		Informer (M)		Autoformer (M)		StemGNN (M)	
	p_o	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Simulated7	0.0	0.004	0.041	-	-	-	-	**0.001**	**0.015**	0.004	0.043	0.008	0.079	0.017	0.096	0.036	0.122
	0.2	**0.009**	**0.081**	0.010	0.093	0.082	0.236	0.012	0.176	0.162	0.291	0.071	0.237	0.319	0.430	0.097	0.266
	0.4	**0.011**	**0.073**	0.018	0.121	0.123	0.341	0.022	0.213	0.368	0.465	0.095	0.242	0.591	0.612	0.181	0.385
	0.6	**0.018**	**0.100**	0.095	0.277	0.196	0.394	0.133	0.338	0.479	0.502	0.231	0.380	0.752	0.994	0.362	0.488
	0.8	**0.023**	**0.127**	0.216	0.323	0.395	0.470	0.365	0.439	0.515	0.596	0.372	0.478	0.925	1.106	0.436	0.614
ILI	0.0	**0.724**	**0.557**	-	-	-	-	1.379	0.762	1.228	0.730	4.265	1.329	2.249	0.967	4.013	1.437
	0.2	**1.153**	**0.662**	1.946	0.901	1.837	0.894	2.028	0.933	2.737	1.102	3.624	1.127	3.250	1.292	4.372	1.503
	0.4	**1.646**	**0.793**	3.110	1.142	2.837	1.137	3.248	1.134	4.180	1.267	3.908	1.351	4.308	1.419	4.752	1.694
	0.6	**2.453**	**1.012**	3.552	1.364	3.409	1.285	3.871	1.257	4.355	1.449	3.991	1.277	4.571	1.515	5.170	1.863
	0.8	**3.136**	**1.139**	3.878	1.293	4.126	1.377	5.102	1.451	5.676	1.647	4.180	1.363	4.745	1.536	5.337	1.836
Exchange	0.0	0.048	0.157	-	-	-	-	0.031	0.102	**0.024**	**0.100**	0.472	0.534	0.049	0.167	0.102	0.251
	0.2	0.091	0.223	**0.072**	**0.208**	0.093	0.230	0.295	0.342	0.362	0.498	1.013	0.772	0.758	0.523	1.021	0.599
	0.4	**0.158**	**0.292**	0.163	0.313	0.160	0.311	0.321	0.534	1.057	0.590	1.162	0.905	0.782	0.606	0.826	0.608
	0.6	**0.367**	**0.418**	0.492	0.661	0.503	0.692	0.549	0.745	1.451	0.925	1.564	0.932	1.346	0.849	1.991	1.002
	0.8	**1.125**	**0.785**	1.308	0.860	1.217	0.812	1.792	1.188	2.890	1.302	2.530	1.281	2.520	1.231	2.778	1.318
Solar	0.0	**0.007**	**0.054**	-	-	-	-	0.008	0.061	0.008	0.060	0.011	0.065	0.018	0.095	0.015	0.077
	0.2	**0.012**	**0.058**	0.015	0.060	0.017	0.075	0.016	0.063	0.018	0.079	0.016	0.083	0.025	0.116	0.016	0.084
	0.4	**0.012**	**0.062**	0.017	0.069	0.021	0.082	0.017	0.068	0.023	0.119	0.020	0.102	0.023	0.118	0.023	0.103
	0.6	**0.013**	**0.068**	0.020	0.071	0.022	0.090	0.022	0.075	0.025	0.126	0.021	0.107	0.027	0.130	0.031	0.115
	0.8	**0.014**	**0.072**	0.025	0.089	0.025	0.103	0.025	0.089	0.035	0.148	0.035	0.156	0.036	0.127	0.065	0.192
ETTm2	0.0	0.136	0.212	-	-	-	-	0.116	0.203	**0.107**	**0.202**	0.366	0.462	0.171	0.275	0.154	0.273
	0.2	0.155	0.260	0.145	0.226	**0.140**	**0.221**	0.656	0.420	0.542	0.492	0.895	0.704	0.512	0.450	0.963	0.581
	0.4	0.228	0.316	0.211	0.282	**0.200**	**0.274**	1.191	0.637	1.154	0.631	0.923	0.679	1.078	0.683	1.542	0.795
	0.6	**0.500**	**0.442**	0.563	0.460	0.535	0.458	1.976	0.918	1.763	1.002	1.784	0.999	1.643	0.891	2.151	0.978
	0.8	**0.725**	**0.528**	0.986	0.624	0.801	0.593	2.019	1.230	2.315	1.071	1.982	1.115	1.992	1.062	2.595	1.162
Weather	0.0	0.112	0.193	-	-	-	-	**0.109**	**0.127**	0.131	0.124	0.218	0.283	0.186	0.263	0.116	0.152
	0.2	0.154	0.218	**0.148**	**0.155**	0.163	0.165	0.181	0.205	0.190	0.233	0.287	0.350	0.324	0.389	0.179	0.223
	0.4	**0.205**	**0.269**	0.216	0.283	0.229	0.297	0.264	0.279	0.292	0.308	0.320	0.385	3.105	1.456	0.296	0.311
	0.6	**0.329**	**0.387**	0.417	0.412	0.431	0.454	0.420	0.414	0.486	0.501	0.761	0.628	3.772	1.660	0.495	0.569
	0.8	**0.439**	**0.463**	0.506	0.550	0.509	0.546	0.517	0.503	0.531	0.517	0.915	0.889	4.204	1.859	0.760	0.702

into disjoint segments of length s. Second, each feature m in each segment is occluded with probability p_o. We repeat the experiment with different probabilities: 0% (no missing values), 20%, 40%, 60%, and 80%. The size of segments s is fixed at ten timestamps for ILI and at 100 for all other datasets. Figure 3.3 shows an example of the setting for **Simulated7** with 80% missing data (occluded data in grey).

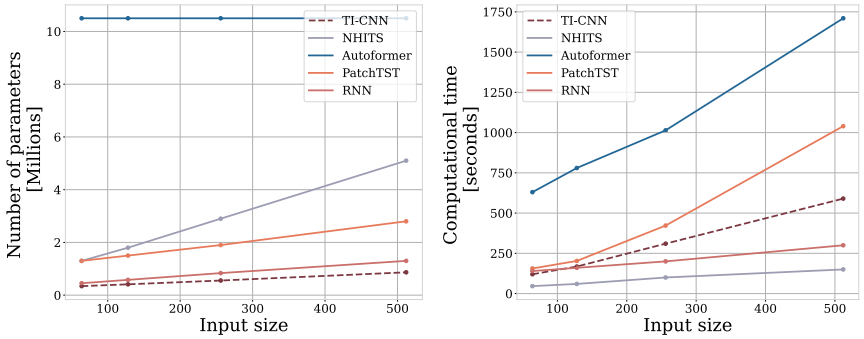
3.5.4 Key results

Forecasting accuracy on full data. TIN-CNN achieves SoTA performance on settings with full data, outperforming all multivariate models consistently across all datasets. TIN-CNN is also the only multivariate approach to improve over the recent NHITSand PatchTSTmodels in several datasets.

Forecasting accuracy with missing data. While all models perform similarly well on **Simulated7** with full data, the accuracy of baselines degrades as the missing data increases. On the contrary, as seen in Figure 3.3 (a), our method can produce accurate forecasts even with 80% of missing values. This controlled experiment allows for isolating the effects of missing data on the models' performance, demonstrating that SoTA forecasting methods failed to accurately forecast simple and predictable time series. Appendix B.8 demonstrates how TIN-CNN is robust to changes over time of the *missing data regime*. The superior robustness of TIN-CNN

Table 3.2: Imputation accuracy on the test set for TIN and baselines with different proportion of missing data (p_o). Lower scores are better, best model highlighted in bold.

		TIN-CNN		CSDI		Mean		Naive		Linear	
	p_o	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Simulated7	0.2	**0.008**	**0.070**	0.062	0.184	0.535	0.642	0.188	0.344	0.069	0.213
	0.6	**0.022**	**0.116**	0.218	0.334	0.551	0.656	0.468	0.518	0.225	0.341
Solar	0.2	**0.004**	**0.042**	0.022	0.096	0.038	0.182	0.052	0.139	0.032	0.116
	0.6	**0.006**	**0.048**	0.034	0.103	0.038	0.182	0.056	0.145	0.039	0.128
Exchange	0.2	**0.018**	0.058	0.045	0.093	1.560	1.002	0.071	0.105	0.023	**0.055**
	0.6	**0.035**	**0.084**	0.101	0.196	1.573	1.010	0.099	0.138	0.042	0.089
ILI	0.2	**0.592**	**0.603**	2.153	0.903	4.085	1.412	1.878	0.871	0.630	0.642
	0.6	**2.568**	**1.349**	5.131	1.508	7.351	1.880	7.840	1.895	7.245	1.605



(a) Memory Efficiency **(b)** Time Efficiency

Figure 3.4: Memory efficiency and train time analysis on ETTm2. Memory efficiency is measured as the number of parameters; train time includes the complete training procedure. We use the best hyperparameter configuration for each model based on model accuracy.

to missing values is evident across all real datasets, with the relative performance of our method improving with the proportion of missing data. For example, with 80% of the data occluded, the average MSE across datasets is 48% lower than the NHITS. Finally, while adding CSDI to impute missing values improves the performance of NHITS and PatchTST, TIN-CNN consistently outperforms both pipelines.

Imputation accuracy. Table 3.2 presents the results for the imputation task. We include a SoTA imputation diffusion model, CSDI (Tashiro et al., 2021), and simple baselines: (i) imputation with the mean of each feature, (ii) imputation with the last available value (Naive) and (iii) linear interpolation between past and future available values. TIN-CNN consistently achieves the best performance across all datasets. CSDI outperforms baselines on **Simulated7** and **Solar**, but its performance degrades on **ILI** dataset due to the distribution shift (see Appendix B.6).

Memory and time complexity. We compare the training time and memory usage as a

p_o	TIN-CNN$_1$ MSE	TIN-CNN$_1$ MAE	TIN-CNN$_2$ MSE	TIN-CNN$_2$ MAE	TIN-CNN MSE	TIN-CNN MAE
0.0	0.691	0.461	0.237	0.283	**0.213**	**0.251**
0.2	0.839	0.623	0.382	0.328	**0.310**	**0.284**
0.4	1.412	0.666	0.492	0.427	**0.449**	**0.341**
0.6	1.952	0.974	0.916	0.620	**0.752**	**0.467**
0.8	2.541	1.082	1.547	0.859	**1.126**	**0.579**

Table 3.3: Average forecasting accuracy across five real benchmark datasets for TIN and two versions without main components. Lower scores are better, best highlighted in bold.

function of the input size on the ETTm2 dataset in Figure 3.4, using the optimal hyperparameters. Panel (a) shows our method has the lowest memory footprint, with up to **85%** fewer parameters than the NHITS and **69%** fewer than PatchTST. With the improvements discussed in section 3.4, training times are comparable to baseline models. Appendix B.10 compares prediction times, analyzing the performance and computation trade-off of the number of iterations for the latent factors inference process.

Ablation studies. Finally, we test the contribution of the two new components proposed in this work, the *Latent Factors Inference* and TDM. We compare TIN-CNN against two versions: TIN-CNN$_1$ adds a CNN encoder to map inputs to latent factors, and TIN-CNN$_2$ keeps the inference step but does not have the TDM. Table 3.3 presents the average performance across the five datasets. As expected, TIN-CNN$_1$ with the parametric encoder is not robust to missing data, with similar performance to other SoTA models. On top of improving the performance over TIN-CNN$_2$, the TDM reduces the number of parameters by 70% since fewer layers are needed to generate the complete window.

3.6 Discussion and Conclusion

Multivariate vs Univariate. Several recent studies have raised questions on the effectiveness of multivariate forecasting models (Zeng et al., 2023; Challu et al., 2023b), as simple univariate approaches often outperform them. We believe that our approach constitutes a significant improvement in multivariate architectures. TIN-CNN achieved better accuracy than multivariate baselines in all datasets and outperformed SoTA univariate models in several cases. In particular, we hypothesize our model improves over univariate models in **Solar** by leveraging *Granger causal relations* (Granger, 1969) between nodes [1]. The benefits of modeling multivariate relations are evident in scenarios with missing data, as the available features provide information to forecast and impute missing features.

Impact of missing data. Our experiments show the performance of current SoTA forecasting models significantly degrades under missing data, a common setting in high-stakes

[1]Panels in the east receive sunlight earlier due to the sun movement, providing useful information and current conditions to forecast western nodes better.

settings such as healthcare and finance. Designing robust algorithms can significantly improve their adoption and potential benefits across multiple domains. Using imputation models to recover missing data partially alleviates the performance degradation, but this approach also adds complexity to pipelines and computational costs, as several models need to be trained and maintained.

Distribution Shifts. We believe that TIN's capability to dynamically adjust the forecasts to the latest temporal dynamics provides additional flexibility to handle distribution shifts in the data. TIN-CNN's performance on **ILI**, which exhibits larger spikes on the test set, supports this claim. In particular, the unconstrained latent factors can extrapolate beyond training regions to produce forecasts with unseen patterns and scales, such as **ILI**'s larger spikes. We formally define this setting and perform additional experiments in Appendix B.6. Related to this, we believe our approach can also have applications in transfer learning. The inference procedure will allow a pre-trained generator model to adapt the forecasts to unseen temporal patterns on the target dataset. Exploring TIN's transfer learning and domain adaptation capabilities can be a promising line of research for future work.

Limitations. While our approach demonstrates improvements over the current SoTA in several tasks and settings, we identify several limitations. Similarly to all windows-based models, TIN-CNN can only model dynamics present within the reference window. Additionally, while our approach can handle some forms of distribution shifts, it assumes the temporal dynamics are constant between the reference and forecast windows. Finally, in this paper, we focused on point forecasting and imputation tasks. Extending TIN to produce probabilistic predictions will be explored in future work.

This chapter presents *Temporal Inference Networks*, a family of neural models for simultaneous forecasting and imputation of time series. A TIN network consists of three components: the *Latent Factors Inference* procedure, a *Temporal Dynamic Module*, and the Generator. The Generator generates complete time series windows from latent factors, interpolating missing data and forecasting future values. The temporal latent factors are inferred from the first two components by matching the Generator's output on past observations. We compare the accuracy of our method against SoTA models from both forecasting and interpolation literature on several benchmarks. TIN-CNN achieves SoTA on both tasks simultaneously, with comparable training times and as much as 92% fewer parameters.

Part II

Scalable and Interpretable Multi-step Forecasting

The impressive accuracy of deep learning models in the long-horizon multi-step forecasting task is usually accompanied by high computational cost, given that modeling long temporal dependencies requires complex models and longer inputs. The first chapter presents the NHITS, a novel model with hierarchical interpolation and multi-rate sampling techniques to decompose the forecast in frequency bands, extending on the classic Fourier decomposition. The second chapter presents multivariate extensions with exogenous covariates to enhance forecasting accuracy. The proposed methods are tested in two important domains: energy and healthcare.

Chapter 4

NHITS: Long-Horizon Multi-Step Forecasting

4.1 Motivation

The recent advancements in neural forecasting methods have steadily improved the performance and capabilities of forecasting systems. Machine learning methods won recent large-scale competitions, such as the M4 (Makridakis et al., 2020), improving over classic statistical methods. However, their impressive accuracy is usually accompanied by a drastic increase in computational cost. This increase is exacerbated in long-horizon forecasting since accurately predicting hundreds of timestamps requires modeling longer temporal dependencies, which in turn requires longer inputs. Current architectures scale poorly with the input and output dimensionality. Designing accurate and efficient methods is essential for many applications, particularly in domains where computational resources are constrained. To this end, this chapter proposes NHITS, a novel global model with *hierarchical interpolation* and *multi-rate sampling* techniques to decompose the forecast in frequency bands, which achieves SoTA performance in several benchmark datasets with a fraction of the computational cost.

Long-horizon multi-step forecasting is challenging due to the computational complexity and performance degradation of modeling longer temporal dependencies. Most of the latest academic work has focused on transformer-based models, given their capabilities to automatically select relevant features and timestamps among large input sequences. However, transformers are inherently large models and expensive to train. Some recent work has questioned using transformer-based models for time series, claiming the additional computational cost is not worth the small accuracy gains (Zhou et al., 2022a; Zeng et al., 2022).

A popular strategy of many forecasting models is to decompose the time series representations or predictions based on interpretable definitions for humans. These decompositions can have multiple benefits: incorporating domain knowledge in the form of inductive bias, simplifying architectures, reducing parameters, and producing interpretable forecasts. The most common decomposition is the trend and seasonal components (Hyndman and Athanasopoulos,

2018b). For example, the NBEATS (Oreshkin et al., 2020) learns projections to a pre-defined basis of polynomial and harmonic functions, and the ESRNN (Smyl, 2019) uses the classic Holt-Winters (Holt, 1957) model to deseasonalize and standardize the time series automatically. A separate popular approach is decomposing the time series by frequency, incorporating Fourier transforms in the architecture. Some examples are Fredo (Sun and Boning, 2022), FEDFormer (Zhou et al., 2022b), and ETSFormer (Woo et al., 2022a). One limitation of current methods is they assume constant dynamics along a frequency band.

The contributions are summarized below:

1. **Multi-Rate Data Sampling**: incorporate sub-sampling layers in front of fully connected blocks, significantly reducing the memory footprint and the amount of computation needed while maintaining the ability to model long-range dependencies.
2. **Hierarchical Interpolation**: to enforce the smoothness of the multi-step predictions by reducing the dimensionality of the neural network's prediction and matching its time scale with that of the final output via multi-scale hierarchical interpolation. This novel technique is not unique to the proposed model and can be incorporated into different architectures.
3. **NHITS architecture**: hierarchically synchronizing the input sampling rate with the scale of output interpolation across blocks, which induces each block to specialize in forecasting its own frequency band of the time series signal.
4. **State-of-the-art results** on six large-scale benchmark datasets from the long-horizon forecasting literature: electricity transformer temperature, exchange rate, electricity consumption, San Francisco Bay area highway traffic, weather, and influenza-like illness.

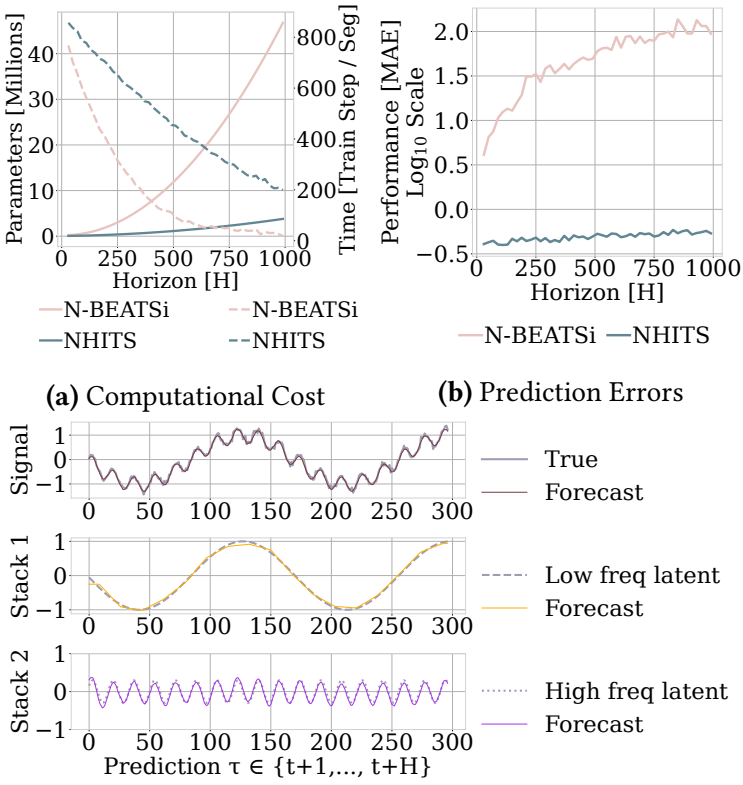
4.2 Related Work

4.2.1 Multi-step forecasting

Multi-step forecasting consists of predicting a sequence of future values of the target time series. Two classic strategies for producing multi-step forecasts are iterating one-step-ahead models, named *recursive strategy*, and allocating separate models for each forecast horizon, named *direct strategy*. Several recent studies, such as (Atiya and Taieb, 2016), demonstrated the direct method achieves lower bias by avoiding error accumulation through the forecast horizon. However, this approach usually has a larger variance due to the increased number of parameters. The high expressivity of deep learning approaches allowed for a third strategy: a single large model simultaneously produces forecasts for all steps. This approach achieves a better balance between variance and bias by also avoiding error accumulation while leveraging shared parameters.

4.2.2 Long-horizon forecasting

Long-horizon forecasting is a case of multi-step forecasting where the predicted sequence spans hundreds or thousands of timestamps. The recent surge in long-horizon forecasting



(a) Computational Cost **(b)** Prediction Errors

(c) Neural Hierarchical Interpolation

Figure 4.1: (a) The computational costs in time and memory (b) and mean absolute errors (MAE) of the predictions of a high capacity fully connected model exhibit evident deterioration with growing forecast horizons. (c) Specializing a flexible model's outputs in the different signal frequencies through hierarchical interpolation combined with multi-rate input processing offers a solution.

methods originated from the `Informer` (Zhou et al., 2020) work, which proposed ProbSparse, an efficient attention mechanism with sparse query approximation to achieve O(L log L) in time complexity. After the initial success of the `Informer`, many other transformer-based approaches have been proposed to further reduce computational complexity and improve accuracy. Some examples include the `Autoformer` (Wu et al., 2021), `ETSformer` (Woo et al., 2022a), and `FEDformer` (Zhou et al., 2022b).

4.2.3 Multi-rate sampling and interpolation

The earliest forms of multi-rate sampling for time series forecasting correspond to the *mixed data sampling regression* (`MIDAS`; Ghysels et al. 2007). The regression model in `MIDAS` incorporates input time series data sampled at different frequencies to extract relevant information in higher frequencies efficiently. The multi-rate sampling technique proposed in the approach is based on the pooling layers commonly used in Convolution Neural Networks (CNN) for Computer Vision (CV) (Krizhevsky et al., 2017). In time series forecasting, interpolation has a wide range of applications, from completing unevenly sampled data and noise filters (Chow and Lin, 1971;

Fernandez, 1981; Rubanova et al., 2019) to fine-grained quantile-regressions with recurrent networks (Gasthaus et al., 2019). To my knowledge, this work is the first approach to use temporal interpolation to induce multi-scale forecasts.

4.3 Methodology

The method extends the *Neural Basis Expansion Analysis* approach (NBEATS; Oreshkin et al. 2020) in several important respects, making it more accurate and computationally efficient, especially in the context of long-horizon forecasting. In essence, it uses multi-rate sampling of the input signal and multi-scale synthesis of the forecast, resulting in a hierarchical construction of the forecast, greatly reducing computational requirements and improving forecasting accuracy. Fig. 4.2 presents a high-level diagram and main principles of operation.

Similarly to NBEATS, NHITS performs local nonlinear projections onto basis functions across multiple blocks. Each block consists of a *multilayer perceptron* (MLP), which learns to produce coefficients for the backcast and forecast outputs of its basis. The backcast output is used to clean the inputs of subsequent blocks, while the forecasts are summed to compose the final prediction. The blocks are grouped in stacks, each specialized in learning a different data characteristic using a different set of basis functions. The overall network input, $\mathbf{y}_{t-L:t}$, consists of L lags.

NHITS is composed of S stacks, B blocks each. Each block contains an MLP predicting forward and backward basis coefficients. The next subsections describe the novel components of the architecture. Note that the s stack index is not included for brevity in the following section.

4.3.1 Multi-Rate Signal Sampling

At the input to each block ℓ, we propose to use a MaxPool layer with kernel size k_ℓ to help it focus on analyzing components of its input with a specific scale. Larger k_ℓ will tend to cut more high-frequency/small-time-scale components from the input of the MLP, forcing the block to focus on analyzing large-scale or low-frequency content. We call this *multi-rate signal sampling*, referring to the fact that the MLP in each block faces a different effective input signal sampling rate. Intuitively, this helps the blocks with larger pooling kernel size k_ℓ focus on analyzing large-scale components critical for producing consistent long-horizon forecasts.

Additionally, multi-rate processing reduces the width of the MLP input for most blocks, limiting the memory footprint and the amount of computation and reducing the number of learnable parameters, hence alleviating the effects of overfitting while maintaining the original receptive field. Given block ℓ input $\mathbf{y}_{t-L:t,\ell}$ (the input to the first block $\ell = 1$ is the network-wide input, $\mathbf{y}_{t-L:t,1} \equiv \mathbf{y}_{t-L:t}$), this operation can be formalized as follows:

$$\mathbf{y}^{(p)}_{t-L:t,\ell} = \mathbf{MaxPool}\left(\mathbf{y}_{t-L:t,\ell},\ k_\ell\right) \tag{4.1}$$

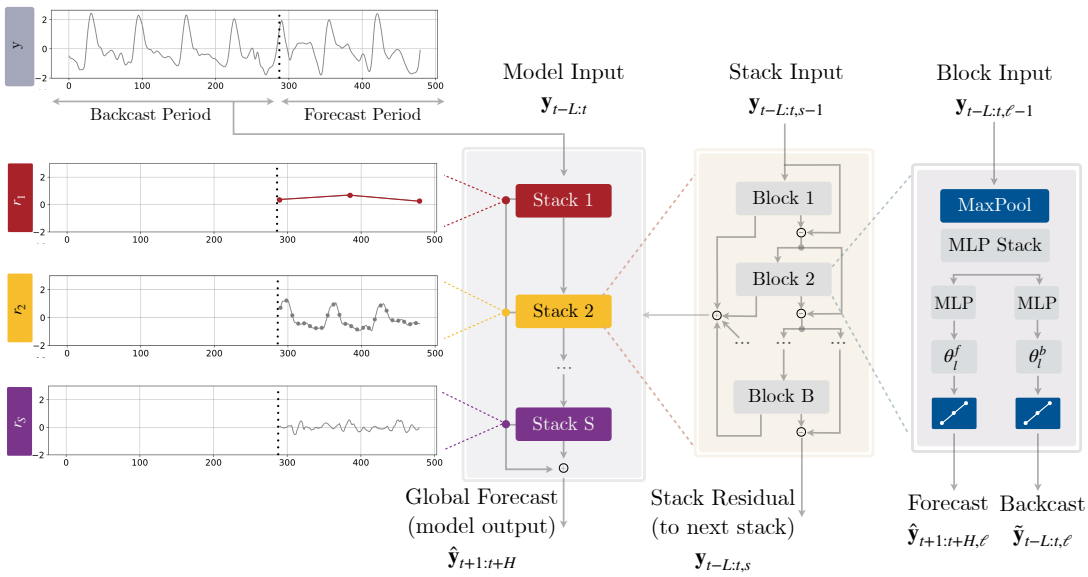


Figure 4.2: NHITS architecture. The model is composed of several MLPs with ReLU nonlinearities. Blocks are connected via doubly residual stacking principle with the backcast $\tilde{\mathbf{y}}_{t-L:t,\ell}$ and forecast $\hat{\mathbf{y}}_{t+1:t+H,\ell}$ outputs of the ℓ-th block. Multi-rate input pooling, hierarchical interpolation, and backcast residual connections induce the specialization of the additive predictions in different signal bands, reducing memory footprint and compute time and improving architecture parsimony and accuracy.

4.3.2 Non-Linear Regression

Following subsampling, block ℓ looks at its input and non-linearly regresses forward θ_ℓ^f and backward θ_ℓ^b interpolation MLP coefficients that learns hidden vector $\mathbf{h}_\ell \in \mathbb{R}^{N_h}$, which is then linearly projected:

$$\begin{aligned} \mathbf{h}_\ell &= \mathbf{MLP}_\ell\left(\mathbf{y}_{t-L:t,\ell}^{(p)}\right) \\ \theta_\ell^f &= \mathbf{LINEAR}^f\left(\mathbf{h}_\ell\right) \quad \theta_\ell^b = \mathbf{LINEAR}^b\left(\mathbf{h}_\ell\right) \end{aligned} \tag{4.2}$$

The coefficients are then used to synthesize backcast $\tilde{\mathbf{y}}_{t-L:t,\ell}$ and forecast $\hat{\mathbf{y}}_{t+1:t+H,\ell}$ outputs of the block, via the process described below.

4.3.3 Hierarchical Interpolation

In most multi-horizon forecasting models, the neural network prediction's cardinality equals the horizon's dimensionality, H. For example, in NBEATS-I $|\theta_\ell^f| = H$; in Transformer-based models, decoder attention layer cross-correlates H output embeddings with L encoded input embeddings (L tends to grow with growing H). This leads to quick inflation in compute requirements and unnecessary explosion in model expressiveness as horizon H increases.

