Faculdade de Engenharia da Universidade do Porto

Integrating N-BEATS with a Mixture-of-Experts Layer for Time Series Forecasting

Ricardo André Araújo de Matos



Master in Informatics and Computing Engineering

Supervisor: Vítor Cerqueira

Second Supervisor: Luís Roque

July 22, 2025

© Ricardo André Araújo de Matos, 2025

Integrating N-BEATS with a Mixture-of-Experts Layer for Time Series Forecasting

Ricardo André Araújo de Matos

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

President: Henrique Lopes Cardoso
Referee: Bruno Veloso
Supervisor: Vítor Cerqueira

July 22, 2025

Abstract

Time series forecasting plays a crucial role in numerous real-world applications. While traditional statistical methods have long been the standard, deep neural networks are increasingly demonstrating strong performance on benchmark datasets and competitions.

Deep learning approaches have shown particular success in forecasting problems involving multiple time series. These models can effectively learn patterns not only across time but also across collections of time series. In real-world forecasting scenarios, datasets often contain heterogeneous time series with varying characteristics such as distinct trend or seasonal patterns, or varying noise and scale levels. This heterogeneity motivates research into new neural architectures that can effectively handle such diversity. Additionally, having interpretable models that can explain their predictions is also a desirable property.

A notable state-of-the-art architecture in deep learning-based time series forecasting is `N-BEATS` (Neural Basis Expansion Analysis for Time Series). In its interpretable configuration, it is composed of trend and seasonal stacks, each formed by sequentially connected blocks linked via a residual structure. These blocks project the time series onto basis functions modelling the respective components, allowing the model to produce forecasts that are decomposable and, thus, more interpretable.

In this work, we present `N-BEATS-MOE`, an extension of `N-BEATS` based on a Mixture-of-Experts (MoE) layer. `N-BEATS-MOE` employs a dynamic block weighting strategy based on a gating network which allows the model to better adapt to the characteristics of each time series. We also hypothesize that the gating mechanism provides additional interpretability by identifying which expert is most relevant for each series.

We evaluate the proposed approach, and some variations, by conducting experiments on 12 benchmark datasets comprising a total of 100.141 time series. The results indicate that `N-BEATS-MOE` demonstrates competitive performance with N-BEATS, particularly on datasets composed of heterogeneous time series. Moreover, a decomposition-based analysis of the results provided insights into the behaviour of the gating mechanism.

Keywords: Time series, Forecasting, Mixture-of-Experts, `N-BEATS`

Resumo

A previsão de séries temporais é crucial em diversas aplicações práticas. Os métodos estatísticos convencionais, que se mantiveram como o paradigma dominante por um extenso período, têm sido superados pelas redes neurais profundas, que têm apresentado um desempenho cada vez mais notável em conjuntos de dados e competições de referência.

As abordagens baseadas em aprendizagem profunda têm revelado um desempenho particularmente eficaz em problemas de previsão que envolvem múltiplas séries temporais. Estes modelos não só conseguem aprender padrões ao longo do tempo, como também entre diferentes séries temporais. Na prática, os conjuntos de dados de previsão frequentemente contêm séries temporais heterogéneas, com características variadas, como diferentes tendências, padrões sazonais, níveis de ruído e escalas distintas. Esta heterogeneidade motiva a investigação de novas arquiteturas neurais que consigam lidar de forma eficaz com tal diversidade. Além disso, é desejável que os modelos sejam interpretáveis, ou seja, capazes de explicar as suas previsões.

Uma arquitetura de ponta notável na previsão de séries temporais com aprendizagem profunda é o `N-BEATS` (Neural Basis Expansion Analysis for Time Series). Na sua configuração interpretável, é composto por stacks de tendência e sazonalidade, cada uma formada por blocos sequencialmente ligados por via de uma estrutura de resíduos. Estes blocos são responsáveis por projetar as séries em funções base que modelam esses componentes, permitindo gerar previsões decomponíveis e, por conseguinte, mais interpretáveis.

Neste trabalho, apresentamos o `N-BEATS-MOE`, uma extensão do `N-BEATS` baseada numa camada Mixture-of-Experts (MoE). O `N-BEATS-MOE` recorre a uma estratégia dinâmica de ponderação dos blocos, fundamentada numa rede de gating, que possibilita ao modelo adaptar-se de forma mais precisa às características de cada série temporal. Além disso, propõe-se a hipótese de que o mecanismo de gating proporciona uma interpretabilidade adicional, ao identificar qual bloco é mais pertinente para cada série.

Procedeu-se à avaliação da abordagem proposta e de algumas das suas variações, por meio de experimentos em 12 conjuntos de dados de referência, num total de 100.141 séries temporais. Os resultados obtidos indicam que o `N-BEATS-MOE` apresenta um desempenho competitivo com o `N-BEATS`, especialmente em conjuntos de dados compostos por séries temporais heterogéneas. Adicionalmente, uma análise à decomposição da previsão do modelo forneceu uma compreensão aprofundada sobre o comportamento do mecanismo de gating.

Palavras-chave: Séries temporais, Previsão, Mistura de Especialistas, `N-BEATS`

Funding

This work is a result of Agenda “Center for Responsible AI”, nr. C645008882-00000055, investment project nr. 62, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU. Funded by the European Union – NextGenerationEU. Funded by the European Union under the Horizon Europe Framework Programme Grant Agreement Nº: 101095387. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Health and Digital Executive Agency can be held responsible for them; UID/00027 of the LIACC - Artificial Intelligence and Computer Science Laboratory - funded by Fundação para a Ciência e a Tecnologia, I.P./ MCTES through the national funds.

Ricardo de Matos

Acknowledgements

First and foremost, heartfelt thanks and warm appreciation to Vítor and Luís for their guidance, insightful feedback, and ideas that were invaluable to me throughout this work.
This marks the end of a long and grateful journey. Looking back, it's hard to believe just how much progress was made and how much effort it took. I'm truly thankful to have had this opportunity.
I would like to express my sincere gratitude to my homeland, Portugal, and to all its people for giving me this opportunity. In a world that is ever-changing and often unforgiving, I feel incredibly fortunate to have been born in a country where hard work and determination still open doors.
To my family **... beijinhos lá para casa ...** for supporting, guiding, and educating me throughout my life.
To my friends, Pedro, Bárbara, Carolina, José, Mariana, **... os velhinhos ...** João, Luís, Eva, António, Rubén, and Marco **... da universidade para a vida ...** for being there through my happiest moments and toughest times.
Thank you all!

Ricardo de Matos

"Long enough have you dream'd contemptible dreams,
Now I wash the gum from your eyes,
You must habit yourself to the dazzle of the light
and of every moment of your life."

Walt Whitman

Contents

Abbreviations x

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Motivation . . . 2
- 1.3 Objectives and Research Questions . . . 3
- 1.4 Contributions . . . 3
- 1.5 Document Outline . . . 3

2 Background and State-of-the-Art 5
- 2.1 Time Series . . . 5
- 2.2 Time Series Forecasting . . . 6
  - 2.2.1 Definition and Challenges . . . 6
  - 2.2.2 Classical Forecasting Methods . . . 8
  - 2.2.3 Evaluating Forecasts . . . 9
- 2.3 Deep Learning . . . 9
  - 2.3.1 Neural Networks . . . 10
  - 2.3.2 Training Neural networks . . . 10
  - 2.3.3 Deep learning for Forecasting . . . 11
- 2.4 Mixture of Experts . . . 12
  - 2.4.1 Background on MoE . . . 13
  - 2.4.2 Taxonomy of MoE . . . 14
  - 2.4.3 Scaling LLMs with MoE architectures . . . 16
  - 2.4.4 Considerations on MoE for TSF . . . 17
  - 2.4.5 Considerations on the routing collapse problem . . . 18

3 Methodology 19
- 3.1 Problem Definition . . . 19
- 3.2 Research Hypothesis . . . 20
- 3.3 Approach . . . 21
- 3.4 Method Variants . . . 22

4 Experiments 25
- 4.1 Experimental Design . . . 25
  - 4.1.1 Datasets and Evaluation . . . 25
  - 4.1.2 Training Procedure . . . 26
  - 4.1.3 Hyperparameter Configuration . . . 27
  - 4.1.4 Experimental Protocol . . . 27
- 4.2 Results . . . 28
- 4.3 Expert Specialization Analysis . . . 29
  - 4.3.1 Specialization Analysis via STL Decomposition . . . 29
  - 4.3.2 Forecast Decomposition in the M1 dataset . . . 31
- 4.4 Aspect-based Analysis . . . 31
- 4.5 Discussion . . . 33

4.6 Threats to Validity 35

5 Conclusion **37**
5.1 Research Questions 37
5.2 Main Contributions 38
5.3 Future Work 38

References **40**

A Hyperparameter Sensitivity and Findings **48**

B Forecast on Aspect-based Series **50**

List of Figures

2.1 A univariate time series . . . 6
2.2 A multivariate time series . . . 7
2.3 Neural networks as Autoregressive models with non-linear hidden layers . . . 11
2.4 Architecture of the `N-BEATS` interpretable configuration . . . 13
2.5 Training step in a Mixture of MLP-experts . . . 14
2.6 Inference in dense Mixture-of-Experts . . . 15
2.7 A sparse Moe consisting of 3 experts . . . 16

3.1 The proposed architecture, `N-BEATS-MOE` . . . 22
3.2 `N-BEATS-MOE` variations . . . 24

4.1 Experimental setup to analyze specialization in `N-BEATS-MOE`. . . . 30
4.2 Forecast decomposition in the M1 monthly dataset . . . 32
4.3 Aspect-based Evaluation . . . 34

B.1 Aspect-based forecast on the M1 monthly dataset . . . 50
B.2 Aspect-based forecast on the M1 quarterly dataset . . . 51
B.3 Aspect-based forecast on the M1 yearly dataset . . . 51
B.4 Aspect-based forecast on the Tourism monthly dataset . . . 52
B.5 Aspect-based forecast on the Tourism quarterly dataset . . . 52
B.6 Aspect-based forecast on the Tourism yearly dataset . . . 53
B.7 Aspect-based forecast on the M3 monthly dataset . . . 53
B.8 Aspect-based forecast on the M3 quarterly dataset . . . 54
B.9 Aspect-based forecast on the M3 yearly dataset . . . 54
B.10 Aspect-based forecast on the M4 monthly dataset . . . 55
B.11 Aspect-based forecast on the M4 quarterly dataset . . . 55
B.12 Aspect-based forecast on the M4 yearly dataset . . . 56

List of Tables

2.1 Forecasting evaluation metrics 10

4.1 Summary of datasets by frequency 26
4.2 Hyperparameter search space configuration used in experiments 28
4.3 Experiment results 28
4.4 Expert selection ratios for STL decomposition components 31
4.5 Mean Rank across aspect-evaluation metric 33

A.1 Optimized Hyperparameters 49

Abbreviations

ADAM Adaptive Moment Estimation. 27

AI Artificial Intelligence. 1, 9

AR Autoregressive Model. 8

ARIMA Autoregressive Integrated Moving Average Model. 9

ARMA Autoregressive Moving Average Model. 8, 9

CVar Conditional Value at Risk. 32

DL Deep Learning. 9–11

FC Fully Connected. 12, 22–24

FNN Feed-Forward Neural Network. 27

GFM Global Forecasting Model. 1, 2, 7, 20

GN Gating Network. 13, 14

LLM Large Language Model. 2, 13, 16, 17, 20, 33, 39

LSTM Long Short-Term Memories. 11, 12

MA Moving Average Model. 8

MAE Mean Absolute Error. 10, 26

MAPE Mean Absolute Percentage Error. 10

MASE Mean Absolute Scaled Error. 10

MELE Mixture of Explicitly Localized Experts. 15

MILE Mixture of Implicitly Localized Experts. 15

ML Machine Learning. 9, 11

MLP Multilayer Perceptron. 11

MoE Mixture of Experts. 2, 3, 12–18, 20–23, 27, 33, 34, 38, 39

MSE Mean Square Error. 10

N-BEATS Neural Basis Expansion Analysis for Time Series. 2–4, 11, 12, 20, 22, 24, 27, 29, 31, 33, 36, 38, 39, 48

N-BEATSx Neural Basis Expansion Analysis with Exogenous. 12

N-HITS Neural Hierarchical Interpolation for Time Series Forecasting. 12

NN Neural Network. 10, 11, 17

OOS Out-of-Sample. 9

ReLU Rectified Linear Unit. 23

RMSE Root Mean Squared Error. 10

SARIMA Seasonal ARIMA. 9

SARIMAX Seasonal Autoregressive Integrated Moving-Average with Exogenous Variables. 9

SES Simplest Exponential Smoothing. 9

sMAPE Symmetric Mean Absolute Percentage Error. 9, 10, 26, 28, 29, 31, 32

STL Seasonal-Trend decomposition using LOESS. 29, 30, 35

TCN Temporal Convolution Network. 12

TFT Temporal Fusion Transformers. 12

TPE Tree-structured Parzen Estimator. 27

TS Time Series. 5, 7

TSF Time Series Forecasting. 1, 7, 9, 10, 18, 20, 38

Chapter 1

Introduction

This chapter presents an overview of the context and motivation behind this thesis. Section 1.1 offers a concise introduction to the forecasting problem. Section 1.2 elaborates on the rationale driving our research. Section 1.3 delineates the primary objectives and research questions of the study. Lastly, section 1.5 outlines the structure and organization of the document.

1.1 Context 1
1.2 Motivation 2
1.3 Objectives and Research Questions 3
1.4 Contributions 3
1.5 Document Outline 3

1.1 Context

Artificial Intelligence (AI) is set to play a transformative role in the global economy, with the potential to contribute an additional $13 trillion to global GDP by 2030 [6]. Its ability to automate complex tasks, analyse vast amounts of data and improve decision-making has far-reaching implications for sectors ranging from healthcare to finance and manufacturing. By improving productivity and driving innovation, AI is expected to reshape industries, optimise resource use, and create new economic opportunities, making it a key driver of technological progress and economic growth in the coming decades.

One of the most notable tasks in AI is Time Series Forecasting (TSF). A time series denotes a collection of time-ordered observations. This data type can be used to represent various phenomena in domains such as finance [16], healthcare [65], energy [67], and retail [41]. TSF involves using historical time series data to predict the values of future observations. While traditional statistical methods such as ARIMA [31] have long been the standard approach, deep neural networks are increasingly showing their effectiveness in benchmark datasets and competitions [73].

When tackling forecasting problems, deep neural networks are typically trained on collections of time series. The hypothesis is that, by integrating disparate time series datasets, it will be possible to discern common patterns and intricate relationships [28]. Ultimately, this leads to better forecasts. These types of models are called Global Forecasting Models (GFMs), and contrast with local models that fit each time series individually (e.g. ARIMA [31]).

However, in real-world forecasting scenarios, datasets often consist of heterogeneous time series that display varying characteristics, such as distinct trend and seasonal patterns, as well as differences in noise levels and scale [57]. This diversity within the data can pose significant challenges to traditional deep learning models, which may struggle to capture the unique characteristics of each time series within a heterogeneous dataset and make incorrect generalizations. Montero-Manso and Hyndman [64] show that regardless of the degree of heterogeneity in a given dataset, there exists a GFM that can perform with equal or greater effectiveness than the collection of local models. Furthermore, one of the key factors contributing to the performance and complexity of the model is its efficiency in the partitioning of the dataset. Despite the advances in deep learning approaches for time series forecasting, opportunities remain for improving how GFM handle diverse real-world time series datasets [57].

1.2 Motivation

The challenges of real-world time series forecasting, particularly when dealing with multiple and heterogeneous time series, often expose the limitations of traditional models in capturing diverse patterns such as varying trends, seasonalities, noise levels, and scales.

In machine learning, a common approach to handle datasets with heterogeneous patterns is the Mixture of Experts (MoE) framework. This method involves training multiple specialized models, or "experts", where each is tailored to handle specific patterns or characteristics in the data. MoE also include a gating network that dynamically determines the most appropriate expert for each input. While MoE approaches have been around for a while [33, 59], they have gained popularity in the last few years, particularly in the context of Large Language Models (LLMs) [18, 35, 22], where they improve model performance and reduce inference cost, especially when working with large heterogeneous datasets [66, 95].

A notable state-of-the-art architecture in deep learning-based time series forecasting is Neural Basis Expansion Analysis for Time Series (N-BEATS) [73, 1]. N-BEATS has been shown to outperform various neural-based time series forecasting approaches, including architectures based on transformers or recurrent neural networks [15]. In its interpretable configuration, N-BEATS comprises trend and seasonal stacks, each consisting of sequentially connected blocks integrated through a residual structure. These blocks compute basis coefficients, which are then used to project the time series onto predefined basis functions that model the respective components. This architecture enables the generation of forecasts that are decomposable and, consequently, more interpretable. N-BEATS has been effectively applied to datasets involving multiple time series [73, 15, 58].

Motivated by the need for models that can both adapt to heterogeneous time series and remain interpretable, we propose extending the N-BEATS architecture with a MoE framework. Our hypothesis is that this integration will allow different components of the model to specialize in distinct structural patterns (e.g., different types of seasonality or trend), enhancing the model's flexibility and generalization capability. In this way, each expert can focus on distinct time series patterns, with the gating mechanism learning how to best combine their outputs at inference time.

The successful development of this research will have practical implications for time series forecasting in heterogeneous datasets. For instance, in organizations that operate across different sectors or geographical regions, the ability to handle and adapt to diverse time series patterns could significantly enhance planning and resource allocation without having to train and maintain

multiple models. Overall, this adaptability could lead to more efficient and informed decision-making processes in areas such as supply chain management, financial planning, and resource allocation.

1.3 Objectives and Research Questions

The primary objective of this research is to extend the N-BEATS architecture by integrating a MoE mechanism, with the goal of improving its performance on datasets composed of heterogeneous univariate time series. This objective is addressed through the following three key research questions:

1. How can we augment the `N-BEATS` architecture with a MoE approach to better accommodate the *heterogeneous patterns* (e.g., varying trends and seasonal profiles) inherent in datasets comprising multiple univariate time series?
2. To what extent do `N-BEATS` MoE-augmented approaches achieve better performance in terms of forecast accuracy?
3. How does the inclusion of a MoE layer improve the interpretability of `N-BEATS`?

The responses to these research questions will contribute to a deeper understanding of how MoEs approaches can be applied in the context of univariate time series forecasting, particularly in datasets composed of heterogeneous time series [57, 56].

1.4 Contributions

We address the research questions by proposing `N-BEATS-MOE`, an extension of N-BEATS that features a MoE layer. According to the proposed method, the blocks in the N-BEATS stacks act as experts, with a gating network controlling their combination. We validate the proposed method on a set of 12 benchmark datasets, and show that it performs competitively with N-BEATS, especially in datasets composed with heterogeneous time series. In summary, the main contributions of this work are:

- A novel architecture combining N-BEATS with MoE for improved handling of heterogeneous time series.
- A set of experiments[1] that show that N-BEATS-MOE performs competitively with N-BEATS. These also include an analysis of the behaviour of the gating network in an attempt to understand the behaviour of the introduced MoE layer.

1.5 Document Outline

This dissertation is organized into five chapters. Chapter 2 reviews the background and related work, covering time series, forecasting methods, deep learning, and MoE, including their associated challenges. This chapter focuses on presenting the fundamentals of the task, summarizing the current state-of-the-art approaches, and discussing related work.

[1] All implementations are available at `https://github.com/zaai-ai/mixture_of_experts_time_series/tree/main/models`.

Chapter 3 outlines the problem statement, formulates the research hypotheses, and details the methodology applied. It restates the problem addressed in this dissertation and provides an overview of the approach, and its variants, adopted to solve it.

Chapter 4 describes the experimental setup, presents the results, and analyses expert specialization and forecasting aspects, concluding with a discussion and considerations on validity. We present the datasets, evaluation metrics, training procedure, hyperparameter configuration, and experimental protocol. This is followed by an analysis of expert specialization along two dimensions: its alignment with standard series decompositions and its impact as demonstrated in a case study on the M1 dataset. Next, an aspect-based analysis is conducted to identify where our model achieves improvements over N-BEATS. The chapter concludes with a discussion of all results and potential threats to validity.

Chapter 5 summarizes the research contributions, answers the research questions, and outlines directions for future work. Additional materials are included in the appendices, including the final hyperparameter settings and examples of forecasts across different datasets and aspects.

Chapter 2

Background and State-of-the-Art

In this section, an overview of the main topics of this thesis is provided. Section 2.1 presents a definition of the key concepts in time series. Section 2.2 addresses the problem of time series forecasting, focusing on the framing and definition of the problem, as well as classical forecasting methods and evaluation techniques. Section 2.3 offers a brief introduction to deep learning, highlighting notable deep learning architectures used for time series forecasting. Finally, Section 2.4 introduces the Mixture of Experts framework, with a discussion on relevant related works.

2.1 Time Series 5
2.2 Time Series Forecasting 6
2.3 Deep Learning 9
2.4 Mixture of Experts 12

2.1 Time Series

A Time Series (TS) is a sequence of data points indexed in time order, typically measured at uniform intervals. Formally, it can be represented as $TS = y_1, y_2, y_3, ...y_t$, where t denotes increasingly chronological time steps.

A univariate time series is characterized by having a single variable at each time step. It opposes to multivariate time series which are characterized for having more than one variable per each time step, for example $y_t = (a_t, b_t)$ at each time step, where a_t and b_t represent distinct time series variables. Figures 2.1 and 2.2 illustrate examples of univariate and multivariate time series, respectively. The former shows temperature alone over time, while the latter includes temperature, humidity, and pressure evolving together.

In many scenarios, datasets are composed of multiple time series $\{TS^1, \ldots, TS^Z\}$, where Z is the number of time series in the dataset. For example, retail organizations typically aim to forecast the sales of various products simultaneously, where each product is represented as a time series. In this work, we focus on datasets composed of multiple univariate time series.

