People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
KASDI MERBAH UNIVERSITY - OUARGLA
Faculty of New Technologies of Information and Telecommunication
Department of Computer Science and Information Technology

MASTER

Domain : Computer Science

Field : Artificial Intelligence and Data Science

Submitted by : Sellami Mohammed Abdelhadi and Hamzi Oussama Seyf elislam

Thesis :

Time Series Forecasting Using Linear Models:

A Comprehensive Investigation and Empirical Analysis

Evaluation Date : 24/06/2024

Before the Jury :

Khaldi Belal	Supervisor	UKM Ouargla
Merabti Houcine	President	UKM Ouargla
Bouanane Khadra	Examiner	UKM Ouargla

Academic year: 2023/2024

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to our supervisor, Dr. Khaldi Belal, for his invaluable guidance, support, and encouragement throughout the course of this research. His insightful feedback and dedication to excellence have been instrumental in the successful completion of this thesis.

We are also profoundly grateful to all our teachers and professors over the years. Their commitment to education and their passion for their subjects have inspired and motivated us to pursue and achieve our academic goals. Each of them has contributed significantly to our personal and professional growth.

A special thank you goes to our families for their unwavering support and belief in us. Their love, patience, and sacrifices have been a constant source of strength and inspiration. We are deeply thankful for their encouragement, which has helped us persevere through the challenges of this journey.

Lastly, we would like to thank our friends for their companionship, understanding, and support. Their constant encouragement and the joy they bring into our lives have been invaluable throughout our academic endeavors.

To all who have contributed to our education and personal development, we extend our heartfelt thanks. This accomplishment would not have been possible without your support.

Abstract

Transformer models, renowned for their exceptional performance in natural language processing and computer vision, encounter challenges when applied to time series forecasting. This is attributed to the permutation invariance of their self-attention mechanism, which hinders their ability to capture temporal dependencies effectively.

Consequently, researchers have explored adapting simpler models, such as those based on convolutional neural networks (CNNs), recurrent neural networks (RNNs), and multilayer perceptrons (MLPs), to time series forecasting. MLP-based models, in particular, have demonstrated success in capturing moving averages and seasonal patterns in data. However, they often struggle to accurately forecast trends and sudden changes, limiting their overall forecasting performance.

To address this limitation, we propose a novel Univariate Multi-Scale Linear (UMS-Linear) model that leverages timestamps, multi-scale decomposition, and a new loss function to enhance the forecasting ability of MLP-based models. UMS-Linear decomposes the input time series into multiple scales, capturing intricate dynamics and patterns that may be missed by traditional MLP models. By incorporating timestamps, UMS-Linear explicitly models the temporal relationships within the data, further improving forecasting accuracy.

Empirical results across multiple benchmark datasets demonstrate that UMS-Linear outperforms existing methods, including transformer-based models, CNNs, RNNs, and MLPs. This superior performance highlights the effectiveness of our approach in capturing the complex dynamics of time series data, paving the way for improved forecasting accuracy in various applications. important field.

Keywords: Long-term Time series forecasting, Linear Models, MLPs, Series Decomposition, Univariate forecasting.

ملخص

نماذج الترانزفورماتور، التي تشتهر بأدائها الاستثنائي في معالجة اللغة الطبيعية والرؤية الحاسوبية، تعاني من صعوبات عند تطبيقها على استخدامها للتنبؤ بالسلاسل الزمنية. يحدث ذلك بسبب عدم تاثر آلية الاهتمام الذاتي بالترتيب واستقلالها عنه (permutation invariance) ، والذي يعيق قدرتها على التقاط الانماط الزمنية بشكل فعال.

وبناءً على ذلك، قام الباحثون بتبني نماذج أبسط لحل هذه المشكلة، مثل تلك التي تعتمد على شبكات الالتفاف العصبية (CNNs) والشبكات العصبية المتكررة (RNNs) والشبكات العصبونية متعددة الطبقات (MLPs) لتوقع السلاسل الزمنية. وقد حققت النماذج المستندة إلى MLP على وجه الخصوص نجاحًا في التقاط متوسطات متحركة وأنماط موسمية في البيانات. ومع ذلك، فإنها غالبًا ما تواجه صعوبة في توقع الاتجاهات والتغييرات المفاجئة بدقة، مما يحد من أدائها العام في التوقع.

لمعالجة هذا الاشكال، نقترح نموذجًا جديدًا خطيا متعدد الاحجام أحادي المتغير (UMS-Linear) والذي يستفيد من طوابع الوقت وتحليل متعدد الاحجام ودالة تعلم جديدة لتحسين قدرة التوقع لنماذج MLP . يقوم UMS-Linear بتحويل السلسلة الزمنية المدخلة إلى احجام متعددة، مما يوفر ديناميكيات وأنماطًا معقدة قد تفوتها نماذج MLP التقليدية. من خلال دمج طوابع الوقت، يقوم UMS-Linear بنمذجة العلاقات الزمنية داخل البيانات، مما يحسن دقة التوقع بشكل أكبر.

تظهر النتائج التجريبية عبر مجموعات بيانات معيارية متعددة أن UMS-Linear يتفوق على الأساليب الحالية، بما في ذلك النماذج المستندة إلى Transformer و CNNs و RNNs و MLPs . يسلط هذا التفوق في الأداء الضوء على فعالية نموذجنا في التقاط الديناميكيات المعقدة للسلاسل الزمنية، مما يمهد الطريق لتحسين دقة التوقع في تطبيقات مختلفة.

الكلمات المفتاحية : التنبؤ بالمتسلسلات الزمنية على المدى الطويلة، النماذج الخطية، الشبكة العصبونية متعددة الطبقات،تفكيك السلاسل الزمنية، التنبؤ أحادي المتغير.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
GENERAL INTRODUCTION

1.1 Background And Motivation

The 21^{st} century has witnessed the rise of artificial intelligence (AI) as a transformative force across diverse fields, ranging from healthcare and finance to education and transportation. The characteristic that allowed artificial intelligence to emerge in this era is its ability to deal with various types of data, such us Text data, Image and video, Audio, Numerical, Categorical, where each of these has its own properties. Time series (TS) is a data type that consists in a sequence of observations from domains that include temporal measurements. TS, usually indexed by time stamps, can be captured via physical sensors, censuses or transaction [1]. Unlike other types of data, time series samples are related to each other in time, indicating that changes occurring in the future are effected by many interconnected changes in the past. Furthermore, this special relation in the data leads researchers to introduce the time series forecasting (TSF) task, which aims to predict future changes based on past information.

Time series data is prevalent in numerous domains, encompassing any recorded series with temporal observation. Moreover, as depicted in Figure 1.1, some of the most well-known domains associated with time series data include:

- **Finance**. It is one of the most well-known domains, which has many subfields including: stock prices, which are numerical data that show a stock's price at a given point in time; loan defaults indicating if a loan was repaid or defaulted on; currency exchange rates, which show the rate at which two currencies exchange; interest rates showing the cost of borrowing money over a given period of time; and consumer spending, which shows the total amount of money that consumers spend on goods

and services over time.

- **Healthcare**. Time series data plays a crucial role in today's healthcare applications. This includes patient health monitoring, which involves recording vital signs at specific times (e.g., heart rate, blood pressure, temperature); medication history indicates medications administered to a patient; disease outbreaks consist of categorical and numerical data (e.g., disease type, number of cases) recorded over time to track outbreaks.

- **Weather**. The temperature, precipitation, wind speed, humidity, which are numerical data representing weather conditions at specific locations and times, are all part of time series data. Additionally, satellite imagery, which represents cloud cover, weather patterns, and other visual information captured by satellites, contributes to the time series analysis of weather phenomena.

- **Retail**. In the retail industry, there are different types of data crucial for understanding sales and customer behavior. Sales data provides information about the amount of products sold at specific times, while customer behavior data gives insights into how customers interact with products, such as browsing history and purchases. Lastly, inventory levels data shows the quantity of products available in stock at specific times.

- **Other Domains**. There are other domains that are worth mentioning such as social media which consists of the user engagement, content popularity, and sentiment on various topics; astronomy represents the movement and characteristics of celestial bodies over time; engineering is monitoring the performance and status of machines, structures, and systems; transportation represents the travel times; climate science and ecology are the monitoring of the populations of plants and animals, and the environmental changes over time; and in energy domain which consists of analyzing energy consumption patterns, optimizing energy production, and forecasting

demand.

Figure 1.1: Domains engaging in analysis, forecasting, and processing of time series data

From the aforementioned domains, it appears that TS plays a crucial role across various fields including finance, healthcare, weather analysis, retail, etc. A lot of real-life applications have been developed to leverage the cues gained from TS data across different domains. The main categories for applications of TS are forecasting, monitoring, and analytics. AFV

1.2 Applications of Time Series Analysis and Forecasting

In now day's data-driven world, TS analysis has become crucial across various industries. It grants businesses extracting valuable insights, make informed decisions, and optimize processes. Finance, healthcare, technology, energy, retail and transportation companies

worldwide exploit the power of TS analysis to forecast outcomes, and drive innovation [1]. We list some of examples of the largest worldwide companies leveraging time series analysis:

Finance Industry: Big investment Banks such as Goldman Sachs, J.P. Morgan, and Morgan Stanley opted for TS analysis to forecast stock prices, manage risks, and employs algorithmic trading. Other firms of Hedge funds and asset management, such as Bridgewater Associates and Renaissance Technologies, utilize TS analysis to develop quantitative trading strategies and manage portfolios effectively.

Technology Industry: Google, Facebook, and Amazon are amongst the major technology companies that rely on TS analysis for a range of purposes including: predicting demand, identifying anomalies, and analyzing user behavior. TS analysis in companies such as Intel and Qualcomm plays a crucial role in various aspects of their manufacturing processes of semiconductors, including: prediction, equipment maintenance, and quality control.

Healthcare Industry: The Biggest pharmaceutical companies on the planet, including Pfizer and Merck, utilize TS analysis in order to enhance drug discovery, optimize clinical trials, and develop disease models. Healthcare providers such as Mayo Clinic and Kaiser Permanente monitor patients, conduct disease surveillance, and plan resources using TS.

Energy Industry: Oil and gas companies, such as ExxonMobil and Shell, utilize TS analysis to enhance reservoir modeling, optimize production, and improve equipment maintenance. Likewise, renewable energy companies like NextEra Energy and Tesla utilize TS analysis to predict solar and wind power generation.

Retail Industry: Major retail companies in the word like Walmart, Target, or Amazon are relying on time series analysis to predict demand, manage inventory, and optimize pricing. Time series analysis is utilized by e-commerce giants such as Alibaba and eBay to enhance their recommendation systems, personalize marketing strategies, and detect fraudulent activities.

The present thesis focuses on time series forecasting, emphasizing its significance in addressing real-life challenges and issues. Forecasting in TS is a crucial aspect of machine learning when it comes to solving real-life problems. It enables us to avoid unforeseen danger that humans cannot anticipate, such as natural disasters and sudden epidemic diseases. It doesn't only grant avoiding negative risks, but also possess the ability to predict changes in market prices and improve the experience across industries.