We propose to use *temporal interpolation* to combat these issues. We define the dimensionality of the interpolation coefficients in terms of the *expressiveness ratio* r_ℓ that controls the number

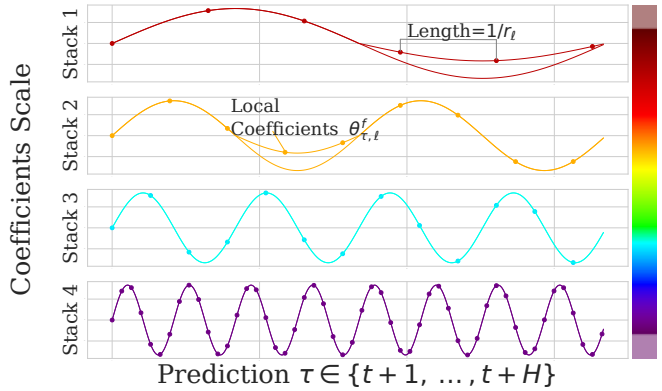


Figure 4.3: NHITS composes its predictions hierarchically using blocks specializing on different frequencies, through *expressiveness ratios*, and interpolation. The coefficients are locally determined along the horizon, allowing NHITS to reconstruct non-periodic/stationary signals beyond constant Fourier transform.

of parameters per unit of output time, $|\theta^f_\ell| = \lceil r_\ell H \rceil$. To recover the original sampling rate and predict all H points in the horizon, we use temporal interpolation via the interpolation function g:

$$\begin{aligned} \hat{y}_{\tau,\ell} &= g(\tau, \theta^f_\ell), \quad \forall \tau \in \{t+1, \ldots, t+H\}, \\ \tilde{y}_{\tau,\ell} &= g(\tau, \theta^b_\ell), \quad \forall \tau \in \{t-L, \ldots, t\}. \end{aligned} \tag{4.3}$$

Interpolation can vary in *smoothness*, $g \in \mathcal{C}^0, \mathcal{C}^1, \mathcal{C}^2$. In Appendix C.7, we explore the nearest neighbor, piece-wise linear, and cubic alternatives. For concreteness, the linear interpolator $g \in \mathcal{C}^1$, along with the time partition $\mathcal{T} = \{t+1, t+1+1/r_\ell, \ldots, t+H-1/r_\ell, t+H\}$, is defined as

$$\begin{aligned} g(\tau, \theta) &= \theta[t_1] + \left(\frac{\theta[t_2] - \theta[t_1]}{t_2 - t_1} \right) (\tau - t_1) \\ t_1 &= \arg \min_{t \in \mathcal{T}: t \leq \tau} \tau - t, \quad t_2 = t_1 + 1/r_\ell. \end{aligned} \tag{4.4}$$

The *hierarchical* interpolation principle is implemented by distributing expressiveness ratios across blocks in a manner synchronized with multi-rate sampling. Blocks closer to the input have smaller r_ℓ and larger k_ℓ, implying that input blocks generate low-granularity signals via more aggressive interpolation, being also forced to look at more aggressively sub-sampled (and smoothed) signals. The resulting hierarchical forecast $\hat{\mathbf{y}}_{t+1:t+H}$ is assembled by summing the outputs of all blocks, essentially composing it out of interpolations at different time-scale hierarchy levels.

Since each block specializes in its own scale of input and output signal, this induces a clearly structured hierarchy of interpolation granularity, the intuition conveyed in Fig. 4.1 and 4.3. We propose to use *exponentially increasing expressiveness ratios* to handle a wide range of frequency bands while controlling the number of parameters. Alternatively, each stack can specialize in modeling a different known cycle of the time series (weekly, daily, etc.) using a matching r_ℓ. Finally, the backcast residual formed at the previous hierarchy scale is subtracted from the input of the next hierarchy level to amplify the focus of the next level block on signals outside of the band that have already been handled by the previous hierarchy members.

$$\hat{\mathbf{y}}_{t+1:t+H} = \sum_{l=1}^{L} \hat{\mathbf{y}}_{t+1:t+H,\ell}$$
$$\mathbf{y}_{t-L:t,\ell+1} = \mathbf{y}_{t-L:t,\ell} - \tilde{\mathbf{y}}_{t-L:t,\ell}$$

Hierarchical interpolation has advantageous theoretical guarantees. We show in Appendix C.1, that it can approximate infinitely/dense horizons. As long as the interpolating function g is characterized by projections to informed multi-resolution functions V_w, and the forecast relationships are smooth.

Theorem 4.1: Neural Basis Approximation.

Let a forecast mapping be $\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]}) : [0,1]^L \to \mathcal{F}$, where the forecast functions $\mathcal{F} = \{\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]}) : [0,1] \to \mathbb{R}\} = \mathcal{L}^2([0,1])$ representing a infinite/dense horizon, are square integrable. If the multi-resolution functions $V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0,\ldots,2^w]\}$ can arbitrarily approximate $\mathcal{L}^2([0,1])$. And the projection $\mathrm{Proj}_{V_w}(\mathcal{Y}(\tau))$ varies smoothly on $\mathbf{y}_{[t-L:t]}$. Then the forecast mapping $\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]})$ can be arbitrarily approximated by a neural basis expansion learning a finite number of multi-resolution coefficients $\hat{\theta}_{w,h}$. That is $\forall \epsilon > 0$,

$$\int |\mathcal{Y}(\tau \mid \mathbf{y}_{[t-L:t]}) - \sum_{w,h} \hat{\theta}_{w,h}(\mathbf{y}_{[t-L:t]})\phi_{w,h}(\tau)| d\tau \leq \epsilon \tag{4.5}$$

Examples of multi-resolution functions $V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0,\ldots,2^w]\}$ include piece-wise constants, piece-wise linear functions and splines with arbitrary approximation capabilities.

4.4 Experiments

The experimental setting is based on (Wu et al., 2021; Zhou et al., 2020) (NeurIPS 2021 and AAAI 2021 Best Paper Award). We first describe datasets, baselines, and metrics used to evaluate the model quantitatively. Table 4.1 presents the key results, demonstrating the SoTA performance

of the method relative to existing work. We then carefully describe the details of training and evaluation setups. We conclude the section by describing ablation studies.

Table 4.1: Main empirical results in long-horizon forecasting setup, lower scores are better. Metrics are averaged over eight runs, best results are highlighted in bold. In Appendix E we complement the main results with standard deviations.

		NHITS (Ours)		NBEATS		FEDformer		Autoformer		Informer		LogTrans		Reformer		DilRNN		ARIMA	
	H.	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_2$	96	**0.176**	**0.255**	0.184	0.263	0.203	0.287	0.255	0.339	0.365	0.453	0.768	0.642	0.658	0.619	0.343	0.401	0.225	0.301
	192	**0.245**	**0.305**	0.273	0.337	0.269	0.328	0.281	0.340	0.533	0.563	0.989	0.757	1.078	0.827	0.424	0.468	0.298	0.345
	336	**0.295**	**0.346**	0.309	0.355	0.325	0.366	0.339	0.372	1.363	0.887	1.334	0.872	1.549	0.972	0.632	1.083	0.370	0.386
	720	**0.401**	**0.413**	0.411	0.425	0.421	0.415	0.422	0.419	3.379	1.388	3.048	1.328	2.631	1.242	0.634	0.594	0.478	0.445
ECL	96	0.147	0.249	**0.145**	**0.247**	0.183	0.297	0.201	0.317	0.274	0.368	0.258	0.357	0.312	0.402	0.233	0.927	1.220	0.814
	192	**0.167**	**0.269**	0.180	0.283	0.195	0.308	0.222	0.334	0.296	0.386	0.266	0.368	0.348	0.433	0.265	0.921	1.264	0.842
	336	**0.186**	**0.290**	0.200	0.308	0.212	0.313	0.231	0.338	0.300	0.394	0.280	0.380	0.350	0.433	0.235	0.896	1.311	0.866
	720	0.243	**0.340**	0.266	0.362	**0.231**	0.343	0.254	0.361	0.373	0.439	0.283	0.376	0.340	0.420	0.322	0.890	1.364	0.891
Exchange	96	**0.092**	**0.202**	0.098	0.206	0.139	0.276	0.197	0.323	0.847	0.752	0.968	0.812	1.065	0.829	0.383	0.45	0.296	0.214
	192	**0.208**	**0.322**	0.225	0.329	0.256	0.369	0.300	0.369	1.204	0.895	1.040	0.851	1.188	0.906	1.123	0.834	1.056	0.326
	336	**0.301**	**0.403**	0.493	0.482	0.426	0.464	0.509	0.524	1.672	1.036	1.659	1.081	1.357	0.976	1.612	1.051	2.298	0.467
	720	**0.798**	**0.596**	1.108	0.804	1.090	0.800	1.447	0.941	2.478	1.310	1.941	1.127	1.510	1.016	1.827	1.131	20.666	0.864
Traffic-L	96	0.402	**0.282**	**0.398**	**0.282**	0.562	0.349	0.613	0.388	0.719	0.391	0.684	0.384	0.732	0.423	0.580	0.308	1.997	0.924
	192	0.420	0.297	**0.409**	**0.293**	0.562	0.346	0.616	0.382	0.696	0.379	0.685	0.390	0.733	0.420	0.739	0.383	2.044	0.944
	336	**0.448**	**0.313**	0.449	0.318	0.570	0.323	0.622	0.337	0.777	0.420	0.733	0.408	0.742	0.420	0.804	0.419	2.096	0.960
	720	**0.539**	**0.353**	0.589	0.391	0.596	0.368	0.660	0.408	0.864	0.472	0.717	0.396	0.755	0.423	0.695	0.372	2.138	0.971
Weather	96	**0.158**	**0.195**	0.167	0.203	0.217	0.296	0.266	0.336	0.300	0.384	0.458	0.490	0.689	0.596	0.193	0.245	0.217	0.258
	192	**0.211**	**0.247**	0.229	0.261	0.276	0.336	0.307	0.367	0.598	0.544	0.658	0.589	0.752	0.638	0.255	0.306	0.263	0.299
	336	**0.274**	**0.300**	0.287	0.304	0.339	0.380	0.359	0.395	0.578	0.523	0.797	0.652	0.064	0.596	0.329	0.360	0.330	0.347
	720	**0.351**	**0.353**	0.368	0.359	0.403	0.428	0.419	0.428	1.059	0.741	0.869	0.675	1.130	0.792	0.521	0.495	0.425	0.405
ILI	24	**1.862**	**0.869**	1.879	0.886	2.203	0.963	3.483	1.287	5.764	1.677	4.480	1.444	4.400	1.382	4.538	1.449	5.554	1.434
	36	**2.071**	**0.934**	2.210	1.018	2.272	0.976	3.103	1.148	4.755	1.467	4.799	1.467	4.783	1.448	3.709	1.273	6.940	1.676
	48	**2.134**	**0.932**	2.440	1.088	2.209	0.981	2.669	1.085	4.763	1.469	4.800	1.468	4.832	1.465	3.436	1.238	7.192	1.736
	60	**2.137**	**0.968**	2.547	1.057	2.545	1.061	2.770	1.125	5.264	1.564	5.278	1.560	4.882	1.483	3.703	1.272	6.648	1.656

4.4.1 Datasets

All large-scale datasets used in the empirical studies are publicly available and have been used in the neural forecasting literature, particularly in the context of long-horizon (Lai et al., 2017; Zhou et al., 2019; Li et al., 2019c; Wu et al., 2021). Table A1 summarizes their characteristics. Each set is normalized with the train data mean and standard deviation.

Electricity Transformer Temperature. The **$ETTm_2$** dataset measures an electricity transformer from a region of a province of China, including oil temperature and variants of load (such as high useful load and high useless load) from July 2016 to July 2018 at a fifteen-minute frequency. **Exchange-Rate.** The **Exchange** dataset is a collection of daily exchange rates of eight countries relative to the US dollar. The countries include Australia, UK, Canada, Switzerland, China, Japan, New Zealand and Singapore from 1990 to 2016. **Electricity.** The **ECL** dataset reports the fifteen-minute electricity consumption (KWh) of 321 customers from 2012 to 2014. For comparability, we aggregate it hourly. **San Francisco Bay Area Highway Traffic.** This **Traffic-L** dataset was collected by the California Department of Transportation; it reports road hourly occupancy rates of 862 sensors from January 2015 to December 2016. **Weather.** This **Weather** dataset contains the 2020 year of 21 meteorological measurements recorded every 10 minutes from the Weather Station of the Max Planck Biogeochemistry Institute in Jena, Germany. **Influenza-like illness.** The **ILI** dataset reports weekly recorded influenza-like

illness (ILI) patients from the Centers for Disease Control and Prevention of the United States from 2002 to 2021. It is a ratio of ILI patients vs. the week's total.

4.4.2 Evaluation Setup

The accuracy of the proposed approach is evaluated using *mean absolute error* (MAE) and *mean squared error* (MSE) metrics, which are well-established in the literature (Zhou et al., 2020; Wu et al., 2021), for varying horizon lengths H:

$$\text{MSE} = \frac{1}{H}\sum_{\tau=t}^{t+H}\left(\mathbf{y}_{\tau}-\hat{\mathbf{y}}_{\tau}\right)^{2}, \qquad \text{MAE} = \frac{1}{H}\sum_{\tau=t}^{t+H}\left|\mathbf{y}_{\tau}-\hat{\mathbf{y}}_{\tau}\right| \tag{4.6}$$

Note that for multivariate datasets, the NHITS forecasts each feature, and metrics are averaged across dataset features. Since the model is univariate, each variable is predicted using only its own history, $\mathbf{y}_{t-L:t}$, as input. Datasets are partitioned into train, validation, and test splits. Train split is used to train model parameters, validation split is used to tune hyperparameters, and test split is used to compute metrics reported in Table 4.1. Appendix C.3 shows partitioning into train, validation, and test splits: seventy, ten, and twenty percent of the available observations, respectively, with the exception of $\mathbf{ETTm}_2$ that uses twenty percent as validation.

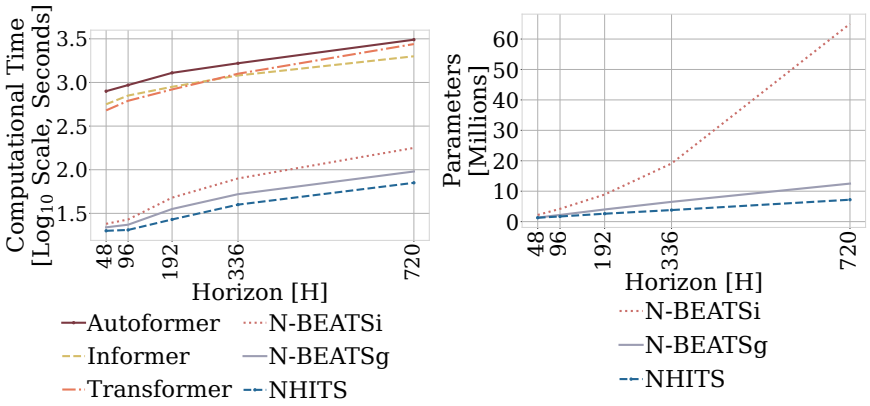
4.4.3 Training and Hyperparameter Optimization

We consider a minimal search space. We tune the kernel size for multi-rate sampling from Equation (4.1) and the number of coefficients from Equation (4.2), some matching common seasonalities and others exponentially increasing. Additionally, we tune the random seed to escape under-performing local minima. Details are reported in Appendix C.4.

During the hyperparameter optimization phase, we measure MAE on the validation set and use a Bayesian optimization library (HYPEROPT; Bergstra et al. 2011), with 20 iterations. We use the optimal configuration based on the validation loss to make predictions on the test set. We refer to the combination of hyperparameter optimization and test prediction as a *run*. NHITS is implemented in PyTorch (Paszke et al., 2019) and trained using ADAM optimizer (Kingma and Ba, 2014), MAE loss, batch size 256 and initial learning rate of 1e-3, halved three times across the training procedure. All the experiments were conducted on a GeForce RTX 2080 GPU.

4.4.4 Key Results

We compare NHITS to the following SoTA multivariate baselines: (1) FEDformer (Zhou et al., 2022b), (2) Autoformer (Wu et al., 2021), (3) Informer (Zhou et al., 2020), (4) Reformer (Kitaev et al., 2020) and (5) LogTrans (Li et al., 2019c). Additionally, we consider the univariate baselines: (6) DilRNN (Chang et al., 2017) and (7) auto-ARIMA (Hyndman and Khandakar, 2008).



(a) Time Efficiency

(b) Memory Efficiency

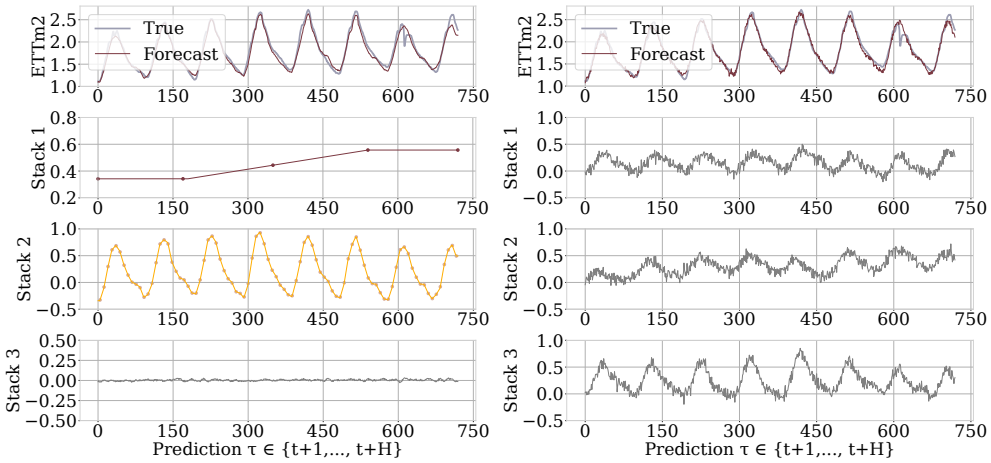
Figure 4.4: Computational efficiency comparison. NHITS exhibits the best training time compared to Transformer-based and fully connected models and the smallest memory footprint.

Forecasting Accuracy. Table 4.1 summarizes the multivariate forecasting results. NHITS outperforms the best baseline, with an average relative error decrease across datasets and horizons of 14% in MAE and 16% in MSE. NHITS maintains a comparable performance to other state-of-the-art methods for the shortest measured horizon (96/24), while for the longest measured horizon (720/60), it decreases multivariate MAE by 11% and MSE by 17%. We complement the key results in Table 4.1, with the additional univariate forecasting experiments in Appendix C.6, again demonstrating state-of-the-art performance against baselines.

Computational Efficiency. We measure the computational training time of NHITS, NBEATS and Transformer-based methods in the multivariate setting and show a comparison in Figure 4.4. The experiment monitors the whole training process for the **ETTm$_2$** dataset. We used hyperparameters reported in (Wu et al., 2021) for the Transformer-based models. Compared to the Transformer-based methods, NHITS is 45× faster than `Autoformer`. In terms of memory, NHITS has less than 26% of the parameters of the second-best alternative since it scales linearly with respect to the input's length. Compared to the original NBEATS, the proposed method is 1.26× faster and requires only 54% of the parameters. Finally, while NHITS is a univariate model, it has *global* (shared) parameters for all time series in the dataset. Just like Oreshkin et al., 2020, the experiments (Appendix C.9) show that NHITS maintains constant parameter/training computational complexity regarding the dataset's size.

4.4.5 Ablation Studies

We believe the advantages of the NHITS architecture are rooted in its multi-rate hierarchical nature. Fig. 4.5 shows a qualitative comparison of NHITS with and without hierarchical interpolation/multi-rate sampling components. Unlike the control model, we clearly see NHITS developing the ability to produce interpretable forecast decomposition, providing valuable information about trends and seasonality in separate channels. Appendix C.7 presents the decomposition for the different interpolation techniques.



(a) hie. interp., multi-rate sampling **(b)** No hie. interp., multi-rate sampling

Figure 4.5: ETTm2 and 720 ahead forecasts using NHITS (left panel), NHITS with hierarchical linear interpolation and multi-rate sampling removed (right panel). The top row shows the original signal and the forecast. The second, third, and fourth rows show the forecast components for each stack. The last row shows the residuals, $y - \hat{y}$. In (a), each block shows scale specialization, unlike (b), in which signals are not interpretable.

Tab. 4.2 presents an ablation study on the individual proposed components, given by: NHITS_2 only hierarchical interpolation, NHITS_3 only multi-rate sampling, NHITS_4 no multi-rate sampling or interpolation (corresponds to the original NBEATS-G (Oreshkin et al., 2020)), finally NBEATS-I, the interpretable version of the NBEATS ((Oreshkin et al., 2020)). Tab. 4.2 clearly shows that combining both proposed components results in the best performance, emphasizing their complementary nature in long-horizon forecasting. The original NBEATS is consistently worse, especially the NBEATS-I. The advantages of multi-rate sampling and interpolation for long-horizon forecasting are not limited to the NHITS architecture. In Appendix C.8, we demonstrate how adding them to a DilRNN improves its performance.

Table 4.2: Empirical evaluation of long multi-horizon multivariate forecasts for NHITS with/without enhancements. MAE and MSE for predictions averaged over eight runs, and five datasets, the best result is highlighted in bold, second best in blue (lower is better).

		NHITS	NHITS_2	NHITS_3	NHITS_4	NBEATS-I
A. MSE	96	0.195	0.196	**0.192**	0.196	0.209
	192	**0.250**	0.261	0.251	0.263	0.266
	336	**0.315**	0.315	0.342	0.346	0.408
	720	**0.484**	0.498	0.518	0.548	0.794
A. MAE	96	0.239	0.241	**0.237**	0.240	0.254
	192	**0.290**	0.299	0.291	0.300	0.307
	336	**0.338**	0.342	0.346	0.352	0.405
	720	**0.439**	0.450	0.454	0.468	0.597

Additional *ablation studies* are reported in Appendix C.7. The MaxPool multi-rate sampling wins over AveragePool. Linear interpolation wins over nearest neighbor and cubic. Finally

and most importantly, we show that the order in which hierarchical interpolation is implemented matters significantly. The best configuration is to have the low-frequency/large-scale components synthesized and removed from analysis first, followed by more fine-grained high-frequency/intermittent signals modeling.

4.5 Discussion and Conclusion

The results indicate the complementary effectiveness of multi-rate sampling and hierarchical interpolation for long-horizon time series forecasting. Table 4.2 indicates that these components enforce a useful inductive bias compared to both the free-form model NHITS$_4$ (plain fully connected architecture) and the parametric model NBEATS-I (polynomial trend and sinusoidal seasonality used as basis functions in two respective stacks). The latter provides a detrimental inductive bias for long-horizon forecasting. We barely scratched the surface in the right direction, and further progress is possible using advanced multi-scale processing approaches in the forecasting context, motivating further research.

NHITS outperforms SoTA baselines and provides an interpretable non-linear decomposition. Fig. 4.1 and 4.5 showcase NHITS perfectly specializing and reconstructing latent harmonic signals from synthetic and real data, respectively. This novel *interpretable* decomposition can provide insights to users, improving their confidence in high-stakes applications, such as healthcare. Finally, NHITS hierarchical interpolation is connected to Wavelet's multi-resolution analysis (Daubechies, 1992). Replacing the interpolation functions with orthogonal Wavelet spaces is a possible research line.

The results in this chapter question the effectiveness of existing long-horizon multi-variate forecasting approaches, as they are substantially outperformed by an univariate algorithm. If these approaches underperform due to overfitting problems at the level of marginals, integrating the proposed components with Transformer-inspired architectures is a promising research direction. However, there is a chance that existing approaches underperform due to their inability to integrate information from multiple variables, which clearly hints at possibly untapped research potential. Whichever is the case, we believe these results provide a strong guidance signal and a valuable baseline for future research in long-horizon multivariate forecasting.

This chapter presents a novel neural forecasting algorithm NHITS that combines two complementary techniques, multi-rate input sampling, and hierarchical interpolation, to produce drastically improved, interpretable, and computationally efficient long-horizon time series predictions. The NHITS model, operating in the univariate regime and accepting only the predicted time series history, significantly outperforms all previous Transformer-based multivariate models using an order of magnitude less computation. This sets a new baseline for all ensuing multivariate work on six popular datasets and motivates research to use information across variables effectively.

Chapter 5

Multivariate extensions and applications

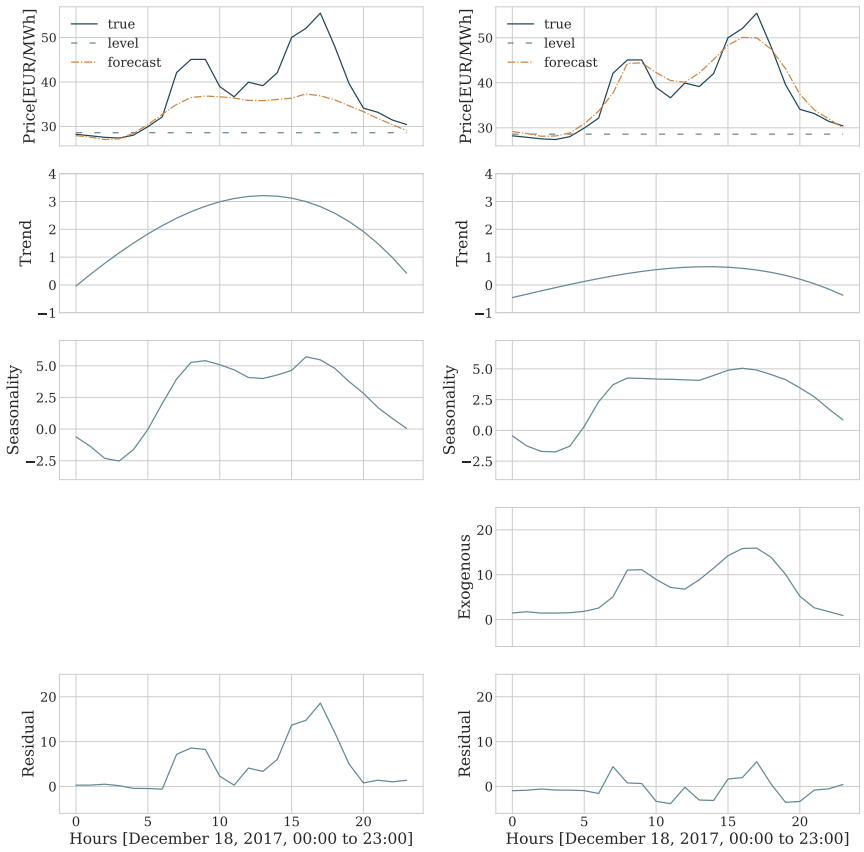
5.1 Incorporating Exogenous Variables for Electricity Price Forecasting

5.1.1 Introduction

Neural networks have proven powerful and flexible, yet there are several situations where our understanding of the model's predictions can be as crucial as their accuracy, which constitutes a barrier to their wider adoption. The interpretability of the algorithm's outputs is critical because it encourages trust in its predictions, improves our knowledge of the modeled processes, and provides insights that can improve the method itself. Additionally, the absence of time-dependent covariates makes these powerful models unsuitable for many applications. For instance, Electricity Price Forecasting (EPF) is a task where exogenous covariates are fundamental to obtaining accurate predictions.

In this chapter, we address the two mentioned limitations by extending the neural basis expansion analysis, allowing it to incorporate temporal and static exogenous variables. We refer to the new method as NBEATSx. The main contributions of this chapter include:

1. **Incorporation of Exogenous Variables:** an extension to the NBEATS model to incorporate time-dependent and static exogenous variables based on a convolutional encoder to clean and learn useful information from these covariates.
2. **Interpretable Time Series Signal Decomposition:** the extended NBEATSx architecture allows it to decompose its predictions into the classic set of level, trend, and seasonality and identify the effects of exogenous covariates.
3. **Time Series Forecasting Comparison:** the proposed method achieves state-of-the-art performance on five EPF tasks, with accuracy improvements of almost 20% in comparison to the original NBEATS, and up to 5% over other well-established EPF-tailored methods (Lago et al., 2021).



(a) NBEATS **(b)** NBEATSx

Figure 5.1: Time series forecast decomposition for electricity price day-ahead forecasts using interpretable variants of NBEATS and NBEATSx. The fourth row presents the partial forecast of the exogenous block with electricity load and production covariates.

5.1.2 Methodology

We propose a new deep learning module to incorporate exogenous variables, compatible with any *decomposition-based model* such as the NHITS and NBEATS. The proposed method learns a context vector $\mathbf{C}_l \in \mathbb{R}^{N_c \times H}$ from temporal exogenous variables $\mathbf{X} \in \mathbb{R}^{N_x \times L+H}$ with a convolutional encoder:

$$\begin{aligned} \mathbf{C}_l &= \mathrm{TCN}(\mathbf{X}) \\ \hat{\mathbf{y}}_l^{exog} &= \sum_{i=1}^{N_c} C_{l,i}\theta_{l,i}^f \equiv \mathbf{C}_l\theta_l^f \end{aligned} \tag{5.1}$$

where $\hat{\mathbf{y}}_l^{exog}$ is the partial forecast of the exogenous block, $\theta_l^f \in \mathbb{R}^{N_c}$ are the learned coefficients for each context variable (see Figure 4.2), and the TCN is a *Temporal Convolutional Network* (Bai et al., 2018b). The TCN encoder mixes the N_x exogenous covariates to produce the context vector. Motivated by the NBEATS, the final forecast is produced by a *basis expansion*

Table 5.1: Benchmark datasets used in the experiments. For the five electricity markets considered, we report the test period and two covariate variables.

Market	Exogenous Variable 1	Exogenous Variable 2	Test Period
NP	day-ahead load	day-ahead wind generation	27-12-2016 to 24-12-2018
PJM	2 day-ahead system load	2 day-ahead COMED load	27-12-2016 to 24-12-2018
EPEX-FR	day-ahead load	day-ahead total France generation	04-01-2015 to 31-12-2016
EPEX-BE	day-ahead load	day-ahead total France generation	04-01-2015 to 31-12-2016
EPEX-DE	day-ahead zonal load	day-ahead wind and solar generation	04-01-2016 to 31-12-2017

operation on the context vector as the learned basis.

We added the exogenous block to the original NBEATS model and tested the proposed approach, named NBEATSx, on five electricity markets commonly used in the EPF literature as benchmarks. NBEATSx achieved SoTA performance, improving the accuracy by 20% over the NBEATSand by 5% over statistical and deep learning methods specialized in EPF.

Another advantage of the proposed method is interpretability. The effects of the exogenous variables on the forecast are isolated in the exogenous block. Figure 5.1 presents the interpretable decomposition of the NBEATSx method.

5.1.3 Experiments

Datasets. To evaluate our method's forecasting capabilities, we consider short-term electricity price forecasting tasks, where the objective is to predict day-ahead prices. Five major power markets[1] are used in the empirical evaluation, all comprised of hourly observations of the prices and two influential temporal exogenous variables that extend for 2,184 days (312 weeks, six years). From the six years of available data for each market, we hold two years out to test the forecasting performance of the algorithms. The length and diversity of the test sets allow us to obtain accurate and highly comprehensive measurements of the robustness and the generalization capabilities of the models.