A time series can be conceptualized as a combination of multiple underlying components, each of which contributes to the formation of the overall pattern:

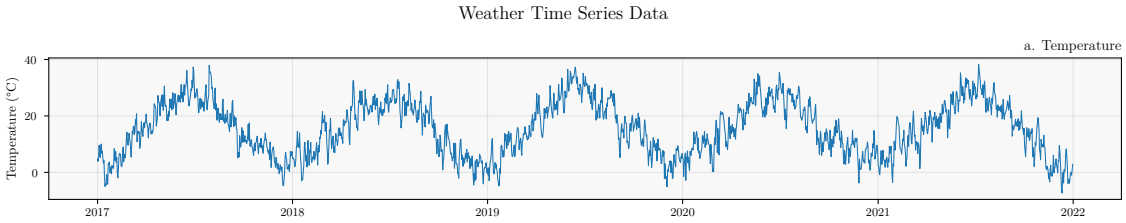


Figure 2.1: An example of a univariate time series of temperature, measured in degrees Celsius.

- **Trend:** represents the long-term, persistent movement or direction in the data level over time. It captures the overall increase, decrease, or stability in the time series, reflecting broad changes that occur gradually rather than in short bursts.
- **Seasonality:** occur when the time series data is influenced by recurring seasonal patterns that are tied to specific, fixed intervals (e.g. monthly).
- **Cyclic patterns:** occur when the time series data exhibits repeating patterns but without a regular interval.
- **Noise or Remainder:** refers to the random variation present in time series data. It reflects the inexplicable or irregular fluctuations that do not conform to the trend, seasonality or cyclical patterns.

Considering this, the series can be decomposed as follows:

$$y_t = S_t + T_t + C_t + R_t \tag{2.1}$$

Where y_t represents the observed data, S_t is the seasonal component, T_t the trend component, C_t the cycle component, and R_t the remainder component, all at time t. Other forms of decomposition, including the multiplicative and mixed decomposition, have their own set of advantages and disadvantages [70].

2.2 Time Series Forecasting

This section provides an overview of the forecasting task. Section 2.2.1 offers a formal definition of the forecasting problem, accompanied by a discussion of the challenges encountered in forecasting. Section 2.2.2 introduces a range of classical methods commonly used in forecasting. Finally, Section 2.2.3 examines various evaluation methods and metrics used to assess the accuracy and effectiveness of forecasts.

2.2.1 Definition and Challenges

Time Series Forecasting plays a crucial role in various domains, including business [24], financial markets [4], weather prediction [40], and industrial manufacturing [62].

Univariate time series forecasting is the process of predicting future values in a single-variable time series. Formally, given a historical time series $\mathbf{X} = \{y_i \mid i = 0, \ldots, t\}$, where $\mathbf{X} \in \mathbb{R}^{1 \times t}$ and t denote the number of historical timestamps, the objective is to learn a function f such that $f(\mathbf{X}) = \mathbf{Y}$ with $\mathbf{Y} = \{y_i \mid i = t+1, \ldots, h\} \in \mathbb{R}^{1 \times h}$, where h is the prediction horizon (i.e., the

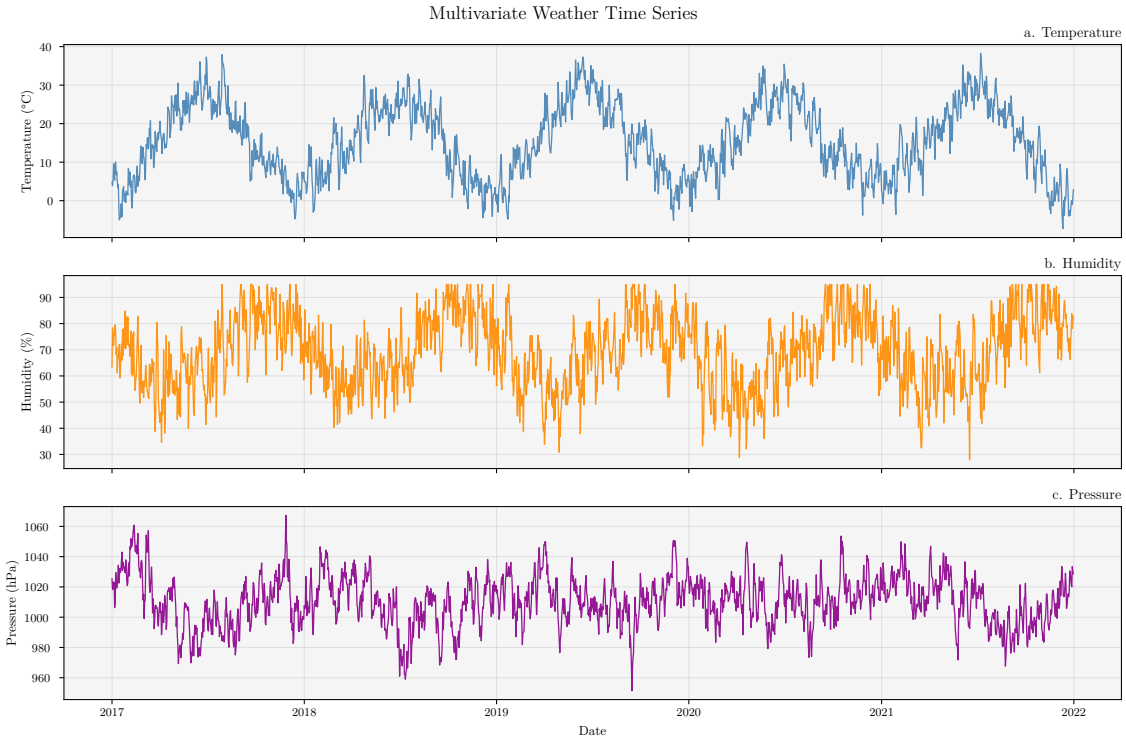


Figure 2.2: An example of a multivariate time series, where three variables, namely temperature, humidity, and pressure, are modelled together.

number of future timestamps to forecast). The goal is to find the function f that minimizes a loss function $loss(\mathbf{Y}, \hat{\mathbf{Y}})$, where $\hat{\mathbf{Y}} \in \mathbb{R}^{1 \times h}$ represents the forecasted values.

For clarity and throughout this dissertation, in some models we make the distinction between the input series, $x_0 \dots x_t$ that serve as input to the model, and the expected forecast values, $y_{t+1} \dots y_{t+h}$. When using a hat, we refer to the model forecast $\hat{y}$ or backcast $\hat{x}$, how the model predicts the future will be and how it comprehends the past.

Forecasts with a horizon of one, $h = 1$, referred to as one-step ahead forecasts. Conversely, forecasts with a horizon of $h = n$ are termed n-step ahead forecasts. Similarly, it is common to distinguish between short-term forecasting, where the forecast horizon is relatively short, and long-term forecasting for longer horizons [31].

When dealing with TSF, a model can be trained using two distinct approaches: local methods, which model each time series in the dataset independently, and global methods, which model each time series jointly [34]. Hybrids approaches, [81, 91, 89], try to leverage the specialization of local methods with the generalization of GFMs.

TSF faces several challenges, including the non-stationarity of real-world data, meaning the data can change in unpredictable ways over time, and the difficulty of distinguishing true causality from correlations driven by coincidence or hidden variables [43]. The difficulty in capturing diverse patterns across heterogeneous time series datasets [52]. And also, due to the inherent nature of TS data, it is susceptible to noise, such as that introduced by sensors [29], or inconsistencies in the data collection process such as missing and duplicated data [17].

2.2.2 Classical Forecasting Methods

The term "classical forecasting method" is typically associated with statistical models [34]. The following section presents a selection of classical forecasting methods.

Naïve A naïve forecast assumes that the value at the next time step is equal to the most recent observed value:

$$\hat{y}_{t+1} = y_t, \tag{2.2}$$

where $\hat{y}_{t+1}$ denotes the forecast for time $t+1$, and y_t is the observed value at time t. This simple model, also known as a random walk forecast, has been extensively applied in financial contexts, such as exchange rates [42, 61] and stock prices [8], where random walk tests are conducted to determine whether future movements are unpredictable and independent of past behaviour.

Another example of a naïve method is the seasonal naïve forecast, which assumes that the value at the next time step will be equal to the value from the same season in the previous cycle. Formally, for a seasonal period m, the forecast is given by:

$$\hat{y}_{t+h} = y_{t+h-m(s+1)}, \tag{2.3}$$

where h is the forecast horizon, m is the length of the seasonal cycle, and s is the number of complete seasonal cycles between the last observed value and the forecast period.

Autoregressive Models (AR) A model where the current value is a linear combination of its past values, a regression of the variable itself. An Autoregressive Model (AR), of order p, can be formally defined as:

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t \tag{2.4}$$

where $\varphi_1, ..., \varphi_p$ are the parameters of the model, c a constant, and ε_t is white noise [31]. Also referred as $\mathbf{AR(p)}$, an autoregressive model of order p. AR models are frequently used in time series analysis [104, 107].

Moving Average (MA) Model A model that expresses the current value of a time series as a linear combination of past random errors. Thus, $\mathbf{MA(q)}$:

$$y_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \tag{2.5}$$

where $\theta_1, ..., \theta_q$ are the parameters of the model, μ is the mean of the series, and $\varepsilon_t, \varepsilon_{t-1}, \ldots, \varepsilon_{t-q}$ are the error terms [31]. Moving Average Model (MA) models are also used for analysis and forecasting, with applications such as the prediction of tourist numbers [32], energy consumption [77], and more [10, 44, 104].

Autoregressive Moving Average Models (ARMA) A commonly used model in time series analysis that combines both AR and MA components to capture a wider range of temporal dependencies [104]. An Autoregressive Moving Average Model (ARMA) of order (p, q) is denoted as $\mathbf{ARMA(p, q)}$, where p is the **AR** order and q the **MA** order. Formally:

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.6}$$

Alternative methodologies include Autoregressive Integrated Moving Average Models (ARIMAs), which introduces differencing in ARMA to address non-stationary data [9]; Seasonal ARIMA (SARIMA), which extends the aforementioned methodology and incorporates seasonal components [74]; and Seasonal Autoregressive Integrated Moving-Average with Exogenous Variables (SARIMAX), which further extends the SARIMA and allows for the introduction of exogenous variables, facilitating the utilization of external variables [2].

Exponential Smoothing Models A set of models that are weighted averages of past observations, with the weights decaying exponentially as the observations become less recent. The Simplest Exponential Smoothing (SES) method can be defined as:

$$\hat{y}_{t+1|t} = \alpha y_t + (1-\alpha) \cdot \hat{y}_{t|t-1} \tag{2.7}$$

where $0 < \alpha < 1$ is a smoothing factor. An increase in the value of α will result in a reduction in the influence of older values on the prediction of $t+1$. Decreasing α will have the opposite effect.

Holt's Linear Trend method extends the SES to include a trend component [30]. The Holt-Winters Seasonal method adds a seasonal component to the latter to handle data with both trend and seasonality [100, 30]. Several other exponential smoothing variants also exist [25, 26].

2.2.3 Evaluating Forecasts

The evaluation of forecasts is of paramount importance for the assessment of a given model's performance and its relative positioning in comparison to other benchmark models.

Evaluation in TSF usually follows the same process of common Machine Learning (ML) algorithms. First, it is essential to partition the data in validation and test sets, the Out-of-Sample (OOS) evaluation. However, in contrast to conventional tasks, time series evaluation necessitates the consideration of the temporal sequence of data, thereby ensuring that the training set is positioned prior to the validation and test sets [14]. This approach is undertaken to prevent data leakage and overfitting.

Once the predictions have been generated by the models, the subsequent step is to compute the errors associated with these predictions. A variety of metrics are used to evaluate point forecasts, which can be classified into distinct categories, including scale-dependent and scale-independent metrics, percentage errors, and relative metrics. Table 2.1 presents a selection of the most pertinent metrics in the context of TSF. In this thesis, special attention is given to the Symmetric Mean Absolute Percentage Error (sMAPE) metric, which played a central role in the evaluation framework of the M4 competition.

2.3 Deep Learning

Deep Learning (DL) is a subfield of AI and ML that tackles problems in many areas, such as healthcare [53, 63, 85], generative ai [85, 36], tasks in natural language processing [21, 35], agriculture [38], and many more. Unlike traditional machine learning methods that often rely on

Table 2.1: Forecasting evaluation metrics

Category	**Name of Metric**	**Formula**
Scale-Dependent	Mean Absolute Error (MAE)	$\frac{1}{H}\sum_{\tau=t+1}^{t+H}\lvert y_\tau - \hat{y}_\tau\rvert$
	Mean Square Error (MSE)	$\frac{1}{H}\sum_{\tau=t+1}^{t+H}(y_\tau - \hat{y}_\tau)^2$
	Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{H}\sum_{\tau=t+1}^{t+H}(y_\tau - \hat{y}_\tau)^2}$
Percentage Errors	Mean Absolute Percentage Error (MAPE)	$\frac{1}{H}\sum_{\tau=t+1}^{t+H}\frac{\lvert y_\tau - \hat{y}_\tau\rvert}{\lvert y_\tau\rvert}$
	sMAPE	$\frac{1}{H}\sum_{\tau=t+1}^{t+H}\frac{\lvert y_\tau - \hat{y}_\tau\rvert}{\lvert y_\tau\rvert + \lvert\hat{y}_\tau\rvert}$
Scale-Independent	Mean Absolute Scaled Error (MASE)	$\frac{1}{H}\sum_{\tau=t+1}^{t+H}\frac{\lvert y_\tau - \hat{y}_\tau\rvert}{MAE(y_\tau, \hat{y}_\tau^{season})}$

handcrafted features, deep learning architectures automatically learn hierarchical and highly complex representations directly from raw data, allowing them to tackle problems of unprecedented scale and complexity [49]. This section begins with an introduction to the fundamentals of DL and then explores deep learning architectures for TSF. Section 2.3.1 provides a brief overview of the origins of Neural Networks (NNs). Section 2.3.2 explains the training process for NNs. Finally, Section 2.3.3 presents a historical overview and detailed explanation of the key deep learning architectures used for TSF.

2.3.1 Neural Networks

The concept of artificial neurons, which form the basis of NNs, was first proposed in 1947 by Warren McCullogh and Walter Harry Pitts [60]. They proposed a model that mimicked the firing of neurons in the human brain. A set of signals, which in the biological context are gathered by dendrites from synapses [7], are combined linearly. Subsequently, an activation function, a threshold unit, is responsible for "firing" the neuron. Later, Frank Rosemblatt introduced a way to learn from data [79] and Bernard Widrow introduced the idea of representing the threshold value as a bias term [99].

NNs are no more than layers of artificial neurons interconnected. Every neuron is responsible for calculating the weighted sum of the inputs, using a set of *weights* and *bias* as parameters, and then applying a non-linear *activation function*. Non-linear activation functions play a crucial role in neural networks, enhancing their performance and enabling the modeling of complex relationships between inputs and outputs [47].

2.3.2 Training Neural networks

To evaluate the performance of a neural network, we compare the predicted output with the expected output using a *loss function*. The *loss function* quantifies the difference, providing a measure of how close the network's current solution is to the desired outcome. The learning process

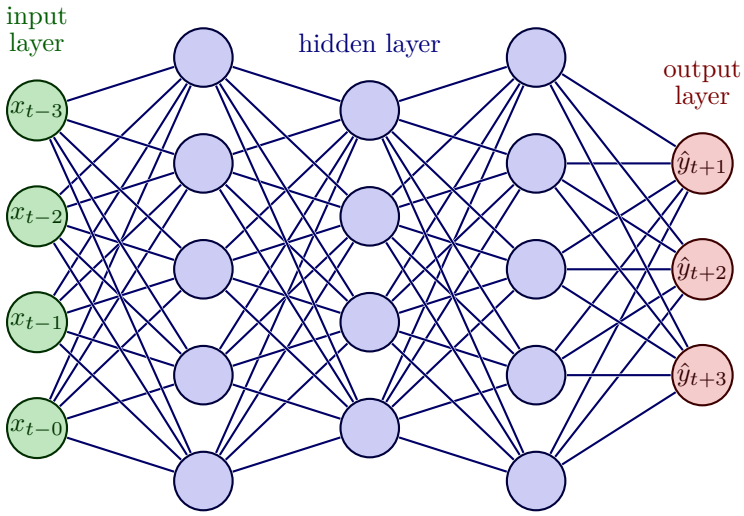


Figure 2.3: Neural networks as Autoregressive models with non-linear hidden layers. In this approach, the time lags $x_{t-3}...x_{t-0}$ are used to predict future occurrences $\hat{y}_{t+1}...\hat{y}_{t+3}$.

involves two main steps: backpropagation, which calculates gradients of the loss with respect to the network's parameters, and optimization, which adjusts these parameters to minimize the loss.

The weights and biases are updated multiple times to allow the model to understand the intricate relationships of the input data. Sometimes this results in overfitting. A number of techniques exist to prevent overfitting, including dropout and regularization. Dropout consists in randomly switching off some neurons [88]. Regularization is a term added to the *loss function* ($\mathcal{L}$) that aims at making the model generalize better [45].

2.3.3 Deep learning for Forecasting

Deep Learning approaches are effective in forecasting problems, as evidenced by the state-of-the-art results on benchmark datasets and competitions such as M4 [57] and M5 [58]. NNs address forecasting tasks via supervised learning typically following an auto-regressive modelling approach, as illustrated in Figure 2.3. In effect, future values are modelled using past lags as input explanatory variables.

Historically, most of the literature focus on MLPs [92] or recurrent approaches (e.g., LSTMs [86]) for time series forecasting using neural networks. More recently, the literature on deep learning for forecasting has explored different architectures besides these, including convolutional neural networks [101], transformers [51], among other approaches.

Despite varied solutions, recent advances have shown that purely feedforward, MLPs-based architectures can achieve state-of-the-art forecasting accuracy [73, 15, 19]. A pioneering architecture in this direction is N-BEATS [73].

Until the release of N-BEATS, pure DL and pure ML architectures had struggled to consistently outperform classical statistical methods in standardized forecasting benchmarks such as M3 and M4 [73, 57, 56]. This was mainly due to challenges with preprocessing and overfitting, which limited their effectiveness [57]. The N-BEATS architecture addresses these issues with a simple, general design that requires no feature engineering or input scaling [73].

N-BEATS consists of multiple stacks, each representing distinct basis functions, as illustrated in Figure 2.4. Every stack comprises N MLP-based blocks, where every MLP computes and projects

the backward $\boldsymbol{\theta}^b$ and forward $\boldsymbol{\theta}^f$ expansion coefficients onto their respective basis functions. This process produces, for each block ℓ, the forecast and backcast outputs denoted by $\hat{\mathbf{y}}_\ell = g_f^\ell(\boldsymbol{\theta}_\ell^f)$ and $\hat{\mathbf{x}}_\ell = g_b^\ell(\boldsymbol{\theta}_\ell^b)$, respectively. Afterwards, the backcast $\hat{\mathbf{x}}_\ell$ is subtracted from the block input signal $\mathbf{x}_\ell$, and the resulting residual is passed to the subsequent block $\ell+1$. The forecast $\hat{\mathbf{y}}_\ell$ is aggregated with previous forecasts as $\hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2 + \cdots + \hat{\mathbf{y}}_\ell$.

In its interpretable configuration, N-BEATS is composed of a trend and seasonal stacks. In the trend stack, the model uses a polynomial basis $\mathbf{B}(t)$, while in the seasonal stack it uses a harmonic Fourier basis $\mathbf{F}(t)$, both applied over normalized time windows [73]:

$$\mathbf{t} = \frac{1}{H}[0,1,2,\ldots,H-2,H-1],$$

$$\mathbf{B}_{i,:} = \begin{bmatrix} 1 & \mathbf{t}_i & \mathbf{t}_i^2 & \cdots & \mathbf{t}_i^d \end{bmatrix},$$

$$\mathbf{F}_{i,:} = \begin{bmatrix} 1 & \cos(2\pi\mathbf{t}_i) & \cdots & \cos\left(2\pi\left\lfloor\frac{H}{2}+1\right\rfloor\mathbf{t}_i\right), \sin(2\pi\mathbf{t}_i) & \cdots & \sin\left(2\pi\left\lfloor\frac{H}{2}+1\right\rfloor\mathbf{t}_i\right) \end{bmatrix}$$

Here, $i \in \{0,\ldots,H-1\}$ indexes the time steps, and $\mathbf{t}_i \in \{0, \frac{1}{H}, \ldots, \frac{H-1}{H}\}$ represents normalized time. The parameter d is the degree of the polynomial in the trend basis.

In contrast to the interpretable configuration, the generic configuration does not impose any predefined time series feature–specific basis functions. Instead, it learns the basis functions directly from the data within the FCs layers of the blocks. In this configuration, there is no need to distinguish between stacks and blocks because each stack uses the same basis function. These stacks are referred to as identity stacks since the basis function simply replicates the basis coefficients from the block, allowing the block to learn how to decompose the forecast.

N-BEATS is primarily designed for univariate forecasting, which is the main focus of this dissertation. In contrast, Neural Basis Expansion Analysis with Exogenous (N-BEATSx) extends this framework by incorporating exogenous variables, enabling the model to account for external factors that influence the time series [71].

Another architecture, built upon N-BEATS, is Neural Hierarchical Interpolation for Time Series Forecasting (N-HITS). Incorporating hierarchical interpolation and multi-rate data sampling techniques, this model addresses the issues of volatility and computational complexity associated with long-term forecasts [15].

It is also pertinent to mention some other significant architectural styles. Temporal Fusion Transformers (TFT) incorporates multi-head attention to highlight relevant features for interpretability [51]. Although many transformer architectures have been proposed [84, 51, 105], their effectiveness is debated [103]. Temporal Convolution Network (TCN) denotes the utilisation of 1D convolutional architectures in the context of temporal data. Empirical evidence suggests that this approach yields superior outcomes compared to conventional recurrent networks, such as LSTMs, in specific tasks and datasets [5]. Another notable approach is DeepAR, which trains an autoregressive recurrent neural network on related time series to generate accurate probabilistic forecasts [80].