1.3 Challenges With Time Series Forecasting

Time series data can be found in a multitude of domains, making it exhibiting diverse behaviors and characteristics. Therefore, understanding and modeling the dynamics of different types of TS faces several challenges.

- **Stationary vs. Non-Stationary Data:** Time series data can be categorized as a) Stationary data where the data has a constant mean and variance over time (e.g., Examples include daily temperature fluctuations), and b) Non-stationary data which represents the majority of time series where the mean and variance change over time (e.g., stock prices, global population growth, and daily website traffic)

- **Unknown Inter-Domain Influences:** In some domains, the behaviors of TS are affected by unknown factors from other domains; for instance, changes in a clothing store's sales can be affected by the weather conditions at a particular time. This unknown relationship between domains leads to sudden changes in the data itself; this phenomenon is called change points [2]. Change points led to the emergence of a separate research branch of time series, which was called Change Point Detection, aiming to identify change points in data. However, this is applied only in the known past data, where the current solutions still can't detect future coming change points.

- **Frequency Specification:** Other than that, time series data are usually specified at a fixed, regular frequency. Furthermore, most approaches show high performance in

low-frequency captured data (e.g., Second, Minute), but in contrast, these approaches are still struggling with high-frequency captured data (e.g., Hourly, Daily).

- **Complex Relationship Capture:** Capturing complex relationships within the data, including long-term dependencies, is also one of the challenges that time series forecasting models try to enhance, especially in long-term time series forecasting tasks.

Many approaches have been presented in the time series forecasting task, the diversity of these models results in some interesting observations regarding the accuracy, overfitting, interpretability and the complexity. Usually, as model complexity increases, accuracy on the training data often improves. However, this can lead to overfitting, where the model performs poorly on unseen data. Striking a balance between capturing the underlying trend and avoiding overfitting is crucial. Generally, simpler models are easier to interpret, while complex models often become opaque and difficult to understand. Choosing the right level of complexity depends on the specific needs of the application. If interpretability is crucial, a simpler model with slightly lower accuracy might be preferable.

When evaluating a time series prediction model, one may be interested in assessing its prediction capability and robustness in handling diverse data scenarios. The main aim is to achieve certain capacity that provides accurate forecasting while being able to navigate through complex and dynamic data structure. This means inspecting the model's capability to capture intricate patterns and dependencies within TS, while ensuring that it can effectively generalize to unseen instances. Moreover, the evaluation necessitates an exploration of interpretability of the model, elucidating how well it communicates its predictions and underlying reasoning. Through rigorous assessment, it becomes possible to discern the model's proficiency in delivering reliable predictions consistently across various temporal contexts, thereby facilitating informed decision-making in real-world applications. It is worth mentioning that throughout the literature review, we didn't encounter works that took into consideration all these criteria in evaluating TS models.

1.4 Rise Of Transformers And Role Of Simpler Models Like MLPs

The year 2017 is a significant milestone in the history of artificial intelligence, as it was marked by the emergence of the famous Transformer architecture [3]. Initially, Transformers were presented as new method for different tasks of Natural language processing (NLP) (e.g., Text Generation, Classification, Summarization) [4, 5, 6]. The use of Transformers in NLP demonstrated great performance, leading to its introduction in various artificial intelligence applications such as speech recognition [7], computer vision [8, 9, 10, 11], and music generation [12].

Time series forecasting is one of the applications that Transformer architecture has been modified to be used with [13], as there is a remarkable number of models introduced to this end including Informer [14], AutoFormer [15], and FedFormer [16]. However, Transformer-based methods have encountered many difficulties when applied for TS forecasting, due to the main core of these models, which is the multi-head self-attention mechanism. This mechanism showed a great performance in most NLP tasks, where it handles complex text semantics. The permutation invariant and anti-order nature of the self-attention causes a limitation when it comes to TS. Unlike semantic rich application, any change in the order of the time series causes total loss of temporal information [17].

In 2022, a breakthrough occurred by the introduction of the LTSF-Linear models, namely Linear, DLinear and NLinear [17]. Surprisingly, these models achieved the state-of-the-art results by significantly outperforming transformers. The performance of LTSF-Linear rises a question about the need for complex modeling of time series, as simple MLP models can outperform other complex models by a large margin. The linear relationship between the look-back and the future forecast in the TSF task underpins this assumption in LTSF-Linear models. Such models are designed as direct multi-step (DMS) techniques to circumvent the accumulation of errors. This issue is considered significant in the iterative multi-step (IMS) techniques used by most transformer models.

The LTSF-Linear models employ various effective preprocessing methods, ranging from normalization to series decomposition [17]. These simple techniques caused diverse reactions and performances in various test scenarios. Regardless, the LTSF-Linear models are still too simple to track the different trend changes leading to less accurate forecasts, especially for TS in domains that involve many fluctuations. Subsequently, this drove us to propose our new MLP-based solution, which combines different techniques to give better performance than the standard simple MLP models.

This work delves into the different techniques LTSF-Linear models utilize to outperform other complex models. We begin with the premise that a combination of different techniques in a simple MLP-based model could potentially enhance the performance of such an approach. Through experiencing different types of kernels, the present thesis investigates diverse decomposition technique of TS showing their effectiveness on the model's learning process.

1.5 Uni-Variate Multi-Scale Linear (*UMS-Linear*) Model

To validate the stated hypothesis stated above, we put forward a new scheme named **U**nivariate **M**ulti **S**cale Linear (*UMS-Linear*) that's based on MLP incorporating multi-scaling decomposition technique. Moreover, our proposed *UMS-Linear* uses a simple timestamps embedding method, furnishing the model with additional and valuable temporal information. Leveraging the MLP inherent characteristics and the techniques within *UMS-Linear*, allows integrating more intricate information. These enhancements enable the latter to outperform other MLP-based models, leading to state-of-the-art accuracy in univariate long-term time series forecasting.

One must know that for any forecasting technique to be deemed effective, various enhancement aspects need to be investigated. However, we found out that prior methods in the field often focus narrowly on specific techniques or aspects of assessment. In this thesis, we delve into a broad spectrum of factors including multi-scale decomposi-

tion, timestamp integration, loss function optimization, look-back length variations, and training parameter analysis. In addition, The proposed scheme has been evaluated on real worlds scenarios, its robustness and sensitivity has be analyzed, along with a discussion on the interpretability and the explainability of the provided results. By examining this wide array of considerations, the study provides a comprehensive understanding of model performance and effectiveness in long-term time series forecasting, addressing a gap in existing research where such extensive analysis is lacking. Additionally, the investigation into different univariate benchmarks commonly used for model comparison reveals that *UMS-Linear* surpasses other MLP models in both Mean Squared Error (MSE) and Mean Absolute Error (MAE) metrics, particularly evident in fluctuated forecasts where *UMS-Linear* demonstrates heightened robustness. Notably, *UMS-Linear* exhibits proficiency in both short-term forecasts (less than 96 future points) and long-term time series forecasting, with the horizon extending beyond 96 future points.

In conclusion, this introduction provides a comprehensive overview of the significance of TS forecasting across various domains, emphasizing its role in addressing real-world challenges. We outlined the prevalent use of time series data in domains such as finance, healthcare, weather analysis, and retail, underscoring the importance of accurate forecasting for informed decision-making. We also discussed the associated challenges, including modeling complexities and the trade-offs between model complexity and interpretability. Furthermore, we discussed the emergence of Transformer architectures and the success of simpler models like LTSF-Linear models, setting the stage for the proposed Uni-Variate Multi-Scale Linear (*UMS-Linear*) model. This introduction serves to establish the motivation, context, and objectives for the subsequent chapters of the thesis, which will delve deeper into later on. In the next chapter chapter 2, a literature review will be given where we investigate the aspects of different approaches that have been recently proposed, starting from the statistical to MLP-Based solutions. The third chapter (Link to the chapter) discusses our proposed scheme by detailing each used technique. The fourth chapter (Link to

the chapter) is a comprehensive experiment section discusses the assessment aspects have been taken into account in evaluating our proposed scheme against state-of-the-art. Finally, we synthesize some concluding remarks, reflect on the contributions of the thesis, discuss implications for future research.

CHAPTER 2

LITERATURE REVIEW

The inherent challenges of time series forecasting, such as capturing non-stationary data, long-term dependencies, and complex relationships, have driven the development of diverse forecasting models. These models can be broadly categorized into: statistical models, which utilize statistical methods like regression to capture trends, seasonality, and other patterns in the data; these models are generally interpretable and computationally efficient. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) are leveraged neural networks with specialized architectures to handle sequential data and long-term dependencies. Convolutional neural networks (CNNs) capture spatial and temporal features from the data, potentially suited for specific time series structures. Transformers utilize attention mechanisms to learn long-range dependencies in the data, which is potentially effective for complex relationships. Multi-layer perceptrons (MLPs) employ feed-forward neural networks for learning the different relationships in the data. This chapter will serve as an inclusive literature review, providing details on each category, highlighting the strengths and weaknesses of well-performing models within each, and discussing their suitability for different forecasting tasks.

2.1 Statistical Techniques (Parametric Techniques)

Statistical methods have served as the bedrock of time series forecasting for decades, providing a robust and interpretable framework for predicting future values based on historical data. Despite the increasing prominence of machine learning models, statistical methods retain their significance in forecasting owing to several key advantages: interpretability, transparency, and efficiency. Among the most popular statistical methods in time series forecasting are moving average (MA), which smooths out noise by taking the average of

past observations to predict future values, autoregressive (AR), which forecasts future values based on past values within the time series itself. Autoregressive integrated moving average (ARIMA) [18], a powerful method that combines AR and MA to account for trends and seasonality in the data. Seasonal ARIMA (SARIMA) [19], an extension of ARIMA, designed to handle data exhibiting periodic patterns, such as monthly sales figures. Statistical models offer valuable insights into the factors influencing time series data. They can elucidate how past values, trends, and seasonality contribute to future outcomes, fostering trust in predictions and aiding in issue diagnosis. Additionally, the documented assumptions and limitations of statistical models enable careful evaluation of their suitability for specific time series. However, it's essential to acknowledge their limitations. Statistical models may struggle to capture intricate, non-linear relationships within data, leading to less accurate forecasts for highly dynamic or volatile time series. Moreover, these methods often necessitate manual feature engineering, such as trend or seasonality identification, which can be time-consuming and reliant on domain expertise. Furthermore, many statistical models assume stationary data, meaning their statistical properties (like mean and variance) remain constant over time, which may not always hold true in real-world scenarios.

2.2 Techniques based on Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs)

RNN [20] and LSTM networks [21] have emerged as powerful tools for time series forecasting tasks. Unlike traditional statistical methods, these deep learning models can effectively capture complex and non-linear patterns in time series, making them well-suited for forecasting applications across various domains, including finance, weather, energy, etc.

RNN [20] is a neural network architecture designed to process sequential data, such as time series. Unlike feedforward neural networks, RNNs have a feedback loop that allows them to maintain an internal state, enabling them to model the temporal dependencies

present in time series data. This internal state, also known as the hidden state, acts as a memory, carrying information from previous time steps and influencing the network's predictions at the current time step. While traditional RNNs [20] are capable of learning and modeling sequential patterns, they suffer from the vanishing/exploding gradient problem, which makes it difficult for them to capture long-term dependencies in the data. This limitation can be particularly problematic in time series forecasting, where patterns may span over extended periods.