Table 5.1 summarizes the key characteristics of each market. The Nord Pool electricity market (**NP**), which corresponds to the Nordic countries exchange, contains the hourly prices and day-ahead forecasts of load and wind generation. The second dataset is the Pennsylvania-New Jersey-Maryland market in the United States (**PJM**), which contains hourly zonal prices in the Commonwealth Edison (COMED) and two day-ahead forecasts of load at the system and COMED zonal levels. The remaining three markets are obtained from the integrated European Power Exchange (**EPEX**). Belgium (**EPEX-BE**) and France (**EPEX-FR**) markets share the day-ahead forecast generation in France as covariates since it is known to be one of the best predictors for Belgian prices (Lago et al., 2018b). Finally, the German market (**EPEX-DE**) contains the hourly prices, day-ahead load forecasts, and the country-level wind and solar generation day-ahead forecast.

[1]For the sake of reproducibility, we only consider datasets that are openly accessible in the EPFtoolbox library https://github.com/jeslago/epftoolbox (Lago et al., 2021).

Table 5.2: Forecast accuracy measures for day-ahead electricity price predictions of *ensembled models*. The ESRNN and NBEATS do not include time-dependent covariates. The smallest errors in each row are highlighted in bold.

		AR1	ESRNN	NBEATS	ARx1	LEARx*	DNN	NBEATSx-G	NBEATSx-I
NP	MAE	2.26	2.09	2.08	2.01	1.74	1.68	**1.58**	1.62
	rMAE	0.71	0.66	0.66	0.63	0.55	0.53	**0.50**	0.51
	sMAPE	6.47	6.04	5.96	5.84	5.01	4.88	**4.63**	4.70
	RMSE	4.08	3.89	3.94	3.71	3.36	3.32	**3.16**	3.27
PJM	MAE	3.83	3.59	3.49	3.53	3.01	**2.86**	2.91	2.90
	rMAE	0.79	0.74	0.72	0.73	0.62	**0.59**	0.60	0.60
	sMAPE	14.5	14.12	13.57	13.64	11.98	**11.33**	11.54	11.61
	RMSE	6.24	5.83	5.64	5.74	5.13	5.04	5.02	**4.84**
EPEX-BE	MAE	7.2	6.96	6.84	7.19	6.14	**5.87**	5.95	6.11
	rMAE	0.88	0.85	0.83	0.88	0.75	**0.72**	0.73	0.75
	sMAPE	16.26	15.84	15.80	16.11	14.55	**13.45**	13.86	14.02
	RMSE	18.62	16.84	17.13	18.07	15.97	15.97	**15.76**	15.80
EPEX-FR	MAE	4.65	4.65	4.74	4.56	3.98	3.87	3.81	**3.79**
	rMAE	0.78	0.78	0.80	0.76	0.67	0.65	**0.64**	**0.64**
	sMAPE	13.03	13.22	13.30	12.7	11.57	10.81	**10.59**	10.69
	RMSE	13.89	11.83	12.01	12.94	**10.68**	11.87	11.50	11.25
EPEX-DE	MAE	5.74	5.60	5.31	4.36	3.61	3.41	3.31	**3.29**
	rMAE	0.71	0.70	0.66	0.54	0.45	0.42	**0.41**	**0.41**
	sMAPE	21.37	20.97	19.61	17.73	14.74	14.08	**13.99**	**13.99**
	RMSE	9.63	9.09	8.99	7.38	6.51	5.93	5.72	**5.65**

Baselines We conducted an empirical study involving two types of *Autoregressive Models* (AR1 and ARx1; (Weron, 2014)), the *Lasso Estimated Auto-Regressive* (LEARx; (Uniejewski et al., 2016)), a parsimonious *Deep Neural Network* (DNN; (Lago et al., 2018a; Lago et al., 2021)), the original *Neural Basis Expansion Analysis* without exogenous covariates (NBEATS; (Oreshkin et al., 2020)), and the *Exponential Smoothing Recurrent Neural Network* (ESRNN; (Smyl, 2019)).

Key Results Table 5.2 summarizes the performance of the ensembled models where NBEATSx ensemble shows prevailing performance. It improves 18.77% on average for all metrics and markets when compared with the original NBEATS and 20.6% when compared to ESRNN without time-dependent covariates. For the ensembled models, NBEATSx RMSE improved on average 4.68%, MAE improved 2.53%, rMAE improved 1.97%,and sMAPE improved 1.25%. When comparing NBEATSx ensemble against DNN ensemble on individual markets, NBEATSx improved by 5.38% on the NordPool market, by 2.48% on the French market, and 2.81% on the German market. There was a non-significant difference of NBEATSx performance on **PJM** and **EPEX-BE** markets of 0.24% and 1.1%, respectively.

5.1.4 Conclusion

This chapter presents the NBEATSx, an extension to the NBEATS model to incorporate exogenous variables. The model relies on a novel Convolutional encoder that learns a useful representation of the exogenous variables to forecast the target time series. This encoder can be combined with any *decomposition-based* model to produce an interpretable decomposition of the marginal effects of each component.

The NBEATSx is tested on a set of benchmark datasets from the electricity price forecasting domain, but it can be straightforwardly applied to forecasting problems in other domains. The

qualitative evaluation shows that the interpretable configuration of NBEATSx can provide valuable insights to the analyst, as it explains the variation of the time series by separating it into a trend, seasonality, and exogenous components in a fashion analogous to classic time series decomposition. At the same time, NBEATSx improves over NBEATS by nearly 20% and up to 5% over LEAR and DNN models specialized for the Electricity Price Forecasting tasks.

5.2 Blood Glucose Forecasting with Pharmacokinetic Priors

5.2.1 Introduction

While including exogenous variables can significantly enhance prediction accuracy, as shown in the previous chapter, there are instances in which their integration without considering the domain knowledge yields minimal to no improvement McShinsky and Marshall, 2020; Rubin-Falcone et al., 2020. Forecasting healthcare time series is crucial for early detection of adverse outcomes and patient monitoring (Churpek et al., 2016; Gerry et al., 2020). Notably, healthcare time series data are often affected by exogenous factors, such as medication, exercise, or diet.

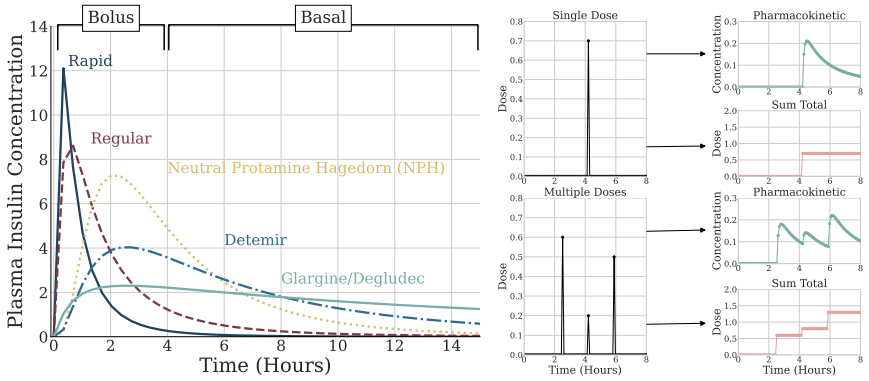
Leveraging the predictive power of exogenous variables presents unique challenges, especially in healthcare, where patient signals and exogenous factors often exhibit a temporal resolution mismatch. Furthermore, while the medication dose is usually known, we cannot directly observe the drug absorption and elimination parameters or plasma concentration of the drug. Consequently, their impact on the predictive signal remains hidden.

This chapter showcases an extension of the NHITS model with a novel pharmacokinetic encoder with local parameters for the blood glucose forecasting task with hidden medication effects. In this setting, blood glucose measurements are taken every five minutes, and exogenous variables such as carbohydrate ingestion or application of insulin are recorded as instantaneous points in time. The encoder leverages pharmacokinetic knowledge to generate patient-specific plasma insulin concentration profiles. Global parameters of the forecasting architecture are shared across multiple patients Semenoglou et al., 2021, but the parameters of the encoder are learned from the individual patient's historical data to address inter-subject variability in response to treatment. The main contributions are:

- **Pharmacokinetic Encoder.** A novel architecture module that generates plasma drug concentration profiles to capture time-dependent medication treatment effects.
- **Hybrid Global-Local Architecture.** Leverage global architectures (with parameters shared across patients) with a learnable patient-specific pharmacokinetic encoder.
- **State-of-the-Art Results.** Significant accuracy improvements on large-scale simulated and real-world blood glucose forecasting datasets.

5.2.2 Related Work

Deep learning models have permeated into the field of blood glucose level prediction, outperforming the accuracy of classic statistical methods (Xie and Wang, 2020). Past research includes various neural network designs ranging from *Multi Layer Perceptron* (MLP; Jahangir et al. 2017), *Recurrent Neural Networks* (RNN; Fox et al. 2018; Rabby et al. 2021; Rubin-Falcone et al. 2022) to *Temporal Convolutional Networks* (TCN; Li et al. 2019b).



(a) Pharmacokinetic modeling **(b)** Insuling featurization

Figure 5.2: (a) Pharmacokinetic modeling of plasma insulin concentration over time. (b) Instantaneous medication administrations are represented as sparse variables in time series data. We propose a pharmacokinetic encoder to effectively capture time-dependent plasma drug concentration.

Despite these advancements, two significant challenges remain. Firstly, existing architectures struggle to effectively utilize sparse exogenous variables, as shown in Rubin-Falcone et al., 2022. Medication and carbohydrate values are recorded as sparse values in data, which increases the difficulty in learning the effects of treatment on blood glucose levels. Early solutions propose adding exogenous values to the current point within an input window to replace sparse features, as shown in Fig. 5.2. However, this *sum-total* approach does not model time-dependent changes in medication. Secondly, many studies (Xie and Wang, 2020; Rubin-Falcone et al., 2022; Zaidi et al., 2021) rely on individualized networks for each patient, with limited adoption of cross-learning approaches that could enhance forecasting models by leveraging shared knowledge across patients.

5.2.3 Methodology

Our proposed approach addresses both challenges in a novel hybrid global-local modeling approach that learns patient-specific pharmacokinetics to improve deep learning model capabilities in forecasting blood glucose levels. The pharmacokinetic encoder informs the deep learning base model of time-dependent plasma drug concentration to enable more accurate forecasting of treatment outcomes.

The encoder C takes sparse exogenous medication variables as inputs and generates a concentration-time profile for a specified dose, d, using the following equation:

$$C(t, d, k_a^{(i)}) = \frac{d}{t k_a^{(i)} \sqrt{2\pi}} \exp\{-\frac{1}{2(k_a^{(i)})^2}(\log(t) - 1)^2\} \tag{5.2}$$

where t is the time and $k_a^{(i)}$ is the absorption rate constant for patient, i. Here, C is equivalent to the log-normal probability density function with $\sigma = k_a$ and $\mu = 1$ and a scaling factor,

d. Concentration profiles are generated under the assumption that all insulin administrations adhere to 100% bio-availability. As bio-availability equates to the area under the concentration curve, the scaled curve thus integrates to the specified dose, d.

Let N be the number of subjects in the data. The absorption rate constant, $k_{a_{\text{bolus}}} \in [0,1]^{Nx1}$ and $k_{a_{\text{basal}}} \in [0,1]^{Nx1}$ are represented as network embedding weights. During model training, sparse basal and bolus features are passed into the pharmacokinetic encoder along with their respective $k_a^{(i)}$ parameters, as shown in Fig. 5.3. The encoder then replaces sparse medication features in the training window with the corresponding concentration curve, as shown in 5.2. The values for $k_{a_{\text{bolus}}}^{(i)}$ and $k_{a_{\text{basal}}}^{(i)}$ are optimized jointly with all the models' parameters with gradient descent optimization.

Multiple insulin doses may occur at close intervals in an event often referred to as "insulin stacking" as shown in Fig. 5.2. The encoder leverages a matrix, $W \in \mathbb{R}^{L \times L}$, where L refers to the lag time, to perform vectorized operations that enable generating concentration curves for multiple doses in an input window in $\mathcal{O}(1)$ time. Each row of W represents a time count up to the input time, where the count shifts one-time step for each subsequent row. Concentration curves are generated for each row of W using the dose at the corresponding time index. The final concentration curve feature, which can reflect the impact of multiple doses, is generated by aggregating the individual concentration curves across time steps.

$$W = \begin{bmatrix} 0 & 1 & 2 & 3 & \cdots & L-1 \\ 0 & 0 & 1 & 2 & \cdots & L-2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{L \times L} \tag{5.3}$$

$$x'^{(i)}_{bolus} = \sum_{t=0}^{L} C(W_{t,j} \cdot f, x^{(i)}_{t_{bolus}}, k^{(i)}_{a_{bolus}}) \tag{5.4}$$

$$x'^{(i)}_{basal} = \sum_{t=0}^{L} C(W_{t,j} \cdot f, x^{(i)}_{t_{basal}}, k^{(i)}_{a_{basal}}) \tag{5.5}$$

Here, $x \in \mathbb{R}^{L \times 1}$ a sparse exogenous variable, where x_{bolus} and x_{basal} correspond to bolus and basal insulin doses, respectively, $f \in \mathbb{R}$ is the frequency of the data, and $x'^{(i)} \in \mathbb{R}^{1 \times L}$ is the new plasma insulin concentration feature.

For each time step within the forecast horizon, H, forecasts, $\hat{y}$, are generated by the model, f_θ, as a function of concentration curves, x', other non-medication time series, x, and static features s,

$$\hat{y}^{(i)}_{t:t+H} = f_\theta(y^{(i)}_{t-L:t}, x'^{(i)}_{t-L:t}, x^{(i)}_{t-L:t}, s^{(i)}). \tag{5.6}$$

In the proposed context, $x'_{t-L:t}$ consists of x'_{basal} and x'_{bolus}, x consists of x_{CHO}, and s includes patient age, weight, one-hot encoded subject identification number, and insulin pump type, if applicable.

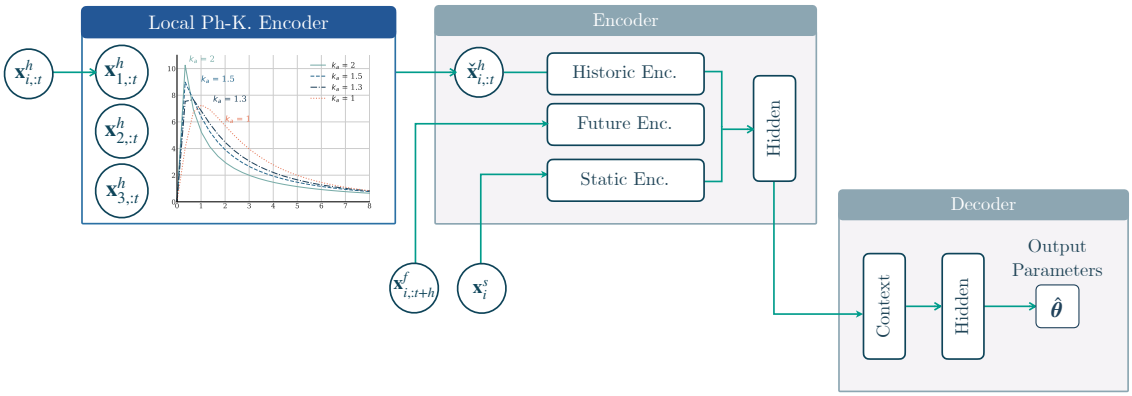


Figure 5.3: Hybrid Global-Local architecture. Global architectures with parameters shared across patients' time series with local parameters from a patient-specific pharmacokinetic (Ph-K.) encoder. Sparse medication dose time series are passed into the pharmacokinetic encoder and their respective k_a parameters to generate concentration curves. Concentration curves are then passed into a deep learning model that learns relevant hidden states (hidden) and output parameters, $\hat{\theta}$.

5.2.4 Experiments

Evaluation The task is to forecast blood glucose levels 30 minutes into the future consistent with prior work (Rubin-Falcone et al., 2022; Xie and Wang, 2020), based on a 10-hour history of blood glucose, carbohydrate, insulin basal, and insulin bolus values. Trained models are used to generate rolling window forecasts at one-step intervals. In accordance with prior work (Rubin-Falcone et al., 2022; Xie and Wang, 2020), forecasts are evaluated using mean absolute error (MAE), and root mean squared error (RMSE), computed across 8 trials of individually trained models. Given the critical importance of healthcare outcomes related to blood glucose levels exceeding or falling below safe thresholds, the model's performance is evaluated exclusively at time points where blood glucose levels reach or fall below certain thresholds relevant to hypoglycemic and hyperglycemic events (i.e., $\leq$70 mg/dL and $\geq$180 mg/dL).

Simulated data An open-source python implementation of the FDA-approved UVa/Padova Simulator (2008 version) (Xie, 2018; Visentin et al., 2016) provides clinical and biological parameters for 30 patients (10 adults, 10 adolescents, and 10 children) with Type-I diabetes. Clinical parameters include information on age and weight, and biological parameters include information on insulin-glucose kinetics, such as absorption constants. We generate 54 days of data for each patient to match the median training set of the OhioT1DM, and approximately 9 days for the test set. In a similar approach to (Rubin-Falcone et al., 2022), the meal schedule used to generate simulated data was based on the Harrison-Benedict equation (Arthur Harris and Gano Benedict, 21919) with approximately 3 meals per day and no additional snacks.

OhioT1DM 12 de-identified individuals with Type 1 diabetes are included in the OhioT1DM 2018 and 2020 datasets (Marling and Bunescu, 2020). Each patient has approximately 8 weeks of data that includes blood glucose (mg/dL) measurements recorded with a CGM at a frequency of 5-minutes. The dataset consists of both female (n=5) and male (n=8) subjects with unspecified

Table 5.3: Mean absolute error (MAE) computed for model predictions across all values in the forecast horizon and only across critical values in the forecast horizon (blood glucose $\leq 70| \geq 180$)

Simulated Dataset				
Model	**All Values**		**Critical Values**	
	Baseline	Exogenous	Baseline	Exogenous
ETS	10.040	–	13.531	–
MLP	9.734	8.653	12.189	9.887
LSTM	8.731	8.372	10.863	9.996
TCN	16.772	12.234	28.913	18.122
TFT	7.788	6.762	9.664	7.828
NHITS	8.333	7.282	10.184	8.353
TFT- Sum-Total	–	6.698	–	7.634
NHITS- Pharmacokinetic	–	7.027	–	7.917
TFT- Pharmacokinetic	–	**6.466**	–	**7.514**
OhioT1DM Dataset				
ETS	10.126	–	11.781	–
MLP	10.568	10.275	11.805	11.434
LSTM	10.604	9.606	12.999	11.144
TCN	11.011	11.090	12.693	13.212
TFT	9.605	9.515	10.686	10.555
NHITS	9.234	8.954	10.436	10.117
NHITS- Sum-Total	–	8.988	–	10.092
NHITS- Pharmacokinetic	–	**8.769**	–	**10.007**
TFT- Pharmacokinetic	–	9.492	–	10.484

ages within the ranges of 20-40, 40-60, and 60-80. Sparse exogenous features in the data include basal rate (in units per hour), temporary basal rate (units per hour), bolus insulin (units), and carbohydrate intake. We use the specified train and test partitions provided in the dataset. The maximum number of test samples consistent across all patients (2691 samples) was used to obtain equal test sets of approximately 9 days.

Key Results Table 5.3 presents the main results. The pharmacokinetic encoder significantly improves forecasting accuracy for the top-performing deep learning model by approximately 4.4% and 2.1% for simulated and OhioT1DM datasets, respectively. For the OhioT1DM dataset, the NHITS-Pharamacokinetic model has significantly lower MAE than the NHITS-Sparse Exogenous model for forecasts evaluated at all values (t-test p-value: 4.44e-05) and critical values (t-test p-value: 1.6e-2).

We showcase the efficacy of our novel hybrid global-local model in producing more accurate forecasts than models trained per patient. Global models consistently outperform local models, on average, for each OhioT1DM subject, as shown in Fig. 5.4. For the OhioT1DM dataset, the hybrid global-local pharmacokinetic model achieves a significantly lower (t-test p-value: 1.29e-2) MAE by 14.6% over patient-specific pharmacokinetic models.

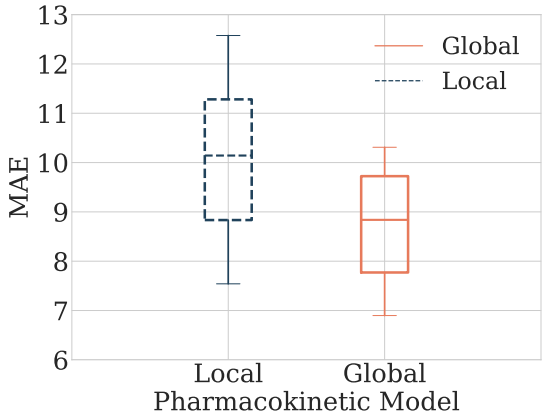


Figure 5.4: The boxplots show average MAE computed across 8 trials for global (orange) and local (blue) models for OhioT1DM subjects. Models trained on all subject data, or global models, have lower MAE, on average, than models trained on individual subjects or local models.

5.2.5 Other applications and extensions

The proposed pharmacokinetic encoder can be easily extended to other applications in healthcare beyond glucose forecasting. The functions and parametrization of the encoder can be replaced to model other phenomena, while the global-local architecture will incorporate this knowledge and simultaneously learn general patterns and particular dynamics of each time series.

As a concrete example, it can be adapted for epidemic forecasting as follows. It is important to forecast the onset, magnitude, and temporal evolution of outbreaks of seasonal influenza, as this knowledge can inform preparedness and reduce the impact of this disease on the society. The encoder described above can incorporate prior knowledge of epidemiology experts regarding the ranges of expected dynamics of the outbreak events and the seasonal probability priors of their occurrence, conditioned whenever possible on relevant exogenous covariates encoding societal factors such as people's mobility, or environmental factors such as weather. A complementary approach would be to further improve the models by including other relevant covariates, such as over-the-counter medication sales, search engine queries, or social media mentions, that can inform the estimation of flu incidence rates (Farrow et al., 2015).

5.3 Conclusion

The proposed pharmacokinetic encoder enables deep learning models, such as the NHITS and TFT, to improve their accuracy. Current models that only include sparse features may be under-leveraging the potential utility of exogenous information. The pharmacokinetic encoder can have multiple beneficial applications in clinical practice, such as issuing early warnings about unexpected treatment responses or helping to characterize patient-specific treatment effects regarding drug absorption and elimination characteristics. Characterizing patient-specific pharmacokinetic parameters holds substantial potential for advancing the development of more effective treatments by tailoring medication to individual patients.

Part III

Towards Time Series Foundation Models

Transfer learning is a popular technique in machine learning that can provide huge benefits in both performance and computational cost when solving tasks. It has been successfully exploited in domains such as Computer Vision (CV) and Natural Language Processing (NLP), constituting the basic principle behind the recent foundational models. The goal of this chapter is to advance applications of transfer learning in time series tasks, consisting of a large-scale study on the conditions that enable the transferability of models between tasks on two aspects: the model's architecture and data characteristics. First, we compare state-of-the-art models with multiple architecture types, number of parameters, forecasting strategy, and fine-tuning steps, on source datasets of various sizes. Second, we propose a distance metric between tasks that strongly correlates with the performance of pre-trained models.

Chapter 6

Transfering Neural Forecast Models

6.1 Motivation

Neural networks are renowned for their capacity to learn hierarchical representations of inputs. They acquire higher-level features by composing simple functions across layers, learning intricate relationships, and enabling accurate predictions. In the forecasting field, these models excel due to their improved accuracy in large data settings and their ability to simplify the forecasting pipelines, as shown by their success in recent competitions (Makridakis et al., 2020; Makridakis et al., 2021) and their industry adoption (Wen et al., 2017; Lim et al., 2021b).

Despite the recent successes of neural networks in time series forecasting, their full expressivity often remains underutilized in scenarios with small groups of series, and the considerable computational costs remain an adoption barrier. Transfer learning offers a solution by pre-training deep network models on large-scale datasets to capture expressive representations and domain knowledge. These pre-trained models can be fine-tuned on smaller datasets, facilitating effective generalization to specific forecasting tasks with limited data. In addition to knowledge sharing, transfer learning also enables lightning-fast predictions at a fraction of the computational cost while significantly enhancing the models' generalization capabilities (Yosinski et al., 2014; Zhuang et al., 2021).

The objective of this chapter is to advance the applications of transfer learning in time series forecasting tasks. We aim to enhance our understanding of the technique and make these tools widely accessible. The contributions are summarized as follows:

- **A Unified Model Implementation.** To ensure consistent experimental settings and facilitate direct comparisons, we introduced a unified implementation of various neural network-based forecasting models, allowing us to control variables such as architecture size and optimization and standardize the transfer learning tasks.
- **Accessible Transfer Learning Comparison.** We explore the impact of neural forecasting innovations on transfer learning tasks and conduct an extensive comparative analysis of various models and datasets with zero-shot and fine-tuning. We make the comparison experiments' code, along with the pre-trained models, publicly accessible.

- **Transferability's Enabling Conditions Exploration.** We document insights from several transferability-enabling conditions. First, we focus on model-related aspects, such as architecture, with an emphasis on the number of parameters and forecasting strategy. Second, we propose a measure of the *distance* between the source and target tasks strongly correlating with the pre-trained models' performance.

6.2 Related Work

6.2.1 Transfer Learning Literature

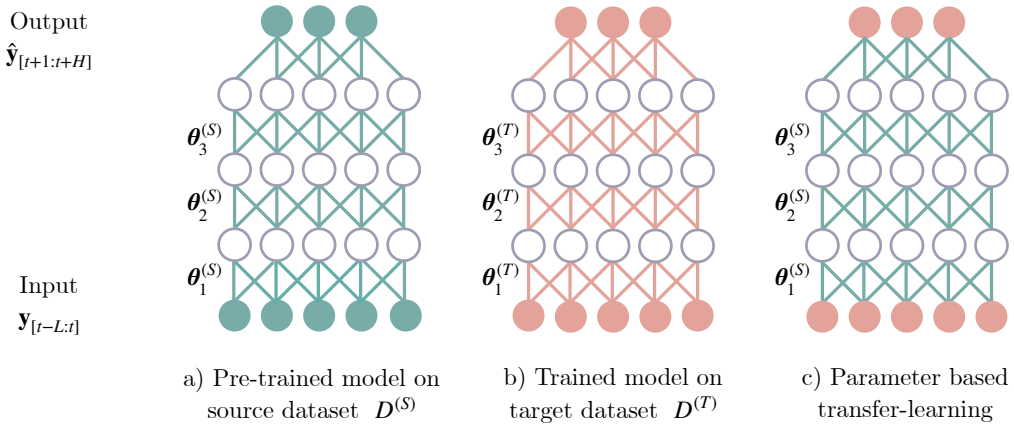
Transfer learning refers to the techniques that aim to improve a model on a target domain by transferring information or knowledge from another related training/source domain. It has been widely studied for tabular data, computer vision, and natural language processing. There are several different transfer-learning solutions, with data-based solutions focusing on transferring knowledge via the distribution adjustment and transformation of data; and model-based approaches focusing on model regularization, model ensembling, and model parameter sharing. For a throughout review, we refer to the following surveys (Weiss et al., 2016; Zhuang et al., 2021).

Notable examples of the distribution adjustment solutions include sample weighting strategies for domain adaptation (Huang et al., 2006; Dai et al., 2007), feature transformation strategies based on minimization of distribution differences (Shen et al., 2018) and feature mappings (Wang et al., 2018). Notable examples of model-based approaches to transfer learning include model parameter sharing (Yosinski et al., 2014), model consensus regularization, and model ensembling (Yao and Doretto, 2010). In this work, we mostly focus on model parameter sharing, which is the most popular transfer learning technique in neural network applications.

6.2.2 Cross and Transfer Learning Relationship

In recent years, neural forecasting models have improved over classical methods, overcoming previous computational and accuracy limitations (Makridakis et al., 2018). This progress can be attributed to the widespread adoption of the cross-learning technique and the use of global models leveraging data from large collections of related time series. Notable applications include the M4 and M5 competition top performers (Smyl, 2019; Montero-Manso et al., 2020; Oreshkin et al., 2020) and popular industry models such as DeepAR, MQCNN, and TFT (Salinas et al., 2020; Wen et al., 2017; Lim et al., 2021b). This success has renewed interest in industry and academia, leading to numerous neural forecasting innovations (Längkvist et al., 2014; Benidis et al., 2020b).

The cross-learning and transfer-learning via model parameter-sharing are closely related, with their main differences in two key aspects. Firstly, transfer learning refers to leveraging the pre-trained global model to different target datasets. Secondly, transfer learning offers the flexibility to minimize the re-training procedure or entirely skip it, which is particularly suited for



a) Pre-trained model on source dataset $D^{(S)}$

b) Trained model on target dataset $D^{(T)}$

c) Parameter based transfer-learning

Figure 6.1: The figure shows a three-layer fully connected network predictive function. Classic forecasting applications optimize distinct model parameters for source $D^{(S)}$ and target $D^{(T)}$ datasets, a) and b) columns. Parameter-based transfer-learning leverages source dataset knowledge by using a pre-trained model's parameters $\boldsymbol{\theta}_l^{(S)}$, to initialize another model's parameters $\boldsymbol{\theta}_l^{(T)}$ that can specialize on a target dataset.

small dataset applications. On the time series front, the technique's use has been limited but has shown very promising early results in classification (Fawaz et al., 2018), anomaly detection (Wen and Keyes, 2019), contrastive learning pre-training (Eldele et al., 2021) and forecasting meta learning (Oreshkin et al., 2021).

In neural forecasting applications, an immediate approach for transfer learning via model parameter-sharing is using pre-trained neural networks that can be updated using the smaller target dataset. The updating degree determines the scenarios, ranging from zero-shot to k-shot and fine-tuning [1].