2.4 Mixture of Experts

The MoE framework involves training several specialized models, or "experts", each focused on capturing different patterns within the data. Additionally, it incorporates a gating network that is

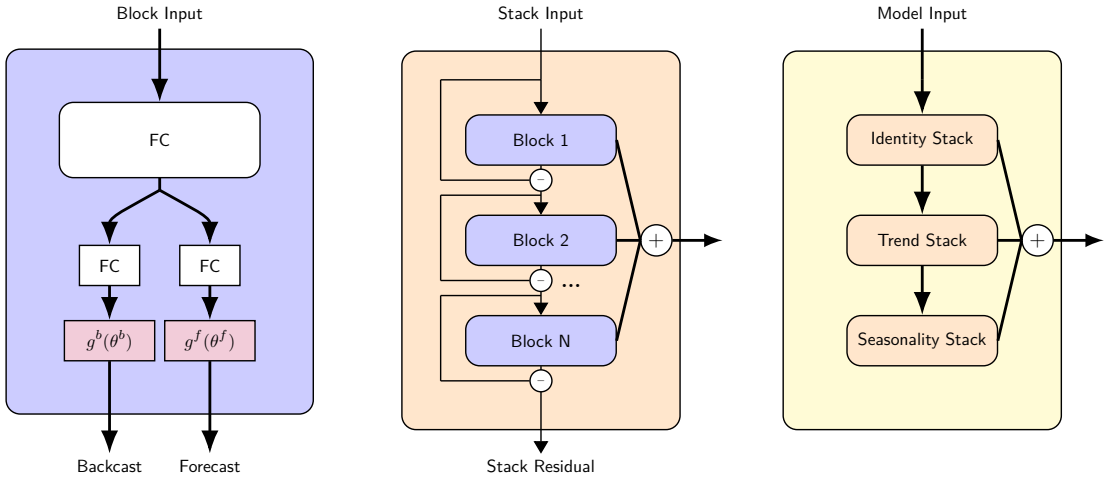


Figure 2.4: Architecture of the `N-BEATS` interpretable configuration originally proposed by Oreshkin et al. [73]. In the original N-BEATS architecture, the identity stack is not included. However, adding an identity stack can enhance forecasting performance in certain cases without compromising the interpretability of the results. Another important feature of this configuration is that the parameters of the blocks are shared within each stack, which the authors argue contributes to improved overall performance.

capable of dynamically determining the most appropriate expert for each input [33]. This approach has recently been successfully applied in LLMs such as Mixtral-8x7B [35] and DeepSeek [18]. In this section, MoE is introduced and discussed in the context of this thesis. Section 2.4.1 outlines the key characteristics of MoE. Section 2.4.2 presents an overview of the taxonomy, focusing on gating mechanisms and training paradigms. Section 2.4.3 provides a brief overview of how MoEs are utilized in LLMs. Finally, Section 2.4.4 reviews related work, highlighting existing gaps in the current research, while Section 2.4.5 addresses the routing collapse problem.

2.4.1 Background on MoE

MoE methods address machine learning problems using an ensemble-based approach. A MoE typically consists of N expert networks $\{f_1, \ldots, f_N\}$ trained alongside a gating network $\mathcal{G}$. Originally proposed by Jacobs et al. [33], MoE aligns closely with the divide-and-conquer approach, a strategy that breaks down a complex problem into smaller, more manageable subproblems that are easier to solve. Unlike conventional models that activate all parameters for every input, MoEs dynamically select the active parameters depending on the input. This enables the model to better adapt to datasets that involve multiple domains such as time series datasets composed of heterogeneous time series [57]. Consequently, the MoE framework can be defined as follows:

- **Expert Models,** $\{f_1, \ldots, f_N\}$: A set of models, each of which specializes in learning and solving a particular subset of the data. In most cases, the expert model can be any type of model, although it is more commonly a neural network. Each expert is defined by a set of parameters W_i and generates a prediction for an input x, denoted as $f_i(x; W_i)$.
- **Gating Network,** $\mathcal{G}$: The Gating Network (GN) is responsible for redirecting the input, x, to the appropriate experts. As such, it typically computes a *linear-softmax* distribution that assigns probabilities to all experts, N. While the gate is normally implemented as

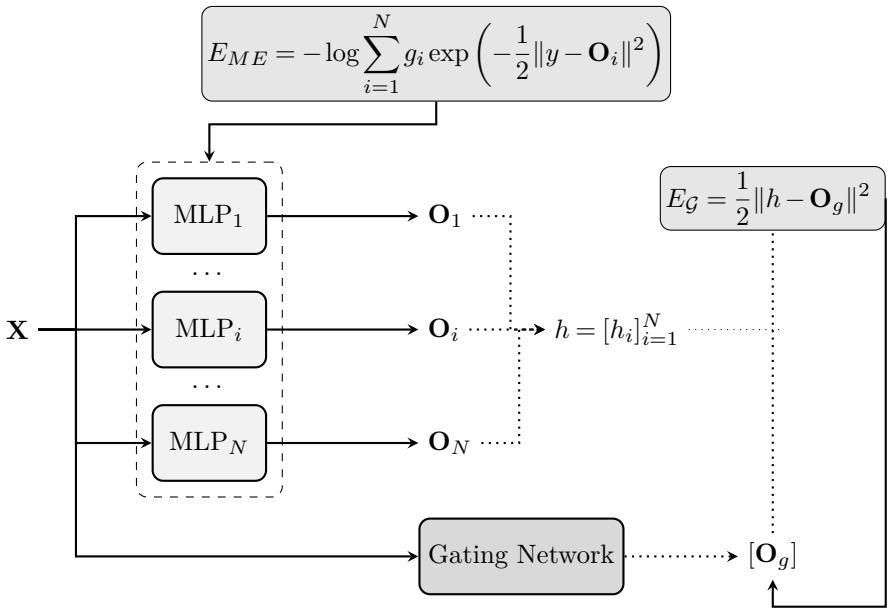


Figure 2.5: Training step in a Mixture of MLP-experts. In this particular example, the gating network partitions the input space in response to the performance of the various experts, assigning softmax-based weights to each expert. This gives the MLP-experts a competitive environment in which to identify and specialize in different patterns. The training process is guided by two loss functions: the main objective function E_{ME}, which encourages experts to specialize based on performance, and the auxiliary gating loss $E_{\mathcal{G}}$, which aligns the gating network's output O_g with the ideal gating targets h. Image adapted from [59].

a linear projection [35, 18], other approaches exist, such as domain-specific gating or non-linear gating mechanisms [66]. GNs can be described as a set of parameters, Θ, which take an input, x, and produce an output, $\mathcal{G}(x;\Theta)$.

- **Loss Function, $\mathcal{L}$**: The loss function is responsible for quantifying how well the MoE is performing in terms of its predictions. For MoE, the total loss is typically a combination of the individual losses from each expert network weighted by the gating network's output. The loss function enables the GNs to learn how to route each input to the most relevant expert(s), thereby allowing the experts to specialize in the subset. However, as discussed later, this mechanism can lead to imbalances in expert utilization and requires special attention.

- **Pooling method**: The pooling methods refers to the mechanism used to combine the expert(s) output, $f_i(x;W_i)$, and the GN output, $\mathcal{G}(x;\Theta)$, in order to produce a prediction. The pooling output, and thus the MoE prediction, can then be defined as $\mathcal{F}^{\text{MoE}}(\mathbf{x};\Theta,\{\mathbf{W}_i\}_{i=1}^{N})$.

In Figure 2.5, an example of the training process for a MoE is illustrated. Similarly, Figure 2.6 presents an example of the inference process.

2.4.2 Taxonomy of MoE

Masoudnia and Ebrahimpour [59] describe MoE as a data-dependent combining method where the input directly influences the ensemble by determining the combining weights. In the context of ensemble learning, they distinguish two main strategies: fusion, where all classifiers are trained on the entire feature space, and selection, where each classifier is trained on a specific region or

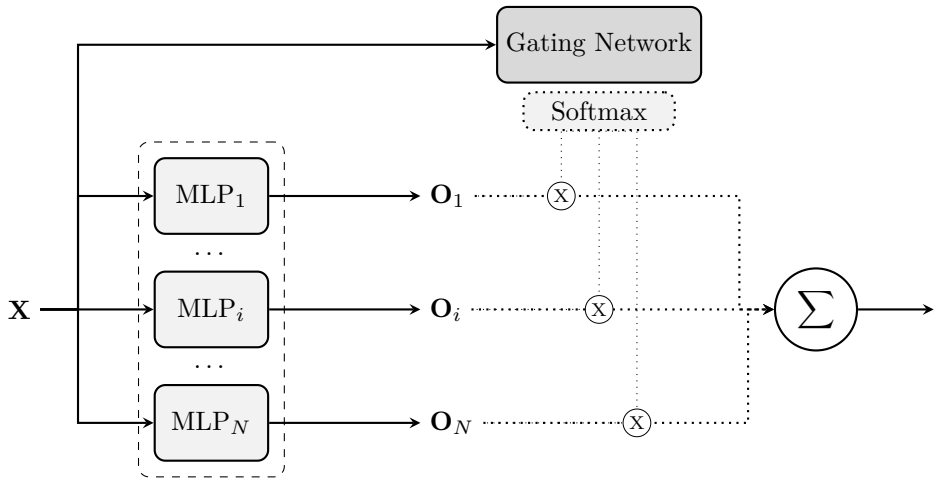


Figure 2.6: Inference in a Mixture-of-Experts model. The input, x, is passed to each expert. The gating network computes a softmax distribution over the gating values, and the output is obtained by calculating the weighted average of the expert outputs. Image adapted from [59].

subset. MoE is presented as a hybrid of these approaches, combining aspects of both by dynamically weighting classifier outputs according to their local competence for a given input.

Masoudnia and Ebrahimpour [59] further distinguish how the partitioning of subspaces is performed. In the Mixture of Implicitly Localized Experts (MILE) approach, the problem space is dynamically divided during training, with experts competing and adapting to different regions based on the gating network's decisions. Conversely, in Mixture of Explicitly Localized Experts (MELE), the problem space is pre-partitioned using clustering techniques prior to training, and each expert is assigned to a specific subspace based on this predefined division.

In regard to the pooling method, two principal approaches can be identified:

- **Dense MoE**: In this approach, all experts are activated for each input and their outputs are aggregated without any selection. Dense MoE models tend to be computationally expensive because they use the full capacity of all experts at each forward pass [82]. As a result, the output of a dense MoE can be considered as:

$$\mathcal{F}^{\text{MoE}}_{\text{dense}}(\mathbf{x};\Theta,\{\mathbf{W}_i\}_{i=1}^{N}) = \sum_{i=1}^{N} \mathcal{G}(\mathbf{x};\Theta)_i f_i(\mathbf{x};\mathbf{W}_i)$$

  where $\mathcal{G}(\mathbf{x};\Theta)_i$ represents the *linear-softmax* of the gating values for input x, while $f_i(\mathbf{x};\mathbf{W}_i)$ represents the output of expert i for input x.

- **Sparse MoE**: In Sparse MoE, the gating network selects only a subset of experts to activate for each input, typically the top-k experts with the highest gating values. This approach significantly reduces computational cost while maintaining performance by focusing on the most relevant experts for a given input [35, 18, 82]. The output of a Sparse MoE can be expressed as:

$$\mathcal{F}^{\text{MoE}}_{\text{sparse}}(\mathbf{x};\Theta,\{\mathbf{W}_i\}_{i=1}^{N}) = \sum_{i\in\mathcal{S}} \mathcal{G}(\mathbf{x};\Theta)_i f_i(\mathbf{x};\mathbf{W}_i),$$

  where $\mathcal{S}$ denotes the set of selected top-k experts based on their pre-softmax gating values, and $f_i(\mathbf{x};\mathbf{W}_i)$ represents the output of expert i for the input $\mathbf{x}$.

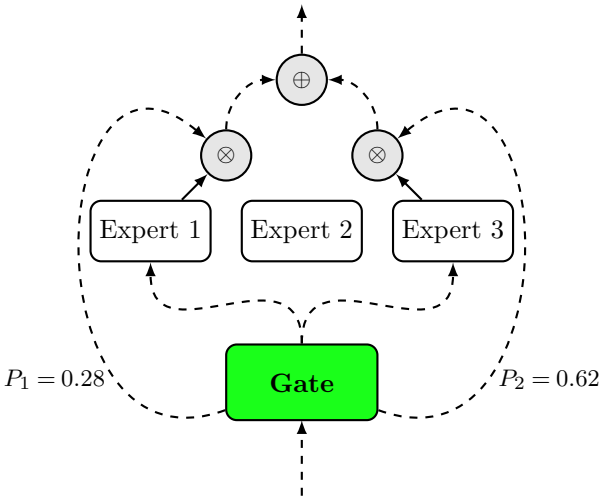


Figure 2.7: A sparse MoE consisting of 3 experts. The gating mechanism first selects the 2 most appropriate experts for the input based on their gate function scores. Then, a linear combination is performed, using the *softmax* values of the gating scores as weights. The key distinction between sparse MoE and dense MoE is that, as shown in the image, not all experts are activated.

While the previous two approaches are the most well-known and widely used, there are several other combination methods worth highlighting. For instance, Soft MoE addresses the scalability, training stability, and related challenges faced by Sparse MoE. It performs an implicit soft assignment by passing different weighted combinations of all input tokens to each expert, making the entire operation continuous and differentiable [75]. Another variant, Hierarchical MoE, extends MoE models into a tree-structured form where inputs are routed through multiple gating levels to selectively activate specialized experts [50, 37]. Additionally, there are methods such as Shared MoE that ensure one or more experts are always used for all inputs [18].

Figure 2.7 illustrates the core process and motivation behind sparse MoE.

2.4.3 Scaling LLMs with MoE architectures

In recent years, LLMs have gained widespread popularity and have become important tools for productivity, knowledge retrieval, code assistance, and a variety of other applications. As the demand for larger and more capable models intensifies, the need for more efficient architectures poses an increasingly significant challenge.

In conventional dense LLMs, performance typically exhibits a log-linear relationship with increases in dataset size, model size, and the amount of compute utilised for training [39]. This indicates that to achieve substantially superior models, there is a necessity for exponentially more parameters and computational resources. However, scaling the model parameters typically results in an increased inference latency and higher computational costs.

To address this issue some architectures, [35] [18], decided to switch the feed-forward layer of their models with a sparse MoE layer. This modification enabled computational costs to remain manageable as model size increased and performance improved.

In a sparse MoE layer, only a subset of available experts is activated for each input token, typically selected via a gating mechanism. This design allows the model to scale up the total

number of parameters without proportionally increasing the computational burden during inference or training [35]. A critical component of making this architecture viable at scale is the use of all-to-all dispatch and combine operations, which facilitate the dynamic routing of tokens to the appropriate experts across distributed hardware [11]. Although this approach introduces additional communication overhead, it enables substantial gains in efficiency and scalability. However, the effectiveness of communication depends on various factors, such as hardware configurations, bandwidth limitations of communication channels, and potential load imbalances [11, 66]. These imbalances, which are inherent to MoE architectures, can exacerbate inefficiencies by introducing synchronization delays [11, 66].

To mitigate this, DeepSeek-V3, a recently popularized model, employs a DualPipe algorithm, which overlaps communication with computation to minimize pipeline stalls and hide most of the all-to-all overhead [20]. Other researchers have also proposed different strategies to address this problem [76, 83]. Although this issue is not encountered in this dissertation, managing all-to-all communication remains a crucial challenge when training LLMs based on MoE.

2.4.4 Considerations on MoE for TSF

The interplay between global and local modeling approaches is a recurring topic in the field of time series forecasting. Global models, such as NNs, have shown strong performance in capturing complex patterns, especially with limited data [94, 28]. However, this superiority depends on a set of factors, such as heterogeneity of the dataset, complexity of patterns and techniques, and the amount of data [64, 28]. In order to achieve a balance between globality and locality, a number of hybrid approaches have been developed. Some of the most notable applications of this technique, which have produced state-of-the-art results, include MCNet [89] for long-term time series forecasting, DeepGLO [81] for high-dimensional time series, and LGnet [91] for multivariate time series with missing values.

Although the potential of this hybrid approach has been demonstrated, further research is required to determine how MoE frameworks can be used to improve the performance of hybrid models. More recently, foundational models based on the MoE framework have proven state-of-the-art performance in zero-shot forecasting and in-distribution forecasting [54, 84]. Despite their notable success, these foundational models possess a considerable number of parameters, reaching the order of billions. Moreover, given their recent publications, the accuracy of these models in real-world scenarios and their respective inference costs remain uncertain [93, 102].

Also recently, Ni et al. proposed a MoE framework, named MoLE, that can augment long-term linear forecasting models to address non-stationary and seasonal patterns [69]. While the authors demonstrated a clear improvement in the addition of MoE, several questions remained unanswered. Firstly, it would be beneficial to ascertain whether the technique can be adapted for use in other forecasting horizons. Secondly, as alluded to in the paper, in what types of datasets does the MoLE framework have the potential to be beneficial? In this regard, Ni et al. hypothesis that this is related to the heterogeneity of the dataset [69].

The field of traffic flow prediction is a widely studied area of research, characterized by substantial heterogeneity and complex patterns. Wang et al. put forth a heterogeneous MoE model for traffic flow prediction, which demonstrates state-of-the-art performance by employing five distinct experts to address the intricacies of the task domain [96]. Another research problem where MoE is successfully employed to tackle data heterogeneity is in addressing sudden or gradual shifts in time

series data [90]. Finally, using a MoE framework, pre-trained experts and an arbitrated approach as the routing mechanism has also been explored for time series forecasting [13].

2.4.5 Considerations on the routing collapse problem

In sparsely activated neural networks, such as sparse MoE architectures, it is common for the gating mechanism to assign disproportionately high probabilities to a single expert, thus neglecting the remaining experts. This imbalance can significantly restrict training opportunities for underutilized experts, further degrading the problem known as routing collapse [82].

To address this issue, it is common to introduce an auxiliary loss function term, as proposed by Dai et al. [18] and later adapted for TSF by Shi et al. [84]. This term promotes load balancing among experts by penalizing uneven expert utilization.

However, this latter approach introduces significant instability during training and may result in suboptimal solutions that hinder expert specialization. A common strategy to address this limitation is the introduction of a shared expert, which is responsible for capturing general or "common sense" knowledge, thereby allowing the remaining experts to specialize more effectively across distinct domains [18].

Another alternative involves the use of *loss-free* balancing strategies, which aim to achieve expert balancing without relying on auxiliary loss terms. For example, Zhou et al. [106] propose to reverse the traditional routing mechanism by allowing experts to select *top-k* tokens, rather than having a gate select the *top-k* experts. More recently, Wang et al. [97] introduce expert-wise biases applied directly to the routing scores, promoting balanced expert selection through architectural modifications rather than explicit loss penalties.

Although a number of strategies have been proposed to address this issue, as discussed above, there remains limited research on how load balancing mechanisms can be effectively applied within the context of TSF. Existing approaches are predominantly designed for Transformer-based architectures, such as those presented in [84] and [35], which are not inherently well-suited for the forecasting task [103]. Furthermore, in these cases, the gating mechanisms typically operate on rich, high-dimensional embedding representations, the interpretability of which remains challenging.

At first glance, it may seem intuitive to assume that the MoE gating mechanism learns to route different domains (e.g. coding question, healthcare question) to corresponding experts. However, in practice, the gating function dynamically routes inputs based on latent similarities within the data, which may, but do not necessarily, align with predefined domains [11].

Chapter 3

Methodology

This chapter restates the problem addressed in the dissertation and provides an overview of our approach to solving it. Section 3.1 briefly revisits the problem definition. Section 3.2 presents the hypothesis and research questions. Finally, section 3.3 presents the proposed approach and section 3.4 its variants.

3.1 Problem Definition 19
3.2 Research Hypothesis 20
3.3 Approach 21
3.4 Method Variants 22

3.1 Problem Definition

Univariate time series forecasting involves predicting future values of a single observed variable over time. When addressing forecasting problems, deep neural networks are typically trained on collections of univariate time series. The underlying hypothesis is that by jointly learning from multiple series, the model can discover common patterns and complex relationships that improve overall predictive performance [28].

However, real-world datasets often contain heterogeneous collections of time series exhibiting diverse behaviours, dynamics, and noise levels. This heterogeneity poses significant challenges for traditional deep learning models, which may struggle to adequately capture the unique characteristics of each individual series within the dataset [28]. Consequently, models risk making inaccurate generalizations by failing to account for the varying temporal patterns and statistical properties present.

The key difficulty arises when a single global model attempts to simultaneously represent series with differing trends, seasonalities, variances, and noise characteristics. This can lead to a trade-off where the model overfits dominant patterns shared by many series while underfitting rare or atypical behaviours exhibited by particular time series. As a result, forecasting accuracy for less common time series deteriorates, undermining the model's overall utility.

For example, consider a large-scale retail platform [23], which maintains demand histories for millions of products. Each product's sales data forms a univariate time series. Some products,

such as everyday essentials, may have stable and non-seasonal demand patterns, whereas others, such as holiday items or seasonal apparel, exhibit strong seasonal fluctuations. Moreover, niche products may show irregular, bursty sales with limited historical data. While it is impractical to model each series independently, a global model that ignores these heterogeneities may perform well on typical products but poorly on seasonal or niche items.

This research addresses the challenge of modelling heterogeneous univariate time series datasets by coupling the state-of-the-art N-BEATS model with a MoE, enabling expert specialization and adaptive pattern combination across diverse series.

3.2 Research Hypothesis

As previously discussed, to improve the generalization of GFM in TSF, we propose to augment N-BEATS with a MoE architecture. Based on the problem outlined in the preceding section, the following hypothesis can be formulated:

> By coupling N-BEATS with a MoE, we can further improve its accuracy and interpretability, especially on *heterogeneous* datasets comprising multiple univariate time series.

To address this hypothesis, it is essential to explore how a MoE approach can be incorporated into `N-BEATS`. What specific challenges arise compared to other MoE implementations, and what advantages might this integration offer in terms of forecast accuracy?

As mentioned previously, one of our motivations for this research comes from the effective use of MoE in LLMs, where they improve model performance and reduce inference cost, especially when working with large heterogeneous datasets [66, 95]. It is also important to acknowledge the difference between the forecasting domain and natural language.