LSTM [21] is designed to address the vanishing/exploding gradient problem faced by traditional RNN. LSTM [21] is a specific type of RNN architecture that includes a complex gating mechanism, allowing them to selectively remember and forget information from the input sequence. The key components of an LSTM [21] cell are the forget gate, input gate, and output gate. These gates regulate the flow of information into and out of the cell state, which acts as the network's long-term memory. This gating mechanism enables LSTMs [21] to effectively capture long-term dependencies in the data, making them particularly well-suited for time series forecasting tasks involving long sequences or patterns spanning over extended periods.

Segment-wise Recurrent Neural Network (SegRNN)[22] is a recently proposed RNN model designed specifically for long-term time series forecasting. It utilizes two key strategies to reduce the number of recurrent iterations required for effective convergence: 1) Segment-wise Iterations: The original time point-wise iterations are replaced with sequence segment-wise iterations, reducing the number of iterations from L to L/w, where L is the length of the input sequence and w is the segment length. 2) Parallel Multi-step Forecasting (PMF): Single-step predictions are replaced with parallel multi-step forecasting, further reducing the number of iterations from H/w to 1, where H is the forecasting horizon. The authors of SegRNN demonstrated that this novel RNN architecture can significantly reduce the number of recurrent iterations required for LTSF while still achieving competitive performance compared to state-of-the-art transformer-based models. Additionally, they

provide strong evidence that the vanishing/exploding gradient problem, which has traditionally limited the effectiveness of RNNs in LTSF, can be overcome through the use of segment-wise iterations and PMF.

2.3 CNN-Based Techniques

Convolutional neural networks (CNNs) have traditionally been successful in computer vision tasks, but their ability to capture local patterns and temporal dependencies has also made them valuable for time series forecasting problems. In recent years, numerous CNN-based architectures have been proposed specifically for time series data, offering various advantages and addressing different challenges. This section focuses on several prominent CNN models for time series forecasting, highlighting their main ideas, advantages, and potential limitations.

Temporal Convolutional Networks (TCN) [23] were designed for sequence modeling tasks such as time series forecasting. They employ causal convolutions, which ensure that the output at any time step depends only on the current and previous inputs, making them suitable for real-time or online forecasting. They employ dilated convolutions, which enable capturing long-range dependencies while maintaining a reasonable computational cost. Additionally, TCNs can handle sequences of varying length and support parallelization during training and inference. However, TCNs may struggle with capturing long-range dependencies in very long sequences and may not be as effective as attention-based models in certain cases.

Multi-scale Isometric Convolution Network (MICN) [24] combines the modeling perspectives of CNNs and transformers to efficiently extract local features and global correlations of time series. It utilizes multiple branches with different convolution kernels to capture diverse temporal patterns and employs a novel local-global module based on downsampling convolution and isometric convolution. This approach reduces computational complexity to linearity and achieves competitive accuracy.

Sample Convolution and Interaction Network (SCINet) [25] is a novel neural network architecture for time series modeling and forecasting. It adopts a hierarchical downsample-convolve-interact architecture that effectively captures dynamic temporal dependencies at multiple resolutions with a rich set of convolutional filters. The downsampling procedure allows for a larger receptive field at each convolutional layer, and the interactive learning strategy enables information interchange between the downsampled sub-sequences. This design allows SCINet to extract distinct yet valuable temporal features from multiple resolutions, leading to enhanced representation capabilities and improved forecasting accuracy.

TimesNet [26] is a novel approach to temporal variation modeling for time series analysis. It leverages the concept of multi-periodicity to decompose time series into multiple temporal 2D-variations, which capture both intra- and inter-period variation. By transforming 1D time series into 2D space, TimesNet enables the use of parameter-efficient inception blocks for representation learning, which can effectively aggregate multi-scale temporal variations. This approach addresses the limitations of previous methods that attempted to model temporal variations directly from 1D time series, which is extremely challenging due to the intricate temporal patterns. The main disadvantage of TimesNet is that it requires the estimation of periods, which can be challenging for time series with complex or irregular periodicities. Additionally, the model's performance may be sensitive to the choice of hyperparameters, such as the number of periods to consider and the size of the inception block.

PatchMixer [27] is a novel CNN-based model for long-term time series forecasting. It employs a patch-mixing design that combines depthwise separable convolutions and point-wise convolutions to capture both local and global features in the time series. The dual forecasting heads, consisting of one linear and one MLP flatten head, enabling to effectively model both linear and nonlinear trends in the data. PatchMixer outperforms state-of-the-art transformer and CNN-based models on seven long-term forecasting benchmarks. It achieves this performance while being 2-3 times faster than the most recent methods.

2.4 Transformer-based techniques

As mentioned before, Transformers [3] first appeared in 2017, where these models have made a unique leap in the field of artificial intelligence. The presented Transformer architecture was able to handle the different natural language processing (NLP) tasks; some examples of these tasks that are worth mentioning are as follows: text summarization [28], where Transformers can condense lengthy texts into concise summaries by identifying the most important information. Machine Translation [3], where Transformers excel at translating text from one language to another by effectively capturing long-range dependencies within sentences. Sentiment Analysis [29], where Transformers can determine the emotional tone of text, such as positive, negative, or neutral. In addition, Named Entity Recognition (NER), Text Generation, and Question Answering are also some of the many tasks that the Transformer model can handle [4], where there is a large performance gap between the standard methods (e.g., RNN, LSTM) and these Transformers. The spread of transformers was not limited to the NLP field, but their use also appeared in other artificial intelligence fields, where the great performance of this last model shows that these models are worth transferring to many other fields. Some of these fields are as follows: computer vision [8, 9, 10, 11], where by some adjustments, Transformers can analyze images and videos. Speech Recognition [7], by analyzing the sequence of sounds in speech and identifying the words spoken. Recommendation Systems [30], where transformers can analyze user-item interactions and learn the relationships between different items. Music Generation [12], where transformers can be adapted to generate music by analyzing existing musical sequences and learning the patterns and relationships between notes, rhythms, and other musical elements.

Likewise, time series forecasting has witnessed significant advancements with the adaptation of transformers, showcasing remarkable performance in capturing long-term dependencies, particularly suited for long-term forecasting tasks. However, the main multi-head

self-attention mechanism inherent in transformers results in considerable computational cost and complexity, presenting challenges in testing and manipulating these models. To address these limitations, recent advancements have focused on enhancing transformer architectures to reduce complexity and computational costs, with some models achieving a decrease from $\mathcal{O}(L^2)$ to $\mathcal{O}(L)$ [16]. This evolution has led to the emergence of several distinguished works demonstrating excellent results in time series forecasting. This section explores some of the most unique and renowned Transformer-based models, providing a succinct overview of their functionalities.

Informer: Informer [14] was proposed to solve the quadratic complexity and memory usage that the traditional self-attention mechanism produces. In addition, for more accurate predictions, Informer uses a generative style decoder where it predicts long sequences in a single forward operation, avoiding cumulative error and speed plunges. In order to reduce the quadratic complexity of the self-attention, Informer introduced a new ProbSparse self-attention technique, which replaces the canonical self-attention with a sparse mechanism that achieves $\mathcal{O}(LlogL)$ time and memory complexity. The authors of this paper created the well-known ETT (Electricity Transformer Temperature) dataset. Experiments on this dataset with three other large-scale datasets demonstrate that Informer significantly outperforms existing methods in 2021.

Autoformer: Autoformer [15] is a novel decomposition architecture with an auto-correlation mechanism proposed to address the long-term forecasting problem of time series. Autoformer aims to overcome the challenges of traditional transformer-based models, where they struggle with intricate temporal patterns and computational efficiency in long-term forecasting. Autoformer consists of a series decomposition block, an auto-correlation mechanism, and a corresponding encoder-decoder structure. The decomposition block progressively separates long-term trend information from input series by using the moving average kernel. Auto-Correlation discovers period-based dependencies and aggregates similar sub-series from underlying periods, expanding information utilization beyond point-

wise self-attention. The auto-correlation mechanism is an interesting technique that allows to enhance the traditional transformer-based model. The auto-correlation calculates the autocorrelation of the series to identify period-based dependencies. It then utilizes time-delay aggregation to roll and aggregate similar sub-series, providing series-wise connections. Autoformer is evaluated on six real-world benchmarks covering energy, traffic, economics, weather, and disease. The results demonstrate significant improvements in accuracy over state-of-the-art models, with a 38% relative improvement under long-term forecasting settings.

Pyraformer: Pyraformer [31] is a novel transformer-based model that addresses the challenge of capturing long-range dependencies in time series data with low time and space complexity. It introduces the pyramidal attention module (PAM), which leverages a pyramidal graph to summarize features at different resolutions and model temporal dependencies of different ranges. The PAM employs a multi-resolution representation of the time series, with coarser scales capturing long-range patterns and finer scales focusing on short-term dependencies. This allows Pyraformer to capture long-range correlations efficiently, reducing the maximum length of the signal traversing path to $\mathcal{O}(1)$ without increasing the time and space complexity, which is $\mathcal{O}(L)$ with regard to the sequence length L. In single-step forecasting experiments on three real-world datasets, Pyraformer achieves good prediction accuracy with the least amount of time and memory consumption.

FedFormer: Transformer-based methods for long-term series forecasting struggle to capture a global view of time series and are computationally expensive. To address these issues, FedFormer [16] introduced a combination of the Transformer architecture with seasonal-trend decomposition and Fourier analysis. FEDformer significantly improves prediction accuracy in comparison with state-of-the-art methods on six benchmark datasets, where it reduces prediction error by 14.8% for multivariate and 22.6% for univariate time series forecasting. Also, Fedformer has linear computational complexity with respect to sequence length, making it suitable for long-term forecasting. This model uses different tech-

niques, which are summarized as follows: Fourier Enhanced Blocks (FEBs) and Fourier Enhanced Attention (FEA) can efficiently capture global properties and reduce computational costs. FEBs replace self-attention blocks with the Fourier transform to handle frequency components and enhance learning, while FEA performs attention in the frequency domain for information exchange between the encoder and decoder. Additionally, Mixture of Experts Decomposition (MOEDecomp) extracts seasonal and trend patterns, and Random Fourier Component Selection demonstrates the effectiveness of using a chosen subset of Fourier components for efficient representation. By combining these innovative techniques, FEDformer offers a powerful and efficient approach for long-term time series forecasting.

PatchTST: PatchTST [32] is a novel transformer-based model for multivariate time series forecasting and self-supervised representation learning. To enhance locality and reduce computational complexity, PatchTST uses a patching technique where time series are segmented into subseries-level patches, which are used as input tokens to the Transformer. In addition, PatchTST uses the channel-independence method, where each input token contains data from a single channel, allowing for channel-specific embeddings and weights. PatchTST achieves significant improvements in long-term forecasting accuracy compared to state-of-the-art transformer models and the DLinear model [17].