6.3 Transfer Learning Notation

We now present a formal definition of transfer learning, which will be utilized throughout the rest of the paper.

Definition 6.1

(Domain and Task). Two components define a domain, a feature space $\mathcal{X}$ and a marginal probability distribution $\mathbb{P}(\mathbf{x})$, with realizations $\mathbf{x} \in \mathcal{X}$. A task $\mathcal{T} = \{\mathcal{Y},\ f(\cdot)\}$ is defined by two components, a dependent variable space $\mathcal{Y}$ and a *predictive function* $f : \mathcal{X} \mapsto \mathcal{Y}$. In the case of a simple forecasting task, the dependent variable space is $\mathcal{Y} = \{\mathbf{y}_{[t+1:t+H]}\}$, which corresponds to the set of H future values of a series $\mathbf{y}$ at time t, and the feature space $\mathcal{X} = \{\mathbf{y}_{[t-L:t]}\}$ is the set of L past observed values (lags) of the series[a]. The forecasting task

[1] It is important to note that in classification settings, the k-shot category refers to the number of labeled data on the target task; k-shot refers to the number of training iterations in the forecasting domain.

is to learn or estimate the parameters $\boldsymbol{\theta}$ in the following regression:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \mathbb{P}\left(\mathbf{y}_{[t+1:t+H]} | \ \mathbf{y}_{[t-L:t]}, \boldsymbol{\theta}\right) \tag{6.1}$$

[a]Note that the feature space is not restricted to autoregressive features can also include exogenous variables predictors such as static data, past information, or information available at the time of the predictions.

Definition 6.2

(Transfer Learning). A training dataset D is defined as the set of realization pairs $D = \{(\mathbf{x}, \mathbf{y}) | \ \mathbf{x} \in \mathcal{X}, \ \mathbf{y} \in \mathcal{Y}\}$. Transfer learning breaks the traditional assumption that the training and the test/target datasets are taken from the same domain. Formally, consider two different source and target dataset ($D^{(S)} \neq D^{(T)}$) along their corresponding learning tasks; transfer-learning is the process of improving the predictive function $f^{(T)}(\cdot)$ by using information directly from the source domain $D^{(S)}$ or indirectly through $f^{(S)}(\cdot)$.

In this work, we consider a case of *homogeneous transfer-learning* case where the (autoregressive) feature space $\mathcal{X}^{(S)} = \mathcal{X}^{(T)}$, and the forecasting tasks are the same $\mathcal{T}^{(S)} = \mathcal{T}^{(T)}$, but the distribution of the features are only related $\mathbb{P}^{(S)}(\mathbf{x}) \neq \mathbb{P}^{(T)}(\mathbf{x})$. We focus, in particular, on the parameter-based transfer learning approach.

6.3.1 Zero-shot, K-shot, and Finetuning

In general, the forecasting learning task can be translated into a function estimation problem using the classic Empirical Risk Minimization (ERM) framework (Vapnik, 1999), where the objective is to minimize the expected loss function at the dataset D level:

$$f^* := \underset{f \in \mathcal{F}(\Theta)}{\arg\min}\, \mathbb{E}_D \left[\mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) \right] \quad \Longleftrightarrow \quad \hat{\theta} := \underset{\boldsymbol{\theta} \in \Theta}{\arg\min}\, \mathbb{E}_D \left[\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}(\mathbf{y}_{[t-L:t]} | \, \boldsymbol{\theta})) \right] \tag{6.2}$$

In this work pre-training and retraining the considered models is performed using stochastic gradient descent updates (Robbins and Monro, 1951). We used pre-trained model parameters $\boldsymbol{\theta}^{(S)} = \boldsymbol{\theta}_0^{(T)}$ as the initial condition, and update the parameters for the target task using the following rule:

$$\text{Sample } (\mathbf{x}, \mathbf{y}) \sim \mathbb{P}(D^{(T)}) \quad \text{update} \quad \boldsymbol{\theta}_k^{(T)} = \boldsymbol{\theta}_{k-1}^{(T)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{y}, f(\mathbf{x}, \boldsymbol{\theta})) \tag{6.3}$$

Where the re-trained model's parameters after k steps are denoted by $\boldsymbol{\theta}_k^{(T)}$, the learning rate is denoted by α. The difference between zero-shot, k-shot, and finetuning dwells in the amount of updating steps performed. Zero-shot requires no re-training, finetuned parameters are re-trained until an early stopping halt, and k-shot performs exactly k-updates.

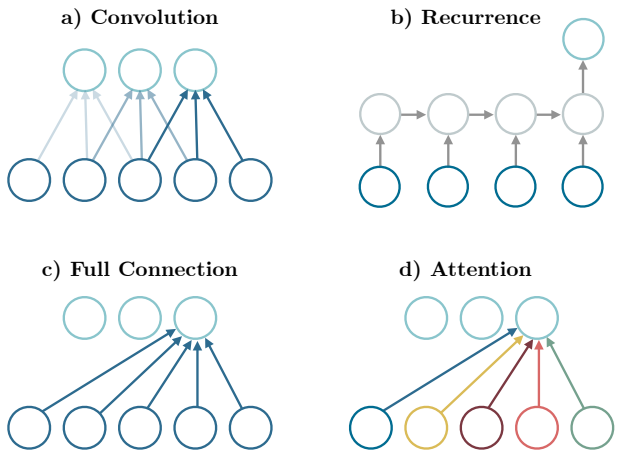


Figure 6.2: A taxonomy of neural forecasting models, defined by their fundamental building blocks.

6.4 Forecasting Baselines

6.4.1 Neural Forecast Models

We chose well-performing neural forecasting architectures identified by their fundamental building blocks to establish experiment baselines: fully connected layers, recurrent and convolutional decoders, and attention mechanisms.

In the fully connected architectures model group, we consider the basic multi-layer perceptron (MLP) (Rosenblatt, 1961), the neural basis expansion network (NBEATS) (Oreshkin et al., 2020) and the neural hierarchical interpolation network (NHITS) (Challu et al., 2023b). For architectures with recurrent and convolutional encoders, we include the classic long-short-term memory network (LSTM) (Gers et al., 2000; Sak et al., 2014), a temporal convolution network (TCN) (Oord et al., 2016; Bai et al., 2018b), and the deep autoregressive network (DeepAR) (Salinas et al., 2020). Finally, for architectures that integrate the attention mechanism, we consider the temporal fusion transformer (TFT) (Lim et al., 2021b) and the patch time series transformer (PatchTST) (Nie et al., 2023).

All architectures are implemented and trained in PyTorch (Paszke et al., 2019), with the code publicly available at the NeuralForecast repository.

6.4.2 Automated Statistical Forecast

We include in our experiments two workhorse statistical forecasting baselines: the Autoregressive Integrated Moving Average model (ARIMA) and the simple Naive baseline, which surprisingly provides accurate performance reference. For ARIMA, we rely on a modern implementation tool that automatically explores its hyperparameters (Hyndman and Khandakar, 2008; Garza et al., 2022).

Table 6.1: Summary of datasets used in the empirical study. All the datasets are used in the forecasting transfer-learning experiments. We use larger datasets as source domains and smaller as target domains.

Dataset	Purpose	Time Series	Frequencies
Wikipedia	Source	5e5	{Daily}
M4	Source	1e5	{Yearly, Quarterly, Monthly, Daily}
M3	Source	3003	{Yearly, Quarterly, Monthly, Daily}
M1	Target	1001	{Yearly, Quarterly, Monthly}
M3	Target	3003	{Yearly, Quarterly, Monthly, Daily}
Tourism	Target	1311	{Yearly, Quarterly, Monthly}

Table 6.2: Model evaluation based on **sMAPE** on **zero-shot** forecasting transfer learning (lower is better). The source dataset is the first vertical column, while the target dataset is the second column. Metrics are averaged over eight runs and standard deviation in brackets, best results are highlighted in bold.

		Freq	Naive	ARIMA	MLP	LSTM	TCN	DeepAR	TFT	Trans	NHITS	PatchTST
M3	**M1**	Y	22.43	19.53	**15.96**	36.51	32.49	29.28	18.08	51.48	17.15	16.50
		Q	18.38	17.57	18.15	24.46	22.93	18.31	17.42	23.24	17.30	**15.74**
		M	18.67	13.72	16.35	16.31	16.37	17.27	15.40	15.37	14.43	**13.27**
	Tour	Y	34.80	39.76	36.02	60.18	37.79	**29.55**	35.41	106.13	38.59	32.34
		Q	54.19	49.74	58.02	80.07	70.30	48.17	52.55	78.67	56.16	**46.76**
		M	72.24	65.42	65.74	64.77	64.78	71.47	**63.52**	66.31	66.28	63.76
M4	**M1**	Y	22.43	19.53	15.70	24.15	25.41	22.94	15.62	21.50	**15.62**	16.52
		Q	18.38	17.57	15.53	17.09	18.39	17.98	15.48	16.47	15.33	**15.18**
		M	18.67	13.72	13.38	15.78	16.01	17.62	13.24	12.94	**12.71**	12.80
	M3	Y	17.88	18.13	16.12	19.55	21.33	20.59	15.64	22.87	15.68	**15.30**
		Q	11.32	10.57	9.32	10.47	11.20	12.46	9.37	10.37	**8.95**	9.06
		M	16.85	13.47	13.46	13.60	14.48	15.40	13.24	13.13	12.97	**12.82**
		D	8.56	**6.91**	8.44	10.79	10.03	15.54	8.47	8.27	9.55	8.88
	Tour	Y	34.80	39.76	35.25	31.38	32.15	**29.50**	31.74	51.32	31.19	31.13
		Q	54.19	49.74	45.41	44.32	44.92	50.70	45.49	51.47	45.09	**43.62**
		M	72.24	65.42	64.80	62.88	65.09	71.12	**62.40**	63.96	63.26	62.85
Wiki	**M3**	D	8.56	**6.91**	7.91	9.15	9.57	16.71	7.45	7.91	7.72	9.88

6.5 Empirical Evaluation

6.5.1 Datasets

All the datasets used in our empirical studies are publicly available and have been previously utilized in the neural forecast literature. In Table 6.1, we concisely summarize each dataset's sources, time frequencies, and purpose for our transfer learning tasks. Below we describe in greater detail the origins and dataset characteristics.

- Wikipedia (**Wikipedia**): This large dataset comprises time series data from Wikipedia articles' visits and logs. With approximately half a million series, this dataset provides valuable insights into the temporal patterns of Wikipedia usage.

Table 6.3: Model evaluation based on **sCRPS** on **zero-shot** forecasting transfer learning; lower values are better. The source dataset is the first vertical column, while the target dataset is the second column. Metrics are averaged over eight runs and standard deviation in brackets, best results are highlighted in bold.

		Freq	Naive	ARIMA	MLP	LSTM	TCN	DeepAR	TFT	Trans	NHITS	PatchTST
M3	**M1**	Y	0.184	0.160	0.116	0.216	0.264	0.198	**0.093**	0.393	0.130	0.134
		Q	0.102	**0.088**	0.124	0.149	0.115	0.095	0.125	0.148	0.105	0.112
		M	0.197	0.134	0.125	0.133	0.130	0.164	0.119	0.121	**0.107**	0.110
	Tour	Y	0.112	0.114	0.112	0.161	0.112	**0.096**	0.114	0.272	0.123	0.098
		Q	0.206	0.121	0.163	0.279	0.194	0.127	0.146	0.224	0.157	**0.109**
		M	0.412	**0.145**	0.202	0.200	0.207	0.208	0.197	0.182	0.172	0.173
M4	**M1**	Y	0.184	0.160	0.128	0.211	0.205	0.133	0.119	0.186	**0.118**	0.140
		Q	0.102	0.088	0.077	0.092	0.110	0.085	0.076	0.080	0.079	**0.072**
		M	0.197	0.134	0.118	0.127	0.133	0.160	0.119	**0.103**	0.111	0.110
	M3	Y	0.138	0.162	0.139	0.159	0.170	0.156	0.134	0.176	0.136	**0.122**
		Q	0.086	0.079	0.069	0.081	0.086	0.094	0.070	0.075	**0.066**	0.067
		M	0.138	0.088	0.087	0.093	0.099	0.106	0.088	0.085	**0.085**	0.085
		D	0.067	**0.053**	0.061	0.094	0.078	0.137	0.063	0.061	0.071	0.068
	Tour	Y	0.112	0.114	0.107	0.098	**0.091**	0.096	0.099	0.148	0.101	0.102
		Q	0.206	0.121	0.107	0.104	0.119	0.150	0.104	0.142	0.102	**0.096**
		M	0.412	**0.145**	0.170	0.189	0.198	0.217	0.164	0.151	0.148	0.169
Wiki	**M3**	D	0.067	**0.053**	0.068	0.082	0.085	0.147	0.064	0.069	0.066	0.079

- Makridakis Competitions (**M1**, **M3**, **M4**): The **M4** dataset (Makridakis et al., 2020) is a comprehensive one hundred thousand time series collection from diverse domains such as finance, industry, macroeconomics, and microeconomics. It offers a wide range of frequencies from yearly to hourly intervals. The **M3** dataset (Makridakis and Hibon, 2000) shares similarities with **M4**, although it consists of a smaller set of 3003 series. Notably, the **M3** dataset (Fiorucci et al., 2016; Spiliotis et al., 2020; Hyndman and Khandakar, 2008) has been extensively utilized for research on statistical methods. Additionally, the **M1** dataset (Makridakis et al., 1982) comprises 1001 time series representing demography, industry, and economics. **M1** dataset's time series is only available in yearly, quarterly, and monthly frequencies.
- The **Tourism** dataset (Athanasopoulos et al., 2011) consists of yearly, quarterly, and monthly records of tourist visits and other indicators sourced from tourism bodies or academics. This dataset has been utilized in previous tourism forecasting studies.

6.5.2 Evaluation Metrics

To assess the models' probabilistic and point forecast accuracy, we employ the Scaled Continuous Ranked Probability Score (sCRPS; Makridakis et al. 2022) and the Symmetric Mean Average

Percentage Error (sMAPE; Hyndman and Koehler 2006).

$$\begin{aligned}\mathrm{sCRPS}(\hat{\mathbb{P}}, \mathbf{y}) &= \frac{2}{H \times |[i]|} \sum_{i} \sum_{\tau=t+1}^{t+H} \frac{\int_0^1 \mathrm{QL}(\hat{\mathbb{P}}_{i,\tau}, y_{i,\tau})_q dq}{\sum_i |y_{i,\tau}|} \\ \mathrm{sMAPE}(\mathbf{y}, \hat{\mathbf{y}}) &= \frac{200}{H \times |[i]|} \sum_{i} \sum_{\tau=t+1}^{t+H} \frac{|y_\tau - \hat{y}_\tau|}{|y_\tau| + |\hat{y}_\tau|}\end{aligned} \tag{6.4}$$

where $\mathrm{QL}(\hat{\mathbb{P}}_{i,\tau}, y_{i,\tau})_q$ stands for the quantile loss at the q level, between the estimated forecast probability $\hat{\mathbb{P}}_{i,\tau}$ and the observation $y_{i,\tau}$. We use a Riemann approximation to the sCRPS with dq quantile intervals of 1 percent.

In addition to measuring forecast accuracy, we also evaluate the computational time complexity of the methods. We conduct the experiments on an EC2 g5.4xlarge instance with NVIDIA Tesla M60 GPUs and 16 CPU cores.

6.5.3 Key Results

We define a transfer-learning task by combining a source and a target dataset. We report the performance of zero-shot models, that is, pre-trained models with no additional training. We use large datasets as sources, for which we select **Wikipedia**, and **M4**. The target datasets, **M3**, **M1**, and **Tourism** are smaller.

6.5.4 Zero-shot

Our findings summarized in Table 6.2 and Table 6.2 confirm that neural networks can acquire general forecasting knowledge and effectively apply it in zero-shot transfer. Pre-training on larger source datasets improves performance on downstream target tasks. In addition, the latest architectures, including NHITS, TFT, and PatchTST, are better suited for leveraging large datasets' information. The best pre-trained PatchTST model shows improvements over automated ARIMA of 8.89% on average across all tasks. On the other side, other architectures such as LSTM do not manage to improve on ARIMA's performance on average.

These results also show that, in most cases, the accuracy of transferred neural forecast models improves when the source dataset is larger. For example, the best pre-trained model on Tourism monthly is the NHITS, with a sCRPS of 0.172 when trained in M3 and 0.148 when trained in M4, representing a 14% improvement. In section 6.6.1 we explain this finding based on the *distance* between tasks.

6.5.5 Inference Time

One of the most significant advantages of zero-shot transfer learning is the ability to produce forecasts from a model with a fraction of the computational cost of complete training procedures.

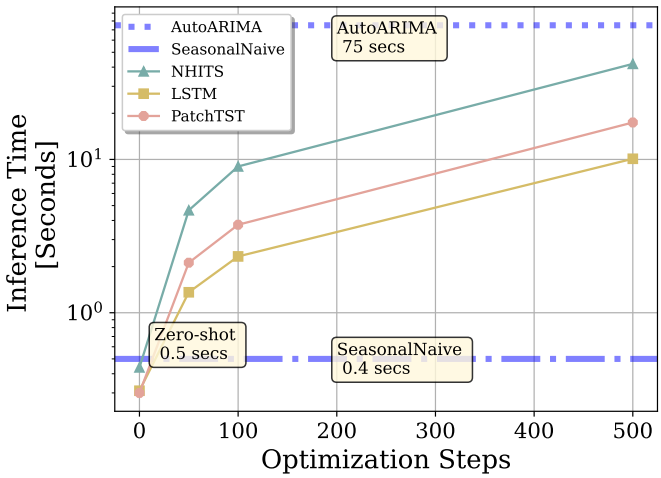


Figure 6.3: Inference time comparison for selected neural method and baselines. Values reported correspond to the average inference time for the complete target datasets from Table 6.2.

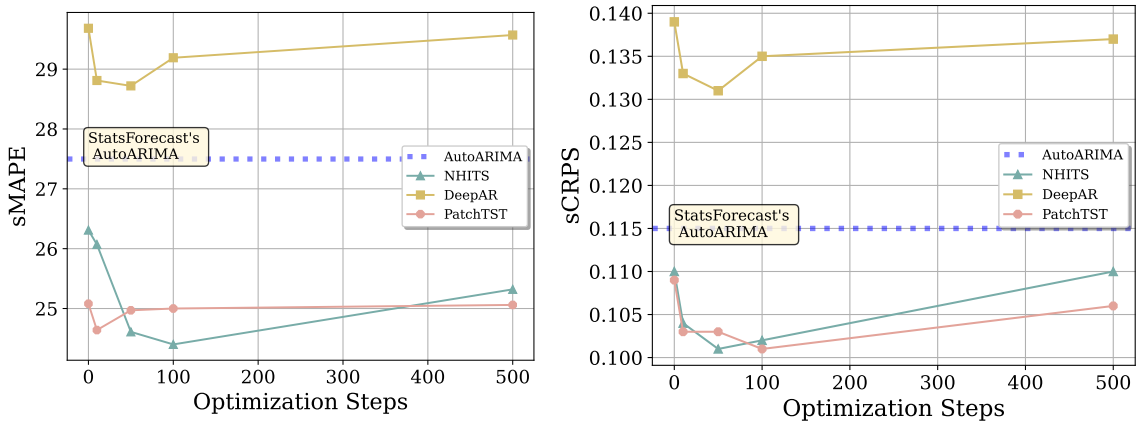


Figure 6.4: Finetuning forecasting performance for selected neural forecast models, measured by average sMAPE and sCRPS across all transfer-learning tasks.

In Figure 6.3, we report the average inference time to forecast a complete target dataset. Zero-shot inference requires almost the same computational time as the simplest baseline, the `Naive`, and 145 times less time to fit and predict with the highly efficient `ARIMA` from StatsForecast (Garza et al., 2022). Moreover, neural methods can be fine-tuned for up to 500 training iterations, requiring less time than the `ARIMA`.

6.5.6 Finetuning

6.4 presents the results of fine-tuning neural methods in the target task for multiple optimization steps. The results show that it is possible to improve the zero-shot performance by customizing the pre-trained models on the task at hand. Updating the model weights on the smaller datasets tends to be beneficial. However, the benefits diminish with the training iterations, and in the case

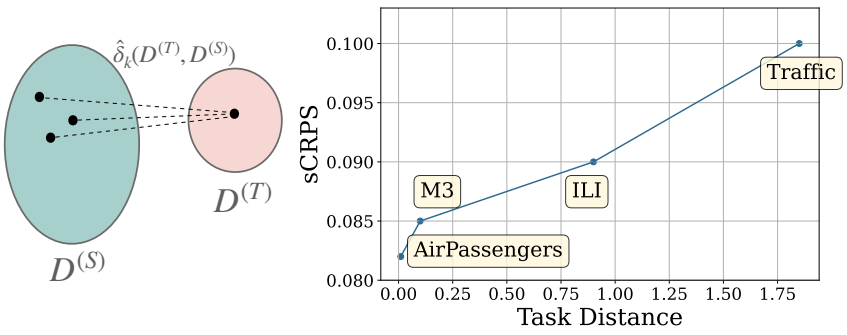


Figure 6.5: Transferability decreases as the distance between tasks increases. (a) Distance nearest neighbor estimation between forecasting tasks. (b) NHITS zero-shot performance as a function of task distance.

of large models, there is a potential risk of overfitting, for which early stopping is necessary. In this experiment, the finetune NHITS model achieves average sMAPE improvements considering all tasks of 11.26% over the ARIMA with just 100 optimization steps and under 10 seconds of fine-tuning per task.

6.6 Transferability Enabling Conditions

6.6.1 Effects of Tasks Distance

In this section, we explore how differences in the temporal patterns of the time series between the source and target datasets can affect the transferability of pre-trained models, regardless of their architecture. We first propose a measure of the *distance* between the source and target datasets that strongly correlates with their transferability and, therefore, the performance of pre-trained models.

Let a window of autoregressive features and forecast horizon be denoted $\mathbf{w} = \mathbf{y}_{[t-L:t+H]}$. Let the source domain and target forecasts datasets be $D^{(S)} = \{(\mathbf{y}_{[t-L:t]}, \mathbf{y}_{[t+1:t+H]}) \,|\mathbf{y} \in \mathcal{Y}^S\}$ and $D^{(T)} = \{(\mathbf{y}_{[t-L:t]}, \mathbf{y}_{[t+1:t+H]}) \,|\mathbf{y} \in \mathcal{Y}^T\}$. We estimate the distance between the target and the source tasks as follows:

$$\hat{\delta}_k \left(D^{(T)},\ D^{(S)}\right) = \frac{1}{k \times |\mathcal{W}_t|} \sum_{\mathbf{w} \in \mathcal{W}_t} \sum_{\mathbf{w}_\kappa \in \mathrm{N}_k(\mathbf{w})} ||\mathbf{w} - \mathbf{w}_\kappa||_2 \tag{6.5}$$

where $\mathrm{N}_k(\mathbf{w})$ is the neighborhood of $\mathbf{w}_\kappa$ closest source windows to the $\mathbf{w}$ target window. This proposed distance is motivated by the fact that most deep learning models can be seen as functions that map a fixed amount of historical values (lags) L, to the forecasting horizon of interest H. By setting the window size $\mathbf{w}$ to $L + H$, $\hat{\delta}_k \left(D^{(T)},\ D^{(S)}\right)$ simultaneously measures if the source dataset contains windows with similar input-to-output mappings as the target dataset.

For this experiment, we selected additional target datasets exhibiting varying differences in frequency, seasonality, and trends from the nature of the source datasets in Table 6.1.

- Box-Jenkins Airline Passengers (**AirPassengers**): This classic dataset (Box et al., 2015a) captures monthly totals of international passengers from 1949 to 1960, exhibiting prominent trends and seasonal patterns.
- Influenza-like Illness (**ILI**): This dataset reports weekly recorded influenza-like illness patients from the Centers for Disease Control and Prevention of the United States from 2002 to 2021 (US-CDC, 2017).
- San Francisco Bay Area Highway Traffic (**Traffic**): The California Department of Transportation collected this dataset (Dua and Graff, 2017), reporting hourly road occupancy rates of 862 sensors from January 2015 to December 2016.

Figure 6.5 presents the zero-shot performance of the `PatchTST`pre-trained in **M4**, the model with the best overall performance, against the task distance for four target datasets. Our experiment shows that the zero-shot performance of `PatchTST` declines as the target datasets deviate from the source dataset. We expect reasonable transfer learning performance when the source domain contains some highly similar series to the target task.

6.7 Discussion

Pre-training models. The default approach in Natural Language Processing (NLP) uses pre-trained models due to the considerable advantages in accuracy and efficiency. Large language models can be used without needing to train them from scratch, saving considerable computation and time; such models trained on large and diverse datasets can learn complex representations useful for plenty of smaller tasks. This work may serve as a starting point for the broader adoption of pre-trained models for the univariate forecasting task; we believe that pre-trained models and transfer learning will become the default solution for univariate forecasting projects.

Limitations and opportunities. While pre-trained models show impressive results, they still have limitations. A potential challenge can emerge when the target domain differs substantially from the source domains, distribution shifts, or the need to include particular exogenous variables due to a lack of information in autoregressive futures. Forecasting domain adaptation and heterogeneous transfer learning are exciting lines of research, as they will allow pre-trained models to be of service beyond simple univariate forecasting tasks. Some neural forecasting architectures have begun incorporating a shared latent feature space, enabling their application across diverse input scenarios. However, further research is required to unlock their potential and broaden their applicability.

Beyond Transformers. As is the case of related fields dealing with sequential data such as Natural Language Processing (NLP), almost all recent foundation models (FM) for time series rely on some forms of attention mechanisms (Garza and Mergenthaler-Canseco, 2023; Ansari et al., 2024; Goswami et al., 2024; Liang et al., 2024). Our empirical results also show that Transformers, particularly `PatchTST`, tend to achieve the highest on average performance.

However, even purely feed-forward methods such as NHITS and even a simple MLPcan achieve similar performance with lower computational times and can, therefore, serve as viable options.

A promising alternative to Transformers is State Space Models (SSM), with approaches such as S4 or Mamba (Gu et al., 2021; Gu and Dao, 2023). Similarly to RNNs, the SSM concept relies on modeling the system with *state* variables that sequentially evolve based on a *state equation*, while an *output equation* determines the prediction based on the states and inputs. The latest models, such as Mamba, solve the high inference computation cost of transformers, which scale quadratically with the sequence length, while achieving better or similar performance than Transformer-based FMs on several language modeling tasks.

Several extensions to Mamba have been proposed, particularly for image and video tasks. A recent study showed a simple extension with a linear tokenization layer, Simple-Mamba (Wang et al., 2024), which achieves state-of-the-art performance in time series forecasting. In our experiments on pre-pretrained models, RNNs tend to perform poorly on zero-shot scenarios. We hypothesize their poor performance is driven by their limited expressivity compared to Transformers. The novel selection mechanism of Mamba might alleviate this issue and, therefore, yield an interesting alternative for future time series foundation models.

Part IV

Conclusion

Chapter 7

Conclusion

This dissertation studies improving deep learning (DL) time series models for anomaly detection and forecasting. Recent advancements in the field have shown promising results and adoption in academic and industry applications. Despite this progress, I identified several limiting factors to adoption, including scalability, interpretability, and robustness. The models proposed in this thesis push the SoTA performance and tackle these challenges.

The work on this thesis began raising the following question:

> *Can deep learning algorithms leverage efficient and informed representations of multivariate time series to yield accurate, robust, interpretable, and scalable models to increase their adoption?*

The proposed approach builds on the core principle of DL models: their capability to learn useful data representations and leverage them to complete a task. Multivariate time series are complex objects often exhibiting temporal dependencies and feature correlations. Therefore, we can design models that leverage them to learn better representations and improve their performance and capabilities. In this work, I presented seven research projects that support this claim.

In Part I, I proposed two multivariate time series methods, DGHL and TIN, based on dynamic latent space inference with tailored latent temporal representations. These models achieve SoTA performance in forecasting, imputation, and anomaly detection tasks while remaining robust to challenges such as missing values. This part also introduced SAAT, an automatic anomaly threshold selection algorithm based on augmenting the training data with synthetic anomalies that consider the reference data's distribution and maximize the target evaluation metric.

Part II introduces NHITS, a scalable and decomposition-based interpretable model for long-horizon forecasting. The method uses the proposed *hierarchical interpolation* and *multi-rate sampling* techniques to produce a non-linear frequency decomposition of the forecast. The second chapter presents two extensions of the NHITS and the related NBEATS model to incorporate exogenous covariates efficiently. The NBEATSx includes a convolutional decoder that learns a representation of the exogenous covariates to isolate their effect on the target

time series. Second, we proposed a novel pharmacokinetics prior encoder for treatment effect estimation using domain knowledge of drug absorption.

Finally, in Part III, I presented a large-scale study on enabling conditions, on both model design and data characteristics, for transferability of pre-trained models on time series tasks to foster the use of this technique. Transfer learning and representation learning are intrinsically related, as models exploit commonalities between tasks by learning shared representations that capture common underlying factors. The results show that pre-trained deep learning methods can outperform classic *statistical* models trained on the target task. The conclusions of this study can guide practitioners on how to design transferable models and improve data quality.

7.1 Open Source

While academic studies are important for formalizing and disseminating ideas, it is crucial to make their results and tools accessible for practitioners' use. In that regard, the code for all models and evaluations presented in this thesis is publicly available in open-source libraries and dedicated repositories. The NHITS and NBEATSx models are included in Neuralforecast (Olivares et al., 2022a) and other popular open-source libraries. These open-source implementations of the models presented in this thesis have been impactful, aiding various academic and industry practitioners in their time series tasks. Through open-source code, researchers can provide practitioners with better tools to solve their challenges and, ultimately, assist them in making more informed decisions. In turn, users can offer valuable feedback, which is invaluable for identifying and prioritizing the most impactful research directions.