For instance, current MoE approaches in the literature are typically trained on tokens and mostly within transformer architectures [35, 18]. The load balancing mechanism, as discussed earlier, distributes these tokens across the experts, enabling them to maintain a "global" view of the data while also allowing each expert to specialize. In LLMs, this poses no significant challenge, as individual tokens can be routed to multiple experts according to their contextual embeddings, effectively capturing the variability of word usage across diverse domains and semantic meanings.

In contrast, TSF typically deals with continuous windows of data rather than discrete tokens. Unlike language, where words can be flexibly rearranged or substituted, time series data depends on preserving sequential patterns. This makes it less straightforward to route parts of the input independently, as doing so may disrupt the temporal coherence needed for accurate prediction. As a result, applying MoE models to time series requires careful consideration of how expert specialization interacts with temporal structure.

Another important aspect is the gating network. In current MoE approaches [35, 18, 84], the gating network typically consists of a simple linear layer. However, its suitability for the forecasting task can be different. This concern comes from the fact that, in transformer-based models, the gating mechanism usually operates on highly expressive embeddings that are more amenable to linear separability [68]. In contrast, non-transformer models such as N-BEATS often rely on raw windowed inputs or representations that lack the richness of transformer-derived embeddings, which may impair the effectiveness of a simple linear gate.

Finally, it is important to acknowledge the difficulty of the forecasting task, where unrelated and unpredictable events (e.g., pandemics or wars) can significantly impact the series predictions. This contrasts with the natural language domain, where, even without exposure to the most recent data, models can still convincingly mimic conversation. These factors make the application of MoE approaches more challenging in the forecasting domain and have not been thoroughly addressed in previous literature.

3.3 Approach

Our proposed approach, dubbed `N-BEATS-MOE`, aims to improve `N-BEATS` by introducing an adaptive weighting mechanism that learns to combine the outputs of these stacks.

In our proposed method (Figure 3.1), we replace the standard summation of block outputs in `N-BEATS` with a MoE layer. In the original `N-BEATS` architecture, each block generates a backcast and a forecast by projecting learned coefficients onto predefined basis functions, and the final output is obtained by summing the forecasts from all blocks. In this setup, each block acts as an expert, and the gating mechanism learns to determine the appropriate contribution or importance of each to the final output.

Formally, let the outputs of each of the B blocks in the model be denoted by $\hat{\mathbf{y}}_\ell$, where $\ell \in \{1, \ldots, B\}$, corresponding to forecast contributions from stacks such as *identity*, *trend*, and *seasonal*. In the original `N-BEATS` architecture, these outputs are simply summed to produce the final forecast:

$$\hat{\mathbf{y}} = \sum_{\ell=1}^{B} \hat{\mathbf{y}}_\ell \tag{3.1}$$

In our proposed `N-BEATS-MOE` approach, this summation is replaced by a weighted combination using a learned gating mechanism, yielding:

$$\hat{\mathbf{y}} = \sum_{\ell=1}^{B} \hat{\mathcal{G}}_\ell \cdot \hat{\mathbf{y}}_\ell, \tag{3.2}$$

where the block gating weights $\hat{\mathcal{G}}_\ell$ are defined as:

$$\hat{\mathcal{G}}_\ell = \text{softmax}_\ell \left(\text{LINEAR}_\ell(\text{LayerNorm}(\mathbf{x}_0)) \right). \tag{3.3}$$

Here, LINEAR_ℓ denotes a learnable affine transformation applied to the model input $\mathbf{x}_0$, which is first normalized through a LayerNorm to prevent mode collapse. The gating weights assign different importance to each block output, allowing the model to adaptively combine them.

This dynamic weighting mechanism enables the model to prioritize stack types based on the characteristics of the input series aiming at improving performance on heterogeneous datasets with diverse temporal patterns. Besides a better adaptability, this approach also allows for a better interpretability and explainability, since the output of the gating network can provide insights about which stack types are most important for a given input.

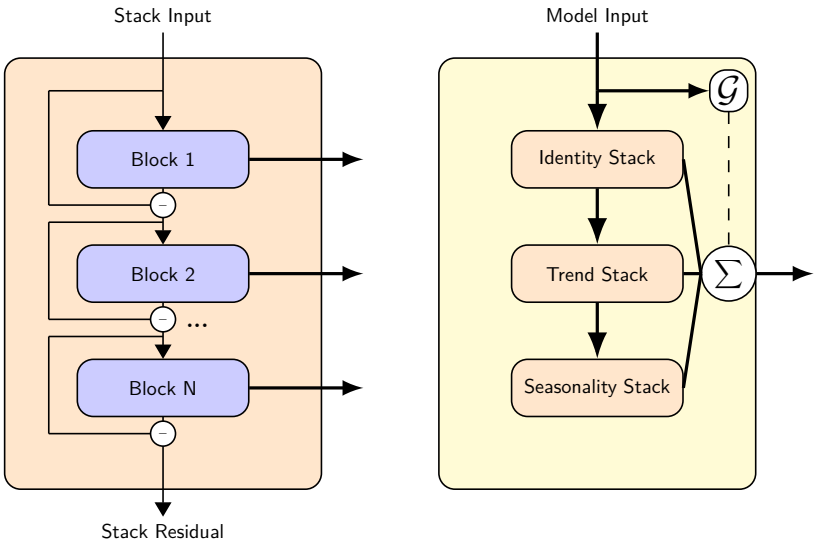


Figure 3.1: The proposed architecture, `N-BEATS-MOE`. The gate $\mathcal{G}$ is responsible for determining the importance of each block based on the input data.

3.4 Method Variants

The MoE layer can be integrated with N-BEATS in several different ways. In this section, we introduce a few possible variants, illustrated in Figure 3.2, of our approach, which are evaluated and compared in detail in the following chapter:

`N-BEATS-`(MoEBlock): In this architecture, we replace the Fully Connected (FC) layer in the N-BEATS basic building blocks, shown on the left of Figure 2.4, with a MoE layer consisting of n expert FC layers, where each expert models different basis coefficients corresponding to subsets of series patterns.

Formally, and following the definition in the original N-BEATS paper [73], the operations of the ℓ-th block can be described as consisting of two parts. The first part, where this approach diverges from the original, computes the forward $\boldsymbol{\theta}_\ell^f$ and backward $\boldsymbol{\theta}_\ell^b$ basis expansion coefficients from the input $\mathbf{x}_\ell$. The second part projects these coefficients through the backward basis layer g_b^ℓ and the forward basis layer g_f^ℓ onto the corresponding basis functions to generate the backcast $\hat{\mathbf{x}}_\ell$ and the forecast $\hat{\mathbf{y}}_\ell$.

Afterwards, the backcast $\hat{\mathbf{x}}_\ell$ is subtracted from the input signal $\mathbf{x}_\ell$, and the resulting residual is passed to the subsequent block $\ell+1$. The forecast $\hat{\mathbf{y}}_\ell$ is aggregated with previous forecasts as $\hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2 + \cdots + \hat{\mathbf{y}}_\ell$.

The modified operations in the first stage, based on a sparse MoE, are defined as follows[1]:

$$\mathbf{m}_\ell = \sum_{i=1}^{N} \mathcal{G}_{\ell,i} \cdot \mathcal{E}_{\ell,i}(\mathbf{x}_\ell), \tag{3.4}$$

[1] See [73] for the original basis coefficient computation, which can be roughly seen as using a single expert instead of a mixture.

where each expert $\mathcal{E}_{\ell,i}$ is a stack of FC layers with ReLU non-linearity:

$$\mathcal{E}_{\ell,i}(\mathbf{x}_\ell) = \mathrm{FC}_{\ell,i}^{(4)}\left(\mathrm{FC}_{\ell,i}^{(3)}\left(\mathrm{FC}_{\ell,i}^{(2)}\left(\mathrm{FC}_{\ell,i}^{(1)}(\mathbf{x}_\ell)\right)\right)\right), \tag{3.5}$$

and the gating weights $\mathcal{G}_{\ell,i}$ are defined by:

$$\mathcal{G}_{\ell,i} = \begin{cases} s_{\ell,i}, & \text{if } s_{\ell,i} \in \mathrm{TopK}(\{s_{\ell,j} \mid 1 \le j \le N\}, K), \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

with softmax gating scores computed as:

$$s_{\ell,i} = \mathrm{softmax}_i\left(\mathrm{LINEAR}_{\ell,i}^{\mathrm{gate}}(\mathrm{LayerNorm}(\mathbf{x}_\ell))\right). \tag{3.7}$$

The resulting mixture output $\mathbf{m}_\ell$ is then separated into forward and backward expansion coefficients:

$$\boldsymbol{\theta}_\ell = \mathbf{m}_\ell, \quad \boldsymbol{\theta}_\ell^b = \boldsymbol{\theta}_\ell[0:\mathcal{D}], \quad \boldsymbol{\theta}_\ell^f = \boldsymbol{\theta}_\ell[\mathcal{D}:] \tag{3.8}$$

where $\mathcal{D}$ is the dimension of either the polynomial or harmonic basis used for the backward and forward expansion coefficients, respectively.

After that, the basis coefficients are projected onto the corresponding basis functions using the basis layers, yielding the forecast and backcast outputs: $\hat{\mathbf{y}}_\ell = g_f^\ell(\boldsymbol{\theta}_\ell^f)$ and $\hat{\mathbf{x}}_\ell = g_b^\ell(\boldsymbol{\theta}_\ell^b)$. This projection can be expressed explicitly as a linear combination over fixed basis vectors [73]:

$$\hat{\mathbf{y}}_\ell = \sum_{i=1}^{\dim(\boldsymbol{\theta}_\ell^f)} \theta_{\ell,i}^f \mathbf{v}_i^f, \quad \hat{\mathbf{x}}_\ell = \sum_{i=1}^{\dim(\boldsymbol{\theta}_\ell^b)} \theta_{\ell,i}^b \mathbf{v}_i^b, \tag{3.9}$$

... where $\{\mathbf{v}_i^f\}$ and $\{\mathbf{v}_i^b\}$ are fixed basis functions defined by the architecture, and $\dim(\boldsymbol{\theta}_\ell^f)$ and $\dim(\boldsymbol{\theta}_\ell^g)$ denote the sizes of the forecast and backcast parameter vectors, respectively. Different stacks, shown at the center of Figure 2.4, represent distinct basis functions. For this study we use the *identity*, *trend* and *seasonal* stacks.

`N-BEATS`-(MoEShared): Inspired by DeepSeek's approach [18], this variant uses one shared expert for all inputs, together with specialized experts that the gating network activates based on the input data.

Formally, the definition of the `MoEShared` architecture is identical to that of the `MoEBlock`, with the only difference being the replacement of the sparse MoE in the latter by a shared MoE in the former. Consequently, the only modification occurs in equation 3.4:

$$\mathbf{m}_\ell = \mathrm{SE}_\ell(\mathbf{x}_\ell) + \sum_{i=1}^{N} \mathcal{G}_{\ell,i} \cdot \mathcal{E}_{\ell,i}(\mathbf{x}_\ell), \tag{3.10}$$

where SE_ℓ is a shared expert common to all inputs that pass the block, capturing general knowledge, while $\mathcal{E}_{\ell,i}$ represents specialized experts focusing on specific inputs or patterns.

`N-BEATS`-(MoEScaled): This architecture is similar to `MoEBlock`, with the key difference that the experts in the MoE layer have varying model sizes, enabling scalability and flexibility in model complexity.

Let the list of MLP layers, previously used in `MoEBlock`, dimensions be defined as:

$$\mathcal{M} = \{[u_j, ..., v_j] \in \mathbb{N}^2 \mid j = 1, 2, \ldots, n\}$$

where each pair $[u_j, ..., v_j]$ represents the hidden layer sizes of the j-th stack, and n is the total number of stacks.[2]

Let $i \in \mathbb{N}_0$ be a fixed non-negative integer representing the index of expert i, used for scaling the corresponding expert's model size.

Define a transformation $T_i : \mathbb{N}^2 \to \mathbb{N}^2$ such that:

$$T_i([u_j, ..., v_j]) = \left[\left\lceil \frac{u_j}{i+1} \right\rceil, \left\lceil \frac{v_j}{i+1} \right\rceil \right]$$

where $\lceil \cdot \rceil$ denotes the ceiling function.

Then, the transformed list of MLP units is given by:

$$\mathcal{M}' = \{T_i([u_j, ..., v_j]) \mid [u_j, ..., v_j] \in \mathcal{M}\}$$

For example, if in the `MoEBlock` we have experts with a hidden size of 512 (in the FC layer), then with scaling applied and considering 4 experts, these become:

$$\text{Hidden Sizes} = \left\{ \left\lceil \frac{512}{i+1} \right\rceil \mid i = 0, 1, 2, 3 \right\} = \{512, 256, 171, 128\}$$

With this approach, we aim to investigate whether it is possible to specialize models based on the complexity of the time series, and to assess whether such specialization aligns with improved model performance.

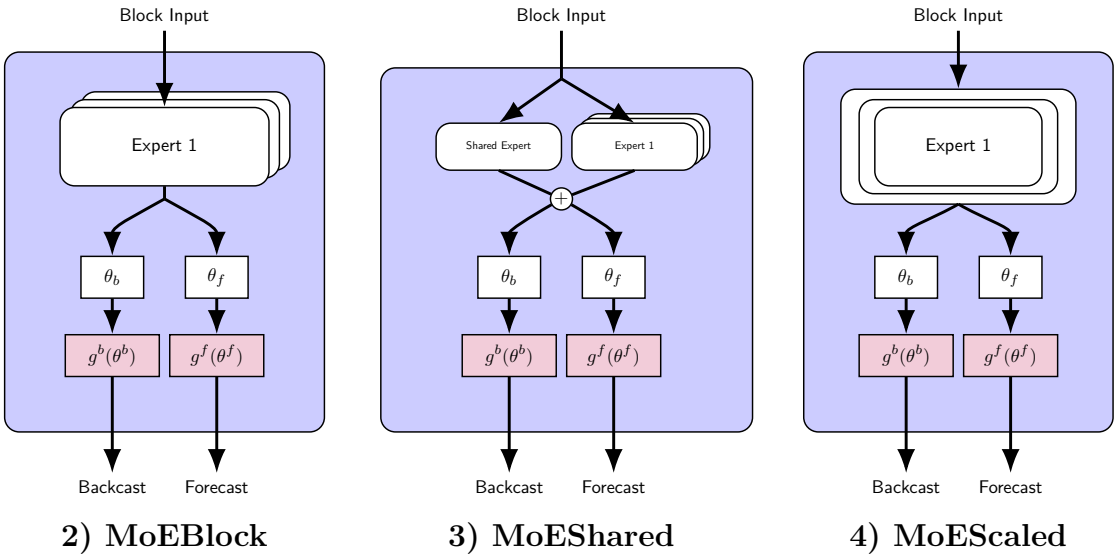


Figure 3.2: `N-BEATS-MOE` variations. In these variations, we focus on applying the MoE at the block level. For clarity, the gating mechanism—a simple linear layer—is omitted.

[2] In the interpretable configuration of N-BEATS, blocks weights are shared within stacks; therefore, we refer to stacks rather than individual blocks. Additionally, in our configuration, we use only four hidden layers, so $n = 4$.

Chapter 4

Experiments

This chapter outlines the experimental design and presents the empirical results of our experiments. We compare various model variants across a range of benchmark datasets and frequency groups to evaluate their effectiveness in terms of forecast accuracy. In addition to standard performance metrics, we provide detailed analyses of expert specialization and examine model outcomes across multiple dimensions.

4.1 Experimental Design 25
4.2 Results 28
4.3 Expert Specialization Analysis 29
4.4 Aspect-based Analysis 31
4.5 Discussion 33
4.6 Threats to Validity 35

4.1 Experimental Design

To effectively address the research questions presented in Section 3.2, it is essential to establish a well-structured experimental setup that supports and validates the findings of this study. To ensure the research is both structured and reproducible, we utilize the *nixtla*[1] NeuralForecast package. The complete implementation of the experiments can be accessed through the following GitHub repository[2]. This section provides a detailed description of the experimental setup, including the data used, model configurations, evaluation metrics, and the procedures followed to conduct and analyze the experiments.

4.1.1 Datasets and Evaluation

We evaluate forecasting performance using 12 time series datasets originating from 4 forecasting competitions: M1 [55], Tourism [3], M3 [56], and M4 [57]. These cover three frequencies: monthly,

[1] https://nixtlaverse.nixtla.io/
[2] Code available at https://github.com/zaai-ai/mixture_of_experts_time_series/tree/main

quarterly, and yearly. The M1, M3 and M4 datasets are from the Makridakis forecasting competitions, which contain time series from domains such as industry, finance, and demographics. Table 4.1 summarizes the datasets used in this study.

Table 4.1: Summary of datasets by frequency

Dataset	Frequency	# Series	Domains
M1	Yearly	181	Mixed
	Quarterly	203	
	Monthly	617	
Tourism	Yearly	518	Tourism
	Quarterly	427	
	Monthly	366	
M3	Yearly	645	Mixed
	Quarterly	756	
	Monthly	1428	
M4	Yearly	23000	Mixed
	Quarterly	24000	
	Monthly	48000	

Three of these four sources contain time series from varied application domains, which provide an adequate test bed for evaluating the performance of the proposed approach in handling heterogeneous time series.

To validate and test the performance of our approach, we exclude the last observations of each time series, as described in Section 2.2.3. Each dataset was divided into training, validation, and test subsets. The test set is composed of the last H observations of each time series in a given dataset. The validation set is partitioned in the same way, being composed of the last H observations of each time series before the test set. We use sMAPE as the evaluation metric, which is defined as follows:

$$\frac{1}{H}\sum_{\tau=t+1}^{t+H}\frac{|y_\tau - \hat{y}_\tau|}{|y_\tau| + |\hat{y}_\tau|} \tag{4.1}$$

where H represents the forecasting horizon, y_τ is the actual value at time τ, and $\hat{y}_\tau$ is the predicted value. This metric is expressed as a percentage error, providing a normalized measure of forecast accuracy.

4.1.2 Training Procedure

When conducting a comparative analysis between models, it is important to ensure that both are provided with an equitable and optimized training configuration. Given that the models under investigation exhibit comparable architectural characteristics, a consistent experimental baseline was established. To maximize the effectiveness of the training process, several loss functions — including sMAPE, MAE, and *Huber* — were evaluated to determine which was the most suitable for each model. As the observed differences in performance across these loss functions were minimal, MAE was ultimately selected as the loss function for subsequent experiments.

For optimization, the Adaptive Moment Estimation (ADAM) optimizer was employed with its default parameters and an initial learning rate of 0.001, consistent with the configuration used in N-BEATS paper. As a learning rate scheduler, we adopted the *StepLR* strategy with a gamma value of 0.5. Alternative scheduling approaches, such as Warm-Up combined with cosine exponential decay, were also evaluated. However, these alternatives did not yield performance improvements and introduced additional hyperparameters requiring fine-tuning. Consequently, the simpler *StepLR* scheduler was retained. It is important to acknowledge, however, that the chosen scheduler configuration may result in extended training durations.

To ensure that the models were neither overfit nor overtrained, an early stopping callback was used, and subsequently the model that achieved the best validation performance during training was used for evaluation. Furthermore, in specific instances aimed at improving the interpretability of the results, additional custom callbacks were introduced to capture crucial information, such as gate values and other relevant metrics.

4.1.3 Hyperparameter Configuration

To tune hyperparameters, we use the train and validation subsets. The test subset is only used to report results of the model. We used a Bayesian Hyperparameter search algorithm [87] named Tree-structured Parzen Estimator (TPE), in *optuna*, as it efficiently balances exploration and exploitation, allowing us to identify optimal hyperparameter configurations with fewer evaluations compared to grid or random search.

The number of trials was set to 20. We also experimented with 100 trials but observed minimal differences in the results. Regarding the hyperparameters to be searched, these included the *mlp_units*, referring to the size of the hidden layers in the Feed-Forward Neural Network (FNN) components. In this case, we set the possible sizes as powers of two, giving the N-BEATS model the option to use more units to compensate for its lower number of parameters compared to the MoE adaptations. Other hyperparameters included in the search space were the *number of blocks per stack*, *number of lags*, and additional architectural parameters. For the MoE models, we also allowed the search to determine the optimal *number of experts* and *top_k* strategy. Table 4.2 presents the hyperparameter search configurations used in our experiments. The hyperparameters obtained from the search are detailed in Annex A.

4.1.4 Experimental Protocol

To fully evaluate the advantages of employing a MoE approach, it is essential to benchmark this methodology against established datasets and architectures. We compare `N-BEATS-MOE` with `N-BEATS`, seasonal naive, and several `N-BEATS-MOE` variants that add a MoE layer to `N-BEATS` using different strategies, previously presented at section 3.4. To ensure both fairness and reproducibility of the results, each model is trained ten times on each dataset frequency using the optimal set of hyperparameters identified during the tuning phase. We report median results over ten trials to account for variability due to random initialization and data sampling.

In addition, the results of the hyperparameter optimization process are preserved. This was made possible through a modification of the *nixtla* package, a critical step in enabling a subsequent analysis of the influence of various hyperparameters on model performance.

Table 4.2: Hyperparameter search space configuration used in experiments.

Parameter	Search Space Values
Model Structure	
`Number of Lags (input size)`	1h, 2h, 3h, 4h, 5h
`Forecast horizon (h)`	Dataset-dependent
`Stack types`	[*"identity"*, *"trend"*, *"seasonality"*]
`MLP units`	$[2^i, 2^i]$ for $i \in \{2,3,\dots,9\}$[a]
`Number of blocks`	$[n,n,n]$ for $n \in \{1,3,6,9\}$
`Scaler type`	*"identity"*
`Shared weights`	True
`Polynomial degree (d)`	2
`Number of experts`	2^j for $j \in \{1,2,3\}$[b]
`Top-k experts`	2^l for $l \in \{0,1,2,3\}$[b]
Training Parameters	
`Max training steps`	1000, 2500, 5000, 10000
`Batch size`	32, 64, 128, 256
`Windows batch size`	128, 256, 512, 1024
`Early stopping patience`	10, 20

[a] For `N-BEATS` and `N-BEATS-MOE` models, $i \in \{2,3,\dots,10\}$.
[b] Not applicable for `N-BEATS` and `N-BEATS-MOE` (set to 0).