Card: Channel **A**ligned **R**obust Blen**d** Transformer (Card) [33], is a novel Transformer model designed for time series forecasting. It addresses key shortcomings of channel-independent Transformer models by introducing channel-aligned attention, a token blend module, and a robust loss function. The channel-aligned attention allows for the capture of correlations among different channels (forecasting variables) and the alignment of local information within each token. The token blend module generates tokens with different resolutions by merging adjacent tokens within the same head, enhancing the extraction of multi-scale knowledge. To alleviate overfitting, CARD introduces a signal decay-based loss function that weights predictions based on their uncertainty. CARD was evaluated on var-

ious long-term and short-term forecasting benchmarks, resulting in a good performance in both MSE and MAE metrics. It also demonstrated superior performance in reconstruction-based anomaly detection tasks.

CrossFormer: Crossformer [34] is a transformer-based model that explicitly captures cross-dimension dependency for multivariate time series (MTS) forecasting. Existing Transformer models mainly focus on cross-time dependency, neglecting the critical cross-dimension dependency. Crossformer utilizes Dimension-Segment-Wise (DSW) embedding to preserve time and dimension information. The Two-Stage Attention (TSA) layer efficiently captures cross-time and cross-dimension dependency. A hierarchical encoder-decoder (HED) is established to leverage information at different scales for forecasting. Extensive experiments on six real-world datasets demonstrate the effectiveness of Crossformer against some other known transformer models.

2.5 Techniques based on Multi layer Perceptron (MLP)

The breakthrough that occurred with the introduction of the LTSF-Linear [17] models paved the way for many researchers in this field to focus on enhancing and developing MLP nature models. The performance of these models is considered one of the best results in time series forecasting, demonstrating their suitability for this domain. This section delves into recent advancements in MLP models specifically designed for TSF.

LTSF-Linear: Long-term Time Series Forecasting Linear models (LTSF-Linear) [17] comprise three MLP-based models, aiming to showcase the limitations of the self-attention mechanism of transformers. These models have demonstrated superior performance compared to most transformer models.

Vanilla Linear: This is a simple standard MLP containing one fully connected layer. The look-back of the time series serves as an input vector for this layer, and the output of this layer constitutes the forecast. Mathematically represented as $Y = WX$, where $X \in \mathbb{R}^L$ is the look-back input, $Y \in \mathbb{R}^T$ is the forecast, and $W \in \mathbb{R}^{T \times L}$. Surprisingly, this

simple MLP model outperforms all transformer models in most benchmarks.

DLinear: To handle complex patterns in data across different domains, the authors introduced the Decomposition Linear model (DLinear). It combines series decomposition used in AutoFormer [15] and Fedformer [16] with a simple vanilla linear layer. DLinear decomposes the input series into a trend and a remainder component using a moving average kernel. These components are then fed into separate linear layers, with their outputs added together to produce the final forecast. This process enables the model to effectively learn series components and trends.

NLinear: Developed to handle distribution shifts between training and test sets in certain datasets, such as ETTh1 and ETTh2. NLinear utilizes a combination of normalization and a linear layer. It first normalizes the input series and then feeds it into a linear layer, resulting in a shifted output series. Finally, the series is shifted back to produce the output forecast series. NLinear excels in handling data shifting, surpassing existing transformers' results on the ETT dataset.

MTS-Mixers: Multivariate Time Series Forecasting via Factorized Temporal and Channel Mixing (MTS-Mixers) [35] employs two factorized modules: temporal and channel mixing. The temporal mixing module captures temporal dependencies, while the channel mixing module captures channel interactions. By leveraging the low-rank property of time series data, MTS-Mixers aims to achieve better prediction accuracy and efficiency. Extensive experiments on several real-world datasets demonstrate its superiority over existing transformer-based models, achieving significant improvements in forecasting performance.

TSMixer: TSMixer [36] is a novel architecture for time series forecasting that combines time-mixing and feature-mixing operations to efficiently capture both temporal patterns and cross-variate information. It stacks multi-layer perceptrons (MLPs) and employs residual connections, normalization, and temporal projection to enhance its performance. TSMixer outperforms state-of-the-art univariate models on popular long-term forecasting benchmarks.

TiDE: TiDE (Time-series Dense Encoder) [37] combines the simplicity of linear models with the non-linearity of MLPs to achieve superior performance on popular forecasting benchmarks. It consists of an encoder-decoder architecture, with dense MLPs used for encoding the past of a time series and covariates into a dense representation. The decoder maps this encoding to future predictions, incorporating future covariates into the predictions using dense MLPs. TiDE surpasses Transformer-based baselines and matches or exceeds the performance of DLinear.

MoLE: Mixture-of-Linear-Experts (MoLE) [38] augments linear-centric LTSF models to improve accuracy. MoLE introduces multiple linear-centric models (experts) and a router model that weighs and combines their outputs. It integrates multiple linear-centric experts with a mixing layer, using a timestamp embedding to learn weights for each expert. This allows MoLE to capture diverse temporal patterns and significantly enhance forecasting accuracy compared to existing linear-centric models.

2.6 Other Techniques

In addition to the aforementioned well-known approaches, other techniques such as pretrained models [39] and probabilistic forecasting methods like ForGAN [40], which utilize GAN architecture, have gained traction. There have also been attempts to develop Language Models (LLMs) for time series, such as timeLLM [41], which leverage pretrained language models in time series forecasting, along with frozen pretrained transformers (FPT) [42]. Here's a brief summary of some of these methods:

ForGan: ForGan [40] proposes a method that enhances the training and prediction of univariate probabilistic time series models by explicitly modeling error autocorrelation within mini-batches. It constructs mini-batches as collections of consecutive time series segments and learns a time-varying covariance matrix to capture the correlated errors. A small neural network is attached to project the hidden state to the weights of the correlation matrix, which is parameterized as a weighted sum of base kernel matrices. This ensures

positive definiteness and unit diagonals. The method improves prediction accuracy and enhances training flexibility without significantly increasing model parameters.

Frozen Pretrained Transformer (FPT): Frozen Pretrained Transformer (FPT) is [42]a novel approach to time series forcasting and analysis that leverages pre-trained language models without modifying their core components. FPT provides a unified framework for diverse time-series tasks,achieving state-of-the-art or comparable results. Its universality is demonstrated by the effectiveness of using pre-trained models from different domains (NLP, CV) for time series forecasting and analysis. However, FPT has limited customization due to the frozen pre-trained layers and requires massive datasets for pre-training, which may not always be available for time series forecasting and analysis.

TimeLLM: TIME-LLM [41] is a novel framework that reprograms large language models (LLMs) for general time series forecasting. Unlike existing methods that specialize in specific tasks and require extensive domain expertise, TIME-LLM can handle a wide range of forecasting tasks with minimal adaptation. This is achieved by reprogramming the input time series into text prototype representations and enriching the input context with declarative prompts, which guide the LLM's reasoning about time series concepts. The transformed time series patches from the LLM are finally projected to obtain the forecasts. TIME-LLM offers several advantages over traditional forecasting models. First, it is generalizable and can be applied to a wide range of forecasting tasks without requiring per-task retraining. Second, it is data-efficient and can perform well even with limited historical data. Third, it leverages the reasoning capabilities of LLMs, which enables it to make highly precise forecasts by leveraging learned higher-level concepts. Fourth, it is multimodal and can fuse different data types, such as text, images, and speech, to enhance forecasting accuracy. Finally, it is easy to optimize and can be deployed with minimal computational resources. TIME-LLM has been evaluated on a variety of forecasting benchmarks and has consistently outperformed state-of-the-art models, especially in few-shot and zero-shot scenarios. This demonstrates the effectiveness of TIME-LLM in

unlocking the untapped potential of LLMs for time series forecasting.

2.7 Comparison With State-Of-The-Art Techniques in Time Series Forecasting

To facilitate insight into the overall field of research, we summarize previous related works by providing Table Table 2.1, which outlines some key specifications of the aforementioned techniques (e.g., Techniques Used, results).

Table 2.1: Comparison With State-Of-The-Art Techniques in Time Series Forecasting

	Author/Year	Title	Model	Techniques Used	Datasets	Results	Reported shortcomings	Conclusion
Transformers	Zhou et al., 2021	Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting	Informer	Transformer using ProbSparse self-attention, self-attention distilling, generative style decoder	Electricity, Traffic, Retail, Weather	Outperforms existing methods on all datasets	Limited to univariate time-series forecasting	Informer model boosts LSTF prediction by capturing long-range dependencies in time series data
	Haixu Wu et al., 2021	Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting	Autoformer	Decomposition, Auto-Correlation, Transformer	ETT, Traffic, Electricity, Exchange, ILI, Weather, Covid19	38% relative improvement under the long-term setting	Not mentioned	Autoformer tops accuracy in forecasting energy, traffic, etc.
	Shizhan Liu et al., 2022	Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting	Pyraformer	Transformer using Pyramidal attention module (PAM), Coarser-scale construction module (CSCM)	Wind, App Flow, Electricity, ETT	Beats Transformers (NRMSE, ND) with fewer parameters & faster runtime	Not mentioned	Pyraformer can simultaneously capture temporal dependencies of different ranges
	Tian Zhou et al., 2022	FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting	FEDformer	Seasonal-trend decomposition, Fourier transform, Wavelet transform, Transformer	ETT, Traffic, Electricity, Exchange, ILI, Weather	14.8% and 22.6% improvement in multivariate and univariate forecasting, respectively	Sensitive to hyperparameter tuning	FEDformer: captures global trends & forecasts efficiently
	Nie et al., 2023	A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers	PatchTST	Transformer, Patching, Channel-independence	Weather, Traffic, Electricity, ILI, ETT	MSE and MAE reduced by 21.0% and 16.7% compared to the best Transformer-based models	Not mentioned	PatchTST tops all methods in self-supervised learning