7.2 Expected Scope and Application Limitations

In this section, I present guidance on utilizing the proposed methods from this thesis on real-world applications and conditions when they might fail to provide accurate predictions.

DGHL. This model excels at reconstructing long multivariate time series with temporal patterns spanning thousands of timestamps. It is suitable for applications where detecting anomalies requires models to understand longer patterns, often required when dealing with high-sampling-rate data. Our experiments show that DGHL has superior robustness to missing data, predominant in domains such as healthcare and IoT. Finally, DGHL is best suited for cases with less than one hundred features. Given the additive nature of the MSE anomaly scores, anomaly detection with a greater number of features is challenging, particularly in cases where anomalies only affect a small subset of features.

The experiments also suggest that the hyperparameters chosen for the inference step of Alternating Back Propagation, such as the number of iterations, significantly impact the trade-off between performance and computational time. Proper hyperparameter optimization, using a validation set with anomalies, can help optimize the model's application performance and help practitioners determine the best configuration for their use case.

As with most other general deep learning anomaly detection models, DGHL can detect most common anomalies, such as changes in frequency, scale, trends, and patterns, as seen in (Goswami et al., 2022). This study also shows that detecting subtle *contextual anomalies*, present in benchmark datasets such as UCR Anomaly Archive (Wu and Keogh, 2021), remains challenging.

TIN. The conditions on the data type best suited for this method are similar to DGHL given the shared principle of dynamic latent inference and the use of a CNN generator.

While TIN networks exhibit superior robustness to missing data and distribution shifts, they still operate with the principle that the information contained in the reference window, with the model's inputs, follows the same temporal relation and distribution as is the data in the forecast window. This is clear in Figure B.2, where the model can forecast under the new distribution once it observes it in the reference window.

TIN models need to perform the inference step of the latent factors when forecasting, significantly increasing computational costs. While these costs are lower than other SoTA imputation alternatives, they might still be prohibitive in some applications, particularly in cases with large amounts of data or applications that require low latency forecasts.

NHITS. This model provides accurate forecasts with low computational complexity, especially in long-horizon settings. Regarding data, NHITS has shown remarkable robustness to data with a low signal-to-noise ratio when trained with robust losses such as the Huber loss. However, its performance is greatly affected by missing values and distribution shifts. In practice, this last limitation can be alleviated by re-training the model frequently, allowed by the lower computational complexity.

The NHITS model can incorporate exogenous covariates by concatenating their history to the target variable's lags and using them as inputs of each MLP block. In practice, we have observed a decrease in performance when adding several exogenous covariates compared to other methods, such as TFT (Lim et al., 2021a). This might limit the applications of NHITS in domains where having many exogenous covariates is required, such as forecasting financial indicators.

The frequency decomposition capabilities can be used for time series analysis tasks different than forecasting. The embeddings of the MLP blocks or the reconstruction of each stack can be used in downstream applications.

The results in Part III show this method can be pre-trained and used in new tasks with zero-shot forecasts, further reducing computational requirements. This result also supports using the model in large-scale settings because it can efficiently leverage information from many different time series.

NBEATSx and **PK Encoder**. Given the similar architecture to the NHITS, these methods have the same limitations and advantages regarding the characteristics of the data.

Regarding exogenous covariates, the NBEATSx has a general convolutional encoder that does not leverage or use prior user's knowledge on the time series. It is best suited for domains such as demand forecasting and related tasks, such as electricity price forecasting (EPF).

On the other hand, using the Pharmacokinetic Prior encoder is recommended for domains

where we have a deeper concrete understanding of the phenomena governing time series behavior, which is often the case in, e.g., healthcare, biology, and physics. While the proposed method was tested in particular for blood glucose forecasting, it provides a general framework that can be used in other tasks. The current method assumes linear stacking of effects (linear pharmacokinetics), which might not apply as well to other than insulin drug types.

7.3 Future Work

The conclusion section of each chapter presents specific future work for each topic. This section briefly presents two research directions derived from the thesis.

A recurrent promising line of research is designing methods that incorporate domain knowledge. In anomaly detection, it will be crucial to enable practitioners to encode their knowledge of potential unseen anomalies during training efficiently. In forecasting, leveraging knowledge of the underlying dynamics, such as physics, economics, or even biology, can further increase their applicability to high-stakes settings by improving performance and avoiding producing unrealistic predictions.

The promising results of transfer learning on time series can lay the path to building foundation models. Some early recent works on this topic include models such as `TimeGPT` and `Lag-Llama` (Garza and Mergenthaler-Canseco, 2023; Rasul et al., 2023). Foundational models can completely change how practitioners build time series pipelines. Some challenges to further develop foundational models include how to model beyond pure autoregressive features and add qualitative context to the task or leverage particular exogenous covariates.

Appendix A

Multivariate Online Anomaly Detection

A.1 Time series generation

DGHL is trained to generate time series windows from a latent space representation. We examine how DGHL learns the representation by interpolating and extrapolating between two latent vectors, $\boldsymbol{Z}_l$ and $\boldsymbol{Z}_u$, inferred from two windows of a real time series from the SMD. The trained *Generator network* is then used to generate new windows across the interpolation subspace.

Figure A.1 presents the generated windows for interpolated and extrapolated vectors for a subset of the original features. The generated time series smoothly transitions between clear patterns on both shape and scale. Moreover, DGHL is able to generate meaningful time series on the extrapolation region. This experiment shows how our approach maps similar time series windows into close points of the latent space, which is a desirable property of latent representations.

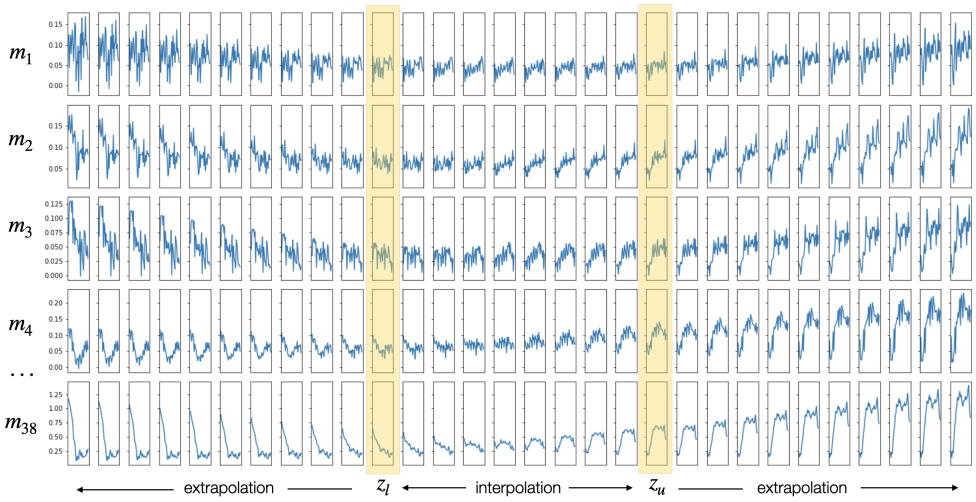


Figure A.1: Generated time series windows from interpolation and extrapolation of latent vectors using DGHL trained on machine-1-1 of the SMD.

Appendix B

Robust Multivariate Forecasting and Imputation

B.1 Generator Network

TIN-CNN uses a Top-Down Convolution Network (CNN) as a generator, which produces the final forecast and reconstruction of the reference window $\hat{\mathbf{Y}}_{t-L:t+H}$ from the temporal embedding $\mathbf{E}$. The full CNN architecture is presented in Figure 3.2, and can be formalized as:

$$\begin{aligned} \mathbf{h_1} &= \text{ReLU}(\text{BN}(\text{Transposed Convolution}(\mathbf{E}))) \\ \mathbf{h_2} &= \text{ReLU}(\text{BN}(\text{Transposed Convolution}(\mathbf{h_1}))) \\ \hat{\mathbf{Y}}_{t-L:t+H} &= \text{Transposed Convolution}(\mathbf{h_2}) \end{aligned} \tag{B.1}$$

where Transposed Convolution is a transpose convolutional layer on the temporal dimension, BN is a batch normalization layer, and ReLU activations introduce non-linearity. The number of filters at each layer, kernel size, and stride are given in Appendix H. The convolutional filters are not *causal* as in a Temporal Convolution Network (TCN) (Bai et al., 2018a). Instead, TIN-CNN's *causality*, which allows forecasting using only past observations, comes from inferring the latent vector $\mathbf{z}^*$ with the reference window.

Figure 3.2 of the paper presents the default recommended configuration for TIN-CNN. The first layers of the CNN learn a common representation for all features from the temporal embedding $\mathbf{E}$. The second layer *refines* the temporal resolution to the final size of the window s_w. The last layer produces the final output for all M features from $\mathbf{h_2}$.

While equation B.1 presents the default configuration used in the experiments, additional layers can be added to increase the expressivity. Table B.1 presents ablation studies on the number of layers. We explore different configurations for the CNN, particularly for the number of hidden layers. For the model TIN-CNN$_{i,j}$, i refers to the number of layers with temporal resolution $s_w/2$, and j to the number of layers with s_w resolution. The following table presents

the results on **ILI** and **Solar** datasets. The results suggest TIN-CNN's performance is not significantly affected by the number of layers in these datasets for several occlusion probabilities. On average, TIN-$CNN_{2,2}$ achieves the best performance.

Table B.1: Forecasting accuracy results of TIN with different CNN architectures on benchmark datasets with different proportions of missing values (p_o), forecasting horizon of 24 timestamps, lower scores are better. Metrics are averaged over five runs. The best model is highlighted in bold.

		TIN-CNN		TIN-$CNN_{2,2}$		TIN-$CNN_{3,2}$		TIN-$CNN_{3,3}$	
	p_o	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ILI	0.0	0.724	0.557	0.719	0.549	0.773	0.560	0.726	0.550
	0.2	1.153	0.662	1.148	0.670	1.164	0.682	1.156	0.658
	0.6	2.453	1.012	2.391	1.002	2.382	0.985	2.369	0.991
Solar	0.0	0.007	0.054	0.007	0.052	0.008	0.058	0.007	0.053
	0.2	0.012	0.058	0.010	0.059	0.012	0.059	0.011	0.054
	0.6	0.013	0.068	0.012	0.064	0.014	0.066	0.015	0.064

B.2 Benchmark datasets

Table B.2 presents summary information for the benchmark datasets, including the frequency, number of features and timestamps, and train/validation/test proportions.

Table B.2: Summary of benchmark datasets

Dataset	Granularity	# of features	# of timestamps	Train/Val/test split
Simulated7	NA	7	20,000	80/10/10
ILI	Weekly	7	966	70/10/20
ETTm2	15 Minute	7	57,600	60/20/20
Exchange	Daily	8	7,588	70/20/10
Solar	Hourly	32	8,760	60/20/20
Weather	10 Minute	21	52,695	70/10/20

All datasets are public and are available in the following links:

- **ILI**: https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
- **Exchange**: https://github.com/laiguokun/multivariate-time-series-data
- **Solar**: https://www.nrel.gov/grid/solar-power-data.html
- **ETTm2**: https://github.com/zhouhaoyi/ETDataset
- **Weather**: https://www.bgc-jena.mpg.de/wetter/

B.3 Simulated dataset

Simulated7is a synthetic dataset that we designed to evaluate forecasting models' robustness to missing data and distribution shifts in a controlled environment. It consists of seven time series, each generated independently as the sum of two cosine functions with different frequencies and a small Gaussian noise. In particular, it is composed as the sum of the three following elements:

$$\begin{aligned} y_t^{\text{low}} &= \cos(it) && , i \sim U(5, 50) \quad t \in [0, 5] \\ y_t^{\text{high}} &= \cos(it) && , i \sim U(100, 300) \quad t \in [0, 5] \\ y_t^{\text{noise}} &\sim N(0, 0.001) && , t \in [0, 5] \end{aligned} \tag{B.2}$$

Each time series consists of 20,000 timestamps, with regular intervals between $[0, 5]$. For the *missing data* we obfuscate random timestamps following the procedure described in section 3.5, using $s = 10$ and $p_o \in \{0, 0.2, 0.4, 0.6, 0.8\}$.

B.4 Hyperparameter optimization

TIN-CNN hyperparameters are tuned on the validation set of each dataset using the HYPEROPT algorithm with 30 iterations; table B.3 presents the hyperparameter grid. To ensure a fair comparison with baseline models, we also tuned their respective hyperparameters with the same procedure, as the default configuration in their implementations might not perform well in the different settings we explore. Next, we detail the hyperparameter grid for each baseline model.

Table B.3: Hyperparameters grid for TIN-CNN.

Hyperparameter	Values
Learning rate	[0.0001, 0.1]
Training steps	{500, 1000}
Batch size	{8, 16, 32}
Random seed	{1,2,...,10}
GD iterations during training	{25, 50}
GD iterations during inference	{300, 500}
GD step size	{0.2, 1, 2}
Ensemble size	{3}
Max degree trend polynomial	{3}
Window size (s_w)	{128, 256, 512}
Kernel size	{4, 8}
Stride	{4}
Dilation	{1}
Convolution filters of two hidden layers (n_1,n_2)	{(256,128), (128, 64)}

For all models, we tune optimization hyperparameters, including learning rate, number of iterations, batch size, and random seed for initialization. The number of iterations for Transformers was larger than that of other models, as recommended in the respective papers. For TIN-CNN, we tune the architecture of the CNN and optimize hyperparameters for the inference step. For NHITS, we use the hyperparameter grid described in their paper (Challu et al., 2023b). The NHITSis a generalization of the NBEATS, which is already included in the hyperparameter grid as a possible configuration. For the Informer, Autoformer, FEDformer, and PatchTST, we explore different values for the dropout probability, number of heads, size of embedding, and sequence length. Finally, for RNN, we explore different dilations, the number of layers, and hidden size. For the NHITS/NBEATS, all Transformers, and RNNmodels, we used the Neuralforecast library (available in PyPI and Conda).

B.5 TIN training algorithm

Algorithm 1 presents the TIN's training procedure. The model is trained for a fixed number of iterations n_{iters}, randomly sampling b windows in each iteration. Parameters $\boldsymbol{\theta}$ are optimized with Adam optimizer.

Algorithm B.1 Training procedure

Input: $\mathbf{Y} \in \mathbb{R}^{m \times T}$, model $F_{\boldsymbol{\theta}}$, learning iterations n_{iters}
Output: $F_{\boldsymbol{\theta}^*}$, inferred latent vectors $\{\mathbf{z}_t^*, t = 0, ..., T\}$
Let $i \leftarrow 0$, initialize $\boldsymbol{\theta}$
Initialize $\mathbf{z}_t$, for $t = 0, ..., T$
while $i < n_{iters}$ **do**
  Sample a random mini-batch of windows: $\{\mathbf{Y}_{j_k - L:j_k+H}, j_k \sim U(L, T-H), k\{1, ..., b\}\}$
  $\mathbf{z}_{j_k}^* \leftarrow \arg\min_{\mathbf{z}} L(\mathbf{Y}_{j_k-L:j_k}, \hat{\mathbf{Y}}_{j_k-L:j_k}(\mathbf{z})), k \in \{1, ..., b\}$
  Update $\boldsymbol{\theta}$ with Adam using $\mathbf{z}^*$ as input.
  Store $\mathbf{z}_{j_k}^*, k \in \{1, ..., b\}$
  $i \leftarrow i + 1$
end while

B.6 Robustness to Distribution Shifts

In this Appendix, we present preliminary evidence on the TIN's robustness to distribution shifts. Distribution shifts are changes in the data-generating process over time. In particular, changes in the behavior of the time series between the training data and the test data (where models are deployed) can considerably degrade the model's performance, as the temporal relation between the input data and the forecast window might differ in both sets. In recent years, we observed an increase in the occurrence of distribution shifts with the COVID-19 pandemic. Similar to the case of missing data, designing robust methods that can intrinsically handle distribution shifts

can provide great benefits by improving their applications in domains where this problem is predominant (Kuznetsov and Mohri, 2014; Du et al., 2021; Ivanovic et al., 2022).

We define distribution shifts between train and test data as follows. Consider a general forecasting task where model F_θ is used to forecast variable $\mathbf{Y}$ and trained on historical data, where $\mathbf{Y} \sim \mathcal{D}_{train}$. The model is then evaluated (deployed) on future observations (test set), with $\mathbf{Y} \sim \mathcal{D}_{test}$. A distribution shift occurs when $\mathcal{D}_{train} \neq \mathcal{D}_{test}$. We present a real example of a distribution shift in Figure B.1 showing the **ILI** dataset. The test set (right of the vertical red line) exhibits larger and more prolonged spikes than the train set.

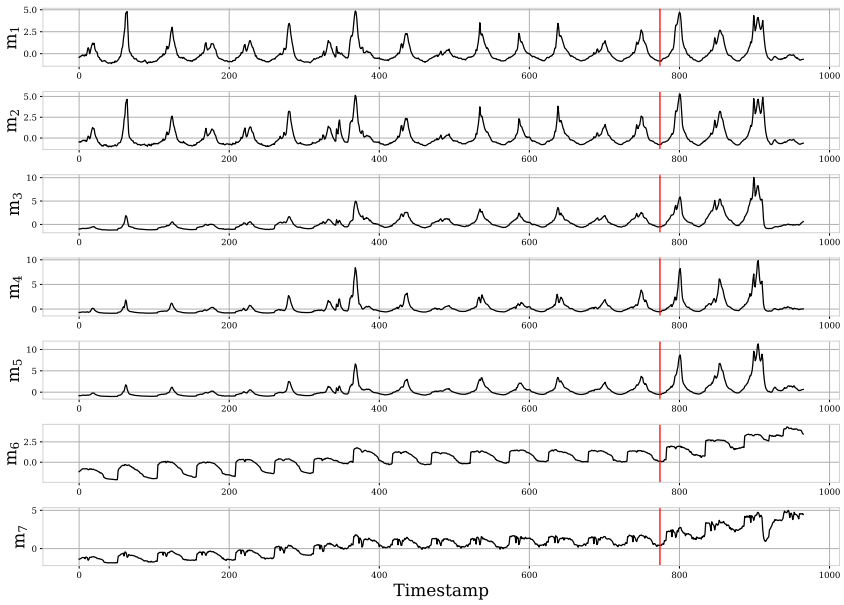
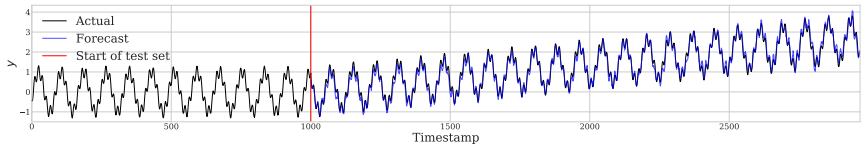


Figure B.1: ILI dataset, the start of the test set marked with a vertical red line.

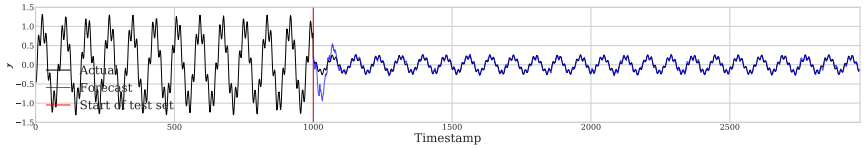
A TIN model has the ability to dynamically adjust the forecasts based on the recent temporal behavior on the reference window, giving the model more robustness to changes in distribution. In particular, the unconstrained latent factors can extrapolate beyond training regions to produce forecasts with unseen patterns and scales, such as **ILI**'s larger spikes.

We first test this hypothesis on synthetic data by perturbing the test set on **Simulated7** dataset with two transformations: adding a linear *trend* with slope 6 and scaling the *magnitude* by 0.5. Figure B.2 presents TIN–CNNś forecasts for this setting, and complete forecasts for all features and baselines are included in Appendix H. The qualitative results are complemented with Table B.4.

These results on **Simulated7** show that our approach has superior robustness to some forms of distribution shifts between the train and test data compared to current SoTA models. The results on **ILI**dataset in Table 3.1 also support this claim, where TIN–CNN reduces forecasting errors by **50%** against the second best alternative. Exploring TIN's domain adaptation capabilities and related transfer learning applications can be a promising line of research for future work.



(a) Trend



(b) Magnitude

Figure B.2: Forecasts for the first feature of **Simulated7** dataset using TIN with (a) change in trend, and (b) change in magnitude. Forecasts are produced every 24 timestamps in a rolling window strategy. We include the last 1,000 of the train set to show the change in distribution.

Table B.4: Forecasting accuracy on **Simulated7**7 dataset with distribution shifts between the train and test sets, forecasting horizon of 24 timestamps. Lower scores are better. Metrics are averaged over five runs, with the best model highlighted in bold.

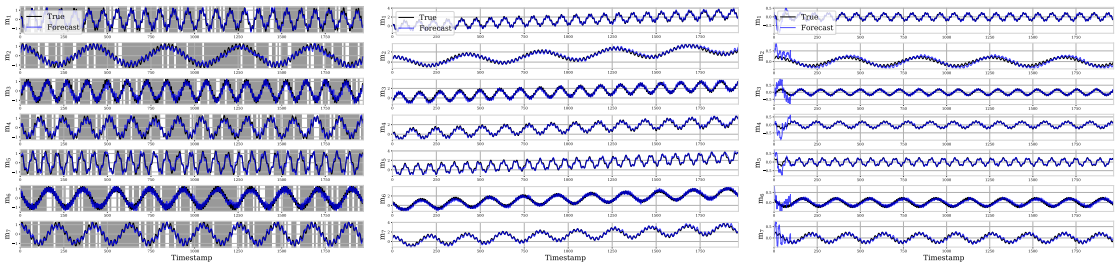
		TIN-CNN		NHITS		Informer		Autoformer		StemGNN		RNN	
	p_o	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Normal	0.0	0.004	0.041	**0.001**	**0.015**	0.008	0.079	0.017	0.096	0.036	0.122	0.001	0.022
Trend	0.0	**0.025**	**0.130**	0.427	0.443	2.821	1.410	0.060	0.19	1.354	0.952	1.214	0.859
Magnitude	0.0	**0.005**	**0.038**	0.009	0.059	0.020	0.094	0.043	0.168	0.060	0.192	0.089	0.248

B.7 Forecasts on Simulated7

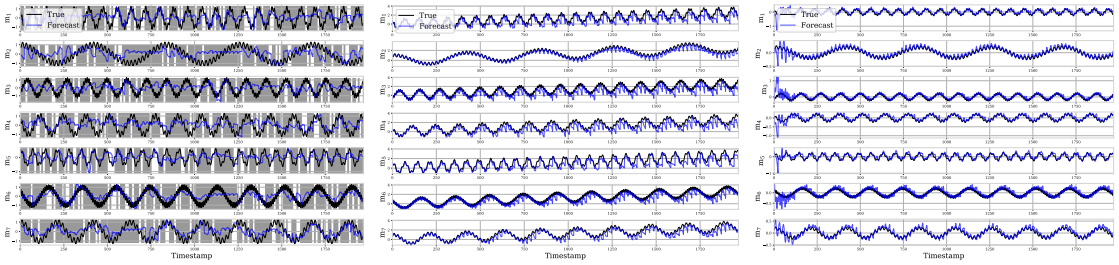
Figure B.3 presents the complete forecasts on the test set for TIN-CNN, NHITS, Informer, and RNNmodels. TIN-CNN is the only model that can accurately forecast with up to 80% missing data and changes in distribution.

B.8 Additional occlusion experiments

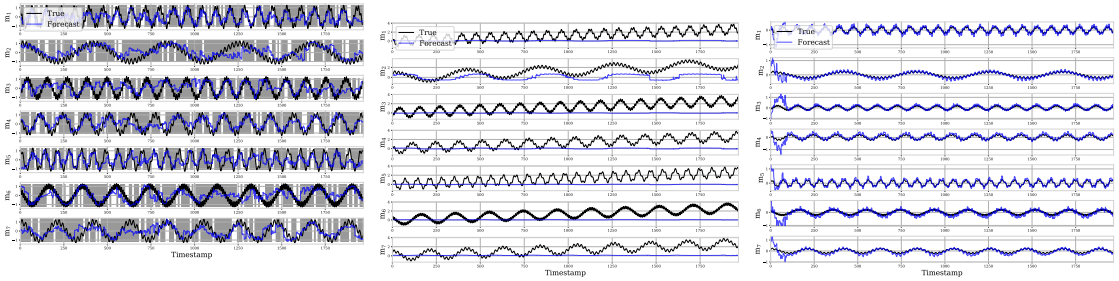
We present additional occlusion experiments to analyze the case where the *missing data regime* (how much or for how long missing data occurs) differs between the training and test sets. **In this setting, we train models on complete data and inject missing values only during inference on the test set**. The following table presents TIN and NHITS(best baseline) results on the **Simulated7** dataset. TIN-CNN performance is almost identical to the original results, demonstrating the method's robustness to changes in the behavior of the presence of missing data.



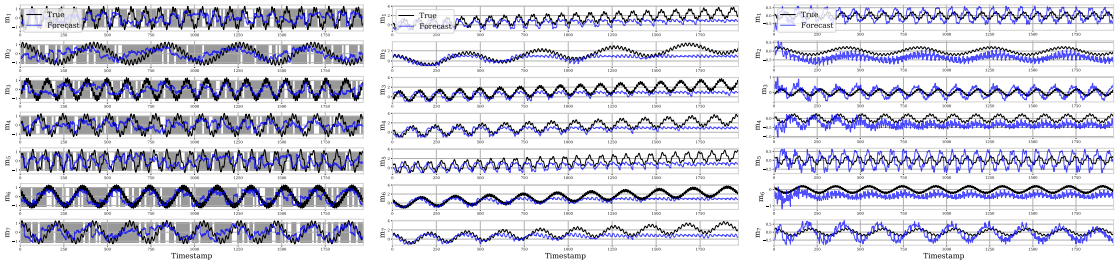
(a) Missing values - TIN-CNN **(b)** Trend - TIN-CNN **(c)** Magnitude - TIN-CNN



(d) Missing values - NHITS **(e)** Trend - NHITS **(f)** Magnitude - NHITS



(g) Missing values - Informer **(h)** Trend - Informer **(i)** Magnitude - Informer



(j) Missing values - RNN **(k)** Trend - RNN **(l)** Magnitude - RNN

Figure B.3: Forecasts on the test set for all features of **Simulated7** dataset with 80% of missing data (missing regions in grey), change in trend, and change in magnitude. Forecasts are produced every 24 timestamps in a rolling window strategy.

B.9 Complete experimental results

Table B.6 presents the extended main forecasting experimental results, reporting accuracy as measured with MSE for all baselines and datasets. MSE is averaged over five runs, varying the random seed for the HYPEROPT algorithm (different configurations are sampled), and the

Table B.5: Forecasting accuracy on **Simulated7** dataset with missing values only on the test set, forecasting horizon of 24 timestamps. Lower scores are better. Metrics are averaged over five runs, with the best model highlighted in bold.

	TIN-CNN		NHITS	
p_o	MSE	MAE	MSE	MAE
0.2	**0.008**	**0.079**	0.013	0.146
0.4	**0.017**	**0.085**	0.031	0.224
0.6	**0.021**	**0.097**	0.257	0.384
0.8	**0.029**	**0.124**	0.463	0.562

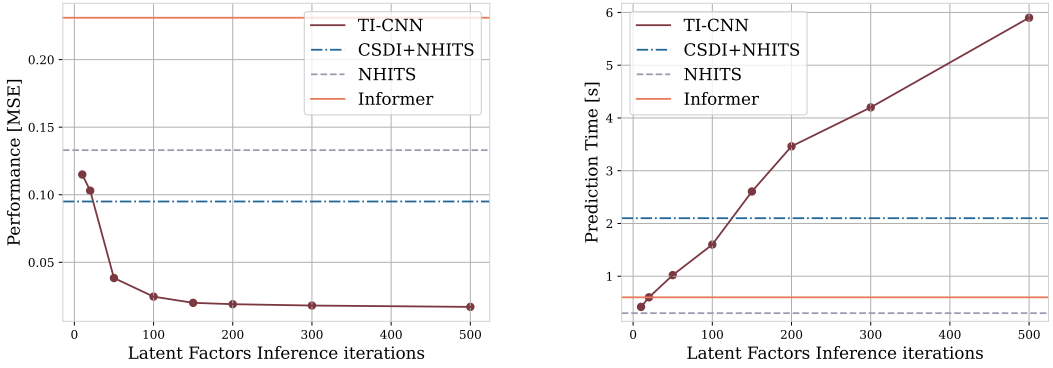
standard deviation is reported in parenthesis. CSDI+N and CSDI+P refer to CSDI+NHITS and CSDI+PatchTST, respectively. MAE is omitted for simplicity; results are consistent with the MSE metric.

B.10 Computational complexity during prediction

Section 5.4 showed that TIN-CNN has training times similar to current SoTA baselines. In this Appendix, we present an analysis of the performance/time trade-off of the number of iterations to infer latent factors during prediction (forecasting). While training usually dominates the overall computational cost and time (for example, training times for TIN-CNN on **Simulated7**was around 150 seconds, compared to only 2 seconds to forecast the complete test set), some applications require performing fast predictions, especially on high-frequency data.

The main TIN's hyperparameter to control prediction times is the number of iterations of the gradient descent algorithm to infer latent factors. Figure B.4 presents how TIN-CNN's performance and prediction times vary with the number of iterations, compared to CSDI+NHITS(best baseline) and other selected baselines, on **Simulated7** with $p_0 = 0.6$. First, TIN-CNN can match CSDI+NHITS performance with only 20 iterations, with a decrease in prediction times of 75%. More importantly, with 100 iterations, our approach can reduce the forecasting error by more than 50% compared to CSDI+NHITSwhile maintaining shorter prediction times. While standalone baselines such as NHITS (fastest baseline) and Informer have very short prediction times, their hindered performance limits their applicability to settings with missing values.

These results also demonstrate that the number of iterations for inference can be controlled to tailor the model behavior to the particular needs and constraints of the application. For example, if resources are constrained during the prediction stage, the number of iterations can be decreased to reduce computation and forecasting times while preserving SoTA performance.