The experimental outcomes are further analysed to compare the performance differences between models. The gate is examined to determine whether its behaviour aligns with the corresponding time series decompositions. Subsequently, the models are evaluated across a range of aspects.

4.2 Results

Table 4.3 summarizes the sMAPE scores across all datasets and frequency groups, comparing `N-BEATS-MOE` with other variants, seasonal naive, and also the original `N-BEATS` method.

Table 4.3: SMAPE across datasets, models, and frequency groups. Values in **bold** (red) represent the best (second best) approach.

Dataset	Freq./H.	N-BEATS	N-BEATS-MOE	MoEBlock	MoEShared	MoEScaled	Seas.Naive
M1	Yearly/2	10.87	**9.76**	10.90	9.97	11.62	11.89
	Quarterly/2	11.92	11.36	11.60	**11.28**	11.44	16.54
	Monthly/8	15.53	**14.84**	15.17	15.57	15.49	16.57
Tourism	Yearly/4	**26.48**	26.51	27.46	26.77	29.78	27.66
	Quarterly/8	19.29	20.96	**19.28**	19.95	19.45	21.10
	Monthly/18	**24.39**	25.13	24.77	24.64	24.76	27.30
M3	Yearly/6	**16.13**	16.27	16.29	16.17	16.28	17.87
	Quarterly/8	**9.01**	9.03	9.12	9.68	9.38	11.07
	Monthly/18	14.03	**13.96**	14.08	14.07	13.99	17.24
M4	Yearly/6	13.45	**13.31**	13.48	13.36	13.34	16.34
	Quarterly/8	9.92	**9.82**	9.87	10.02	9.93	12.52
	Monthly/18	**12.79**	12.99	13.10	13.12	12.95	15.99

Within the M1 dataset, the `N-BEATS-MOE` model consistently achieved the best overall performance, achieving the lowest sMAPE in both the *Yearly* (9.76%) and *Monthly* (14.84%) frequencies, and securing the second-best performance in the *Quarterly* category (11.36%). Notably, the `MoEShared` model also demonstrated strong performance, particularly in the *Yearly* frequency, where it achieved the second-lowest sMAPE (9.97%), and in the *Quarterly* group, where it delivered the best sMAPE (11.28%). Most of MoE-based models perform better than `N-BEATS`.

Turning to the Tourism dataset, the original `N-BEATS` architecture remained highly competitive, yielding the best scores in the *Yearly* (26.48%) and *Monthly* (24.39%) frequencies, and the second-best in the *Quarterly* segment (19.29%). All MoE-based models perform worse than `N-BEATS`.

In the M3 dataset, the standard N-BEATS model achieved the best performance in both the *Yearly* (16.13%) and *Quarterly* (9.01%) frequency groups. In contrast, `N-BEATS-MOE` delivered the strongest result in the *Monthly* category (13.96%) and attained the second-best performance in the *Quarterly* group (9.03%). Although certain mixture-based architectures demonstrated isolated improvements, none exhibited consistent gains across all frequency groups. In several instances, their performance was comparable to, or marginally inferior to, that of the benchmark model.

In the M4 dataset, `N-BEATS-MOE` once again delivered strong results, achieving the best scores in both the *Yearly* (13.31%) and *Quarterly* (9.82%) frequencies. The standard N-BEATS model attained the lowest error in the *Monthly* group (12.79%), followed by `MoEScaled` (12.95%).

4.3 Expert Specialization Analysis

By focusing on the interpretable variant of N-BEATS, which consists of trend and seasonal stacks, our approach enables a clear decomposition of forecasts into their trend and seasonal components. To understand how `N-BEATS-MOE` makes decisions, we look at how its stacks/experts specialize in different parts of a time series—like trend, seasonality, and noise. We ask: Are certain experts consistently chosen for specific structures? And do these choices match known signal decompositions like Seasonal-Trend decomposition using LOESS (STL)?

We explore this by analysing how the model routes inputs through its experts and how these routing decisions correspond to the underlying signal structure. This analysis sheds light on the roles each expert learns and offers insight into the model's internal workings and potential areas for improvement. In particular, we examine how expert selection aligns with the trend, seasonal, and residual components identified by STL decomposition, and we present case studies where `N-BEATS-MOE` outperforms N-BEATS, highlighting how expert specialization contributes to improved performance.

4.3.1 Specialization Analysis via STL Decomposition

To understand how each expert specializes and how the gating mechanism chooses among different time series—especially within the `N-BEATS-MOE` approach—we carried out a structured experiment illustrated in Figure 4.1. First, we trained a `N-BEATS-MOE` model composed of three stacks—*"identity"*, *"trend"*, and *"seasonal"*—each consisting of a single block on a chosen dataset (for example, M1 or M3), and saved the resulting parameters. Next, we applied STL decomposition to that same dataset to separate it into trend, seasonal, and residual components. Finally, we input each of these decomposed components into the pre-trained `N-BEATS-MOE` model and recorded several critical statistics for each expert (or expert stack): (a) the average posterior probability

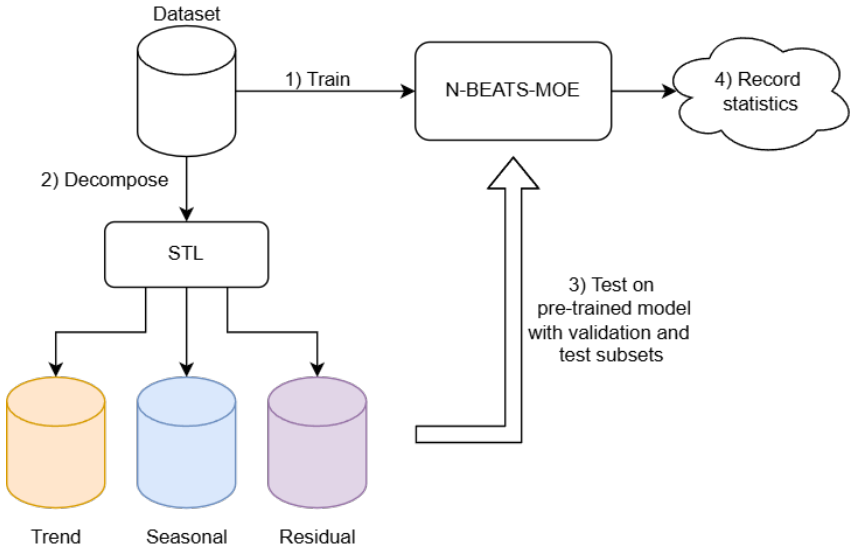


Figure 4.1: Experimental setup to analyse specialization in `N-BEATS-MOE`. We train the model on full time series, decompose the same data using STL, feed decomposed components into the pretrained model, and record how the gating mechanism assigns responsibility to each expert.

assigned to each expert across all series, (b) the number of times an expert's probability exceeded 0.5 (indicating a "dominant" assignment), and (c) how often (in %) each expert achieved the highest gating probability overall. For brevity, we present only the expert selection ratios. When analysing the results, it is important to focus on the differences in selection percentages across the decomposed components and how they change relative to each other.

The results of the expert selection analysis are summarized in Table 4.4.

In the **M3 dataset**, the gating aligns well with the decomposition. The *trend component* mainly selects the trend expert (55%), while the *seasonal component* favors the seasonal expert (45%), though differences are smaller. The *residual component* is mostly assigned to the trend expert (52%), with contributions from all experts.

For the **M4 dataset**, the trend expert dominates the *trend component* (60%). In the *seasonal component*, the trend expert still leads (50%), with the seasonal expert less favoured (10%), suggesting some seasonal patterns are captured by the trend and identity experts. The *residual component* shows a similar trend to M3.

In the **M1 dataset**, the pattern deviates: the *trend component* is mostly assigned to the seasonal expert (68%). For the *seasonal component*, the identity expert leads (44%), followed by trend and seasonal experts. This indicates weaker alignment with decomposition and may reflect dataset-specific gating behaviour. Notably, `N-BEATS-MOE` performs better here, suggesting the gate adapts beyond strict decomposition.

These results indicate that expert assignment is consistent with the intended decomposition in M3 and, to a lesser extent, M4. In contrast, the M1 results reflect different gating behavior, possibly influenced by dataset properties or the model's adaptation strategy.

Table 4.4: Expert selection ratios by dataset and decomposition component for the monthly frequency. Values show the proportion of times each MoE expert block was selected as most relevant for a given input.

Dataset	Component	Identity	Trend	Seasonal
M3	Trend	0.19	0.55	0.26
	Seasonal	0.18	0.37	0.45
	Residual	0.23	0.52	0.25
M1	Trend	0.31	0.01	0.68
	Seasonal	0.44	0.25	0.31
	Residual	0.48	0.24	0.28
M4	Trend	0.40	0.60	0.00
	Seasonal	0.40	0.50	0.10
	Residual	0.36	0.48	0.16

4.3.2 Forecast Decomposition in the M1 dataset

To understand what contributes to the performance of `N-BEATS-MOE` on the M1 dataset, we analyze the decomposition of its three output stacks on the monthly frequency group. We then compute the sMAPE on these series and compare the results with both the decomposition and the performance of a comparable N-BEATS model. Finally, we examine where our approach achieves better results and explore possible reasons for this difference.

Figure 4.2 presents a selection of time series in which our proposed method demonstrates a marked improvement over the baseline model. From it, it is possible to conclude the following: **(1)** Our approach is more capable of understanding the appropriated scaling/significance of each stack output. For instance, in the *ID1* series, our approach correctly identified that the *Trend component* was not relevant in this case, attributing it a score of 16.93% and scaling it down to an interval of $[\sim 0.4, \sim 0.9]$. In contrast, the baseline model wrongly overestimates the impact of this component, $[\sim 1, \sim 3]$, probably due to wrong generalizations. **(2)** Our model demonstrates superior capability in capturing seasonal fluctuations, including both drops and spikes, as exemplified by series *ID1*, *ID121*, and *ID241*. And the improved performance of the model can be attributed to its accuracy on these groups of series. **(3)** A stack with a high gate softmax value does not necessarily correspond to a larger scale of the output, but instead indicates a significant contribution, in %, to the output quality.

The previous analysis can be extended to other series (e.g., *ID110*, *ID112*, *ID113*, *ID115*, *ID119*, *ID120*, *ID123*, *ID242*, *ID268*, ...), and the conclusions would still hold.

4.4 Aspect-based Analysis

Traditional evaluation of forecasting models usually depends on aggregate metrics calculated across the entire dataset. While convenient, these metrics can obscure important variations in model behaviour under different data characteristics or forecast scenarios.

To address this limitation, we utilize *ModelRadar* [12], a recently proposed aspect-based evaluation framework for univariate time series forecasting. *ModelRadar* enables detailed performance assessment across multiple dimensions, such as stationarity, presence of anomalies, forecasting horizon, and problem difficulty. It also incorporates advanced comparison tools, including expected

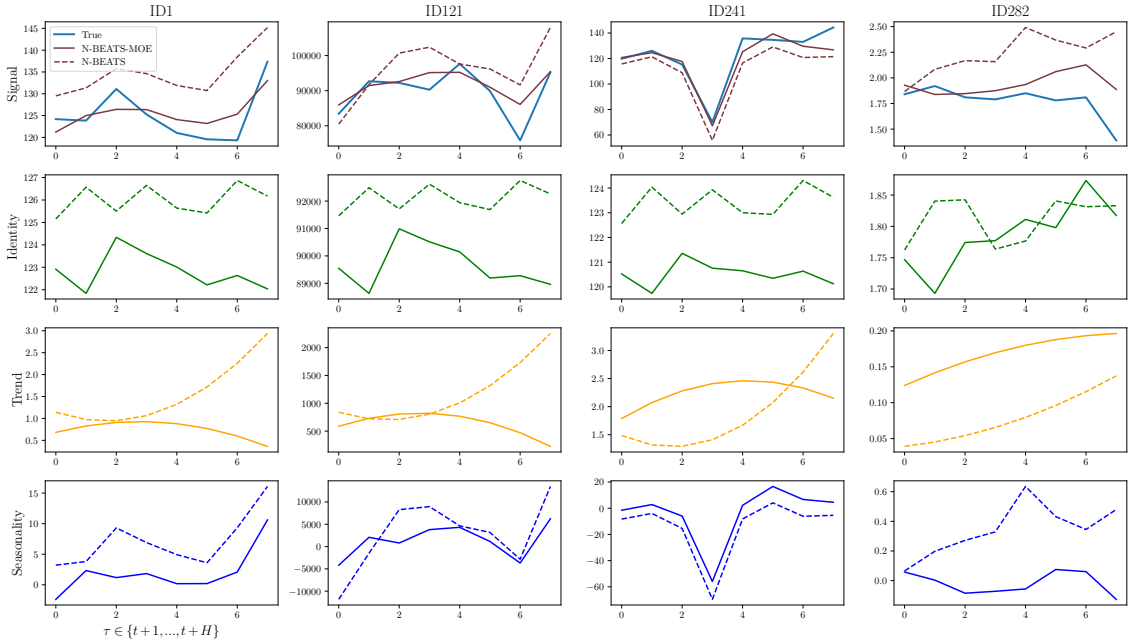


Figure 4.2: Forecast decomposition in the M1 monthly dataset for series *ID1*, *ID121*, *ID241*, and *ID282* using the `N-BEATS-MOE` and `N-BEATS` models stack outputs. The gating values attributed by the `N-BEATS-MOE` model are approximately $[0.275, 0.169, 0.556]$, $[0.319, 0.251, 0.430]$, $[0.196, 0.193, 0.611]$, and $[0.366, 0.232, 0.402]$, respectively. Corresponding SMAPE values for `N-BEATS-MOE` are 2.68%, 3.31%, 4.40%, and 10.15%, while for `N-BEATS` they are 7.34%, 8.03%, 10.26%, and 22.86%. These results illustrate that a high gate softmax value does not necessarily imply greater output numeric scaling but indicates a greater contribution, in %, to the output quality. To align with the *nixtla* decomposition function, the Naïve1 prediction was added to the identity stack, causing its larger numerical scale.

shortfall analysis and win/loss ratio evaluation, to support a more nuanced understanding of model robustness and suitability under varying conditions.

In this analysis, sMAPE is used as the evaluation metric. For the expected shortfall analysis, the Conditional Value at Risk (CVar) quantile is set to 75%, which means we focus on the worst 25% of time series results. Borrowed from financial risk analysis, expected shortfall measures how a model performs on its most difficult cases. In our study, it shows the average forecasting error on the time series where the model does the worst. Likewise, the hardness quantile threshold is set to 75%, where problem difficulty is defined based on the forecasting error of the seasonal naive baseline. Specifically, a time series is considered hard if its *Seasonal Naive* loss ranks in the top 25% across all series, indicating that such series are more challenging to model accurately. Anomalies are defined using the 99% prediction interval from the *Seasonal Naive* model. If the actual observed value falls outside this interval—either below the lower bound or above the upper bound—it is labelled as an anomaly. For stationarity and seasonality, the same method proposed by Cerqueira et al. [12] is used, respectively the Kwiatkowski–Phillips–Schmidt–Shin test [48] and Wang et al. [98].

All results are presented in Figure 4.3 and summarized as ranking averages across all datasets in Table 4.5. Annex B presents selected forecasts illustrating the different aspects of the analysis. Overall, `N-BEATS-MOE` achieved the best performance, with an average ranking of 1.58 across datasets.

In most aspects, N-BEATS and N-BEATS-MOE showed similar results. In particular, the expected shortfall, first and last horizon, and hardness showed almost no difference between the models. In seasonal series, our approach was slightly better than N-BEATS, particularly on the M1 dataset. In contrast, it performed slightly worse in the presence of anomalies. With respect to stationarity, none of the models demonstrated a clear advantage.

Table 4.5: Mean Rank across aspect-evaluation metric. Values in **bold** (red) represent the best (second best) approach.

	Anomalies	Exp. Shortfall	First horizon	Last horizon	On Hard	Seasonal	Stationary	Overall
MoEBlock	2.50	2.58	2.75	2.42	2.08	2.67	2.67	2.67
N-BEATS	**1.67**	2.08	1.83	2.00	**1.83**	1.83	2.33	1.83
N-BEATS-MOE	1.83	**1.83**	1.83	**1.92**	2.08	**1.58**	**2.00**	**1.58**
SeasonalNaive	4.00	3.50	3.58	3.67	4.00	3.92	3.00	3.92

4.5 Discussion

Several approaches and architectures were tested in the course of this study. In particular, adaptations were made to notable models such as Informer [105], N-BEATS [73], Time-MoE [84], among others. Following preliminary experiments, the focus was placed on adaptations of N-BEATS, as these not only yielded the most promising results in this context, but the original architecture itself has a well-established track record of strong performance on heterogeneous datasets.

For the MoE-based models, we used a linear gating mechanism. We also experimented with other types of gates, but none showed consistent improvements over the linear gate. For load balancing, we chose not to incorporate any explicit mechanism. We experimented with several approaches from the literature, including the DeepSeek *loss-free* method [97], Time-MoE [84] auxiliary loss, and a Kullback–Leibler divergence-based regularization [46]. However, these techniques generally led to a decrease in performance, and we observed that adding a simple LayerNorm before the input gate was enough to prevent mode collapse and guarantee a balanced use, which further supported our decision.

The results of this study give several insights into the potential applicability of MoE for time series forecasting, particularly in the context of heterogeneous datasets.

On a suite of 12 benchmark datasets, the results show that the proposed approach N-BEATS-MOE can improve the baseline in some datasets. Overall, the results suggest that our approach provides consistent improvements over N-BEATS, especially in datasets composed of heterogeneous time series (i.e. time series from various domains). In these (M1, M3, and M4), N-BEATS-MOE outperforms N-BEATS in 6 out of 9 cases. However, in the Turism datasets, composed of time series from a single domain, the proposed approach loses in all three frequency variants. Besides these results, N-BEATS-MOE outperforms seasonal naive in all cases, validating its forecasting accuracy. Except for a few cases, the proposed method also shows better performance than its variants that employ a MoE layer differently.

On the other hand, MoEs on the block level did not seem to bring improvements over the baseline. To this matter, it is worth considering the implications of scaling laws, which have been extensively studied in the context of LLMs. In such settings, MoEs are mainly employed as a means to efficiently increase the capacity of the model [18, 35], enabling scaling to larger datasets and parameter counts without a proportional increase in computational cost. However, the

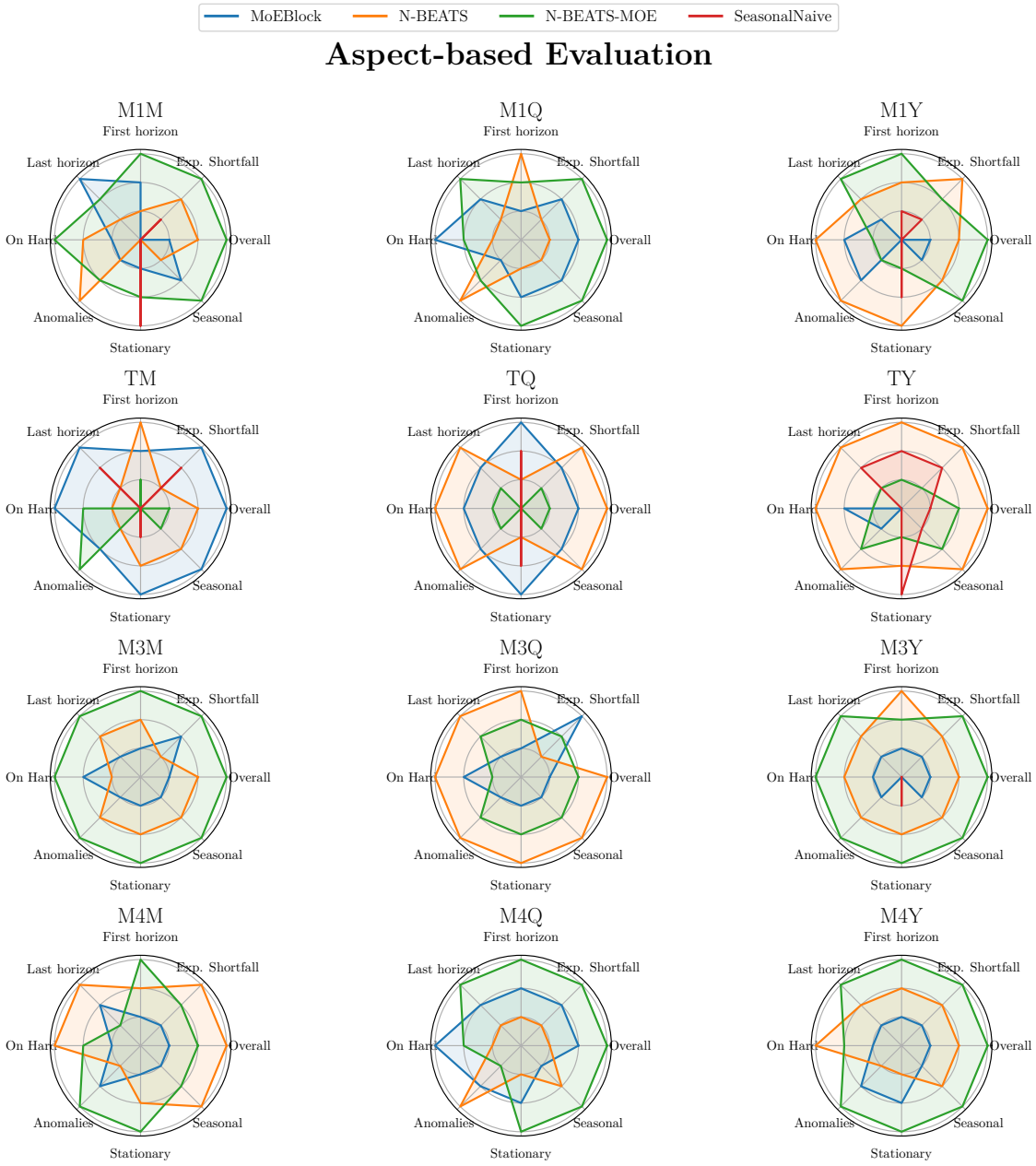


Figure 4.3: Model ranks across datasets–farther from center indicates better performance.

application considered here is fundamentally different: scaling capacity is not necessarily the main objective, nor is the training data sufficiently large to benefit from the conditional computation and specialization that MoEs provide. This mismatch in assumptions may partially explain the lack of gains observed at the block level, as the conditions under which MoEs tend to be most effective are not fully met in this scenario. So, in this matter, in most cases we are only scaling up the number of parameters, without increasing the dataset size.