Transformers	Wang et al., 2024	CARD: Channel Aligned Robust Blend Transformer for Time Series Forecasting	CARD	Transformer, Channel-aligned attention, Token blend module, Robust loss function	ETT, Traffic, Electricity, Exchange, ILI, Weather	Outperforms state-of-the-art methods in long-term forecasting and reconstruction-based anomaly detection	Not mentioned	CARD merges info & scales for better forecasts
	Yunhao Zhang & Junchi Yan., 2023	Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting	Crossformer	Dimension-Segment-Wise (DSW) embedding, Two-Stage Attention (TSA) layer, Hierarchical Encoder-Decoder (HED)	ETT, WTH, ECL, ILI, Traffic	Crossformer ranks top-1 among the 9 models for comparison on 36 out of the 58 settings and ranks top-2 on 51 settings	Crossformer only implicitly utilizes the cross-dimension dependency by embedding, potentially limiting its forecasting capability	Crossformer excels in multivariate forecasting but can improve cross-dimension mining
Multi layer Perceptrons (MLPs)	Ailing Zeng et al. (2024)	Are Transformers Effective for Time Series Forecasting?	LTSF-Linear (Linear, Dlinear, Nlinear)	Fully connected MLP model	ETT, Traffic, Electricity, Weather, ILI, Exchange-Rate	Outperforms existing Transformer-based models on all datasets	the one-layer linear network is hard to capture the temporal dynamics caused by change points	LTSF-Linear can be a new baseline for the LTSF problem
	Zhe Li et al., 2023	MTS-Mixers: Multivariate Time Series Forecasting via Factorized Temporal and Channel Mixing	MTS-Mixers	Factorized temporal and channel mixing	ECL, ETT, Traffic2, PeMS04, Weather3, Exchange, ILI	Outperforms existing Transformer-based models with higher efficiency	Not mentioned	New framework forecasts multiple series, capturing time & channel links
	Chen et al. (2023)	TSMixer: An All-MLP Architecture for Time Series Forecasting	TSMixer	MLP-based model	ETT, Traffic, Electricity, Exchange, ILI, Weather	State-of-the-art results on long-term forecasting benchmarks and M5	Not mentioned	MLPs excel at time series & multi-data, but struggle with extra info
	Abhimanyu Das et al., 2023	Long-term Forecasting with TiDE: Time-series Dense Encoder	TiDE	MLP-based encoder-decoder	Weather, Traffic, Electricity, ETT	Beats Transformers (speed & accuracy) on long-term forecasting	Not mentioned	TiDE: Simple deep learning for long-term forecasts
	Ronghao Ni, Zinan Lin, Shuaiqi Wang, Giulia Fanti, 2023	Mixture-of-Linear-Experts for Long-term Time Series Forecasting	MoLE	Mixture-of-Experts for MLP models	ETT, weather, electricity, traffic, Weather2K	MoLE improves forecasting error in over 78% of the datasets and settings evaluated.	Not mentioned	MoLE boosts linear models for better LTSF forecasts

Convolutional Neural Networks (CNNs)	Shaojie Bai et al., 2018	An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling	TCN	Temporal convolutional networks (TCNs)	no forecasting dataset mentioned	TCNs outperform canonical recurrent networks across a comprehensive suite of tasks and datasets.	Not mentioned	TCNs: Faster training, lower memory, better results than RNNs
	Huiqiang Wang et al., 2023	MICN: Multi-scale Local and Global Context Modeling for Long-term Series Forecasting	MICN	Multi-scale hybrid decomposition, Seasonal prediction block, Trend-cyclical prediction block	ETT, Traffic, Weather, Electricity, ILI, M4	MSE and MAE scores significantly lower than state-of-the-art methods	Not mentioned	MICN tops accuracy on real-world forecasting tasks
	Liu et al., 2022	SCINet: Time Series Modeling and Forecasting with Sample Convolution and Interaction	SCINet	Sample convolution and interaction	Solar-Energy, Traffic, Electricity, Exchange,Rate, ETT, PEMS03, PEMS04, PEMS07, PEMS08	State-of-the-art results on most benchmarks and prediction length settings	SCINet might be affected by the missing data if the ratio exceeds a certain threshold leading to poor prediction performance	SCINet captures short & long-term trends for accurate long-term forecasts
	Wu et al., 2023	TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis	TimesNet	Temporal 2D-Variation Modeling, Multi-Periodicity, 2D Convolution, Inception Block	ETT, Traffic, Electricity, Weather, ILI, Exchange-Rate, M4	State-of-the-art on all five datasets	Not mentioned	TimesNet excels in all time series tasks: forecast, imputation, classify
	Zeying Gong et al., 2023	PatchMixer: A Patch-Mixing Architecture for Long-Term Time Series Forecasting	PatchMixer	Patch-mixing architecture, Depthwise separable convolutions, Dual forecasting heads	Weather, Traffic, Electricity, ETT	Beats Transformers & prior CNNs in accuracy (MSE & MAE).	Patch-based models focus on individual time points, making it difficult to capture long-term temporal patterns and integrate external features that span multiple time points.	PatchMixer is an efficient and accurate model for long-term time series forecasting.
Recurrent Neural Networks	Shengsheng Lin et al., 2023	SegRNN: Segment Recurrent Neural Network for Long-Term Time Series Forecasting	SegRNN	Segment-wise iterations, Parallel Multi-step Forecasting (PMF)	ETT, Electricity, Traffic, Weather	Outperforms SOTA Transformer-based models in 50 out of 56 metrics	SegRNN has a relatively smaller capacity	RNN methods still hold strong potential in the LTSF domain

2.8 Conclusion

At the end of this chapter, we conclude that there are different methods and techniques that are used in time series forecasting. These methods are presented to enhance and outperform the existing forecasting methods. The abundance of these methods has led to the diversity of their categories, such as statistical methods, transformer-based methods, and MLP-based methods. In terms of performance, the MLP-based models are showing the best results, outperforming the existing transformer-based models. However, the state-of-the art results are obtained from models in different categories (e.g., PatchMixer [27], SegRNN [22]). This shows that models in different categories can also be enhanced to resolve the time series forecasting task. In this work, we aim to enhance the MLP-based models to obtain more accurate performance, which will be discussed in the next chapter.

CHAPTER 3

PROPOSED METHOD

3.1 Introduction

In order to reduce the effects of the limitations in the standard MLP-based models, we will propose our own model, *UMS-Linear*, that uses different techniques. In this chapter, we will give an overview of this model, and then we will introduce the different techniques that this model uses, where each of the model's techniques will be explained in detail. And at the end, this chapter will conclude with some theoretical analysis of the computational efficiency of this model.

3.2 Overview of UMS-Linear

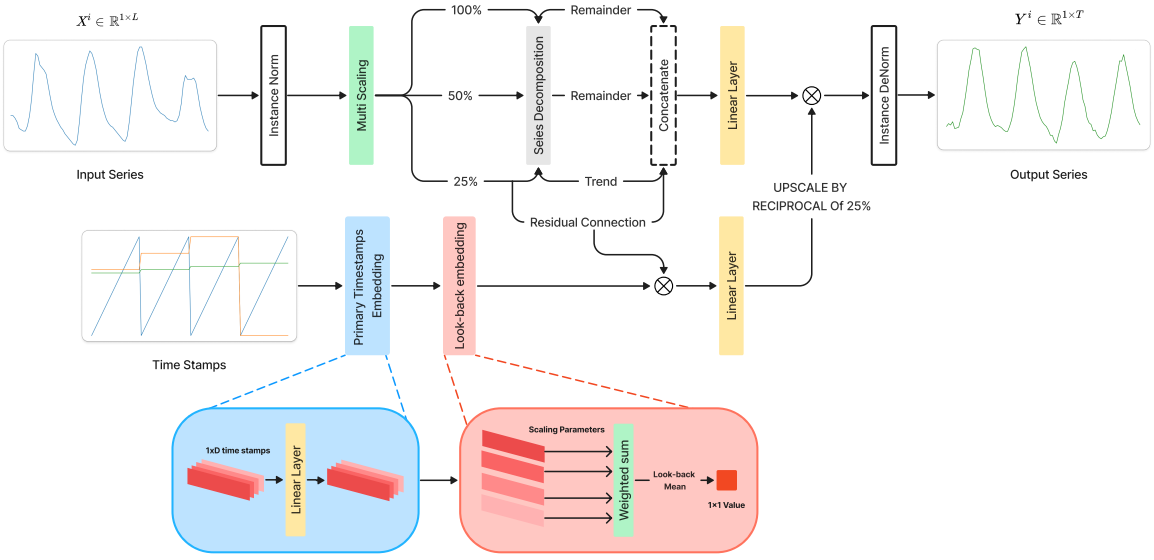


Figure 3.1: Illustration of the *UMS-Linear* model architecture

Inspired by the different linear models of LTSF-Linear [17], we introduce *UMS-Linear*, which is a novel MLP-based model designed for univariate Long-Term Time Series (LTSF)

forecasting. *UMS-Linear* contains different techniques, as shown in Figure 3.1, where it combines three key components to achieve state-of-the-art accuracy:

- **Normalization** : *UMS-Linear* inherits the normalization technique from NLinear [17], which helps mitigate data-shifting issues and improves forecasting performance.
- **Multi-Scale Decomposition Phase**: This phase decomposes the time series in multiple scales using a max pooling kernel. This allows *UMS-Linear* to capture the overall shape of the data and extract the most prominent trends and fluctuations.
- **Base Recover Phase**: The final phase combines the decomposed components with embedded timestamps to produce the forecasted time series. Timestamps provide valuable temporal information, enabling *UMS-Linear* to better track long-term trends and seasonal patterns.

The combination of these techniques allows *UMS-Linear* to effectively capture more complex dynamics and patterns in univariate time series data compared to the standard LTSF-Linear models. This results in improved forecasting accuracy, especially for long-term horizons.

3.3 Incorporating Timestamps and Multi-Scale Decomposition

UMS-Linear incorporates two novel techniques to enhance its forecasting capabilities: timestamp embedding and multi-scale decomposition.

Timestamps Embedding: Timestamps provide valuable temporal information that is often overlooked in time series forecasting. *UMS-Linear* utilizes a simple yet effective dense layer as an embedding for these timestamps. By encoding the timestamps of each data point, the model gains a better understanding of the series' temporal context. This allows it to capture long-term trends and seasonal patterns more accurately.

Multi-Scale Decomposition: The multi-scale decomposition technique in *UMS-Linear* is inspired by the observation that time series exhibit different patterns at different scales.

By decomposing the series into multiple scales, the model can learn the underlying components of the data and focus on the most relevant features for forecasting. The decomposition is performed using a max pooling kernel, which captures the most prominent trends and fluctuations in the data.

The combination of timestamp embedding and multi-scale decomposition enables *UMS-Linear* to capture both the global and local characteristics of time series data. This comprehensive approach enhances the model's ability to predict future values, especially in long-term forecasting scenarios.

3.4 Modeling Time Series for Linear Transformation

UMS-Linear incorporates a linear layer as a key component in its forecasting pipeline. This linear layer plays a crucial role in transforming the decomposed time series components and the timestamps into the final forecasted values. This linear layer is used to model the relationship between the input time series and the forecasted values.

UMS-Linear contains three linear layers to transform different components into different meanings. These transformations are defined by weight matrices and bias vectors. The weight matrices capture the relationships between the input components and the output values. The bias vector adds a constant offset to the output values; this transformation can be mathematically noted as $Y = WX + b$, where Y are the output values and X are the input values, while b is the bias vector and W is the weight matrix. The linear layers in *UMS-Linear* can be simplified into a series of linear regressions. Each linear regression corresponds to one output dimension of the linear layer. The weight matrix of the linear layer can be decomposed into a set of weight vectors, each of which corresponds to one linear regression.

Time series data often exhibits nonlinear patterns. However, a linear layer can still effectively model nonlinear time series by approximating the nonlinear function with a series of linear functions. This is possible because a linear layer can be stacked with multiple

layers, each of which performs a different linear transformation. The combination of these linear transformations can approximate complex nonlinear functions.

The use of linear layers in *UMS-Linear* allows for some advantages, which are listed as follows:

- **Simplicity**: Linear layers are relatively simple to implement and train.
- **Interpretability**: The weight matrix of the linear layer can be interpreted to understand the relationship between the decomposed components and the forecasted values.
- **Efficiency**: Linear layers are computationally efficient, making them suitable for large-scale time series forecasting tasks.