(a) Performance **(b)** Time

Figure B.4: Analysis of TIN-CNN performance and prediction times against the number of iterations to infer latent factors on **Simulated7** dataset with $p_0 = 0.6$. **TIN-CNN achieves the best performance with shorter forecasting times**.

Table B.6: Main forecasting accuracy results measured with **MSE** on benchmark datasets with different proportions of missing values (p_o), forecasting horizon of 24 timestamps, lower scores are better. Metrics are averaged over five runs, and the **standard deviation is in brackets**, the best model highlighted in bold.

	p_0	TIN-CNN	CSDI+N	CSDI+P	NHITS	PatchTST	Informer	Autoformer	FEDformer	StemGNN	RNN
Simulated77	0.0	0.004	-	-	**0.001**	0.004	0.008	0.017	0.013	0.036	0.001
		(0.0003)	-	-	(0.0002)	(0.0009)	(0.0012)	(0.0023)	(0.0015)	(0.0018)	(0.0001)
	0.2	**0.009**	0.010	0.082	0.012	0.162	0.071	0.319	0.141	0.097	0.069
		(0.0003)	(0.0004)	(0.0061)	(0.0016)	(0.0042)	(0.0085)	(0.0455)	(0.0092)	(0.0062)	(0.0008)
	0.4	**0.011**	0.018	0.123	0.022	0.368	0.095	0.591	0.276	0.181	0.172
		(0.0005)	(0.0009)	(0.0156)	(0.0028)	(0.0310)	(0.0079)	(0.0682)	(0.0106)	(0.0095)	(0.0054)
	0.6	**0.018**	0.095	0.196	0.133	0.479	0.231	0.752	0.409	0.362	0.357
		(0.0008)	(0.0012)	(0.0307)	(0.0082)	(0.0598)	(0.0163)	(0.0711)	(0.0351)	(0.0381)	(0.0439)
	0.8	**0.023**	0.216	0.395	0.365	0.515	0.372	0.925	0.586	0.436	0.500
		(0.0012)	(0.0252)	(0.0351)	(0.0260)	(0.0611)	(0.0254)	(0.0740)	(0.0493)	(0.0474)	(0.0592)
ILI	0.0	**0.724**	-	-	1.379	1.228	4.265	2.249	2.103	4.013	3.852
		(0.080)	-	-	(0.073)	(0.062)	(0.327)	(0.147)	(0.106)	(0.293)	(0.249)
	0.2	**1.153**	1.946	1.837	2.028	2.737	3.624	3.250	3.241	4.372	3.647
		(0.102)	(0.092)	(0.095)	(0.119)	(0.107)	(0.339)	(0.155)	(0.138)	(0.315)	(0.258)
	0.4	**1.646**	3.110	2.837	3.248	4.180	3.908	4.308	4.197	4.752	4.360
		(0.116)	(0.137)	(0.103)	(0.130)	(0.155)	(0.360)	(0.293)	(0.190)	(0.396)	(0.293)
	0.6	**2.453**	3.552	3.409	3.871	4.355	3.991	4.571	4.853	5.170	5.561
		(0.175)	(0.184)	(0.167)	(0.234)	(0.200)	(0.352)	(0.310)	(0.247)	(0.352)	(0.316)
	0.8	**3.136**	3.878	4.126	5.102	5.676	4.180	4.745	4.866	5.337	5.063
		(0.193)	(0.206)	(0.217)	(0.279)	(0.294)	(0.384)	(0.390)	(0.251)	(0.370)	(0.375)
Exchange	0.0	0.048	-	-	0.031	**0.024**	0.472	0.049	0.048	0.102	0.086
		(0.003)	-	-	(0.002)	(0.002)	(0.070)	(0.004)	(0.004)	(0.008)	(0.005)
	0.2	0.091	**0.072**	0.093	0.295	0.362	1.013	0.758	0.643	1.021	0.871
		(0.008)	(0.006)	(0.006)	(0.024)	(0.028)	(0.112)	(0.092)	(0.083)	(0.095)	(0.124)
	0.4	**0.158**	0.163	0.160	0.321	1.057	1.162	0.782	0.968	0.826	0.694
		(0.015)	(0.013)	(0.019)	(0.029)	(0.073)	(0.135)	(0.102)	(0.133)	(0.102)	(0.084)
	0.6	**0.367**	0.492	0.503	0.549	1.451	1.564	1.346	1.450	1.991	1.432
		(0.025)	(0.029)	(0.028)	(0.037)	(0.084)	(0.149)	(0.145)	(0.139)	(0.127)	(0.105)
	0.8	**1.125**	1.308	1.217	1.792	2.890	2.530	2.520	2.672	2.778	2.791
		(0.043)	(0.064)	(0.053)	(0.081)	(0.165)	(0.188)	(0.176)	(0.182)	(0.150)	(0.163)
Solar	0.0	**0.007**	-	-	0.008	0.008	0.011	0.018	0.016	0.015	0.012
		(0.0001)	-	-	(0.0001)	(0.0001)	(0.0009)	(0.0010)	(0.0008)	(0.0007)	(0.0004)
	0.2	**0.012**	0.015	0.017	0.016	0.018	0.016	0.025	0.020	0.016	0.015
		(0.0003)	(0.0002)	(0.0003)	(0.0002)	(0.0003)	(0.0008)	(0.0012)	(0.0010)	(0.0009)	(0.0004)
	0.4	**0.012**	0.017	0.021	0.017	0.023	0.020	0.023	0.022	0.023	0.017
		(0.0004)	(0.0005)	(0.0004)	(0.0004)	(0.0005)	(0.0011)	(0.0014)	(0.0014)	(0.0015)	(0.0008)
	0.6	**0.013**	0.020	0.022	0.022	0.025	0.021	0.027	0.024	0.031	0.026
		(0.0007)	(0.0010)	(0.0009)	(0.0012)	(0.0009)	(0.0019)	(0.0013)	(0.0012)	(0.0019)	(0.0013)
	0.8	**0.014**	0.025	0.025	0.025	0.035	0.035	0.036	0.033	0.065	0.028
		(0.0009)	(0.0011)	(0.0015)	(0.0015)	(0.0013)	(0.0021)	(0.0026)	(0.0025)	(0.0031)	(0.0013)
ETTm2	0.0	0.136	-	-	0.116	**0.107**	0.366	0.171	0.156	0.154	0.179
		(0.005)	-	-	(0.003)	(0.003)	(0.054)	(0.020)	(0.021)	(0.013)	(0.041)
	0.2	0.155	0.145	**0.140**	0.656	0.542	0.895	0.512	0.472	0.963	0.574
		(0.009)	(0.008)	(0.008)	(0.042)	(0.044)	(0.073)	(0.031)	(0.027)	(0.055)	(0.053)
	0.4	0.228	0.211	**0.200**	1.191	1.154	0.923	1.078	0.998	1.542	0.807
		(0.012)	(0.009)	(0.010)	(0.059)	(0.053)	(0.072)	(0.049)	(0.045)	(0.058)	(0.067)
	0.6	**0.500**	0.563	0.535	1.976	1.763	1.784	1.643	1.574	2.151	1.163
		(0.024)	(0.013)	(0.015)	(0.066)	(0.071)	(0.080)	(0.062)	(0.064)	(0.074)	(0.061)
	0.8	**0.725**	0.986	0.801	2.019	2.315	1.982	1.992	2.052	2.595	2.085
		(0.035)	(0.069)	(0.054)	(0.065)	(0.076)	(0.083)	(0.081)	(0.083)	(0.096)	(0.086)
Weather	0.0	0.112	-	-	**0.109**	0.131	0.218	0.186	0.175	0.116	0.124
		(0.004)	-	-	(0.002)	(0.003)	(0.019)	(0.012)	(0.009)	(0.010)	(0.012)
	0.2	0.154	**0.148**	0.163	0.181	0.190	0.287	0.324	0.240	0.179	0.189
		(0.008)	(0.005)	(0.005)	(0.009)	(0.010)	(0.021)	(0.019)	(0.017)	(0.016)	(0.015)
	0.4	**0.205**	0.216	0.229	0.264	0.292	0.320	3.105	0.307	0.296	0.291
		(0.010)	(0.006)	(0.007)	(0.013)	(0.013)	(0.021)	(0.085)	(0.018)	(0.015)	(0.014)
	0.6	**0.329**	0.417	0.431	0.420	0.486	0.761	3.772	0.769	0.495	0.435
		(0.016)	(0.019)	(0.018)	(0.025)	(0.022)	(0.034)	(0.096)	(0.034)	(0.021)	(0.023)
	0.8	**0.439**	0.506	0.509	0.517	0.531	0.915	4.204	0.862	0.760	0.480
		(0.017)	(0.027)	(0.025)	(0.029)	(0.028)	(0.037)	(0.112)	(0.035)	(0.038)	(0.021)

Appendix C

NHITS: Long Multi-horizon Forecasting

C.1 Neural Basis Approximation Theorem

In this Appendix, we prove the *neural basis expansion approximation theorem* introduced in Section 4.4.3. We show that the hierarchical interpolation can arbitrarily approximate infinitely long horizons ($\tau \in [0,1]$ continuous horizon), as long as the interpolating functions g are defined by a projection to informed multi-resolution functions, and the forecast relationships satisfy smoothness conditions. We prove the case when $g_{w,h}(\tau) = \theta_{w,h}\phi_{w,h}(\tau) = \theta_{w,h}\mathbb{1}\{\tau \in [2^{-w}(h-1), 2^{-w}h]\}$ are piecewise constants and the inputs $\mathbf{y}_{[t-L:t]} \in [0,1]$. The proof for linear, spline functions and $\mathbf{y}_{[t-L:t]} \in [a,b]$ is analogous.

Lemma 1. Let a function representing an infinite forecast horizon be $\mathcal{Y} : [0,1] \to \mathbb{R}$ a square integrable function $\mathcal{L}^2([0,1])$. The forecast function $\mathcal{Y}$ can be arbitrarily well approximated by a linear combination of piecewise constants:

$$V_w = \{\phi_{w,h}(\tau) = \phi(2^w(\tau - h)) \mid w \in \mathbb{Z}, h \in 2^{-w} \times [0, \dots, 2^w]\}$$

where $w \in \mathbb{N}$ controls the frequency/indicator's length and h the time-location (knots) around which the indicator $\phi_{w,h}(\tau) = \mathbb{1}\{\tau \in [2^{-w}(h-1), 2^{-w}h]\}$ is active. That is, $\forall \epsilon > 0$, there is a $w \in \mathbb{N}$ and $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[t-L:t]}) = \mathrm{Proj}_{V_w}(\mathcal{Y}(\tau|\mathbf{y}_{[t-L:t]})) \in \mathrm{Span}(\phi_{w,h})$ such that

$$\int_{[0,1]} |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau = \int_{[0,1]} |\mathcal{Y}(\tau) - \sum_{w,h} \theta_{w,h}\phi_{w,h}(\tau)| d\tau \leq \epsilon \tag{C.1}$$

Proof. This classical proof can be traced back to Haar's work (1910). The indicator functions $V_w = \{\phi_{w,h}(\tau)\}$ are also referred in literature as Haar scaling functions or father wavelets. Details are provided in Boggess and Narcowich, 2015.Let the number of coefficients for the ϵ-approximation $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[t-L:t]})$ be denoted as $N_\epsilon = \sum_{i=0}^{w} 2^i$.

□

Lemma 2. Let a forecast mapping $\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]}) : [0,1]^L \to \mathcal{L}^2([0,1])$ be ϵ-approximated by $\hat{\mathcal{Y}}(\tau|\mathbf{y}_{[t-L:t]}) = \mathrm{Proj}_{V_w}(\mathcal{Y}(\tau|\mathbf{y}_{[t-L:t]}))$, the projection to multi-resolution piecewise constants. If the relationship between $\mathbf{y}_{[t-L:t]} \in [0,1]^L$ and $\theta_{w,h}$ varies smoothly, for instance $\theta_{w,h} : [0,1]^L \to \mathbb{R}$ is a K-Lipschitz function then for all $\epsilon > 0$ there exists a three-layer neural network $\hat{\theta}_{w,h} : [0,1]^L \to \mathbb{R}$ with $O\left(L(\frac{K}{\varepsilon})^L\right)$ neurons and ReLU activations such that

$$\int_{[0,1]^L} |\theta_{w,h}(\mathbf{y}_{[t-L:t]}) - \hat{\theta}_{w,h}(\mathbf{y}_{[t-L:t]})| d\mathbf{y}_{[t-L:t]} \leq \epsilon \tag{C.2}$$

Proof. This lemma is a special case of the neural universal approximation theorem that states the approximation capacity of neural networks of arbitrary width (Hornik, 1991). The theorem has refined versions where the width can be decreased under more restrictive conditions for the approximated function (Barron, 1993; Hanin and Sellke, 2017). □

Theorem 1. Let a forecast mapping be

$\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]}) : [0,1]^L \to \mathcal{F}$, where the forecast functions $\mathcal{F} = \{\mathcal{Y}(\tau) : [0,1] \to \mathbb{R}\} = \mathcal{L}^2([0,1])$ representing a continuous horizon, are square integrable.

If the multi-resolution functions V_w can arbitrarily approximate $\mathcal{L}^2([0,1])$. And the projection $\mathrm{Proj}_{V_w}(\mathcal{Y}(\tau))$ varies smoothly on $\mathbf{y}_{[t-L:t]}$. Then the forecast mapping $\mathcal{Y}(\cdot \mid \mathbf{y}_{[t-L:t]})$ can be arbitrarily approximated by a neural network learning a finite number of multi-resolution coefficients $\hat{\theta}_{w,h}$.

That is $\forall \epsilon > 0$,

$$\begin{aligned} &\int |\mathcal{Y}(\tau \mid \mathbf{y}_{[t-L:t]}) - \tilde{\mathcal{Y}}(\tau \mid \mathbf{y}_{[t-L:t]})| d\tau \\ &\qquad = \int |\mathcal{Y}(\tau \mid \mathbf{y}_{[t-L:t]}) - \sum_{w,h} \hat{\theta}_{w,h}(\mathbf{y}_{[t-L:t]}) \phi_{w,h}(\tau)| d\tau \leq \epsilon \end{aligned} \tag{C.3}$$

Proof. For simplicity of the proof, we will omit the conditional lags $\mathbf{y}_{[t-L:t]}$. Using both the neural approximation $\tilde{\mathcal{Y}}$ from Lemma 2, and Haar's approximation $\hat{\mathcal{Y}}$ from Lemma 1,

$$\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau = \int |(\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)) + (\hat{\mathcal{Y}}(\tau) - \tilde{\mathcal{Y}}(\tau))| d\tau$$

By the triangular inequality:

$$\begin{aligned} \int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq &\int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| \\ &+ |\sum_{w,h} \theta_{w,h} \phi_{w,h}(\tau) - \sum_{w,h} \hat{\theta}_{w,h} \phi_{w,h}(\tau)| d\tau \end{aligned}$$

By a special case of Fubini's theorem

$$\begin{aligned} &\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq \\ &\int |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} \int_{\tau} |(\theta_{w,h} - \hat{\theta}_{w,h}) \phi_{w,h}(\tau)| d\tau \end{aligned}$$

Using positivity and bounds of the indicator functions

$$\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq$$
$$\int_{\tau} |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}| \int_{\tau} \phi_{w,h}(\tau) d\tau$$
$$< \int_{\tau} |\mathcal{Y}(\tau) - \sum_{w,h} \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}|$$

To conclude, we use both arbitrary approximations from the Haar projection and the approximation to the finite multi-resolution coefficients

$$\int |\mathcal{Y}(\tau) - \tilde{\mathcal{Y}}(\tau)| d\tau \leq$$
$$\int |\mathcal{Y}(\tau) - \hat{\mathcal{Y}}(\tau)| d\tau + \sum_{w,h} |\theta_{w,h} - \hat{\theta}_{w,h}| \leq \epsilon_1 + N_{\epsilon_1} \epsilon_2 \ \leq \epsilon$$

□

C.2 Computational Complexity Analysis

We consider a single forecast of length H for the following complexity analysis, with a NBEATS and a NHITS architecture of B blocks. We do not consider the batch dimension. We consider most practical situations, the input size $L = \mathcal{O}(H)$ linked to the horizon length.

The block operation described by Equation (4.2) has complexity dominated by the fully connected layers of $\mathcal{O}(H\, N_h)$, with N_h the number of hidden units that we treat as a constant. The depth of stacked blocks in the NBEATS-G architecture that endows it with its expressivity is associated with a computational complexity that scales linearly $\mathcal{O}(HB)$, with B the number of blocks.

The block operation described by Equation (4.2) has complexity dominated by the fully connected layers of $\mathcal{O}(H\, N_h)$, with N_h the number of hidden units that we treat as a constant. The depth of stacked blocks in the NBEATS-G architecture, which endows it with its expressivity, is associated with a computational complexity that scales linearly $\mathcal{O}(HB)$, with B the number of blocks.

In contrast, the NHITS architecture that specializes in each stack in different frequencies, through the expressivity ratios, can greatly reduce the number of parameters needed for each layer. When we use *exponentially increasing expressivity* ratios through the depth of the architecture blocks, it allows us to model complex dependencies while controlling the number of parameters used on each output layer. If the *expressivity ratio* is defined as $r_{\ell} = r^{l}$ then the space complexity of NHITS scales geometrically $\mathcal{O}(\sum_{l=0}^{B} Hr^{l}) = \mathcal{O}\left((H(1-r^{B})/(1-r)\right)$.

Table C.1: Computational complexity of deep learning forecasting methods as a function of the output size H. For simplicity, we assume that the input size L scales linearly with respect to H. For NHITS and NBEATS we also consider the network's B blocks.

Model	Time	Memory
LSTM	$\mathcal{O}(H)$	$O(H)$
ESRNN	$\mathcal{O}(H)$	$O(H)$
TCN	$\mathcal{O}(H)$	$O(H)$
Transformer	$\mathcal{O}(H^2)$	$O(H^2)$
Reformer	$\mathcal{O}(H\,logH)$	$O(H\,logH)$
Informer	$\mathcal{O}(H\,logH)$	$O(H\,logH)$
Autoformer	$\mathcal{O}(H\,logH)$	$O(H\,logH)$
LogTrans	$\mathcal{O}(H\,logH)$	$O(H^2)$
NBEATS-I	$\mathcal{O}(H^2B)$	$O(H^2B)$
NBEATS-G	$\mathcal{O}(HB)$	$O(HB)$
NHITS	$\mathcal{O}(H(1-r^B)/(1-r))$	$O(H(1-r^B)/(1-r))$

C.3 Datasets and Partition

Figure C.1 presents one time series for each dataset and the train, validation, and test splits from Table 6.1.

C.4 Hyperparameter Exploration

All benchmark neural forecasting methods optimize the length of the input $\{96, 192, 336, 720\}$ for **ETT**, **Weather**, and **ECL**, $\{24, 36, 48, 60\}$ for **ILI**, and $\{24, 48, 96, 192, 288, 480, 672\}$ for **ETTm**. The Transformer-based models: Autoformer, Informer, LogTrans, and Reformer are trained with MSE loss and ADAM of 32 batch size, using a starting learning rate of 1e-4, halved every two epochs, for ten epochs with early stopping. Additionally, for comparability of the computational requirements, all use two encoder layers and one decoder layer.

We use the adaptation to the long-horizon time series setting provided by Wu et al. 2021 of the Reformer (Kitaev et al., 2020), and LogTrans (Li et al., 2019c), with the multi-step forecasting strategy (non-dynamic decoding).

The Autoformer (Wu et al., 2021) explores with grid-search the top-k auto-correlation filter hyper-parameter in $\{1, 2, 3, 4, 5\}$. And fixes input $L = 96$ for all datasets except for **ILI** in which they use $L = 36$. For the Informer (Zhou et al., 2020) we use the reported best hyperparameters found using a grid-search that includes dimensions of the encoder layers $\{6, 4, 3, 2\}$, the dimension of the decoder layer $\{2\}$, the heads of the multi-head attention layers $\{8, 16\}$ and its output's $\{512\}$.

We considered other classic models, including the automatically selected ARIMA model (Hyndman and Khandakar, 2008). The method is trained with maximum likelihood estimation under normality and independence. It integrates root statistical tests with model selection performed

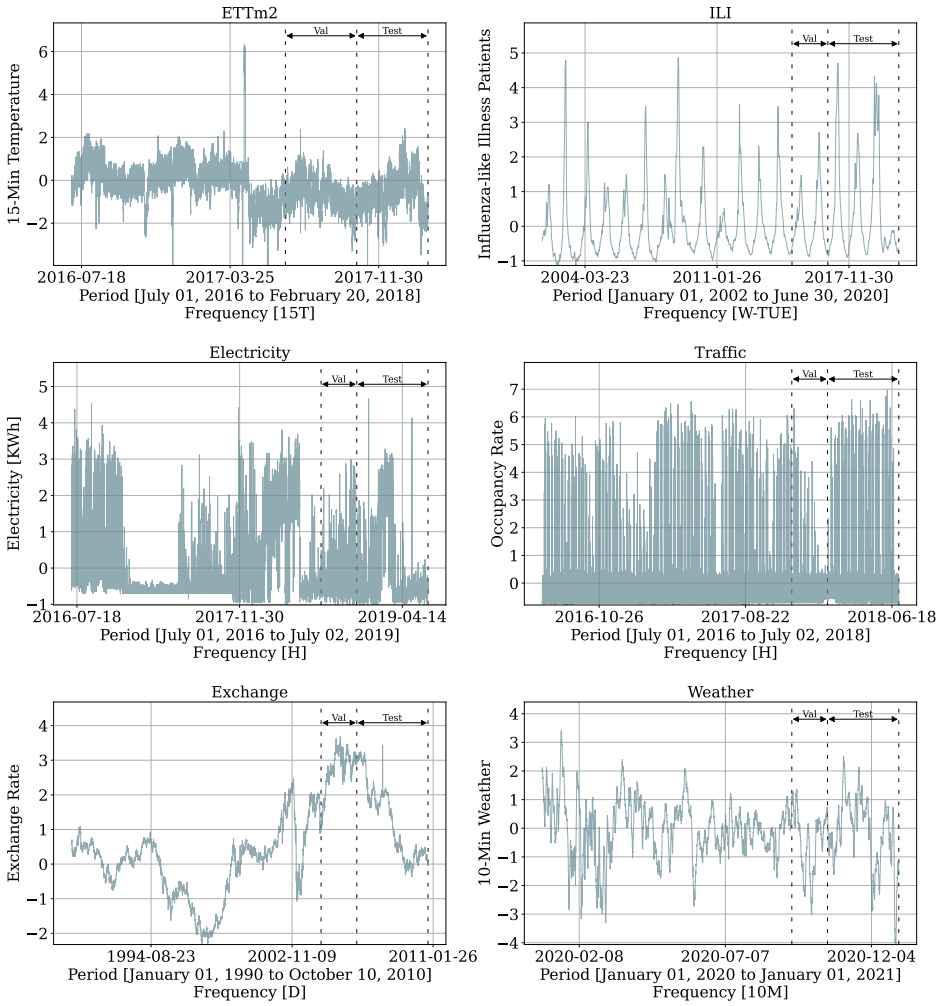


Figure C.1: Datasets partition into the train, validation, and test sets used in our experiments (**ETTm$_2$**, **ECL**, **Exchange**, **ILI**, **Traffic-L**, and **Weather**). All use the last 20% of the total observations as the test set (marked by the second dotted line), and the 10% preceding the test set as validation (between the first and second dotted lines), except for **ETTm$_2$** that also use 20% as validation. Validation provides the signal for hyperparameter optimization. We construct test predictions using rolling windows.

with Akaike's Information Criterion. For the univariate forecasting experiment, we consider `Prophet` (Taylor and Letham, 2018), an automatic Bayesian additive regression that accounts for different frequencies of non-linear trends, seasonal and holiday effects, for this method we tuned the seasonality mode $\{multiplicative, aditive\}$, the length of the inputs.

Finally, as mentioned in Section 4.4.3 for NHITS main results, we limit the exploration to a minimal space of hyperparameters. We only consider the kernel pooling size for multi-rate sampling from Equation (4.1), the number of coefficients in Equation (4.2) and the random seed from Table C.2.

Table C.2: Hyperparameters of the NHITS model

Hyperparameter	Considered Values
Initial learning rate.	{1e-3}
Training steps.	{1000}
Random seed for initialization.	DiscreteRange(1, 10)
Input size multiplier (L=m*H).	$m \in \{5\}$
Batch Size.	{256}
Activation Function.	ReLU
Learning rate decay (3 times).	0.5
Pooling Kernel Size.	$[k_1, k_2, k_3] \in$ {[2,2,2], [4,4,4], [8,8,8], [8,4,1], [16,8,1]}
Number of Stacks.	$S \in \{3\}$
Number of Blocks in each stack.	$B \in \{1\}$
MLP Layers.	{2}
Coefficients Hidden Size.	$N_h \in \{512\}$
Number of Stacks' Coefficients.	$[r_1^{-1}, r_2^{-1}, r_3^{-1}] \in$ {[168,24,1], [24,12,1] [180,60,1], [40,20,1], [64,8,1] }
Interpolation strategy	$g(\tau, \theta) \in$ {Linear}

C.5 Main results standard deviations

Table 4.1 reports the average accuracy measurements to comply with page restrictions. Here, we complement the Table's results with the standard deviation associated with the eight runs of the forecasting pipeline composed of the training and hyperparameter optimization methodologies described in Section 4.4.3. Overall, the standard deviation of the forecasting pipelines accounts for 2.9% of the MSE measurements and 1.75% of the MAE measurements. The small standard deviation verifies the robustness of the results and accuracy improvements of NHITS predictions. We observed that the **Exchange** accuracy measurements present the most variance between each run.

For the completeness of our empirical evaluation, we include in Table C.3 the comparison with the concurrent research including ETSformerand Preformer(Woo et al., 2022b; Du et al., 2022). Table C.3 shows that NHITS maintains MAE 11% and MSE 9% performance improvements across all the benchmark datasets and horizons versus the second-best alternative. The only experiment setting where a concurrent research model outperforms NHITS predictions is short-horizon **Exchange** where ETSformer reports MSE improvements of 9% and MAE improvements of 4%.

C.6 Univariate Forecasting

As a complement of the main results from Section 4.4.4, in this Appendix, we performed univariate forecasting experiments for the **ETTm$_2$** and **Exchange** datasets. This experiment allows us to compare closely with other methods specialized in long-horizon forecasting that also considered this setting (Zhou et al., 2020; Wu et al., 2021).

For the univariate setting, we consider the Transformer-based (1) Autoformer (Wu et al., 2021), (2) Informer (Zhou et al., 2020), (3) LogTrans (Li et al., 2019c) and (4)

Table C.3: Main empirical results in long-horizon forecasting setup, lower scores are better. Metrics are averaged over eight runs and standard deviation in brackets, best results are highlighted in bold, second best results are highlighted in blue.