Regarding interpretability, with this extension we add another layer of interpretability to `N-BEATS`. By looking at the weights assigned by the gate, it is possible to understand which stack the gate considers most important for the series, as demonstrated in section 4.3.2. However, the

behaviour of the gate and how it aligns with the pre-defined STL decomposition is still not well understood. By looking at the M1 decomposition for the monthly set, it was possible to observe that the model recognizes the importance of the seasonal stack for this dataset. Furthermore, the aspect-based evaluation performed in Section 4.4 supports this observation, as the proposed approach `N-BEATS-MOE` is able to outperform the baseline in the seasonal time series group. However, when presented with the seasonal component (from the STL decomposition), the gate struggled to assign the correct stack. We can only speculate why: perhaps the gate is overfitted to the original time series and does not know how to behave in the presence of different data scales (which may result from the data scale shift introduced by the normalization layer). Or, as observed in some trials, the *identity* stack can also act like a seasonal naïve model. And because the STL seasonal component can exhibit deterministic seasonality (that strictly repeats), it becomes very similar to a periodic time series, and is thus assigned to the identity stack. The misalignment between the trend component and the expected gate values may stem from a similar cause. Notably, the proportion of non-seasonal series identified through the aspect-based evaluation is relatively low (around 20%). Given that monthly datasets in these competitions typically exhibit strong seasonality [27], this observation may suggest that the trend component holds limited significance, particularly in the case of the M1 dataset.

Finally, the aspect-based analysis allowed us to better understand the differences in performance between the models. While `N-BEATS` and `N-BEATS-MOE` exhibited similar behaviour across most dimensions, notable differences emerged in the seasonal component. This indicates that our model is better able to forecast such series. While this analysis is presented, it is important to acknowledge potential threats to its validity. In particular, the use of average ranks across different datasets may not accurately reflect the true performance differences between models. For instance, even when differences in performance are negligible, a model may still be ranked higher and consequently perceived as superior, despite the lack of statistical significance. Furthermore, in this analysis, we run the model only once, so its results heavily depend on the random initialization and data sampling.

4.6 Threats to Validity

While we tried to conduct a well-structured experiment, the findings in this study are subject to validity threats that may affect the strength of our conclusions. These threats include:

- **Dataset Bias:** Our evaluation is conducted on a considerable number of datasets. From the 12 datasets used, 9 are heterogeneous. Prior work has shown that cherry-picking datasets can distort the perceived performance of a model [78], concluding that using around 3 to 6 datasets helps reduce the risk of incorrectly identifying an algorithm as the best one. This provides partial validation for our experimental setup, although additional experiments would be necessary to fully assess the performance of the approach under varying time series conditions.

- **Hyperparameter Optimization:** As we report results for a single tuned model, a hyperparameter search was necessary to ensure optimal performance. As discussed in Section 4.1.3, some choices—such as the stack types or loss function—were fixed during this process, potentially introducing bias into the results. Additionally, since hyperparameters were selected based on a simple training-validation split, there is a risk of overfitting to the validation set.

- **Ensemble results:** Oreshkin et al. [73] demonstrate the significant impact of ensembling on the overall performance of the N-BEATS model. While our approach achieves superior results on heterogeneous datasets when comparing individual models, ensembling remains a powerful technique for further improving model accuracy.
- **Reproducibility:** Ensuring that results are reproducible is essential for validating findings, maintaining transparency, and allowing others to build on the work with confidence. We made efforts to achieve reproducibility by controlling hyperparameters, fixing random seeds, and using fully reproducible software. However, differences in computational environments and software versions can still cause slight variations in the results.

To conclude, while our experimental design aimed to be rigorous and our efforts toward reproducibility were thorough, these identified threats to validity highlight the need for cautious interpretation of the results.

Chapter 5

Conclusion

This chapter marks the conclusion of this thesis. It begins by revisiting the research questions posed in Chapter 1 and the main hypothesis, followed by a summary of the key findings. Subsequently, the main contributions of this work are highlighted. Finally, potential directions for future research building upon these findings are discussed.

5.1 Research Questions 37
5.2 Main Contributions 38
5.3 Future Work 38

5.1 Research Questions

We are now in a position to answer the research questions outlined earlier:

RQ1: How can we augment the `N-BEATS` architecture with a MoE approach to better accommodate the *heterogeneous patterns* (e.g., varying trends and seasonal profiles) inherent in datasets comprising multiple univariate time series?

A: To explore this, we modified the original architecture of `N-BEATS` at both the block and stack levels. Our results indicate that the most effective way to handle *heterogeneous patterns* is to replace the standard sum aggregation of block outputs with a MoE mechanism. This allows the model to dynamically select and weight specialized blocks/stacks, leading to improved adaptability across diverse time series behaviours.

By analysing the gate behaviour and the approach results across multiple aspects, our results indicate that the proposed approach `N-BEATS-MOE` is better able to understand the different seasonal profiles of the series.

RQ2: To what extent do `N-BEATS` MoE-augmented approaches achieve better performance in terms of forecast accuracy?

A: The results show that the proposed approach improves upon `N-BEATS`, particularly on datasets composed of heterogeneous time series (M1, M3, M4), where our approach achieved the best performance on **6** out of **9** datasets.

However, in the Tourism datasets, composed of time series from a single domain, the proposed approach loses in all three frequency variants. Besides these results, `N-BEATS-MOE` outperforms seasonal naive in all cases, validating its forecasting accuracy. Except for a few cases, the proposed method also shows better performance than its variants that employ a MoE layer differently.

Specifically, we evaluated alternative designs in which the fully connected layer within each `N-BEATS` block was replaced by different kinds of MoEs layers. Nevertheless, the variants in question did not demonstrate superior performance.

RQ3: How does the inclusion of a MoE layer improve the interpretability of `N-BEATS`?

A: The inclusion of a MoE layer provides hints about which stack is more important for accurately predicting a given time series. This adds a layer of interpretability by showing how the model dynamically assigns importance across stacks. However, in terms of alignment with the STL decomposition, the results were inconclusive. While expert assignments sometimes aligned with the corresponding component (e.g., trend or seasonality), in other cases they did not.

5.2 Main Contributions

This thesis has made several contributions to the field of time series forecasting and to the application of MoEs within it. These contributions not only advance current research in the field, but also open up new avenues and raise important questions for future investigation:

1. **Novel approach for TSF with MoEs**: A novel architecture, `N-BEATS-MOE`, combining N-BEATS with MoE for improved handling of heterogeneous time series. Alongside the main approach, several alternative variants were proposed that, while not outperforming the primary approach, could offer advantages in different application settings.

2. **Set of experiments**: A set of experiments that show that `N-BEATS-MOE` performs competitively with N-BEATS. These also include an analysis of the behaviour of the gating network in an attempt to understand the behaviour of the introduced MoE layer.

In addition to these two main contributions, this work also resulted in a set of software extensions to the *nixtla* package, integrating our models, mixture-of-experts layers, balancing strategies, and training callbacks. The software is publicly available and ready to use[1]. A research paper summarizing the work has also been submitted to the ECML 2nd TempXAI Workshop for Explainable AI in Time Series and Data Streams [2] and is currently under review.

5.3 Future Work

While this thesis has demonstrated the potential of MoE architectures in time series forecasting, several directions remain open for future exploration:

[1] All implementations are available at `https://github.com/zaai-ai/mixture_of_experts_time_series`.
[2] https://lamarr-institute.github.io/tempxai25/

1. **Transfer learning with MoEs:** The N-BEATS architecture has been shown to be relevant for transfer learning, where it outperforms a variety of models even though it was not trained on them (zero-shot learning), as shown by Oreshkin et al. [72]. Building on this idea, we believe that MoEs present a promising avenue for further improving transfer learning capabilities in time series forecasting. In particular, our proposed `MoEBlock` framework allows for the integration of multiple pre-trained models by combining their respective blocks and introducing appropriate gating mechanisms at each block level.

   This approach could be extended to build a unified model that leverages pre-trained components across different temporal resolutions, input frequencies, and source domains, thus supporting more flexible and generalizable forecasting systems.

2. **Train on bigger heterogeneous datasets:** As previously mentioned, it may be worthwhile to test the `MoEBlock` variation on larger datasets. While in the context of LLMs, MoEs are primarily used as a way to efficiently scale model capacity to handle larger datasets and more complex data distributions [18, 35], in our case, the model size is increased without a corresponding increase in dataset size.

3. **Check our approach on the generic N-BEATS configuration:** The generic configuration of N-BEATS, unlike the interpretable variant, does not enforce predefined basis functions and instead learns them directly from the data. This flexibility allows it to potentially capture more complex patterns, particularly important in heterogeneous datasets [73].

   While the interpretable configuration of N-BEATS has shown comparable results against its generic counterpart, we believe it is worth investigating the performance of the generic architecture augmented with the proposed MoE layer. The main reason we did not test this variant was our focus on interpretability. Additionally, the generic configuration typically requires significantly more computational resources (over 30 stacks), making it computationally unfeasible to study during the dissertation period.

References

[1] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Gift-eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024. Cited on page 2.

[2] Fahad Alharbi and Dénes Csala. A Seasonal Autoregressive Integrated Moving Average with Exogenous Factors (SARIMAX) Forecasting Model-Based Time Series Approach. *Inventions*, 7:94, October 2022. Cited on page 9.

[3] George Athanasopoulos, Rob J. Hyndman, Haiyan Song, and Doris C. Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, July 2011. Cited on page 25.

[4] E. Michael Azoff. *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc., 1994. Cited on page 6.

[5] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling, April 2018. arXiv:1803.01271 [cs]. Cited on page 12.

[6] Anita Banerjee, Smeeta Kabadi, and Dostonbek Karimov. The transformative power of ai: Projected impacts on the global economy by 2030. *Review of Artificial Intelligence in Education*, 4:e020, 09 2023. Cited on page 1.

[7] I. A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3–31, December 2000. Cited on page 10.

[8] Maria Rosa Borges. Random walk tests for the lisbon stock market. *Applied Economics*, 43(5):631–639, 2011. Cited on page 8.

[9] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, May 2015. Google-Books-ID: rNt5CgAAQBAJ. Cited on page 9.

[10] WILLIAM BROCK, JOSEF LAKONISHOK, and BLAKE LeBARON. Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, 47(5):1731–1764, 1992. Cited on page 8.

[11] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A Survey on Mixture of Experts, August 2024. arXiv:2407.06204. Cited on pages 17 and 18.

[12] Vitor Cerqueira, Luis Roque, and Carlos Soares. Forecasting with deep learning: Beyond average of average of average performance. In *International Conference on Discovery Science*, pages 135–149. Springer, 2024. Cited on pages 31 and 32.

[13] Vítor Cerqueira, Luís Torgo, Fábio Pinto, and Carlos Soares. Arbitrated ensemble for time series forecasting. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part II 10*, pages 478–494. Springer, 2017. Cited on page 18.

[14] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Model selection for time series forecasting an empirical analysis of multiple estimators. *Neural processing letters*, 55(7):10073–10091, 2023. Cited on page 9.

[15] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza, Max Mergenthaler-Canseco, and Artur Dubrawski. N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting, November 2022. arXiv:2201.12886 [cs]. Cited on pages 2, 11, and 12.

[16] Zhang Cheng, N.N.A. Sjarif, and Roslina Ibrahim. Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14, 09 2023. Cited on page 1.

[17] Andrew A. Cook, Göksel Mısırlı, and Zhong Fan. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal*, 7(7):6481–6494, July 2020. Conference Name: IEEE Internet of Things Journal. Cited on page 7.

[18] Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models, January 2024. arXiv:2401.06066 [cs]. Cited on pages 2, 13, 14, 15, 16, 18, 20, 23, 33, and 39.

[19] Abhimanyu Das, Weihao Kong, Andrew Leach, Shaan Mathur, Rajat Sen, and Rose Yu. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*, 2023. Cited on page 11.

[20] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. Cited on page 17.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. arXiv:1810.04805. Cited on page 9.

[22] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. GLaM: Efficient scaling of language models with mixture-of-experts. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR, 17–23 Jul 2022. Cited on page 2.

[23] Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022. Special Issue: M5 competition. Cited on page 19.

[24] Philip Hans Franses. *Time series models for business and economic forecasting*. Cambridge university press, 1998. Cited on page 6.

[25] Everette S. Gardner and Ed. Mckenzie. Forecasting trends in time series. *Management Science*, 31(10):1237–1246, 1985. Cited on page 9.

[26] Everette S. Gardner Jr. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28, 1985. Cited on page 9.

[27] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive, 2021. Cited on page 35.

[28] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Global models for time series forecasting: A Simulation study. *Pattern Recognition*, 124:108441, April 2022. Cited on pages 1, 17, and 19.

[29] Jennifer N. Hird and Gregory J. McDermid. Noise reduction of NDVI time series: An empirical comparison of selected techniques. *Remote Sensing of Environment*, 113(1):248–258, January 2009. Cited on page 7.

[30] Charles C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, January 2004. Cited on page 9.

[31] Rob J. Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, May 2018. Google-Books-ID: _bBhDwAAQBAJ. Cited on pages 1, 7, and 8.

[32] Zoran Ivanovski, Ace Milenkovski, and Zoran Narasanov. Time series forecasting using a moving average model for extrapolation of number of tourist. *UTMS Journal of Economics*, 9(2):121–132, 2018. Cited on page 8.

[33] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, March 1991. Cited on pages 2 and 13.

[34] Tim Januschowski, Jan Gasthaus, Yuyang Wang, David Salinas, Valentin Flunkert, Michael Bohlke-Schneider, and Laurent Callot. Criteria for classifying forecasting methods. *International Journal of Forecasting*, 36(1):167–177, January 2020. Cited on pages 7 and 8.

[35] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian

Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of Experts, January 2024. arXiv:2401.04088 [cs]. Cited on pages 2, 9, 13, 14, 15, 16, 17, 18, 20, 33, and 39.

[36] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. Cited on page 9.

[37] Michael I. Jordan and Robert A. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 6(2):181–214, March 1994. Cited on page 16.

[38] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture*, 147:70–90, April 2018. Cited on page 9.

[39] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. arXiv:2001.08361 [cs]. Cited on page 16.

[40] Zahra Karevan and Johan A. K. Suykens. Transductive LSTM for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, May 2020. Cited on page 6.

[41] Juan Pablo Karmy and Sebastián Maldonado. Hierarchical time series forecasting via support vector regression in the european travel retail industry. *Expert Systems with Applications*, 137:59–73, 2019. Cited on page 1.

[42] Lutz Kilian and Mark P. Taylor. Why is it so difficult to beat the random walk forecast of exchange rates? *Journal of International Economics*, 60(1):85–107, 2003. Emperical Exchange Rate Models. Cited on page 8.

[43] Jongseon Kim, Hyungjoon Kim, HyunGi Kim, Dongjun Lee, and Sungroh Yoon. A Comprehensive Survey of Time Series Forecasting: Architectural Diversity and Open Challenges, October 2024. arXiv:2411.05793 [cs]. Cited on page 7.

[44] F. Kozin. Autoregressive moving average models of earthquake records. *Probabilistic Engineering Mechanics*, 3(2):58–63, 1988. Cited on page 8.

[45] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for Deep Learning: A Taxonomy, October 2017. arXiv:1710.10686. Cited on page 11.

[46] Neeraj Kumar, Ankur Narang, and Brejesh Lall. Kullback–leibler divergence-based regularized normalization for low-resource tasks. *IEEE Transactions on Artificial Intelligence*, 5(6):2638–2650, 2024. Cited on page 33.

[47] Vladimír Kunc and Jiří Kléma. Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks, February 2024. arXiv:2402.09092. Cited on page 10.

[48] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992. Cited on page 32.

[49] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015. Cited on page 10.

[50] Weikai Li, Ding Wang, Zijian Ding, Atefeh Sohrabizadeh, Zongyue Qin, Jason Cong, and Yizhou Sun. Hierarchical Mixture of Experts: Generalizable Learning for High-Level Synthesis, October 2024. arXiv:2410.19225 [cs]. Cited on page 16.

[51] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, October 2021. Cited on pages 11 and 12.

[52] Xiachong Lin, Arian Prabowo, Imran Razzak, Hao Xue, Matthew Amos, Sam Behrens, and Flora D. Salim. A gap in time: The challenge of processing heterogeneous iot data in digitalized buildings. *IEEE Pervasive Computing*, pages 1–13, 2025. Cited on page 7.

[53] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, December 2017. Cited on page 9.

[54] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. Moirai-MoE: Empowering Time Series Foundation Models with Sparse Mixture of Experts, October 2024. arXiv:2410.10469. Cited on page 17.

[55] 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. *Journal of Forecasting*, 1(2):111–153, 1982. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/for.3980010202. Cited on page 25.

[56] Spyros Makridakis and Michèle Hibon. The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, October 2000. Cited on pages 3, 11, and 25.

[57] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, October 2018. Cited on pages 2, 3, 11, 13, and 25.

[58] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. Cited on pages 2 and 11.

[59] Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42(2):275–293, August 2014. Cited on pages 2, 14, and 15.

[60] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, December 1943. Cited on page 10.

[61] Richard A. Meese and Kenneth Rogoff. Empirical exchange rate models of the seventies: Do they fit out of sample? *Journal of International Economics*, 14(1):3–24, 1983. Cited on page 8.

[62] Nijat Mehdiyev, Johannes Lahann, Andreas Emrich, David Enke, Peter Fettke, and Peter Loos. Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry. *Procedia Computer Science*, 114:242–249, January 2017. Cited on page 6.

[63] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017. Cited on page 9.

[64] Pablo Montero-Manso and Rob J. Hyndman. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37(4):1632–1653, October 2021. Cited on pages 2 and 17.

[65] Mohammad Amin Morid, Olivia R. Liu Sheng, and Joseph Dunbar. Time series prediction using deep learning methods in healthcare. *ACM Trans. Manage. Inf. Syst.*, 14(1), January 2023. Cited on page 1.

[66] Siyuan Mu and Sen Lin. A comprehensive survey of mixture-of-experts: Algorithms, theory, and applications, 2025. Cited on pages 2, 14, 17, and 20.

[67] Alexis Mérigaud and John V. Ringwood. Free-surface time-series generation for wave energy applications. *IEEE Journal of Oceanic Engineering*, 43(1):19–35, 2018. Cited on page 1.

[68] Aishik Nagar, Ishaan Singh Rawal, Mansi Dhanania, and Cheston Tan. How do transformer embeddings represent compositions? a functional analysis, 2025. Cited on page 20.

[69] Ronghao Ni, Zinan Lin, Shuaiqi Wang, and Giulia Fanti. Mixture-of-Linear-Experts for Long-term Time Series Forecasting. In *International Conference on Artificial Intelligence and Statistics*, pages 4672–4680. PMLR, 2024. Cited on page 17.

[70] E. Nwogu, Iheanyi Iwueze, Kelechukwu Dozie, and Mbachu Ifeyinwa. Choice between Mixed and Multiplicative Models in Time Series Decomposition. *International Journal of Statistics and Applications*, pages 153–159, February 2019. Cited on page 6.

[71] Kin G. Olivares, Cristian Challu, Grzegorz Marcjasz, Rafał Weron, and Artur Dubrawski. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. *International Journal of Forecasting*, 39(2):884–900, April 2023. arXiv:2104.05522 [cs]. Cited on page 12.

[72] Boris Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:9242–9250, 05 2021. Cited on page 39.

[73] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, February 2020. arXiv:1905.10437. Cited on pages 1, 2, 11, 12, 13, 22, 23, 33, 36, and 39.

[74] Adhistya Erna Permanasari, Indriana Hidayah, and Isna Alfi Bustoni. Sarima (seasonal arima) implementation on time series to forecast the number of malaria incidence. In *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 203–207, 2013. Cited on page 9.

[75] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. From Sparse to Soft Mixtures of Experts, May 2024. arXiv:2308.00951 [cs]. Cited on page 16.

[76] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale, 2022. Cited on page 17.

[77] Ali Maboudi Reveshti, Elham Khosravirad, Ahmad Karimi Rouzbahani, Saeed Khakshouri Fariman, Hamidreza Najafi, and Ali Peivandizadeh. Energy consumption prediction in an office building by examining occupancy rates and weather parameters using the moving average method and artificial neural network. *Heliyon*, 10(4), 2024. Cited on page 8.

[78] Luis Roque, Carlos Soares, Vitor Cerqueira, and Luis Torgo. Cherry-picking in time series forecasting: How to select datasets to make your model shine, 2024. Cited on page 35.

[79] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. Cited on page 10.

[80] David Salinas, Valentin Flunkert, and Jan Gasthaus. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks, February 2019. arXiv:1704.04110. Cited on page 12.

[81] Rajat Sen, Hsiang-Fu Yu, and Inderjit S. Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019. Cited on pages 7 and 17.

[82] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer, January 2017. arXiv:1701.06538 [cs]. Cited on pages 15 and 18.

[83] Shaohuai Shi, Xinglin Pan, Xiaowen Chu, and Bo Li. Pipemoe: Accelerating mixture-of-experts through adaptive pipelining. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, 2023. Cited on page 17.

[84] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-Scale Time Series Foundation Models with Mixture of Experts, February 2025. arXiv:2409.16040 [cs]. Cited on pages 12, 17, 18, 20, and 33.

[85] Nripendra Kumar Singh and Khalid Raza. *Medical Image Generation Using Generative Adversarial Networks: A Review*, pages 77–96. Springer Singapore, Singapore, 2021. Cited on page 9.

[86] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International journal of forecasting*, 36(1):75–85, 2020. Cited on page 11.

[87] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms, August 2012. arXiv:1206.2944 [stat]. Cited on page 27.