Overall, the linear layer in *UMS-Linear* plays a vital role in transforming the decomposed time series components into accurate forecasted values. The simplicity, interpretability, and efficiency of linear layers make them a valuable component in *UMS-Linear*'s forecasting pipeline.

3.5 Decomposition Phase

Scaling. Given a univariate time series $\mathcal{X} \in \mathbb{R}^L$, scaling is applied to the time series as follows: $\hat{X} = \mathcal{X} * \alpha$ where each point in the time series is scaled with α. This allows the model to learn the shape of the data on different scales. The proposed model applies scaling using a learnable parameter that is first initialized by 0.25 and also produces another fixed version that is scaled by 0.50, These scalars are multiplied to the whole look-back, which makes the whole look-back scaled at the same rate. Note that in the architecture, the scaling by 100% refers to multiplying the input by 1.0 and also for the other scalars (50% $\Leftrightarrow$ 0.5, 25% $\Leftrightarrow$ 0.25).

Moving average. Moving average is a technique that applies average pooling, denoted as $AvgPool$, to a time series $\mathcal{X}$ to extract the trend from the series. The subtraction remains

between the original series and the extracted trend, resulting in the remainder component. In formulation, the series decomposition is known as follows:

$$Trend = AvgPool(\mathcal{X}) \tag{3.1}$$

$$Remainder = \mathcal{X} - Trend \tag{3.2}$$

The time series decomposition technique allows us to perform different approaches on each component, where we impose that the trend is an important part of the series (see the section 4.8 section for more details).

Usually in time series decomposition, the moving average applies the average pooling kernel; the kernel allows for the extraction of a medium trend that equalizes between the maximum values and the minimum values. *UMS-Linear* uses a max pooling kernel; this allows to extract a trend that holds more important overall information, which produces more accurate forecasts. This experiment is detailed in the section 4.8 section with the different pooling methods.

Combination. After scaling the data and producing the $\hat{X}$ series, we perform the series decomposition with the max pooling method in both the original $\mathcal{X}$ series and the scaled $\hat{X}$ series. This will produce the following:

$$T, R = MaxDec(\mathcal{X}) \tag{3.3}$$

$$T_{25}, R_{25} = MaxDec(\hat{X}_{25}) \tag{3.4}$$

$$T_{50}, R_{50} = MaxDec(\hat{X}_{50}) \tag{3.5}$$

Where T is the trend and the Remainder is denoted as R, and $MaxDec$ represents the max pooling decomposition technique.

A combination is made by combining the 4 components T_{25} and R and R_{50} and finally $\hat{X}_{25}$. This combination allows the model to capture the overall movements and fluctuations

from the reminders $\{R/R_{50}\}$ and to learn the general shape of the data from $\hat{T}$ and $\hat{X}$. This combination then enters a linear layer, as follows:

$$C = concatenate(R, R_{50}, \hat{T}, \hat{X}) \tag{3.6}$$

$$F = W_c C \tag{3.7}$$

where $C \in \mathbb{R}^{L*4}$ is the result of the concatenation and $F \in \mathbb{R}^{T}$ is the result of the linear layer $W_c \in \mathbb{R}^{L \times T}$.

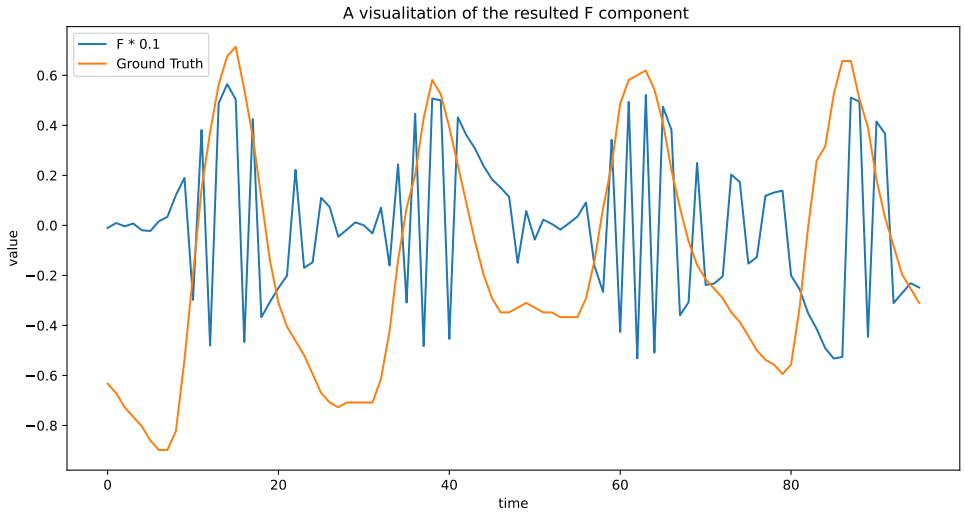


Figure 3.2: A Comparation between the scaled F Component resulted by the Decomposition phase and the ground truth.

Notably, after training the model and visualizing the resulted F component, we see that it is a signal series where the high frequented locations represent a rising or a falling in the ground truth values, and the opposite low frequented parts refer to a falling or a rising location in the ground truth, respectively, as shown in Figure 3.2.

3.6 Timestamps Embedding and Base Wave Forecasting

Timestamps Embedding. Timestamps are a component that the time series consists of; ignoring the timestamps produces a sequential series with no time information; this results

in a model that treats the time series the same as a text line. Positional encoding used in the Transformers [3] will encode the points positions in the sequence. However, it could not encode the repeated time information in the full data. As illustrated in Figure 3.3, the positional embedding will give the same embedding for all the time series points at position i. The data-loading technique used in the LTSF task uses a sliding window with one stride to store the look-backs, which means that a point in series i will get a different embedding in another series.

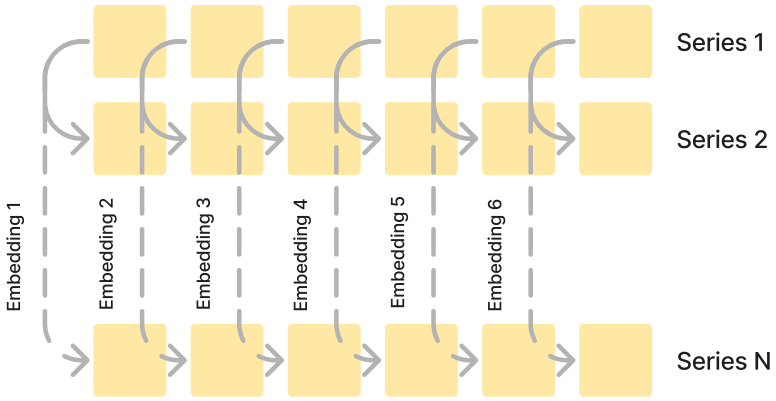


Figure 3.3: An Illustration of the Positional Embedding Technique that has been used in the Basic Architecture of Transformers [3].

Using timestamp embedding will address this problem, where timestamps contain the different time components (e.g., hour, day, weekday, month) in a numerical format that depends on the component's maximum possible value (e.g., hours $\in [1..24]$). These values are normalized as follows: $\hat{h} = \frac{h}{24}$ and for the day number in the month $\hat{d} = \frac{d}{31}$ and for the day number in the week $\hat{wd} = \frac{wd}{7}$ and finally for the month number we got $\hat{m} = \frac{m}{12}$. At the end, we will denote the timestamps as $T \in \mathbb{R}^{L \times D}$, where D is the number of time components. Furthermore, *UMS-Linear* applies a linear transformation and then applies a weighted sum to produce a one feature timestamp embedding as follows:

$$\hat{T} = W_T T$$

where $\hat{T}$ is the product of the linear layer $W_T \in \mathbb{R}^{L \times L}$, And the weighted sum is noted as follows:

$$\hat{T} = WSum(\hat{T}, axis = C)$$

Where $WSum$ is a weighted sum that multiplies each channel by a learnable parameter (a scalar) to identify its importance in the prediction, the final $\hat{T}$ will be of size $L \times 1$.

However, in *UMS-Linear*, we wanted to make a full look-back embedding, where we are going to use the full look-back timestamps to produce a single value that identifies the series location in the full data. This is simply applied by calculating the mean value of the timestamps produced from the last step. Figure 3.4 shows the resulted embedding value of timestamps over the ETTh1 dataset, where it appears that the embedding is produced in a fixed range between [-0.4 and 0.4], which is similar to the timestamp embedding range of MixOfExpert [38]. This changing in values gives the series more flexibility.

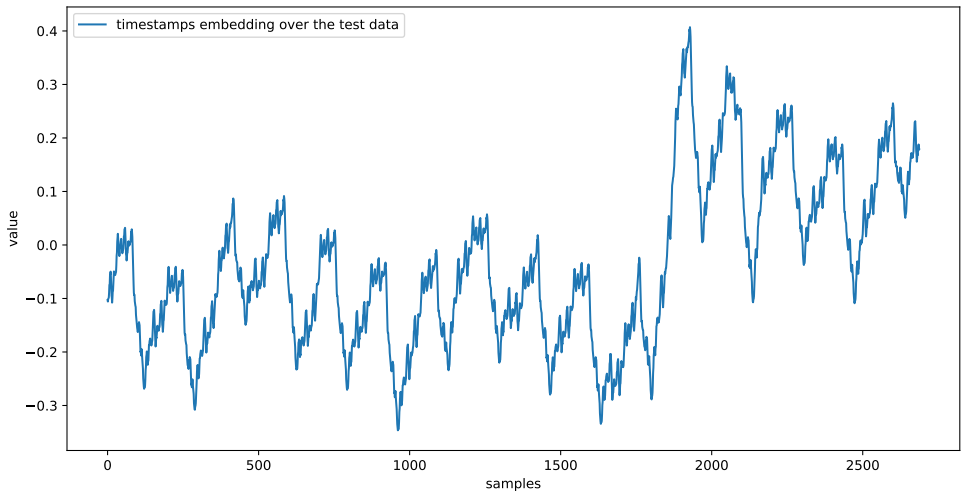


Figure 3.4: The produced timestamps embedding final value over the test samples in ETTh1 dataset

The base wave is produced by feeding the multiplication of the timestamps with the scaled series $\hat{X}$ into a linear layer. The base wave is denoted as B; for more robustness, this base wave is scaled by the learnable scalar α that was used in the decomposition phase.

The resulted wave regularizes the F component from the decomposition phase by scaling it down in different locations to make it look like the base time series format.

The final forecasting result is formulated as follows:

$$Y = (B \otimes F) \oplus LastSequence$$

where $LastSequence$ is the last look-back sequence that is used to normalize the series $\mathcal{X}$ in the first part by $\mathcal{X}_N = \mathcal{X} \ominus LastSequence$ where $\mathcal{X}_N$ is the normalized series.

3.7 Loss Function for Training *UMS-Linear*

While Mean Squared Error (MSE) is the common loss used to train most of the TSF models, the equally 1:1 combination between MSE and Mean Absolute Error (MAE) that is used in PatchMixer [27] is the best choice in our case. Furthermore, applying it makes the model more robust in achieving the best results in more test cases, which is because of the new balance nature of this combined loss.