		NHITS (Ours)		Autoformer		Informer		LogTrans		Reformer		ARIMA		FEDformer		ETSformer		Preformer	
	Horizon	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_2$	96	**0.176**	**0.255**	0.255	0.339	0.365	0.453	0.768	0.642	0.658	0.619	0.225	0.301	0.203	0.287	0.183	0.275	0.213	0.295
		(0.003)	(0.001)	(0.020)	(0.020)	(0.062)	(0.047)	(0.071)	(0.020)	(0.121)	(0.021)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	192	**0.245**	**0.305**	0.281	0.340	0.533	0.563	0.989	0.757	1.078	0.827	0.298	0.345	0.269	0.328	-	-	0.269	0.329
		(0.005)	(0.002)	(0.027)	(0.025)	(0.109)	(0.050)	(0.124)	(0.049)	(0.106)	(0.012)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	336	**0.295**	**0.346**	0.339	0.372	1.363	0.887	1.334	0.872	1.549	0.972	0.370	0.386	0.325	0.366	-	-	0.324	0.363
		(0.004)	(0.002)	(0.018)	(0.015)	(0.173)	(0.056)	(0.168)	(0.054)	(0.146)	(0.0)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	720	**0.401**	**0.413**	0.422	0.419	3.379	1.388	3.048	1.328	2.631	1.242	0.478	0.445	0.421	0.415	-	-	0.418	0.416
		(0.013)	(0.009)	(0.015)	(0.010)	(0.143)	(0.037)	(0.140)	(0.023)	(0.126)	(0.014)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
ECL	96	**0.147**	**0.249**	0.201	0.317	0.274	0.368	0.258	0.357	0.312	0.402	1.220	0.814	0.183	0.297	0.187	0.302	0.180	0.297
		(0.002)	(0.002)	(0.003)	(0.004)	(0.004)	(0.003)	(0.002)	(0.002)	(0.003)	(0.004)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	192	**0.167**	**0.269**	0.222	0.334	0.296	0.386	0.266	0.368	0.348	0.433	1.264	0.842	0.195	0.308	0.196	0.311	0.189	0.302
		(0.005)	(0.005)	(0.003)	(0.004)	(0.009)	(0.007)	(0.005)	(0.004)	(0.004)	(0.005)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	336	**0.186**	**0.290**	0.231	0.338	0.300	0.394	0.280	0.380	0.350	0.433	1.311	0.866	0.212	0.313	0.215	0.330	0.201	0.319
		(0.001)	(0.001)	(0.006)	(0.004)	(0.007)	(0.004)	(0.006)	(0.001)	(0.004)	(0.003)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	720	0.243	**0.340**	0.254	0.361	0.373	0.439	0.283	0.376	0.340	0.42	1.364	0.891	**0.231**	0.343	0.236	0.348	0.232	0.342
		(0.008)	(0.007)	(0.007)	(0.008)	(0.034)	(0.024)	(0.003)	(0.002)	(0.002)	(0.002)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
Exchange	96	0.092	0.02	0.197	0.323	0.847	0.752	0.968	0.812	1.065	0.829	0.296	0.214	0.139	0.276	**0.083**	**0.202**	0.148	0.282
		(0.002)	(0.002)	(0.019)	(0.012)	(0.150)	(0.060)	(0.177)	(0.027)	(0.070)	(0.013)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	192	0.208	0.322	0.300	0.369	1.204	0.895	1.040	0.851	1.188	0.906	1.056	0.326	0.256	0.369	**0.180**	**0.302**	0.268	0.378
		(0.025)	(0.020)	(0.020)	(0.016)	(0.149)	(0.061)	(0.232)	(0.029)	(0.041)	(0.008)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	336	**0.301**	**0.403**	0.509	0.524	1.672	1.036	1.659	1.081	1.357	0.976	2.298	0.467	0.426	0.464	0.354	0.433	0.447	0.499
		(0.042)	(0.030)	(0.041)	(0.016)	(0.036)	(0.014)	(0.122)	(0.015)	(0.027)	(0.010)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	720	**0.798**	**0.596**	1.447.	0.941	2.478	1.310	1.941	1.127	1.510	1.016	20.666	0.864	1.090	0.800	0.996	0.761	1.092	0.812
		(0.041)	(0.013)	(0.084)	(0.028)	(0.198)	(0.070)	(0.327)	(0.030)	(0.071)	(0.008)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
Traffic-L	96	**0.402**	**0.282**	0.613	0.388	0.719	0.391	0.684	0.384	0.732	0.423	1.997	0.924	0.562	0.349	0.614	0.395	0.560	0.349
		(0.005)	(0.002)	(0.028)	(0.012)	(0.150)	(0.060)	(0.177)	(0.027)	(0.070)	(0.013)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	192	**0.420**	**0.297**	0.616	0.382	0.696	0.379	0.685	0.39	0.733	0.42	2.044	0.944	0.562	0.346	0.629	0.398	0.565	0.349
		(0.002)	(0.003)	(0.042)	(0.020)	(0.050)	(0.023)	(0.055)	(0.021)	(0.013)	(0.011)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	336	**0.448**	**0.313**	0.622	0.337	0.777	0.420	0.733	0.408	0.742	0.42	2.096	0.960	0.570	0.323	0.646	0.417	0.577	0.351
		(0.006)	(0.003)	(0.009)	(0.003)	(0.069)	(0.026)	(0.012)	(0.008)	(0.0)	(0.0)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	720	**0.539**	**0.353**	0.660	0.408	0.864	0.472	0.717	0.396	0.755	0.423	2.138	0.971	0.596	0.368	0.631	0.389	0.597	0.358
		(0.022)	(0.012)	(0.025)	(0.015)	(0.026)	(0.015)	(0.030)	(0.010)	(0.023)	(0.014)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
Weather	96	**0.158**	**0.195**	0.266	0.336	0.300	0.384	0.458	0.49	0.689	0.596	0.217	0.258	0.217	0.296	0.189	0.272	0.227	0.292
		(0.002)	(0.002)	(0.007)	(0.006)	(0.013)	(0.013)	(0.143)	(0.038)	(0.042)	(0.019)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	192	**0.211**	**0.247**	0.307	0.367	0.598	0.544	0.658	0.589	0.752	0.638	0.263	0.299	0.276	0.336	0.231	0.303	0.275	0.322
		(0.001)	(0.003)	(0.024)	(0.022)	(0.045)	(0.028)	(0.151)	(0.032)	(0.048)	(0.029)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	336	**0.274**	**0.300**	0.359	0.395	0.578	0.523	0.797	0.652	0.064	0.596	0.330	0.347	0.339	0.380	0.305	0.357	0.324	0.352
		(0.009)	(0.008)	(0.035)	(0.031)	(0.024)	(0.016)	(0.034)	(0.019)	(0.030)	(0.021)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	720	**0.351**	**0.353**	0.419	0.428	1.059	0.741	0.869	0.675	1.130	0.792	0.425	0.405	0.403	0.428	0.352	0.391	0.394	0.393
		(0.020)	(0.016)	(0.017)	(0.014)	(0.096)	(0.042)	(0.045)	(0.093)	(0.084)	(0.055)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
ILI	24	**1.862**	**0.869**	3.483	1.287	5.764	1.677	4.480	1.444	4.4	1.382	5.554	1.434	2.203	0.963	2.862	1.128	3.143	1.185
		(0.064)	(0.020)	(0.107)	(0.018)	(0.354)	(0.080)	(0.313)	(0.033)	(0.177)	(0.021)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	36	**2.071**	**0.934**	3.103	1.148	4.755	1.467	4.799	1.467	4.783	1.448	6.940	1.676	2.272	0.976	2.683	1.029	2.793	1.054
		(0.015)	(0.003)	(0.139)	(0.025)	(0.248)	(0.067)	(0.251)	(0.023)	(0.138)	(0.023)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	48	**2.134**	**0.932**	2.669	1.085	4.763	1.469	4.800	1.468	4.832	1.465	7.192	1.736	2.209	0.981	2.456	0.986	2.845	1.090
		(0.142)	(0.034)	(0.151)	(0.037)	(0.295)	(0.059)	(0.233)	(0.021)	(0.122)	(0.016)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)
	60	**2.137**	**0.968**	2.770	1.125	5.264	1.564	5.278	1.56	4.882	1.483	6.648	1.656	2.545	1.061	2.630	1.057	2.957	1.124
		(0.075)	(0.012)	(0.085)	(0.019)	(0.237)	(0.044)	(0.231)	(0.014)	(0.123)	(0.016)	(-)	(-)	(-)	(-)	(-)	(-)	(-)	(-)

Reformer (Kitaev et al., 2020) models. We selected other well-established univariate forecasting benchmarks: (5) NBEATS (Oreshkin et al., 2020), (6) DeepAR (Salinas et al., 2020) model, which takes autoregressive features and combines them with classic recurrent networks. (7)

Prophet (Taylor and Letham, 2018), an additive regression model that accounts for different frequencies of non-linear trends, seasonal and holiday effects, and (8) an auto ARIMA (Hyndman and Khandakar, 2008).

Table C.4 summarizes the univariate forecasting results. NHITS significantly improves over the alternatives, decreasing in MAE and in MSE across datasets and horizons with respect to the best alternative. As noticed by the community, recurrent-based strategies, such as ARIMA, tend to degrade due to the concatenation of errors phenomenon.

Table C.4: Empirical evaluation of long multi-horizon **univariate** forecasts. *Mean Absolute Error* (MAE) and *Mean Squared Error* (MSE) for predictions averaged over eight runs, the best result is highlighted in bold (lower is better).

		NHITS		Autoformer		Informer		Reformer		NBEATS		DeepAR		ARIMA	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_2$	96	0.066	**0.185**	**0.065**	0.189	0.088	0.225	0.131	0.288	0.082	0.219	0.099	0.237	0.211	0.362
	192	**0.087**	**0.223**	0.118	0.256	0.132	0.283	0.186	0.354	0.120	0.268	0.154	0.310	0.261	0.406
	336	**0.106**	**0.251**	0.154	0.305	0.180	0.336	0.220	0.381	0.226	0.370	0.277	0.428	0.317	0.448
	720	**0.157**	**0.312**	0.182	0.335	0.300	0.435	0.267	0.430	0.188	0.338	0.332	0.468	0.366	0.487
Exchange	96	**0.093**	**0.223**	0.241	0.299	0.591	0.615	1.327	0.944	0.156	0.299	0.417	0.515	0.112	0.245
	192	**0.230**	**0.313**	0.273	0.665	1.183	0.912	1.258	0.924	0.669	0.665	0.813	0.735	0.304	0.404
	336	**0.370**	**0.486**	0.508	0.605	1.367	0.984	2.179	1.296	0.611	0.605	1.331	0.962	0.736	0.598
	720	**0.728**	**0.569**	0.991	0.860	1.872	1.072	1.280	0.953	1.111	0.860	1.890	1.181	1.871	0.935

C.7 Ablation Studies

This section performs ablation studies on the validation set of five datasets that share horizon lengths, **$ETTm_2$**, **Exchange**, **ECL**, **Traffic-L**, and **Weather**. The section's experiments control for NHITS settings described in Table C.2, only varying a single characteristic of interest of the network and measuring the effects in validation.

Pooling Configurations

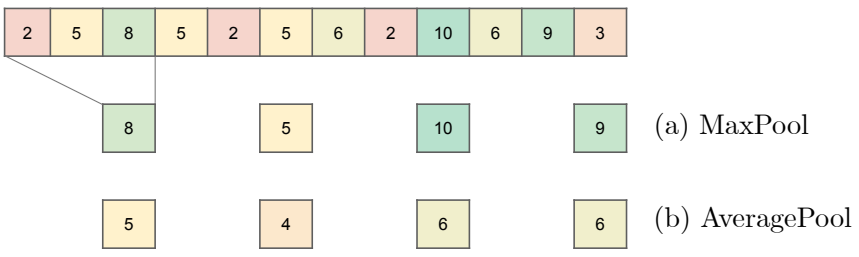


Figure C.2: Proposed pooling configurations

In Section 4.3.1, we described the *multi-rate signal sampling* enhancement of the NHITS architecture. Here, we conduct a study to compare the accuracy effects of different pooling alternatives on Equation (4.1). We consider the MaxPool and AveragePool configurations.

As shown in Table C.5, the MaxPool operation consistently outperforms the AveragePool alternative, with MAE improvements up to 15% and MSE up to 8% in the most extended horizon. On average, the forecasting accuracy favors the MaxPool method across the datasets and horizons.

Table C.5: Empirical evaluation of long-horizon multivariate forecasts for NHITS with different pooling configurations. All other hyperparameters were kept constant across all datasets. MAE and MSE for predictions averaged over eight seeds; the best result is highlighted in bold (lower is better). Average percentage difference relative to average pooling in the last panel.

		MaxPool		AveragePool	
		MSE	MAE	MSE	MAE
$ETTm_2$	96	**0.185**	0.265	0.186	**0.262**
	192	**0.244**	**0.308**	0.257	0.315
	336	**0.301**	**0.347**	0.312	0.356
	720	**0.429**	**0.438**	0.436	0.447
ECL	96	**0.152**	**0.257**	0.181	0.290
	192	**0.172**	**0.275**	0.212	0.320
	336	**0.197**	**0.304**	0.238	0.343
	720	**0.248**	**0.347**	0.309	0.400
Exchange	96	**0.109**	**0.232**	0.112	0.238
	192	0.280	0.375	**0.265**	**0.371**
	336	**0.472**	0.504	0.501	**0.502**
	720	**1.241**	**0.823**	1.610	0.942
Traffic-L	96	**0.405**	**0.286**	0.468	0.332
	192	**0.421**	**0.297**	0.490	0.347
	336	**0.448**	**0.318**	0.531	0.371
	720	**0.527**	**0.362**	0.602	0.400
P. Diff.	96	**-8.911**	**-6.251**	0.000	0.000
	192	**-7.544**	**-6.085**	0.000	0.000
	336	**-8.740**	**-4.575**	0.000	0.000
	720	**-15.22**	**-8.318**	0.000	0.000

Interpolation Configurations

In Section 4.3.3, we described the *hierarchical interpolation* enhancement of the multi-step prediction strategy. Here, we conduct a study to compare the accuracy effects of different interpolation alternatives. To do it, we change the interpolation technique used in the multi-step forecasting strategy of the NHITS architecture. The interpolation techniques considered are nearest neighbor, linear, and cubic. We describe them in detail below.

Recalling the notation from Section 4.3.3, consider the time indexes of a multi-step prediction

$\tau \in \{t+1, \dots, t+H\}$, let $\mathcal{T} = \{t+1, t+1+1/r_\ell \dots, t+H\}$ be the anchored indexes in NHITS layer ℓ, and the forecast $\hat{y}_{\tau,\ell} = g(\tau, \theta^f_\ell)$ and backast $\tilde{y}_{\tau,\ell} = g(\tau, \theta^f_\ell)$ components. Here we define different alternatives for the interpolating function $g \in \mathcal{C}^0, \mathcal{C}^1, \mathcal{C}^2$. For simplicity, we skip the ℓ layer index.

Nearest Neighbor. In the simplest interpolation, we use the anchor observations in the time dimension closest to the observation we want to predict. Specifically, the prediction is defined as follows:

$$\hat{y}_\tau = \theta[t^*] \quad \text{with} \quad t^* = \text{argmin}_{t \in \mathcal{T}}\{|t - \tau|\} \tag{C.4}$$

Linear. An efficient alternative is the linear interpolation method, which uses the two closest neighbor indexes t_1 and t_1, and fits a linear function that passes through both.

$$\hat{y}_\tau = \left(\theta[t_1] + \left(\frac{\theta[t_2] - \theta[t_1]}{t_2 - t_1}\right)(\tau - t_1)\right) \tag{C.5}$$

Cubic. Finally we consider the Hermite cubic polynomials defined by the interpolation constraints for two anchor observations θ_{t_1} and θ_{t_1} and its first derivatives θ'_{t_1} and θ'_{t_1}.

$$\hat{y}_\tau = \theta[t_1]\phi_1(\tau) + \theta[t_2]\phi_2(\tau) + \theta'[t_1]\psi_1(\tau) + \theta'[t_2]\psi_2(\tau) \tag{C.6}$$

With the Hermite cubic basis defined by:

$$\phi_1(\tau) = 2\tau^3 - 3\tau^2 + 1 \tag{C.7a}$$
$$\phi_2(\tau) = -2\tau^3 + 3\tau^2 \tag{C.7b}$$
$$\psi_1(\tau) = \tau^3 - 2\tau^2 + \tau \tag{C.7c}$$
$$\psi_2(\tau) = \tau^3 - \tau^2 \tag{C.7d}$$

The ablation study results for the different interpolation techniques are summarized in Table C.6, we report the average MAE and MSE performance across the five datasets. We found that linear and cubic interpolation consistently outperform the nearest neighbor alternative and show monotonic improvements relative to the nearest neighbor technique along the forecasting horizon.

The linear interpolation improvements over nearest neighbors are up to 15.8%, and up to 7.0% for the cubic interpolation. When comparing between linear and cubic, the results are inconclusive as different datasets and horizons have slight performance differences. On average, across the datasets, both the forecasting accuracy and computational performance favor the linear method, with which we conducted the main experiments of this work with this technique.

Order of Hierarchical Representations

Deep Learning in classic tasks such as computer vision and natural language processing is known to learn hierarchical representations from raw data that increase complexity as the information flows through the network. This automatic feature extraction phenomenon is believed to drive

Table C.6: Empirical evaluation of long multi-horizon multivariate forecasts for NHITS with different interpolation configurations. All other hyperparameters were kept constant across all datasets. MAE and MSE for predictions averaged over eight seeds; the best result is highlighted in bold (lower is better). Percentage difference relative to n. neighbor in the last panel, average across datasets.

		Linear		Cubic		N.Neighbor	
		MSE	MAE	MSE	MAE	MSE	MAE
$ETTm_2$	96	0.185	0.265	**0.179**	**0.256**	0.180	0.259
	192	0.244	0.308	**0.241**	**0.303**	0.252	0.315
	336	**0.301**	**0.347**	0.314	0.358	0.302	0.351
	720	**0.429**	**0.438**	0.439	0.450	0.442	0.455
ECL	96	0.152	0.257	**0.149**	**0.252**	0.151	0.255
	192	**0.172**	**0.275**	0.174	0.279	0.175	0.279
	336	0.197	0.304	**0.190**	**0.295**	0.211	0.318
	720	**0.248**	**0.347**	0.256	0.353	0.263	0.358
Exchange	96	**0.109**	**0.232**	0.1307	0.254	0.126	0.248
	192	0.280	0.375	**0.247**	**0.357**	0.357	0.416
	336	**0.472**	**0.504**	0.625	0.560	0.646	0.560
	720	**1.241**	**0.823**	1.539	0.925	1.740	0.973
Traffic-L	96	0.405	0.286	**0.402**	**0.282**	0.405	0.359
	192	0.421	0.297	**0.417**	**0.295**	0.419	0.201
	336	0.448	0.318	**0.446**	**0.315**	0.445	0.253
	720	0.527	0.362	0.540	0.366	**0.525**	**0.318**
Weather	96	0.164	0.199	0.162	0.203	**0.161**	0.360
	192	0.224	0.255	0.225	0.257	**0.218**	0.928
	336	**0.285**	**0.311**	0.285	0.304	0.298	0.988
	720	**0.366**	**0.359**	0.380	0.369	0.368	1.047
P. Diff.	96	**-0.907**	**-0.717**	0.146	1.61	0.000	0.000
	192	-5.582	-3.259.	**-7.985**	**-4.332**	0.000	0.000
	336	**-10.516**	**-4.199**	-2.108	-1.455	0.000	0.000
	720	**-15.800**	**-7.042**	-5.480	-1.579	0.000	0.000

to a large degree the algorithms' success (Bengio et al., 2012). Our approach differs from the conventions in the sense that we use a *Top-Down* hierarchy where we prioritize in the synthesis of the predictions to low frequencies and sequentially complement them with higher frequencies details, as explained in Section 4.3. We achieve this with NHITS' *expressiveness ratio* schedules. Our intuition is that the *Top-Down* hierarchy acts as a regularizer and helps the model to focus on the broader factors driving the predictions rather than narrowing its focus at the beginning on the details that compose them. To test these intuitions, we designed an experiment where we inverted the expressiveness ratio schedule into *Bottom-Up* hierarchy predictions and compared the validation performance.

Remarkably, as shown in Table C.7, the *Top-Down* predictions consistently outperform the *Bottom-Up* counterpart. Relative improvements in MAE are 4.6%, in MSE of 7.5%, across horizons and datasets. Our observations match the forecasting community practice that addresses long-horizon predictions by first modeling the long-term seasonal components and then its residuals.

Table C.7: Empirical evaluation of long multi-horizon multivariate forecasts for NHITS with different hierarchical orders. All other hyperparameters were kept constant across all datasets. MAE and MSE for predictions averaged over eight seeds, the best result is highlighted in bold (lower is better). Average percentage difference relative to ascending hierarchy in the last panel.

		Top-Down		Bottom-Up	
		MSE	MAE	MSE	MAE
ETTm2	96	**0.185**	**0.265**	0.191	0.266
	192	**0.244**	**0.308**	0.261	0.320
	336	**0.301**	**0.347**	0.302	0.353
	720	**0.429**	**0.438**	0.440	0.454
ECL	96	**0.152**	**0.257**	0.164	0.270
	192	**0.172**	**0.275**	0.186	0.292
	336	**0.197**	**0.304**	0.217	0.327
	720	**0.248**	**0.347**	0.273	0.369
Exchange	96	**0.109**	**0.232**	0.114	0.242
	192	**0.280**	**0.375**	0.436	0.452
	336	**0.472**	**0.504**	0.654	0.574
	720	**1.241**	**0.823**	1.312	0.861
Traffic-L	96	**0.405**	**0.286**	0.410	0.292
	192	**0.421**	**0.297**	0.427	0.305
	336	**0.448**	**0.318**	0.456	0.323
	720	**0.527**	**0.362**	0.557	0.379
Weather	96	0.164	**0.199**	**0.163**	0.200
	192	0.224	0.255	**0.219**	**0.252**
	336	**0.285**	**0.311**	0.288	0.311
	720	0.366	0.359	**0.365**	**0.355**
P. Diff.	96	**-2.523**	**-2.497**	0.000	0.000
	192	**-12.296**	**-6.793**	0.000	0.000
	336	**-11.176**	**-5.507**	0.000	0.000
	720	**-4.638**	**-3.699**	0.000	0.000

C.8 Multi-rate sampling and Hierarchical Interpolation beyond NHITS

Empirical observations let us infer that the advantages of the NHITS architecture are rooted in its multi-rate hierarchical nature, as both the multi-rate sampling and the hierarchical interpolation complement the long-horizon forecasting task in MLP-based architectures. In this ablation experiment, we quantitatively explore the effects and complementarity of the techniques in an RNN-based architecture.

This experiment follows the Table 4.2 ablation study, reporting the average performance across **ETTm**2, **ECL**, **Exchange**, **Traffic-L**, and **Weather** datasets. We define the following set of alternative models: DilRNN$_1$, our proposed model with both multi-rate sampling and hierarchical interpolation, DilRNN$_2$ only hierarchical interpolation, DilRNN$_3$ only multi-rate

sampling, DilRNN with no multi-rate sampling or interpolation (corresponds to the original DilRNN (Chang et al., 2017)).

Tab. C.8 shows that the hierarchical interpolation technique drives the main improvements ($\texttt{DilRNN}_2$), while the combination of both proposed components (hierarchical interpolation and multi-rate sampling) sometimes results in the best performance ($\texttt{DilRNN}_1$), the difference is marginal. Contrary to the clear complementary observed in Table 4.2, the DilRNN does not improve substantially from the multi-rate sampling techniques. We find an explanation in the behavior of the RNN that summarizes past inputs and the current observation of the series, and not the complete whole past data like the MLP-based architectures.

A key takeaway of this experiment is that NHITS' hierarchical interpolation technique exhibits significant benefits in other architectures. Despite these promising results, we decided not to pursue more complex architectures in our work as we found that the interpretability of the NHITS predictions and signal decomposition capabilities was not worth losing.

Table C.8: Empirical evaluation of long multi-horizon multivariate forecasts for DilRNN with/without enhancements. Average MAE and MSE for five datasets. The best result is highlighted in bold, and the second best is in blue (lower is better).

		$\texttt{DilRNN}_1$	$\texttt{DilRNN}_2$	$\texttt{DilRNN}_3$	DilRNN
A. MSE	96	0.346	**0.331**	0.369	0.347
	192	0.539	**0.528**	0.545	0.561
	336	**0.647**	0.691	0.705	0.723
	720	0.765	**0.762**	0.789	0.800
A. MAE	96	**0.343**	0.335	0.352	0.347
	192	0.460	**0.444**	0.462	0.468
	336	**0.513**	0.537	0.539	0.649
	720	0.585	**0.566**	0.600	0.598

C.9 Hyperparameter Optimization Resources

Computational efficiency has implications for the prediction's accuracy and the cost of deployment. Since forecasting systems are constantly retrained to address distributional shifts, orders-of-magnitude improvements in speed can easily translate into orders-of-magnitude price differences deploying the models. This section explores the implications of computational efficiency in the accuracy gains associated with hyperparameter optimization and training economic costs.

Hyperparameter Optimization. Despite all the progress in improving the computation efficiency of Transformer-based methods, see Figure 4.4, their speed and memory requirements make exploring their hyperparameter space unaffordable in practice, considering the amount of GPU computation they still require.

For this experiment, we report the iterations of the *hyperparameter optimization phase*, described in Section 4.4.3, where we explore the hyperparameters from Table C.2 using HYPEROPT,

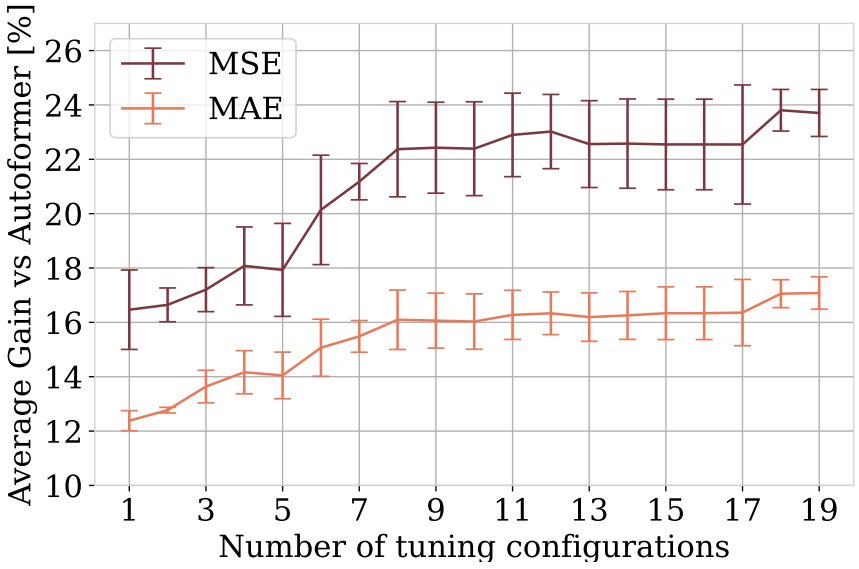


Figure C.3: NHITS performance improvement over Autoformer as a function of explored hyperparameter configurations.

a Bayesian hyperparameter optimization library (Bergstra et al., 2011). As shown in Figure C.3 the exploration exhibits monotonic relative performance gains of NHITS versus the best reported Autoformer (Wu et al., 2021) in the ablation datasets.

Training Economic Costs. We measure the train time for NHITS, NBEATS-G and Transformer-based models on the six main experiment datasets and 8 runs. We rely on an AWS g4dn.2xlarge, with an NVIDIA T4 16GB GPU.

We differentiate between NHITS_1, our method with a single HYPEROPT iteration randomly sampled from Table C.2, and NHITS_{20} to our method after 20 HYPEROPT iterations. For the Transformer-based models, we used optimal hyperparameters as reported in their repositories. Table C.9 shows the measured train time for the models, NHITS_1 takes 1.5 hours while more expensive architectures such as the Autoformer or Informer take 92.6 and 62.1 hours each. Based on hourly prices from January 2022 for the g4dn.2xlarge instance, USD 0.75, the main results of the paper would cost nearly USD 70.0 with Autoformer, USD 46.5 with Informer, while NHITS_1 results can be executed under USD 1.5 and NHITS_{20} with USD 22.8. Figure C.3, shows that NHITS_1 achieves a 17% MSE average performance gain over Autoformer with 1.6% of a single run cost, and NHITS_{20} almost 25% gain with 33% of a single run cost. A single run does not consider hyperparameter optimization.

$$\text{ExptPrice} = \text{GPUPrice} \times \text{HyperOptIters} \times \text{TrainTime} \times \text{Runs}$$

Table C.9: Train time hours on a `g4dn.2xlarge` instance.

	Horizon	NHITS$_1$	NHITS$_{20}$	Autoformer	Informer	NBEATS-G
A. Time	96/24	0.183	3.66	12.156	9.11	0.291
	192/36	0.257	5.14	16.734	11.598	0.462
	336/48	0.398	7.96	22.73	15.237	0.674
	720/60	0.682	13.64	40.987	26.173	1.249
Total		1.523	30.46	92.607	62.118	2.676

Bibliography

[1] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407 (cit. on p. 65).

[2] Charles C Holt. "Forecasting seasonals and trends by exponentially weighted moving averages". In: *(O.N.R. Memorandum No. 52)* (1957). URL: https://doi.org/10.1016/j.ijforecast.2003.09.015 (cit. on p. 39).

[3] Frank Rosenblatt. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc Buffalo NY, 1961 (cit. on p. 66).

[4] Clive WJ Granger. "Investigating causal relations by econometric models and cross-spectral methods". In: *Econometrica: journal of the Econometric Society* (1969), pp. 424–438 (cit. on pp. 26, 35).

[5] Gregory C. Chow and An-loh Lin. "Best Linear Unbiased Interpolation, Distribution, and Extrapolation of Time Series by Related Series". In: *The Review of Economics and Statistics* 53.4 (1971), pp. 372–375. ISSN: 00346535, 15309142. URL: http://www.jstor.org/stable/1928739 (cit. on p. 40).

[6] Christopher A Sims. "Macroeconomics and reality". In: *Econometrica: journal of the Econometric Society* (1980), pp. 1–48 (cit. on p. 26).

[7] Roque B Fernandez. "A Methodological Note on the Estimation of Time Series". In: *The Review of Economics and Statistics* 63.3 (Aug. 1981), pp. 471–476. URL: https://ideas.repec.org/a/tpr/restat/v63y1981i3p471-76.html (cit. on p. 41).

[8] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. "The accuracy of extrapolation (time series) methods: Results of a forecasting competition". In: *Journal of Forecasting* 1.2 (1982), pp. 111–153. DOI: https://doi.org/10.1002/for.3980010202. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.3980010202. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/for.3980010202 (cit. on p. 68).

[9] Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: https://www.sciencedirect.com/science/article/pii/089360809190009T (cit. on p. 91).

[10] Ingrid Daubechies. *Ten lectures on wavelets*. SIAM, 1992 (cit. on p. 49).

[11] Andrew R Barron. “Universal approximation bounds for superpositions of a sigmoidal function”. In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945 (cit. on p. 91).

[12] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140 (cit. on p. 28).

[13] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 1999 (cit. on p. 65).

[14] Felix A. Gers, Fred Cummins, and Jürgen Schmidhuber. “Learning to forget: continual prediction with LSTM”. English. In: *Neural Computation* 12 (2000), pp. 2451–2471. URL: https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218 (cit. on p. 66).

[15] Spyros Makridakis and Michèle Hibon. “The M3-Competition: results, conclusions and implications”. In: *International Journal of Forecasting* 16.4 (2000). The M3- Competition, pp. 451–476. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(00)00057-1. URL: https://www.sciencedirect.com/science/article/pii/S0169207000000571 (cit. on p. 68).

[16] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. “Correcting Sample Selection Bias by Unlabeled Data”. In: *Advances in Neural Information Processing Systems*. Ed. by B. Schölkopf, J. Platt, and T. Hoffman. Vol. 19. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper_files/paper/2006/file/a2186aa7c086b46ad4e8bf81e2a3a19b-Paper.pdf (cit. on p. 63).

[17] Rob J. Hyndman and Anne B. Koehler. “Another look at measures of forecast accuracy”. In: *International Journal of Forecasting* 22.4 (2006), pp. 679–688. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2006.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0169207006000239 (cit. on p. 69).

[18] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. “Boosting for Transfer Learning”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon, USA: Association for Computing Machinery, 2007, pp. 193–200. ISBN: 9781595937933. DOI: 10.1145/1273496.1273521. URL: https://doi.org/10.1145/1273496.1273521 (cit. on p. 63).