[88] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. Cited on page 11.

[89] Jiaqi Sun and Junhai Zhai. MCNet: Multivariate Long-Term Time Series Forecasting with Local and Global Context Modeling. *Information Sciences*, page 120864, 2024. Publisher: Elsevier. Cited on pages 7 and 17.

[90] Yanru Sun, Zongxia Xie, Emadeldeen Eldele, Dongyue Chen, Qinghua Hu, and Min Wu. Learning Pattern-Specific Experts for Time Series Forecasting Under Patch-level Distribution Shift, October 2024. arXiv:2410.09836 [cs]. Cited on page 18.

[91] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5956–5963, 2020. Issue: 04. Cited on pages 7 and 17.

[92] Zaiyong Tang, Chrys De Almeida, and Paul A Fishwick. Time series forecasting using neural networks vs. box-jenkins methodology. *Simulation*, 57(5):303–310, 1991. Cited on page 11.

[93] William Toner, Thomas L Lee, Artjom Joosen, Rajkarn Singh, and Martin Asenov. Performance of zero-shot time series foundation models on cloud data. In *I Can't Believe It's Not Better: Challenges in Applied Deep Learning*, 2025. Cited on page 17.

[94] P. F. Verdes, P. M. Granitto, H. D. Navone, and H. A. Ceccatto. Forecasting chaotic time series: Global vs. local methods. *Novel Intelligent Automation and Control Systems*, 1:129–145, 1998. Cited on page 17.

[95] An Wang, Xingwu Sun, Ruobing Xie, Shuaipeng Li, Jiaqi Zhu, Zhen Yang, Pinxue Zhao, J. N. Han, Zhanhui Kang, Di Wang, Naoaki Okazaki, and Cheng zhong Xu. Hmoe: Heterogeneous mixture of experts for language modeling, 2024. Cited on pages 2 and 20.

[96] Guangyu Wang, Yujie Chen, Ming Gao, Zhiqiao Wu, Jiafu Tang, and Jiabi Zhao. A Time Series is Worth Five Experts: Heterogeneous Mixture of Experts for Traffic Flow Prediction, September 2024. arXiv:2409.17440. Cited on page 17.

[97] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-Loss-Free Load Balancing Strategy for Mixture-of-Experts, August 2024. arXiv:2408.15664 [cs]. Cited on pages 18 and 33.

[98] Xiaozhe Wang, Kate Smith-Miles, and Rob Hyndman. Characteristic-based clustering for time series data. *Data Min. Knowl. Discov.*, 13:335–364, 09 2006. Cited on page 32.

[99] Bernard Widrow. An adaptive'adaline'neuron using chemical'memistors', 1553-1552, 1960. Cited on page 10.

[100] Peter R. Winters. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3):324–342, 1960. Publisher: INFORMS. Cited on page 9.

[101] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022. Cited on page 11.

[102] Kun Yi, Qi Zhang, Wei Fan, Shoujin Wang, Pengyang Wang, Hui He, Defu Lian, Ning An, Longbing Cao, and Zhendong Niu. Frequency-domain mlps are more effective learners in time series forecasting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc. Cited on page 17.

[103] 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*, volume 37, pages 11121–11128, 2023. Cited on pages 12 and 18.

[104] Wenbai Zeng. Application of ar, ma, and arma models in financial time series analysis. *Advances in Economics, Management and Political Sciences*, 141:177–184, 12 2024. Cited on page 8.

[105] 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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:11106–11115, 05 2021. Cited on pages 12 and 33.

[106] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V. Le, and James Laudon. Mixture-of-Experts with Expert Choice Routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, December 2022. Cited on page 18.

[107] Eric Zivot and Jiahui Wang. *Vector Autoregressive Models for Multivariate Time Series*, pages 369–413. Springer New York, New York, NY, 2003. Cited on page 8.

Appendix A

Hyperparameter Sensitivity and Findings

This annex presents and analyses the results of the hyperparameter search mentioned in section 4.1.3. We look deeper at the most important hyperparameters and how they change across different datasets and frequencies. Table A.1 presents the resulting hyperparameters after the search. The main key findings are the following:

- **Input Size Patterns**
  - **Yearly Data:**
    - Typical input size range: 3–6 time steps
    - Reflects limited historical context
  - **Quarterly Data:**
    - Typical input size range: 4–8 time steps
    - Accounts for seasonal context
  - **Monthly Data:**
    - Typical input size range: 12–54 time steps
    - Reflects larger available historical context
  - Overall, the M4 dataset favors larger input sizes relative to other datasets, likely due to the availability of richer temporal context and/or more complex seasonal patterns. Apart from that, the input sizes reflect the expected values and does not change much across models.
- **MLP Hidden Size Patterns**
  - The hidden size of the MLP layers shows notable variation across models and datasets.
  - The `MoEBlock` and or `MoEShared` models tend to use smaller hidden sizes, possibly to compensate for the number of experts.
  - In contrast, models without MoE components at the block level, such as standard N-BEATS and `N-BEATS-MOE`, often use substantially larger hidden sizes.
  - Overall, higher hidden sizes appear to correlate with harder and larger datasets.

- **Sparsity vs Density**
  - In terms of sparsity versus density within the MoE approach at the block level, a larger number of experts is generally preferred in conjunction with sparse selection (i.e., top-k less than 2), rather than adopting a dense configuration. These experts also tend to have smaller individual capacities, which translates into smaller active parameters per input.

Table A.1: Optimized Hyperparameters for N-BEATS and variants across datasets

Dataset	Fq.	Model	Inp. Size	MLP	Blocks	Steps	Batch	WinBatch	Exp.	Top-k
M1	M	N-BEATS	12	16	9	2500	32	256	-	-
		MoEBlock	12	128	3	2500	32	1024	8	2
		MoEShared	12	4	6	5000	32	512	8	4
		N-BEATS-MOE	12	128	6	1000	32	128	-	-
		MoEScaled	12	16	9	10000	32	128	8	8
	Q	N-BEATS	4	128	9	10000	64	256	-	-
		MoEBlock	4	128	1	2500	64	1024	8	4
		MoEShared	4	8	3	2500	64	1024	4	2
		N-BEATS-MOE	4	64	1	2500	32	512	-	-
		MoEScaled	4	128	3	2500	64	1024	8	4
	Y	N-BEATS	3	512	9	10000	32	512	-	-
		MoEBlock	3	4	6	2500	128	128	8	2
		MoEShared	3	8	6	10000	128	128	2	1
		N-BEATS-MOE	3	4	6	2500	64	512	-	-
		MoEScaled	3	32	3	2500	32	1024	8	4
Tourism	M	N-BEATS	24	256	6	1000	64	1024	-	-
		MoEBlock	24	64	6	10000	64	1024	4	4
		MoEShared	24	128	6	10000	64	512	8	8
		N-BEATS-MOE	24	1024	6	10000	128	512	-	-
		MoEScaled	24	512	1	1000	32	1024	8	2
	Q	N-BEATS	4	8	9	5000	32	512	-	-
		MoEBlock	4	64	9	10000	32	128	4	2
		MoEShared	4	64	9	5000	32	512	8	2
		N-BEATS-MOE	4	64	6	10000	256	128	-	-
		MoEScaled	4	256	6	10000	256	256	2	2
	Y	N-BEATS	3	512	1	1000	256	256	-	-
		MoEBlock	3	128	3	2500	256	1024	8	2
		MoEShared	3	8	6	5000	256	256	8	1
		N-BEATS-MOE	3	128	6	1000	128	1024	-	-
		MoEScaled	3	128	6	5000	256	512	4	4
M3	M	N-BEATS	36	128	9	2500	128	256	-	-
		MoEBlock	18	32	9	2500	256	512	4	4
		MoEShared	36	256	1	1000	256	1024	2	2
		N-BEATS-MOE	54	256	9	1000	256	512	-	-
		MoEScaled	18	512	1	1000	128	1024	8	2
	Q	N-BEATS	8	256	3	1000	32	256	-	-
		MoEBlock	8	4	6	5000	128	1024	8	8
		MoEShared	16	32	3	10000	32	512	2	1
		N-BEATS-MOE	8	32	9	2500	128	1024	-	-
		MoEScaled	16	64	1	1000	128	1024	4	2
	Y	N-BEATS	6	8	6	5000	256	256	-	-
		MoEBlock	6	4	1	1000	128	512	8	1
		MoEShared	6	8	6	2500	128	128	8	2
		N-BEATS-MOE	6	16	6	10000	32	1024	-	-
		MoEScaled	6	4	9	5000	256	128	8	1
M4	M	N-BEATS	36	128	6	5000	64	512	-	-
		MoEBlock	54	32	1	10000	256	512	8	2
		MoEShared	36	64	1	5000	32	256	8	8
		N-BEATS-MOE	54	1024	9	10000	32	1024	-	-
		MoEScaled	36	256	6	5000	64	128	8	4
	Q	N-BEATS	40	128	6	2500	32	512	-	-
		MoEBlock	16	64	9	10000	128	128	8	8
		MoEShared	32	512	3	10000	128	512	8	1
		N-BEATS-MOE	16	256	9	5000	64	512	-	-
		MoEScaled	32	512	3	2500	128	1024	8	8
	Y	N-BEATS	18	512	6	5000	256	512	-	-
		MoEBlock	12	8	1	10000	128	128	2	2
		MoEShared	30	256	3	2500	256	128	8	2
		N-BEATS-MOE	18	512	1	5000	128	512	-	-
		MoEScaled	30	256	3	2500	256	128	8	2

Note: Fq. = Frequency. Inp Size = Input size (or number of auto-regressive lags). MLP = hidden size of the mlp. Blocks = number of blocks per stack. Exp. = Experts. Models with "-" do not use MoE on block level. All models use: `shared_weights=True`, `scaler_type="identity"`, `early_stop_patience_steps=10-20`, `start_padding_enabled=True`.