The MSE Loss:

$$\mathcal{L}_{MSE} = \frac{1}{TN} \sum_{i=1}^{N} \sum_{t=1}^{T} (\hat{y}_t^i - y_t^i)^2$$

And the MAE Loss:

$$\mathcal{L}_{MAE} = \frac{1}{TN} \sum_{i=1}^{N} \sum_{t=1}^{T} |\hat{y}_t^i - y_t^i|$$

Where T is the forecast horizon and N is the number of samples. The final loss used to train the model is then noted as follows:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{MAE}$$

However, in a further study, we discovered that this function can be improved to a more effective version, and we suggest that models can perform better if they capture the differences between points. This will reveal the locations of the change points in the data

while also including the patterns in which the data changes between the current point and the next point over time. The differences between points will result in a new series as follows: Let X_t denote the value of the time series at time t. The differenced time series, denoted by ΔX_t, can be calculated as:

$$\Delta X_t = x_{t-1} - x_t$$

We used the mean absolute error (MAE) to include this differenced time series in the loss function, where the differences will be calculated in both the ground truth and the prediction series. This new loss function will be noted as $\mathcal{L}_{MAE_{diff}}$. The final loss function, which includes all the components we previously mentioned, will be noted as follows:

$$\mathcal{L} = \mathcal{L}_{MSE} + \mathcal{L}_{MAE} + \mathcal{L}_{MAE_{diff}}$$

3.8 Theoretical Analysis of Computational Efficiency

UMS-Linear is designed to be computationally efficient, making it suitable for large-scale time series forecasting tasks. The computational efficiency of *UMS-Linear* can be attributed to the following factors:

- **Linear Operations**: *UMS-Linear* primarily relies on linear operations, such as matrix multiplications and vector additions. These operations are highly efficient and can be parallelized for faster computation.

- **Lightweight Decomposition**: The decomposition phase in *UMS-Linear* uses a max pooling kernel, which is a simple and efficient operation. This lightweight decomposition helps reduce the computational cost without compromising forecasting accuracy.

- **Efficient Timestamps Embedding**: The timestamps embedding technique in *UMS-*

Linear involves a linear transformation and a weighted sum. These operations are computationally inexpensive and can be performed efficiently.

Compared to other state-of-the-art time series forecasting models, *UMS-Linear* offers competitive computational efficiency. For example, transformer-based models, such as Informer and Autoformer, have higher computational costs due to their complex self-attention mechanisms. *UMS-Linear*, on the other hand, avoids self-attention and relies on simpler linear operations, resulting in faster training and inference times.

The computational efficiency of *UMS-Linear* makes it scalable for large datasets and long forecasting horizons. As the size of the dataset or the forecasting horizon increases, *UMS-Linear* maintains its efficiency due to its lightweight architecture and efficient operations. This scalability allows *UMS-Linear* to be applied to a wide range of real-world time series forecasting problems.

UMS-Linear's computational efficiency is a key advantage that enables it to handle large-scale time series forecasting tasks effectively. The model's reliance on linear operations, lightweight decomposition, and efficient timestamp embedding contributes to its fast training and inference times. This efficiency makes *UMS-Linear* a practical and scalable solution for real-world time series forecasting applications.

3.9 Conclusion

UMS-Linear, a novel MLP-based model for univariate Long-Term Time Series (LTSF) forecasting, is proposed in this chapter. *UMS-Linear* incorporates timestamp embedding and multi-scale decomposition to capture both the temporal and structural characteristics of time series data. The linear layer in *UMS-Linear* performs a linear transformation on the decomposed components to produce the forecasted values. *UMS-Linear* is computationally efficient due to its lightweight decomposition phase and efficient timestamp embedding technique. This efficiency makes it suitable for large-scale time series forecasting tasks, as will be shown in the next experimental chapter.

CHAPTER 4

EXPERIMENTAL EVALUATION AND DISCUSSION

In the end of this thesis and in order to prove the efficiency of our presented model, we provide different experiments and analyses, where we discuss the different ideas and the advantages that led to the final model. This chapter will first include the experimental setup, dataset and reprocessing, baseline models, and used metrics. It will then contain the different results with multiple visualizations, and finally, this chapter will conclude with a discussion on the interpretability and the explainability.

4.1 Experimental Setup

All the included experiments and results are implemented using PyTorch and executed in colab using one NVIDIA T4 GPU, which is equipped with 16GB of memory. For the model settings, *UMS-Linear* may perform better results using specific settings in different benchmarks, but the general configuration of *UMS-Linear* consists of a look-back size of 512, a batch size of 128, and a learning rate of 0.001. The model is usually trained with 10 to 15 epochs and early stopping with a patience of 3.

4.2 Dataset Description and Preprocessing

Different datasets have been introduced over the years to create new challenges for the TSF models. However, in our model, we only focused on univariate LTSF, where we used the **ETT (Electricity Transformer Temperature)** [14][1] dataset for comparison with other baseline models, where it is the most used dataset for univariate forecasting tasks. This dataset is collected from electricity transformers from July 2016 to July 2018 in two separated counties in China, each data point consists of a target point called *oil temperature*

[1] https://github.com/zhouhaoyi/ETDataset

and six power load features. This dataset has been created to explore the granularity of the Long Sequence Time-Series Forecasting (LSTF) problem; thus, two subsets have been created from it, the first is a two hourly-level dataset (ETTh) and the second is a two 15-minute-level dataset (ETTm). The statistics of these datasets are provided in Table 4.1.

Table 4.1: Statistics on the ETT datasets used for the experiments in *UMS-Linear*

Datasets	ETTh1	ETTh2	ETTm1	ETTm2
Variables	7	7	7	7
Timesteps	17420	17420	69680	69680
Frequencies	1hour	1hour	15min	15min

In addition, we included some other datasets that are used to test our proposed model in real-world applications. These datasets are representative of a wide range of real-world applications, including weather forecasting, financial forecasting, and energy forecasting. The datasets vary in terms of their frequency, from hourly to monthly. These datasets can be summarized as follows:

- **Algeria Exchange rate**[2]. This is a monthly dataset that was collected from 1970 to November 2023 by the Food and Agriculture Organization (FAO). This dataset consists of the different values of the Algerian dinar against the US dollar over the years.

- **Algeria COVID 19**[3]. This dataset is a global daily confirmed case of the COVID-19 virus collected by the Johns Hopkins University Center for Systems Science and Engineering (JHU CCSE) from various sources. The data has been collected since the year 2020, and it was stopped by Jack JHU on March 10, 2023.

- **Algeria Fire alerts** [43][4]. This is a weekly dataset collected by the Global Forest Watch on January 2, 2012. This dataset consists of different confidence types of alerts, which are the high, normal, and low confidence alert levels.

[2]https://data.humdata.org/dataset/faostat-food-prices-for-algeria
[3]https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases
[4]https://www.globalforestwatch.org/dashboards/country/DZA/

- **Algeria Rainfall Indicator**[5]. This dataset is a 10-day frequency Algerian rainfall indicator computed from Climate Hazards Group Infrared Precipitation satellite imagery with in-situ station data (CHIRPS).

- **Algeria Weather Indicators**[6]. This is a daily dataset that provides historical weather data for Algiers, the capital city of Algeria, covering the period from January 2002 to August 2023. The data includes a variety of weather-related variables.

- **Food Price**[7]. The FFPI consists of the average of five commodity group price indices weighted by the average export shares of each of the groups over the period 2014–2016. The five commodity groups are cereals, meat, dairy, oils, and sugar.

- **Diesel Retail Price**[8]. This dataset offers weekly average diesel prices (USD/gallon) from October 2007 for New York State, including breakdowns for eight specific metropolitan regions.

- **Energy Demand Generation**[44][9]. This is an hourly dataset that offers four years of Spanish electrical consumption, generation, and pricing data. The consumption and generation information come from a public source (ENTSOE), while pricing data is from the Spanish TSO (Red Eléctrica España).

Table 4.2: Statistics on the real-world applications datasets used for the experiments in *UMS-Linear*

Datasets	Algeria Exchange rate	Algeria COVID 19	Algeria Fire Alerts	Algeria Rainfall	Algeria Weather	Food Price Index	Diesel Retail Price	Energy Demand Generation
Variables	1	1	3	9	17	6	17	23
Timesteps	753	1142	632	2296812	7913	410	856	35064
Frequencies	1month	1day	1week	10days	1day	1month	1 week	1hour

Table 4.2 provides statistics on these real-world application datasets. We use these datasets to demonstrate the effectiveness of our approach for forecasting real-world time

[5]https://data.humdata.org/dataset/dza-rainfall-subnational
[6]https://www.kaggle.com/datasets/bekkarmerwan/algiers-weather-data-2002-2023
[7]https://www.fao.org/worldfoodsituation/foodpricesindex/en/
[8]https://www.fao.org/worldfoodsituation/foodpricesindex/en/
[9]https://www.kaggle.com/datasets/nicholasjhana/energy-consumption-generation-prices-and-weather

series data, where these datasets vary in terms of their size, complexity, and frequency.

Usually, the existing methods in the time series forecasting task use a normalization technique as the only preprocessing step. The most commonly used normalization techniques are the max-min normalization and the standard normalization (standardization). Both of them aim to transform features within a dataset onto a similar scale. In our model, we used the standardization, which can be mathematically denoted as follows:

$$X = \frac{X - \mu}{\sigma}$$

Where X are the data points, μ is the mean of the full data, and σ is the standard deviation (std) of the data. This technique transforms the data to have a zero mean and a unit standard deviation. This makes the data follow a standard normal distribution.

4.3 Benchmark Methods Selection and Rationale

As a comparison model, we chose the same MLP nature LTSF-Linear [17] models (DLinear, NLinear), and in addition, we compared *UMS-Linear* with some other baseline models, which are PatchTST [32], TimesNet [26], AutoFormer [15], Informer [14] and FedFormer [16]. These baseline models represent a diverse range of approaches to time series forecasting, including transformer-based models, convolutional neural networks, and MLP-based models. By comparing *UMS-Linear* to these baseline models, we can evaluate its performance relative to other state-of-the-art methods.

4.4 Used Metrics

The metrics used to compare the models are the Mean Squared Error (MSE) and Mean Absolute Error (MAE), which are commonly used in the LTSF task. These metrics measure the difference between the forecasted values and the actual values. MSE is the average of the squared differences, while MAE is the average of the absolute differences.

MSE is sensitive to outliers, as it squares the differences between the forecasted values and the actual values. MAE, on the other hand, is not as sensitive to outliers, as it only takes the absolute value of the differences.

In the LTSF task, both MSE and MAE are important metrics to consider. MSE is useful for evaluating the overall accuracy of the forecasts, while MAE is useful for evaluating the robustness of the forecasts to outliers.