[19] Eric Ghysels, Arthur Sinko, and Rossen Valkanov. “MIDAS Regressions: Further Results and New Directions”. In: *Econometric Reviews* 26.1 (2007), pp. 53–90. DOI: 10.1080/07474930600972467. URL: https://doi.org/10.1080/07474930600972467 (cit. on p. 40).

[20] Rob J. Hyndman and Yeasmin Khandakar. “Automatic Time Series Forecasting: The forecast Package for R”. In: *Journal of Statistical Software, Articles* 27.3 (2008), pp. 1–22. ISSN: 1548-7660. DOI: 10.18637/jss.v027.i03. URL: https://www.jstatsoft.org/v027/i03 (cit. on pp. 46, 66, 68, 93, 97).

[21] Yi Yao and Gianfranco Doretto. “Boosting for transfer learning with multiple sources”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, pp. 1855–1862. DOI: 10.1109/CVPR.2010.5539857 (cit. on p. 63).

[22] George Athanasopoulos, Rob J. Hyndman, Haiyan Song, and Doris C. Wu. “The tourism forecasting competition”. In: *International Journal of Forecasting* 27.3 (2011). Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence Special Section 2: Tourism Forecasting, pp. 822–844. ISSN: 0169-2070. DOI: https://doi.

org/10.1016/j.ijforecast.2010.04.009. URL: https://www.sciencedirect.com/science/article/pii/S016920701000107X (cit. on p. 68).

[23] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems* 24 (2011) (cit. on pp. 32, 46, 103).

[24] Radford M Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2 (cit. on p. 13).

[25] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. "Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives". In: *CoRR* abs/1206.5538 (2012). arXiv: 1206.5538. URL: http://arxiv.org/abs/1206.5538 (cit. on p. 100).

[26] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. "Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012". In: *2012 Computing in Cardiology*. IEEE. 2012, pp. 245–248 (cit. on p. 24).

[27] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828 (cit. on p. 2).

[28] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 28, 31, 46).

[29] Vitaly Kuznetsov and Mehryar Mohri. "Generalization bounds for time series prediction with non-stationary processes". In: *International conference on algorithmic learning theory*. Springer. 2014, pp. 260–274 (cit. on pp. 3, 84).

[30] Martin Längkvist, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling". In: *Pattern Recognition Letters* 42 (2014), pp. 11–24. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2014.01.008. URL: https://www.sciencedirect.com/science/article/pii/S0167865514000221 (cit. on p. 63).

[31] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition". In: *Computing Research Repository* abs/1402.1128 (2014). arXiv: 1402.1128. URL: http://arxiv.org/abs/1402.1128 (cit. on p. 66).

[32] Rafał Weron. "Electricity price forecasting: A review of the state-of-the-art with a look into the future". In: *International Journal of Forecasting* 30.4 (2014), pp. 1030–1081. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2014.08.008. URL: https://www.sciencedirect.com/science/article/pii/S0169207014001083 (cit. on p. 53).

[33] Jason Yosinski, J eff Clune, Yoshua Bengio, and Hod Lipson. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG] (cit. on pp. 62, 63).

[34] Albert Boggess and Francis J Narcowich. *A first course in wavelets with Fourier analysis*. John Wiley & Sons, 2015 (cit. on p. 90).

[35] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN: 9781118674925. URL: https://books.google.com/books?id=rNt5CgAAQBAJ (cit. on p. 72).

[36] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (cit. on p. 2).

[37] Lujie Chen, Artur Dubrawski, Gilles Clermont, Marilyn Hravnak, and Michael R Pinsky. "Modelling risk of cardio-respiratory Instability as a heterogeneous process". In: *AMIA Annual Symposium Proceedings*. Vol. 2015. American Medical Informatics Association. 2015, p. 1841 (cit. on p. 24).

[38] David C Farrow, Logan C Brooks, Aaron Rumack, Ryan J Tibshirani, and Roni Rosenfeld. "Delphi epidata API". In: *The Lancet Infectious Diseases. https://github. com/cmu-delphi/delphi-epidata* (2015) (cit. on p. 60).

[39] Amir Atiya and Ben Taieb. "A Bias and Variance Analysis for Multistep-Ahead Time Series Forecasting". In: *IEEE transactions on neural networks and learning systems* 27.1 (2016), pp. 2162–2388. URL: https://pubmed.ncbi.nlm.nih.gov/25807572/ (cit. on p. 39).

[40] Matthew M Churpek, Richa Adhikari, and Dana P Edelson. "The value of vital sign trends for detecting clinical deterioration on the wards". In: *Resuscitation* 102 (2016), pp. 1–5 (cit. on p. 55).

[41] Jose A. Fiorucci, Tiago R. Pellegrini, Francisco Louzada, Fotios Petropoulos, and Anne B. Koehler. "Models for optimising the theta method and their relationship to state space models". In: *International Journal of Forecasting* 32.4 (2016), pp. 1151–1161. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2016.02.005. URL: https://www.sciencedirect.com/science/article/pii/S0169207016300243 (cit. on p. 68).

[42] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016 (cit. on p. 2).

[43] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. "WaveNet: A Generative Model for Raw Audio". In: *Computer Research Repository* abs/1609.03499 (2016). arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499 (cit. on p. 66).

[44] Bartosz Uniejewski, Jakub Nowotarski, and Rafał Weron. "Automated Variable Selection and Shrinkage for Day-Ahead Electricity Price Forecasting". In: *Energies* 9(8),621.8 (2016). ISSN: 1996-1073. DOI: 10.3390/en9080621 (cit. on p. 53).

[45] Roberto Visentin, Chiara Dalla Man, and Claudio Cobelli. "One-Day Bayesian Cloning of Type 1 Diabetes Subjects: Toward a Single-Day UVA/Padova Type 1 Diabetes Simulator". In: *IEEE Transactions on Biomedical Engineering* 63.11 (2016) (cit. on p. 58).

[46] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning". In: *Journal of Big Data* 3.1 (May 2016), p. 9. ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6 (cit. on p. 63).

[47] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. "ST-MVL: filling missing values in geo-sensory time series data". In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 2016 (cit. on p. 24).

[48] Joos-Hendrik Bose, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. "Probabilistic demand forecasting at scale". In: *Proceedings of the VLDB Endowment* 10.12 (2017), pp. 1694–1705 (cit. on p. 24).

[49] US-CDC. *Influenza-like illness patients records*. 2017. URL: https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html (cit. on p. 72).

[50] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. "Dilated recurrent neural networks". In: *Advances in neural information processing systems* 30 (2017) (cit. on pp. 32, 46, 102).

[51] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml (cit. on p. 72).

[52] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "Alternating back-propagation for generator network". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017 (cit. on pp. 8, 10–12, 15, 26, 27, 29).

[53] Boris Hanin and Mark Sellke. *Approximating Continuous Functions by ReLU Nets of Minimal Width*. 2017. URL: https://arxiv.org/abs/1710.11278 (cit. on p. 91).

[54] Maham Jahangir, Hammad Afzal, Mehreen Ahmed, Khawar Khurshid, and Raheel Nawaz. "An expert system for diabetes prediction using auto tuned multi-layer perceptron". In: *2017 Intelligent systems conference (IntelliSys)*. IEEE. 2017, pp. 722–728 (cit. on p. 55).

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90 (cit. on p. 40).

[56] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks". In: *Special Interest Group on Information Retrieval Conference 2018 (SIGIR 2018)* abs/1703.07015 (2017). arXiv: 1703.07015. URL: http://arxiv.org/abs/1703.07015 (cit. on p. 45).

[57] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery". In: *International conference on information processing in medical imaging*. Springer. 2017, pp. 146–157 (cit. on pp. 9, 17).

[58] Alban Siffer, Pierre-Alain Fouque, Alexandre Termier, and Christine Largouet. "Anomaly detection in streams with extreme value theory". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 1067–1075 (cit. on p. 21).

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017) (cit. on p. 26).

[60] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. "A Multi-Horizon Quantile Recurrent Forecaster". In: *31st Conference on Neural Information Processing Systems NIPS 2017, Time Series Workshop*. 2017. arXiv: 1711.11053 [stat.ML]. URL: https://arxiv.org/abs/1711.11053 (cit. on pp. 62, 63).

[61] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271* (2018) (cit. on p. 80).

[62] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". In: *Computing Research Repository* abs/1803.01271 (2018). arXiv: 1803.01271. URL: http://arxiv.org/abs/1803.01271 (cit. on pp. 51, 66).

[63] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. "Brits: Bidirectional recurrent imputation for time series". In: *Advances in neural information processing systems* 31 (2018) (cit. on p. 26).

[64] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. "Transfer learning for time series classification". In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2018. DOI: 10.1109/bigdata.2018.8621990 (cit. on p. 64).

[65] Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. "Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1387–1395 (cit. on p. 55).

[66] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 387–395 (cit. on p. 16).

[67] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018 (cit. on pp. 2, 27).

[68] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018 (cit. on p. 38).

[69] Jesus Lago, Fjo De Ridder, and Bart De Schutter. "Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms". In: *Applied Energy* 221 (2018), pp. 386–405. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2018.02.069. URL: http://www.sciencedirect.com/science/article/pii/S030626191830196X (cit. on p. 53).

[70] Jesus Lago, Fjo De Ridder, Peter Vrancx, and Bart De Schutter. "Forecasting day-ahead electricity prices in Europe: The importance of considering market integration". In: *Applied Energy* 211 (2018), pp. 890–903. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2017.11.098. URL: https://www.sciencedirect.com/science/article/pii/S0306261917316999 (cit. on p. 52).

[71] Dan Li, Dacheng Chen, Jonathan Goh, and See-kiong Ng. "Anomaly detection with generative adversarial networks for multivariate time series". In: *arXiv preprint arXiv:1809.04758* (2018) (cit. on p. 9).

[72] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward". In: *PLoS One* 13(3) (2018), e0194889. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0194889 (cit. on p. 63).

[73] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1544–1551 (cit. on pp. 9, 17).

[74] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. "Deep one-class classification". In: *International conference on machine learning*. PMLR. 2018, pp. 4393–4402 (cit. on p. 17).

[75] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. *Wasserstein Distance Guided Representation Learning for Domain Adaptation*. 2018. arXiv: 1707.01217 [stat.ML] (cit. on p. 63).

[76] Sean J Taylor and Benjamin Letham. "Forecasting at scale". In: *The American Statistician* 72.1 (2018), pp. 37–45 (cit. on pp. 94, 97).

[77] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. "Balanced Distribution Adaptation for Transfer Learning". In: *CoRR* abs/1807.00516 (2018). arXiv: 1807.00516. URL: http://arxiv.org/abs/1807.00516 (cit. on p. 63).

[78] Jinyu Xie. *Simglucose v0.2.1 (2018)*. 2018. URL: https://github.com/jxx123/simglucose (cit. on p. 58).

[79] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications". In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 187–196 (cit. on p. 17).

[80] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. "Deep autoencoding gaussian mixture model for unsupervised anomaly detection". In: *International conference on learning representations*. 2018 (cit. on p. 17).

[81] Raghavendra Chalapathy and Sanjay Chawla. "Deep learning for anomaly detection: A survey". In: *arXiv preprint arXiv:1901.03407* (2019) (cit. on pp. 8, 9).

[82] Sophie Emerson, Ruairi Kennedy, Luke O'Shea, and John O'Brien. "Trends and applications of machine learning in quantitative finance". In: *8th international conference on economics and finance research (ICEFR 2019)*. 2019 (cit. on p. 24).

[83] Jan Gasthaus, Konstantinos Benidis, Bernie Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. "Probabilistic Forecasting with Spline Quantile Function RNNs". In: *AISTATS*. 2019 (cit. on p. 41).

[84] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks". In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 703–716 (cit. on pp. 9, 17).

[85] Kezhi Li, John Daniels, Chengyuan Liu, Pau Herrero, and Pantelis Georgiou. "Convolutional recurrent neural networks for glucose prediction". In: *IEEE journal of biomedical and health informatics* 24.2 (2019), pp. 603–613 (cit. on p. 55).

[86] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. "Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting". In: *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: http://arxiv.org/abs/1907.00235 (cit. on pp. 45, 46, 93, 95).

[87] Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, and Yisong Yue. "Naomi: Non-autoregressive multiresolution sequence imputation". In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 26).

[88] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. "E2gan: End-to-end generative adversarial network for multivariate time series imputation". In: *Proceedings of the*

28th international joint conference on artificial intelligence. 2019, pp. 3094–3100 (cit. on p. 26).

[89] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *arXiv preprint arXiv:1905.10437* (2019) (cit. on pp. 30, 32).

[90] Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on pp. 46, 66).

[91] Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. "Latent ODEs for Irregularly-Sampled Time Series". In: *Advances in Neural Information Processing Systems 33 (NeurIPS 2019)*. 2019. URL: http://arxiv.org/abs/1907.03907 (cit. on p. 41).

[92] Slawek Smyl. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting". In: *International Journal of Forecasting* 36.1 (July 2019), pp. 75–85. DOI: 10.1016/j.ijforecast.2019.03.017. URL: https://www.sciencedirect.com/science/article/abs/pii/S0169207019301153 (cit. on pp. 39, 53, 63).

[93] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. "Robust anomaly detection for multivariate time series through stochastic recurrent neural network". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2828–2837 (cit. on pp. 9, 16, 17).

[94] Tailai Wen and Roy Keyes. *Time Series Anomaly Detection Using Convolutional Neural Networks and Transfer Learning*. 2019. arXiv: 1905.13628 [cs.LG] (cit. on p. 64).

[95] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. "Graph wavenet for deep spatial-temporal graph modeling". In: *arXiv preprint arXiv:1906.00121* (2019) (cit. on p. 26).

[96] S. Zhou, L. Zhou, M. Mao, H. Tai, and Y. Wan. "An Optimized Heterogeneous Structure LSTM Network for Electricity Price Forecasting". In: *IEEE Access* 7 (2019), pp. 108161–108173. DOI: 10.1109/ACCESS.2019.2932999 (cit. on p. 45).

[97] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, et al. "Neural forecasting: Introduction and literature overview". In: *arXiv preprint arXiv:2004.10240* 6 (2020) (cit. on p. 1).

[98] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Bernie Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, Laurent Callot, and Tim Januschowski. "Neural forecasting: Introduction and literature overview". In: *Computing Research Repository* (2020). URL: http://arxiv.org/abs/2004.10240 (cit. on p. 63).

[99] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. "Spectral temporal graph neural network for multivariate time-series forecasting". In: *Advances in neural information processing systems* 33 (2020), pp. 17766–17778 (cit. on pp. 26, 32).

[100] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. "DDSP: Differentiable digital signal processing". In: *arXiv preprint arXiv:2001.04643* (2020) (cit. on p. 30).

[101] Stephen Gerry, Timothy Bonnici, Jacqueline Birks, Shona Kirtley, Pradeep S Virdee, Peter J Watkinson, and Gary S Collins. "Early warning scores for detecting deterioration in adult hospital patients: systematic review and critical appraisal of methodology". In: *bmj* 369 (2020) (cit. on p. 55).

[102] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The Efficient Transformer". In: *8th International Conference on Learning Representations, (ICLR 2020)*. 2020. URL: https://arxiv.org/abs/2001.04451 (cit. on pp. 46, 93, 96).

[103] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M4 Competition: 100,000 time series and 61 forecasting methods". In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 54–74. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.04.014. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301128 (cit. on pp. 4, 38, 62, 68).

[104] Cindy Marling and Razvan Bunescu. "The OhioT1DM Dataset for Blood Glucose Level Prediction: Update 2020". In: *CEUR Workshop Proc.* 2020 (cit. on p. 58).

[105] Richard McShinsky and Brandon Marshall. "Comparison of Forecasting Algorithms for Type 1 Diabetic Glucose Prediction on 30 and 60-Minute Prediction Horizons." In: *KDH@ ECAI*. 2020, pp. 12–18 (cit. on p. 55).

[106] Pablo Montero-Manso, George Athanasopoulos, Rob J. Hyndman, and Thiyanga S. Talagala. "FFORMA: Feature-based forecast model averaging". In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 86–92. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.02.011. URL: https://www.sciencedirect.com/science/article/pii/S0169207019300895 (cit. on p. 63).

[107] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *8th International Conference on Learning Representations, ICLR 2020*. 2020. URL: https://openreview.net/forum?id=r1ecqn4YwB (cit. on pp. 39, 41, 47, 48, 53, 63, 66, 96).

[108] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. "Learning latent space energy-based prior model". In: *arXiv preprint arXiv:2006.08205* (2020) (cit. on pp. 10, 26, 29).

[109] Harry Rubin-Falcone, Ian Fox, and Jenna Wiens. "Deep Residual Time-Series Forecasting: Application to Blood Glucose Prediction." In: *KDH@ ECAI*. 2020, pp. 105–109 (cit. on p. 55).

[110] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301888 (cit. on pp. 63, 66, 96).

[111] Lifeng Shen, Zhuocong Li, and James Kwok. "Timeseries anomaly detection using temporal hierarchical one-class network". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 13016–13026 (cit. on pp. 9, 17).

[112] Evangelos Spiliotis, Vassilios Assimakopoulos, and Spyros Makridakis. "Generalizing the Theta method for automatic forecasting". In: *European Journal of Operational Research* 284.2 (2020), pp. 550–558. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.01.007. URL: https://www.sciencedirect.com/science/article/pii/S0377221720300242 (cit. on p. 68).

[113] Jinyu Xie and Qian Wang. "Benchmarking Machine Learning Algorithms on Blood Glucose Prediction for Type I Diabetes in Comparison With Classical Time-Series Models". In: *IEEE Transactions on Biomedical Engineering* 67.11 (2020), pp. 3101–3124. DOI: 10.1109/TBME.2020.2975959 (cit. on pp. 55, 56, 58).

[114] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. "Multivariate time-series anomaly detection via graph attention network". In: *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2020, pp. 841–850 (cit. on pp. 17, 18).

[115] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting". In: *The Association for the Advancement of Artificial Intelligence Conference 2021 (AAAI 2021)*. abs/2012.07436 (2020). arXiv: 2012.07436. URL: https://arxiv.org/abs/2012.07436 (cit. on pp. 26, 31, 32, 40, 44, 46, 93, 95).

[116] Chris U Carmona, François-Xavier Aubet, Valentin Flunkert, and Jan Gasthaus. "Neural Contextual Anomaly Detection for Time Series". In: *arXiv preprint arXiv:2107.07702* (2021) (cit. on pp. 9, 17).

[117] Cristian Challu, Christian Poppeliers, Predrag Punoševac, and Artur Dubrawski. "Explosion discrimination using seismic gradiometry and spectral filtering of data". In: *Bulletin of the Seismological Society of America* 111.3 (2021), pp. 1365–1377 (cit. on p. 24).

[118] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. "Adarnn: Adaptive learning and forecasting of time series". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 402–411 (cit. on pp. 1, 3, 84).

[119] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. *Time-Series Representation Learning via Temporal and Contextual Contrasting*. 2021. arXiv: 2106.14112 [cs.LG]. URL: https://arxiv.org/abs/2106.14112 (cit. on p. 64).

[120] Albert Gu, Karan Goel, and Christopher Ré. "Efficiently modeling long sequences with structured state spaces". In: *arXiv preprint arXiv:2111.00396* (2021) (cit. on p. 73).

[121] Jesus Lago, Grzegorz Marcjasz, Bart De Schutter, and Rafał Weron. "Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark". In: *Applied Energy* 293 (2021), p. 116983. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2021.116983. URL: https://www.sciencedirect.com/science/article/pii/S0306261921004529 (cit. on pp. 50, 52, 53).

[122] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. "Temporal fusion transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4 (2021), pp. 1748–1764 (cit. on p. 77).

[123] Bryan Lim, Sercan O. Arık, Nicolas Loeff, and Tomas Pfister. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal*

of Forecasting 37.4 (2021), pp. 1748–1764. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0169207021000637 (cit. on pp. 62, 63, 66).

[124] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Predicting/hypothesizing the findings of the M5 competition". In: *International Journal of Forecasting* (2021). ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.09.014. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001631 (cit. on p. 62).

[125] E Nijkamp, B Pang, T Han, L Zhou, SC Zhu, and YN Wu. "Learning multi-layer latent variable model with short run MCMC inference dynamics". In: *European Conference on Computer Vision*. 2021 (cit. on pp. 8, 10, 11).

[126] Boris N. Oreshkin, Dimitri Carpov, Nicolas Chapados, and Yoshua Bengio. "Meta-Learning Framework with Applications to Zero-Shot Time-Series Forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10 (May 2021), pp. 9242–9250. DOI: 10.1609/aaai.v35i10.17115. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17115 (cit. on p. 64).

[127] Bo Pang, Erik Nijkamp, Tian Han, and Ying Nian Wu. "Generative Text Modeling through Short Run Inference". In: *arXiv preprint arXiv:2106.02513* (2021) (cit. on pp. 10, 26).

[128] Md Fazle Rabby, Yazhou Tu, Md Imran Hossen, Insup Lee, Anthony S Maida, and Xiali Hei. "Stacked LSTM based deep recurrent neural network with kalman smoothing for blood glucose prediction". In: *BMC Medical Informatics and Decision Making* 21 (2021), pp. 1–15 (cit. on p. 55).

[129] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE* (2021) (cit. on pp. 8, 9).

[130] Artemios-Anargyros Semenoglou, Evangelos Spiliotis, Spyros Makridakis, and Vassilios Assimakopoulos. "Investigating the accuracy of cross-learning time series forecasting methods". In: *International Journal of Forecasting* 37.3 (2021), pp. 1072–1084. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.11.009. URL: https://www.sciencedirect.com/science/article/pii/S0169207020301850 (cit. on p. 55).

[131] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. "CSDI: Conditional score-based diffusion models for probabilistic time series imputation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 24804–24816 (cit. on pp. 24, 26, 32, 34).

[132] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting". In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2021)*. Ed. by M. Ranzato, A. Beygelzime, P.S. Liang, J.W. Vaughan, and Y. Dauphin. 2021. URL: https://arxiv.org/abs/2106.13008 (cit. on pp. 26, 31, 32, 40, 44–47, 93, 95, 103).

[133] Renjie Wu and Eamonn J Keogh. "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress". In: *IEEE transactions on knowledge and data engineering* 35.3 (2021), pp. 2421–2429 (cit. on pp. 22, 23, 77).

[134] Syed Mohammed Arshad Zaidi, Varun Chandola, Muhanned Ibrahim, Bianca Romanski, Lucy D Mastrandrea, and Tarunraj Singh. "Multi-step ahead predictive model for blood

glucose concentrations of type-1 diabetic patients". In: *Scientific Reports* 11.1 (2021), p. 24332 (cit. on p. 56).

[135] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. "A Comprehensive Survey on Transfer Learning". In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555. URL: https://arxiv.org/abs/1911.02685 (cit. on pp. 62, 63).

[136] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. "Deep Learning for Time Series Forecasting: Tutorial and Literature Survey". In: *ACM Comput. Surv.* (2022). ISSN: 0360-0300. DOI: 10.1145/3533382. URL: https://doi.org/10.1145/3533382 (cit. on p. 24).

[137] Cristian Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. "Deep generative model with hierarchical latent factors for time series anomaly detection". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. PMLR. 2022, pp. 1643–1654 (cit. on p. 5).

[138] Cristian Challu, Christian Poppeliers, Predrag Punoševac, and Artur Dubrawski. "Explosion Discrimination Using Seismic Gradiometry and Spectrally Filtered Principal Components: Controlled Field Experiments". In: *Bulletin of the Seismological Society of America* 112.6 (2022), pp. 3141–3150 (cit. on p. 24).

[139] Dazhao Du, Bing Su, and Zhewei Wei. "Preformer: Predictive Transformer with Multi-Scale Segment-wise Correlations for Long-Term Time Series Forecasting". In: *Computing Research Repository* abs/2202.11356 (2022). URL: https://arxiv.org/abs/2202.11356 (cit. on p. 95).

[140] Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. *StatsForecast: Lightning Fast Forecasting with Statistical and Econometric Models*. PyCon Salt Lake City, Utah, US 2022. 2022. URL: https://github.com/Nixtla/statsforecast (cit. on pp. 66, 70).

[141] Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. "Unsupervised model selection for time-series anomaly detection". In: *arXiv preprint arXiv:2210.01078* (2022) (cit. on p. 77).

[142] Boris Ivanovic, James Harrison, and Marco Pavone. "Expanding the deployment envelope of behavior prediction via adaptive meta-learning". In: *arXiv preprint arXiv:2209.11820* (2022) (cit. on pp. 3, 84).

[143] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L. Winkler. "The M5 uncertainty competition: Results, findings and conclusions". In: *International Journal of Forecasting* 38.4 (2022). Special Issue: M5 competition, pp. 1365–1385. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.10.009. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001722 (cit. on p. 68).

[144] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *arXiv preprint arXiv:2211.14730* (2022) (cit. on p. 32).

[145] Kin G Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. "NeuralForecast: User friendly state-of-the-art neural forecasting models". In: *PyCon Salt Lake City, Utah, US* 2022 (2022), p. 6 (cit. on p. 76).

[146] Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafal Weron, and Artur Dubrawski. "Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx". In: *International Journal of Forecasting* 39.2 (2022), pp. 884–900 (cit. on pp. 5, 24).

[147] Harry Rubin-Falcone, Joyce M Lee, and Jenna Wiens. "Forecasting with Sparse but Informative Variables: A Case Study in Predicting Blood Glucose". In: *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA: ACM, 2022. URL: https://kdd-milets.github.io/milets2022/papers/MILETS_2022_paper_0765.pdf (cit. on pp. 55, 56, 58).

[148] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan. "Differentiable Wavetable Synthesis". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 4598–4602 (cit. on p. 30).

[149] Fan-Keng Sun and Duane S Boning. "Fredo: frequency domain-based long-term time series forecasting". In: *arXiv preprint arXiv:2205.12301* (2022) (cit. on p. 39).

[150] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. "Etsformer: Exponential smoothing transformers for time-series forecasting". In: *arXiv preprint arXiv:2202.01381* (2022) (cit. on pp. 39, 40).

[151] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven C. H. Hoi. "ETSformer: Exponential Smoothing Transformers for Time-series Forecasting". In: *Computing Research Repository* abs/2202.01381 (2022). arXiv: 2202.01381. URL: https://arxiv.org/abs/2202.01381 (cit. on p. 95).

[152] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. "Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=LzQQ89U1qm_ (cit. on p. 21).

[153] Yi Xu, Lichen Wang, Yizhou Wang, and Yun Fu. "Adaptive Trajectory Prediction via Transferable GNN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 6520–6531 (cit. on p. 3).

[154] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. "Are transformers effective for time series forecasting?" In: *arXiv preprint arXiv:2205.13504* (2022) (cit. on p. 38).

[155] Tian Zhou, Ziqing Ma, Qingsong Wen, Liang Sun, Tao Yao, Rong Jin, et al. "Film: Frequency improved legendre memory model for long-term time series forecasting". In: *arXiv preprint arXiv:2205.08897* (2022) (cit. on p. 38).

[156] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27268–27286 (cit. on pp. 32, 39, 40, 46).

[157] Cristian Challu, Peihong Jiang, Ying Nian Wu, and Laurent Callot. "SpectraNet: Multivariate Forecasting and Imputation under Distribution Shifts and Missing Data". In: *ML4IoT Workshop at ICLR, Oral*. 2023 (cit. on p. 5).

[158] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. "Nhits: Neural hierarchical interpolation for time series forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 6. 2023, pp. 6989–6997 (cit. on pp. 5, 31, 32, 35, 66, 83).

[159] Azul Garza and Max Mergenthaler-Canseco. "TimeGPT-1". In: *arXiv preprint arXiv:2310.03589* (2023) (cit. on pp. 72, 78).

[160] Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. "Unsupervised Model Selection for Time-series Anomaly Detection". In: *11th International Conference on Learning Representations, ICLR 2023, Oral*. 2023 (cit. on p. 21).

[161] Albert Gu and Tri Dao. "Mamba: Linear-time sequence modeling with selective state spaces". In: *arXiv preprint arXiv:2312.00752* (2023) (cit. on p. 73).

[162] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. arXiv: 2211.14730 [cs.LG] (cit. on p. 66).

[163] Willa Potosnak, Cristian Challu, Kin G Olivares, and Artur Dubrawski. "Forecasting Response to Treatment with Deep Learning and Pharmacokinetic Priors". In: *Findings Track at ML4H* (2023) (cit. on p. 5).

[164] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Vincent Hassen, Anderson Schneider, et al. "Lag-llama: Towards foundation models for time series forecasting". In: *arXiv preprint arXiv:2310.08278* (2023) (cit. on p. 78).

[165] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. "Are transformers effective for time series forecasting?" In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 9. 2023, pp. 11121–11128 (cit. on p. 35).

[166] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. "Chronos: Learning the language of time series". In: *arXiv preprint arXiv:2403.07815* (2024) (cit. on p. 72).

[167] Cristian Challu, Mononito Goswami, Peihong Jiang, and Laurent Callot. "Automatic Thresholding with Agnostic Synthetic Anomalies". In: *Work in Progress. To be submitted*. 2024 (cit. on p. 5).

[168] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. "MOMENT: A Family of Open Time-series Foundation Models". In: *arXiv preprint arXiv:2402.03885* (2024) (cit. on p. 72).

[169] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. "Foundation Models for Time Series Analysis: A Tutorial and Survey". In: *arXiv preprint arXiv:2403.14735* (2024) (cit. on p. 72).

[170] Kin G Olivares, Cristian Challu, Azul Garza, Max Mergenthaler Canseco, and Artur Dubrawski. "Transferability of Neural Forecast Methods". In: *IIF-SAS Award to be submitted at International Journal of Forecasting* (2024) (cit. on p. 5).

[171] Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Han Zhao, Daling Wang, and Yifei Zhang. "Is Mamba Effective for Time Series Forecasting?" In: *arXiv preprint arXiv:2403.11144* (2024) (cit. on p. 73).

[172] James Arthur Harris and Francis Gano Benedict. *A biometric study of basal metabolism in man*. Carnegie institution of Washington, 21919 (cit. on p. 58).