Appendix B

Forecast on Aspect-based Series

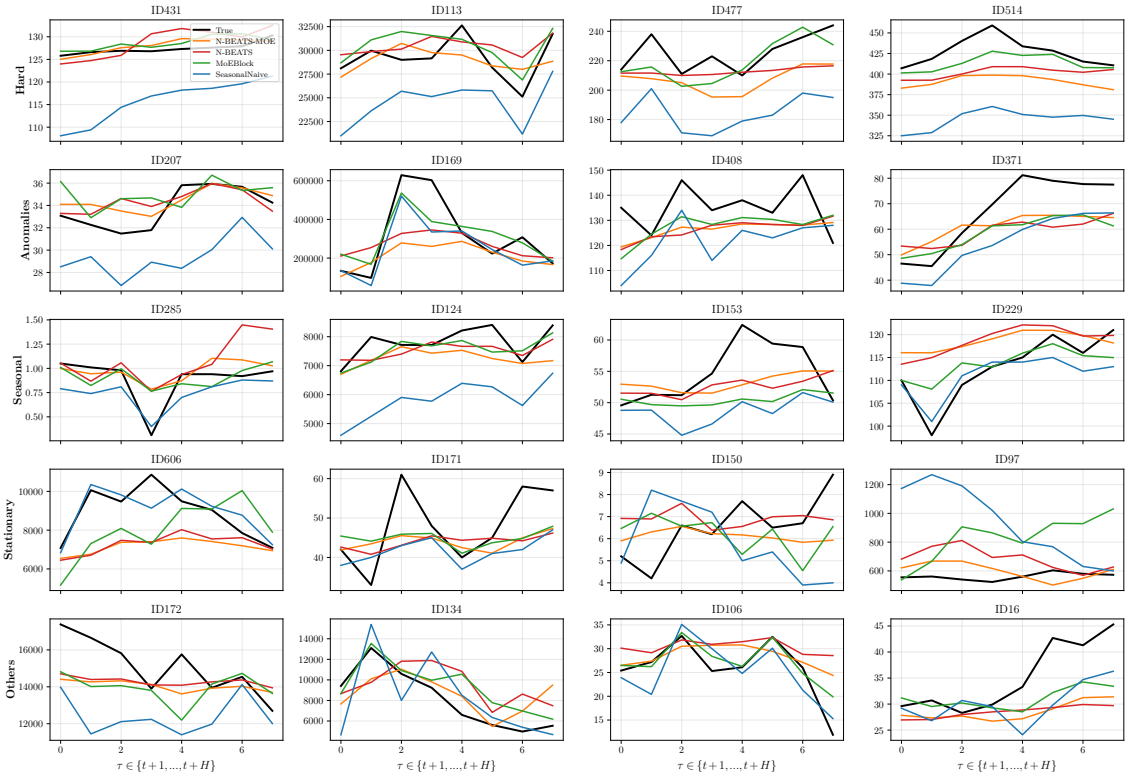


Figure B.1: Aspect-based forecast on the M1 monthly dataset

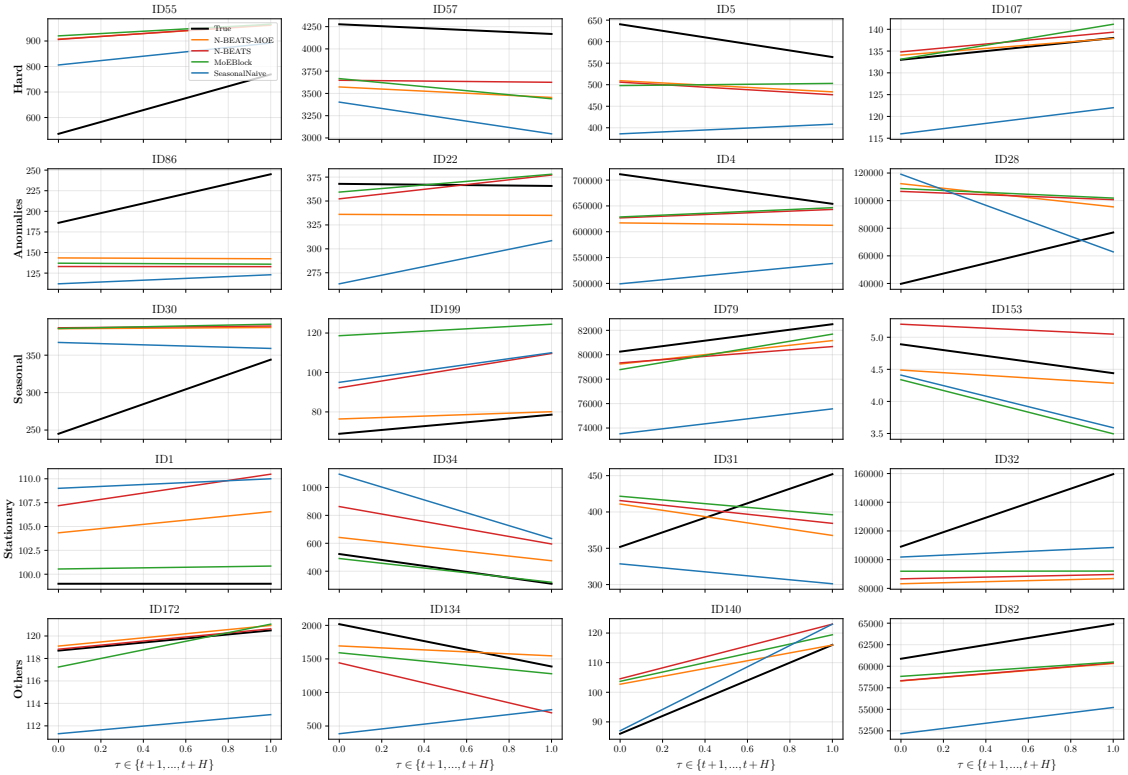


Figure B.2: Aspect-based forecast on the M1 quarterly dataset

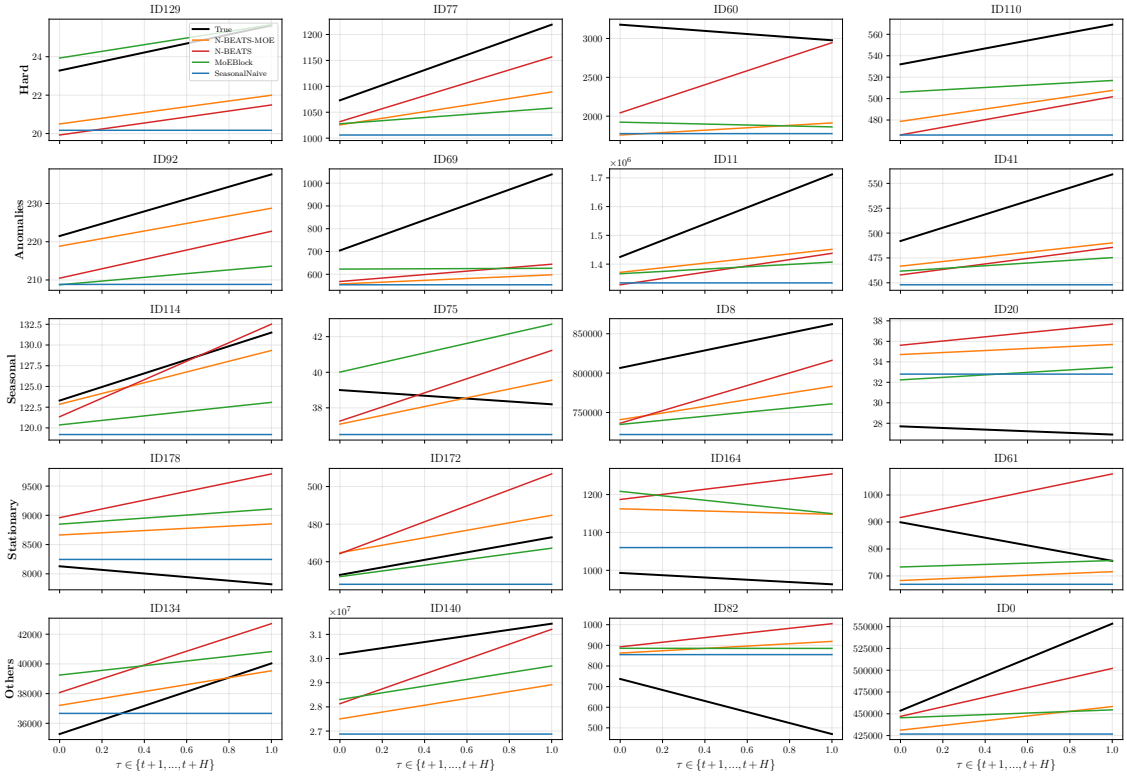


Figure B.3: Aspect-based forecast on the M1 yearly dataset

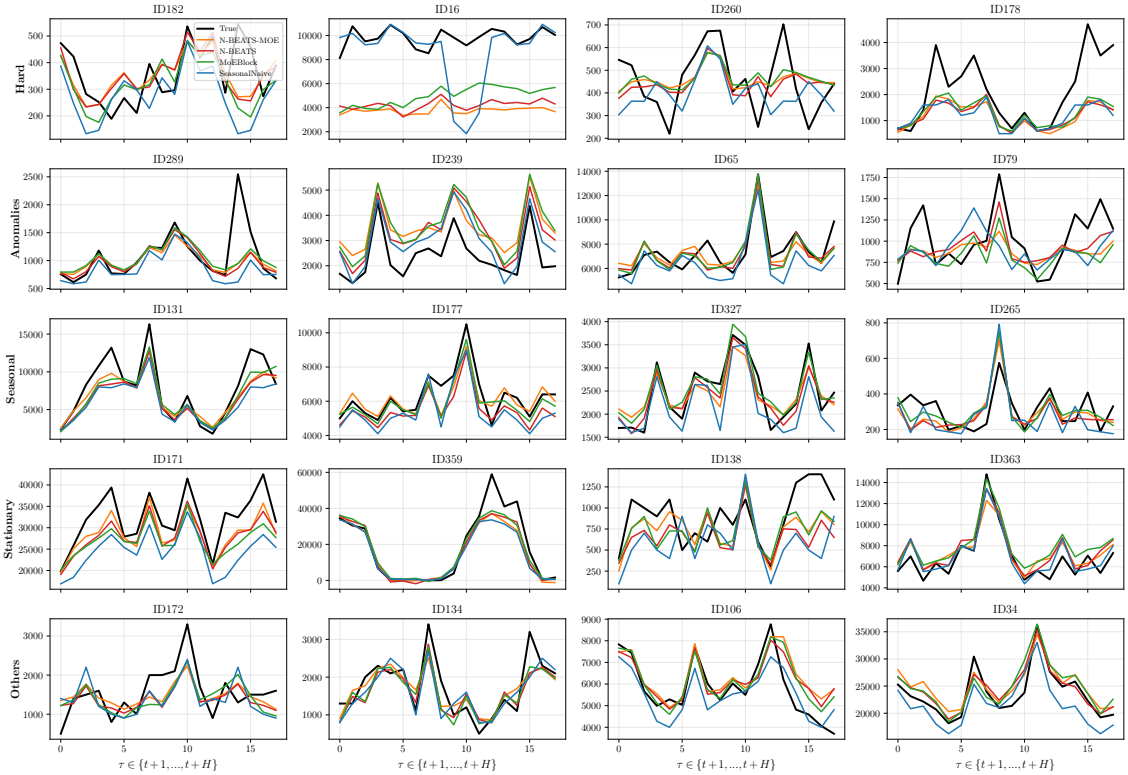


Figure B.4: Aspect-based forecast on the Tourism monthly dataset

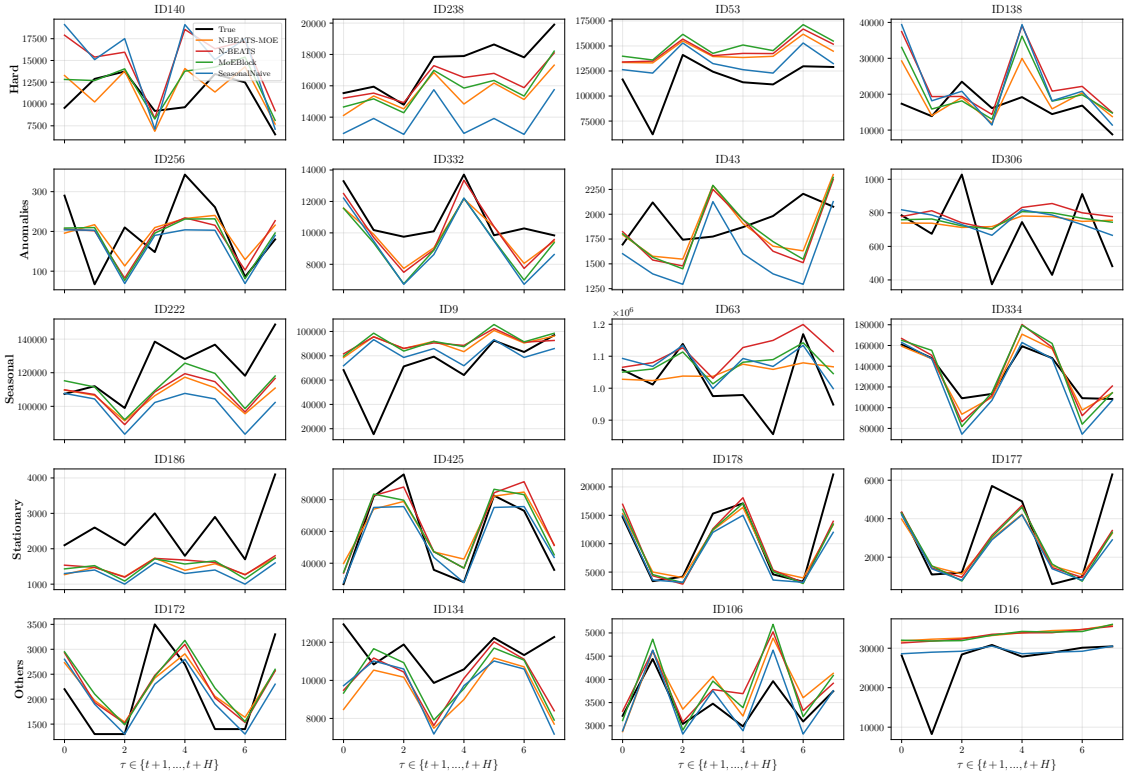


Figure B.5: Aspect-based forecast on the Tourism quarterly dataset

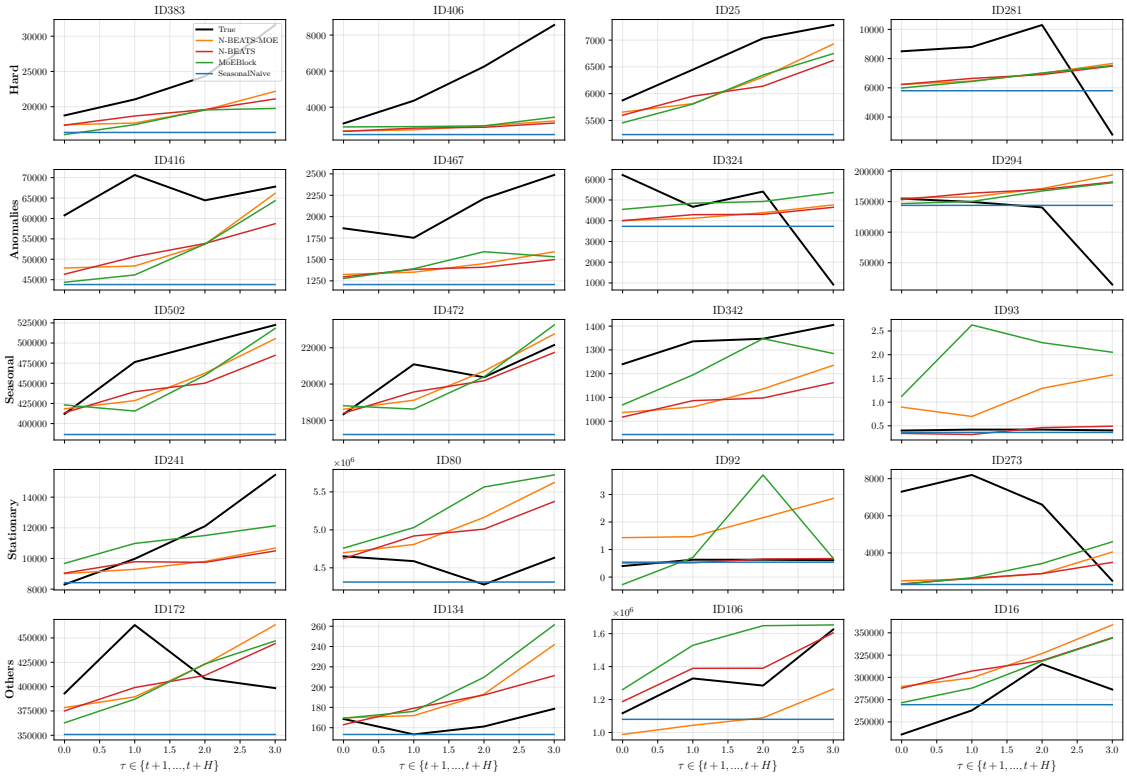


Figure B.6: Aspect-based forecast on the Tourism yearly dataset

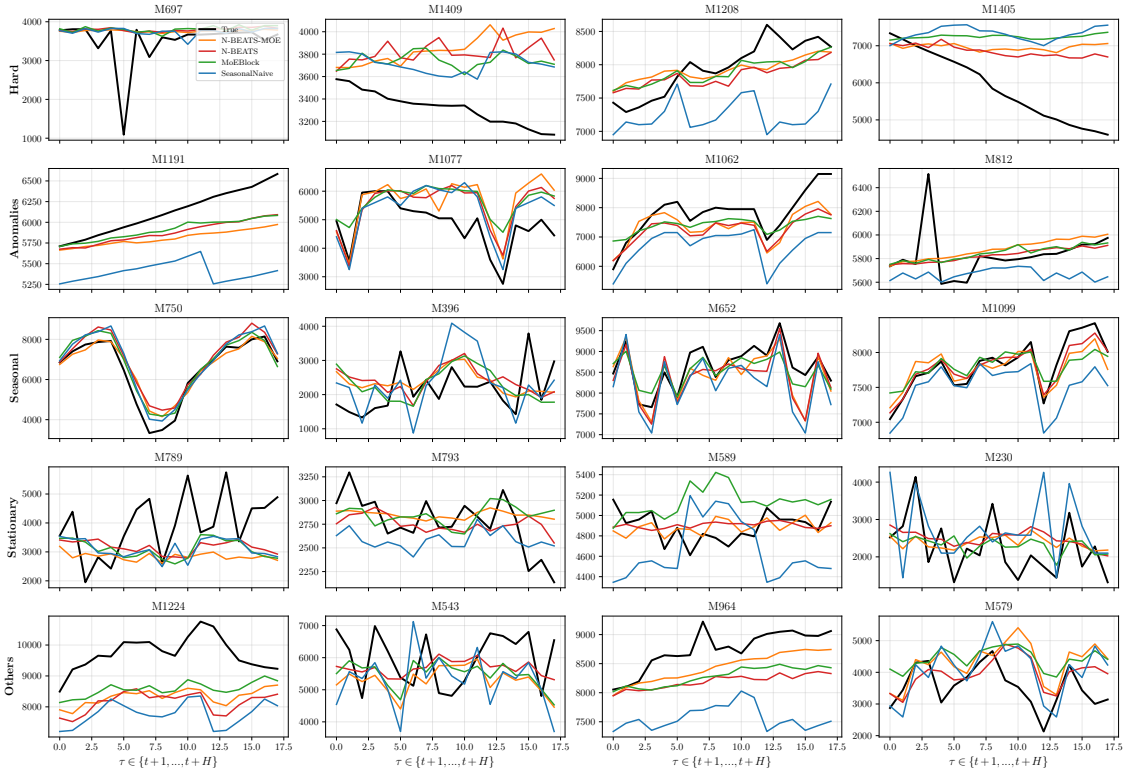


Figure B.7: Aspect-based forecast on the M3 monthly dataset

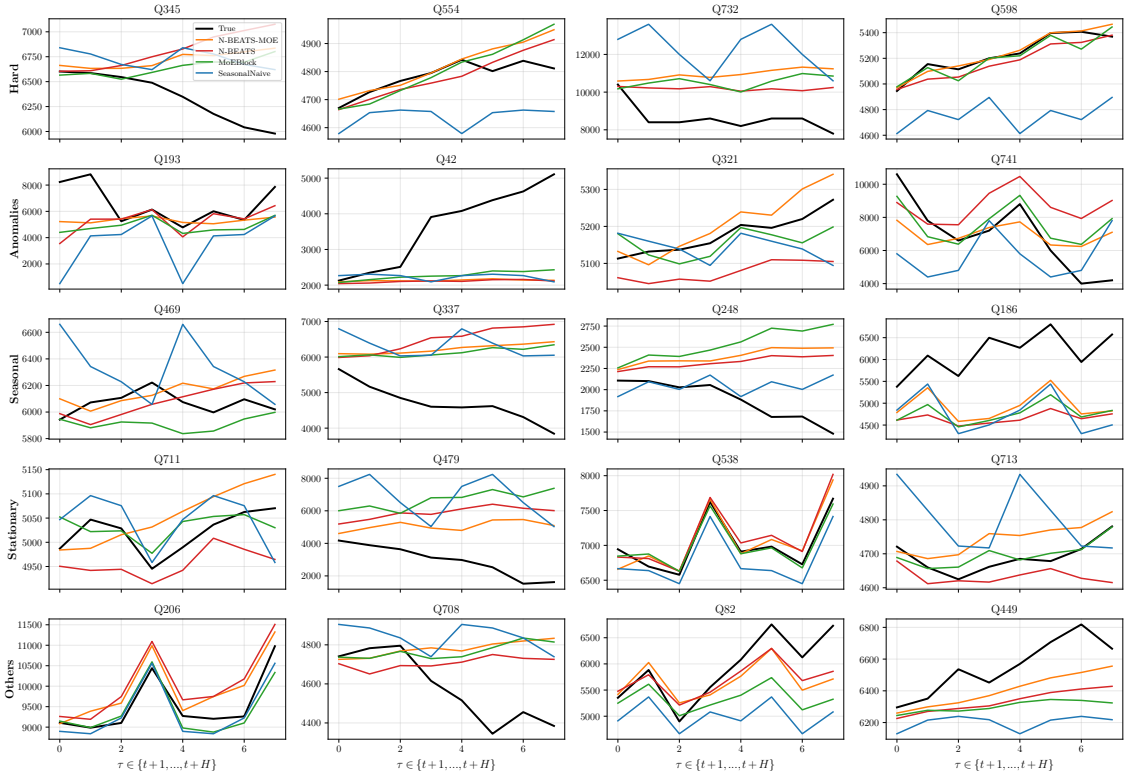


Figure B.8: Aspect-based forecast on the M3 quarterly dataset

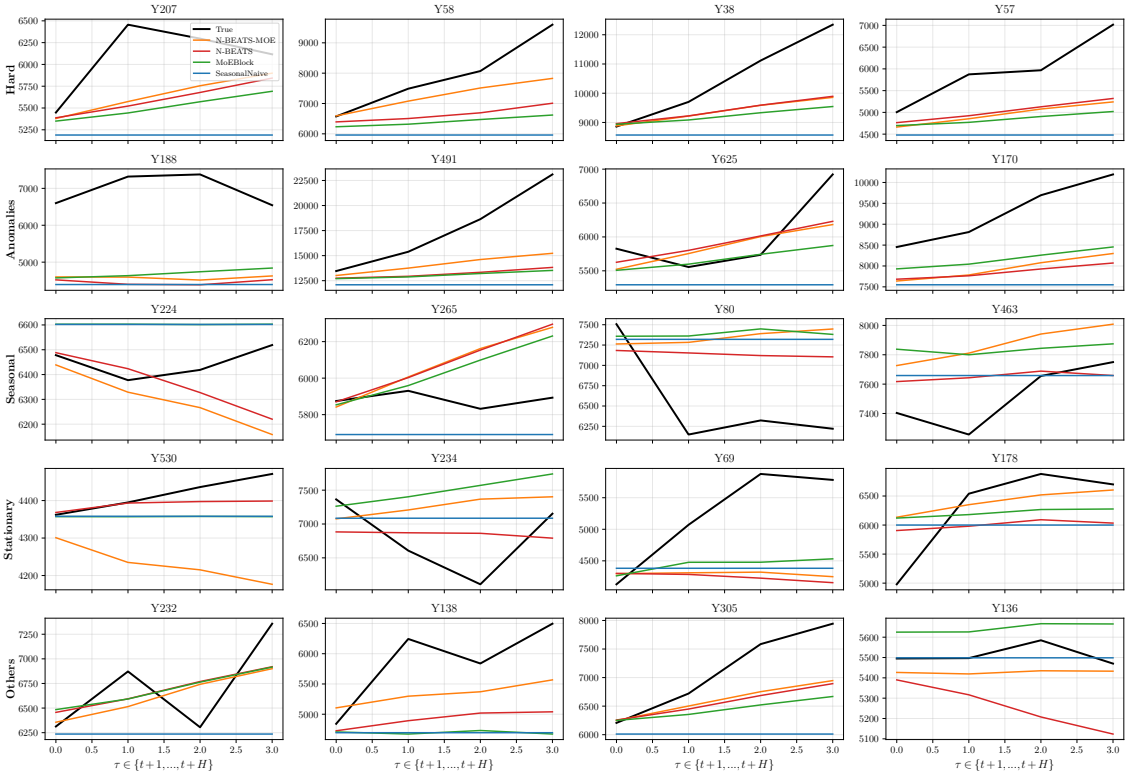


Figure B.9: Aspect-based forecast on the M3 yearly dataset

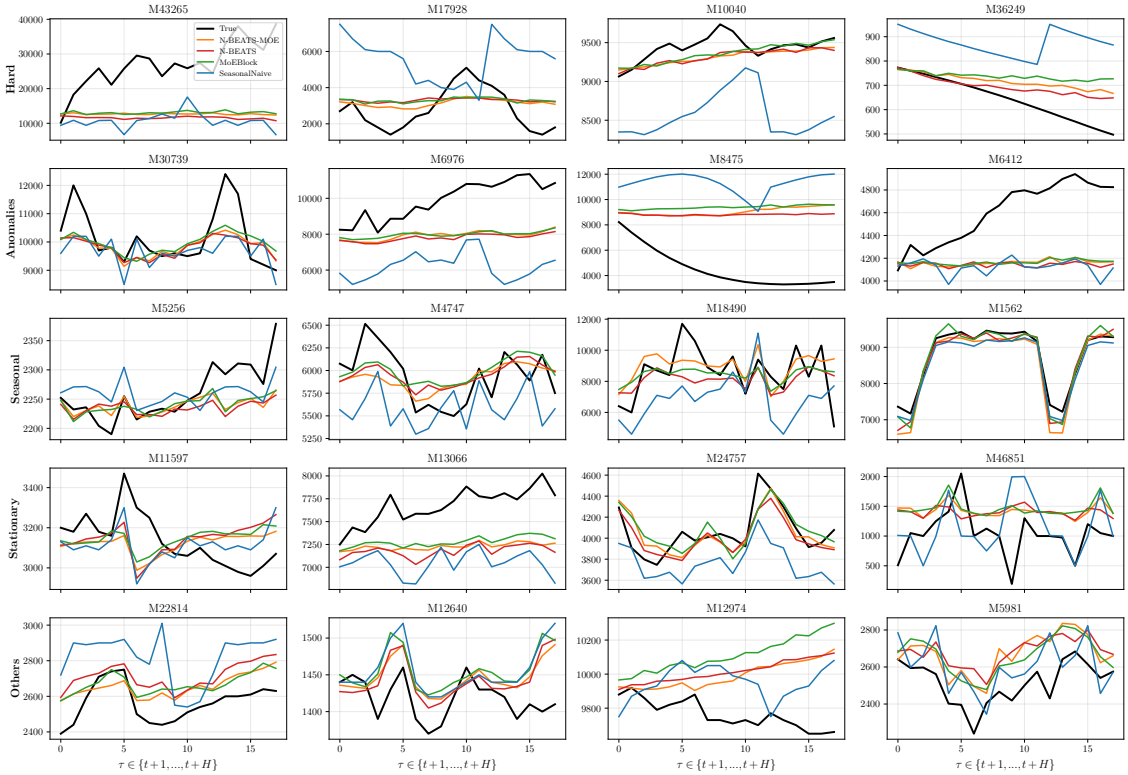


Figure B.10: Aspect-based forecast on the M4 monthly dataset

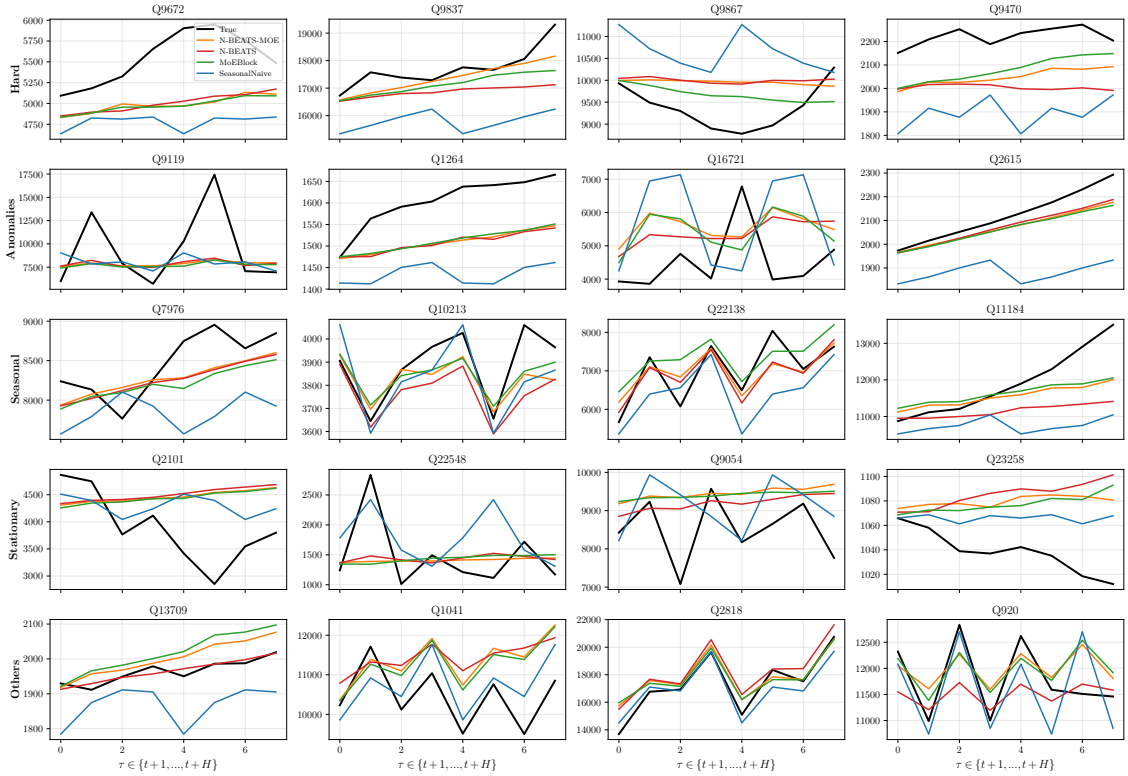


Figure B.11: Aspect-based forecast on the M4 quarterly dataset

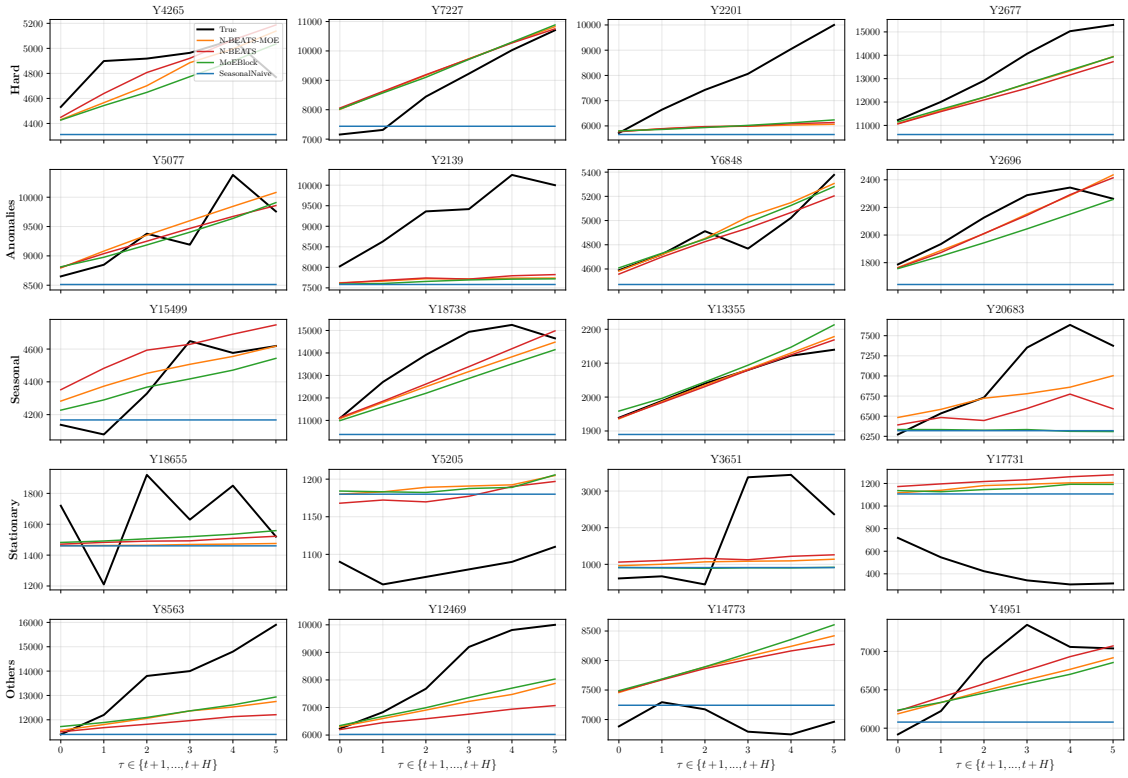


Figure B.12: Aspect-based forecast on the M4 yearly dataset