4.5 Results and Comparative Analysis with Baseline Models

The univariate Long-term time series forecasting results of *UMS-Linear* are shown in Table Table 4.3. Surprisingly, *UMS-Linear* achieved the best results in all 32 metrics. Remarkably, our model manages to achieve better results than the unincluded state-of-the-art (SOTA) models in both the ETTh1 and ETTm1 datasets. This shows that *UMS-Linear* is capable of handling difficult changes, even though its accuracy is still not perfect. However, it provides a promising starting point for future work.

Table 4.3: Univariate long-term forecasting results with *UMS-Linear*. ETT datasets are used with prediction lengths $T \in \{96, 192, 336, 720\}$. The best results are in bold and the second best results are in underlined.

Methods		*UMS-Linear*		NLinear		DLinear		PatchTST		TimesNet		FEDformer		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.048**	**0.168**	0.053	0.177	0.056	0.18	0.059	0.189	0.056	0.182	0.079	0.215	0.071	0.206	0.193	0.377
	192	**0.06**	**0.191**	0.069	0.204	0.071	0.204	0.074	0.212	0.072	0.209	0.104	0.245	0.114	0.262	0.217	0.395
	336	**0.066**	**0.203**	0.081	0.226	0.098	0.244	0.076	0.22	0.086	0.229	0.119	0.27	0.107	0.258	0.202	0.381
	720	**0.072**	**0.213**	0.08	0.226	0.189	0.359	0.087	0.236	0.082	0.228	0.142	0.299	0.126	0.283	0.183	0.355
ETTh2	96	**0.119**	**0.27**	0.129	0.278	0.131	0.279	0.131	0.284	0.136	0.286	0.128	0.271	0.153	0.306	0.213	0.373
	192	**0.155**	**0.312**	0.169	0.324	0.176	0.329	0.171	0.329	0.186	0.34	0.185	0.33	0.204	0.351	0.227	0.387
	336	**0.169**	**0.329**	0.194	0.355	0.209	0.367	**0.171**	0.336	0.22	0.373	0.231	0.378	0.246	0.389	0.242	0.401
	720	**0.184**	**0.345**	0.225	0.381	0.276	0.426	0.223	0.38	0.241	0.392	0.278	0.42	0.268	0.409	0.291	0.439
ETTm1	96	**0.025**	**0.12**	0.026	0.122	0.028	0.123	0.026	0.123	0.029	0.127	0.033	0.14	0.056	0.183	0.109	0.277
	192	**0.038**	**0.149**	0.039	**0.149**	0.045	0.156	0.04	0.151	0.047	0.163	0.058	0.186	0.081	0.216	0.151	0.31
	336	**0.051**	**0.171**	0.052	**0.172**	0.061	0.182	0.053	0.174	0.08	0.214	0.084	0.231	0.076	0.218	0.427	0.591
	720	**0.069**	**0.203**	0.073	0.207	0.08	0.21	0.073	0.206	0.084	0.222	0.102	0.25	0.11	0.267	0.438	0.586
ETTm2	96	**0.059**	**0.174**	0.063	0.182	0.063	0.183	0.065	0.187	0.066	0.187	0.067	0.198	0.065	0.189	0.088	0.225
	192	**0.088**	**0.22**	0.09	0.223	0.092	0.227	0.093	0.231	0.113	0.25	0.102	0.245	0.118	0.256	0.132	0.283
	336	**0.114**	**0.255**	0.117	0.259	0.119	0.261	0.121	0.266	0.133	0.277	0.13	0.279	0.154	0.305	0.18	0.336
	720	**0.157**	**0.311**	0.17	0.318	0.175	0.32	0.172	0.322	0.182	0.333	0.178	0.325	0.182	0.335	0.3	0.435
Count		32		22		2		8		0		2		0		0	

4.6 Case Studies or Real-World Applications

In this section, we present real-world applications of our proposed *UMS-Linear* model. We show how the model can be used to solve real-world problems and provide valuable insights; this will increase the credibility of the model and make it more likely to be adopted by practitioners. The real-world applications that will be included in this chapter will range from Algerian to global applications.

4.6.1 Algeria Exchange rate

The Algerian exchange rate is an important economic indicator that affects a wide range of businesses and individuals. By forecasting the exchange rate, we can help businesses and individuals make better decisions.

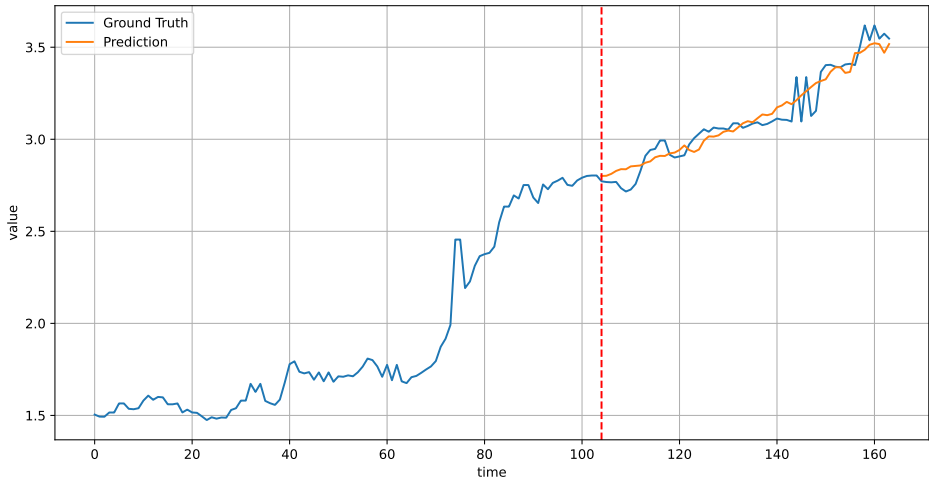


Figure 4.1: Visualisation of the forecasting prediction of *UMS-Linear* in the Algerian Exchange rate dataset using a forecasting horizon of 60 time steps

Figure 4.1 shows the result of using *UMS-Linear* in this Algerian exchange rate dataset, where we can see that the model can perfectly predict the overall trend of the data, but it still has some limitations in predicting the seasonal patterns of this dataset. This problem in predicting seasonal patterns is caused by a lack of data, and this problem cannot be avoided in such datasets.

4.6.2 Algerian COVID 19

The number of daily COVID-19 cases has been an important public health indicator in the past few years and could help policymakers make decisions about how to contain the spread of the virus. By forecasting the number of cases, we can help policymakers prepare for future waves of the virus and allocate resources accordingly.

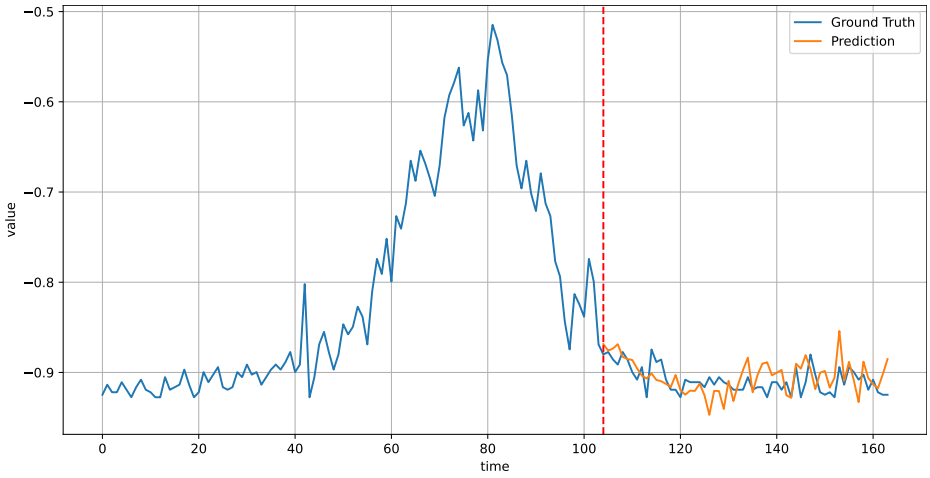


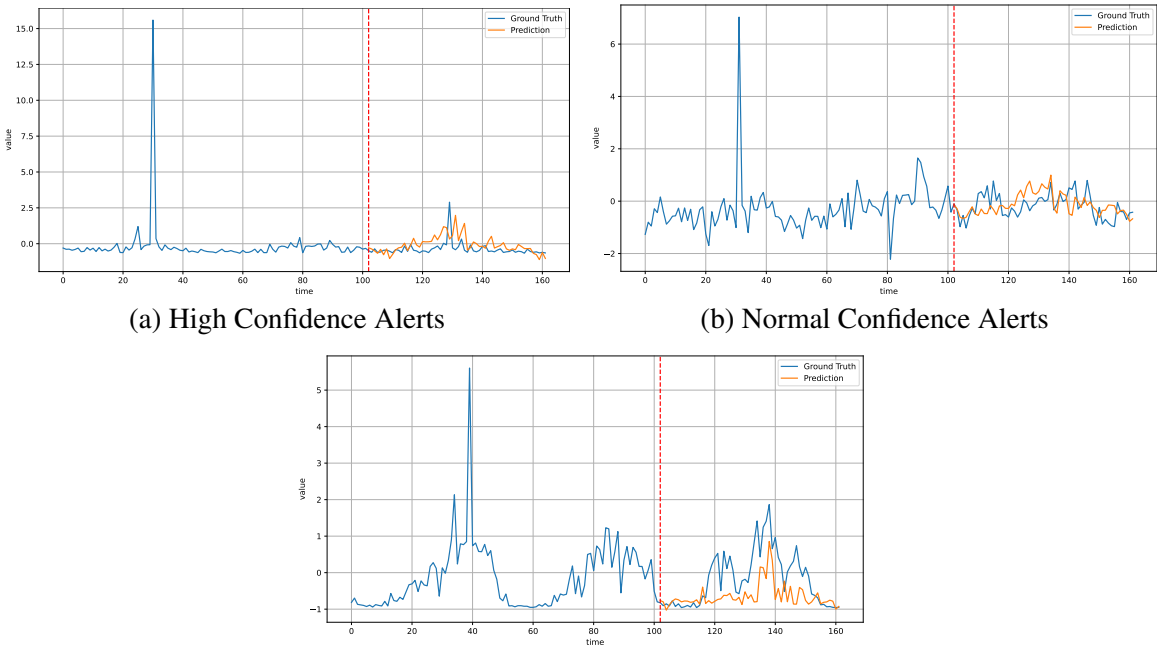
Figure 4.2: Visualisation of the forecasting prediction of *UMS-Linear* in the Algerian COVID 19 dataset using a forecasting horizon of 60 time steps

Figure 4.2 shows using *UMS-Linear* in this COVID 19 dataset, the lack of trend in this dataset caused the model to have problems in predicting the sudden waves of the virus. However, the model can predict the end of this waves normally, it is possible that *UMS-Linear* is able to predict the general shape and general pattern of the data and the times at which these patterns will be repeated.

4.6.3 Algerian Fire alerts

The number of weekly or daily fire alerts is an important indicator of the risk of wildfires. By forecasting the number of fire alerts, we can help policymakers take steps to prevent wildfires and protect lives and property.

This dataset has 3 different confidence level alerts, and as shown in Figure 4.3, our



(a) High Confidence Alerts

(b) Normal Confidence Alerts

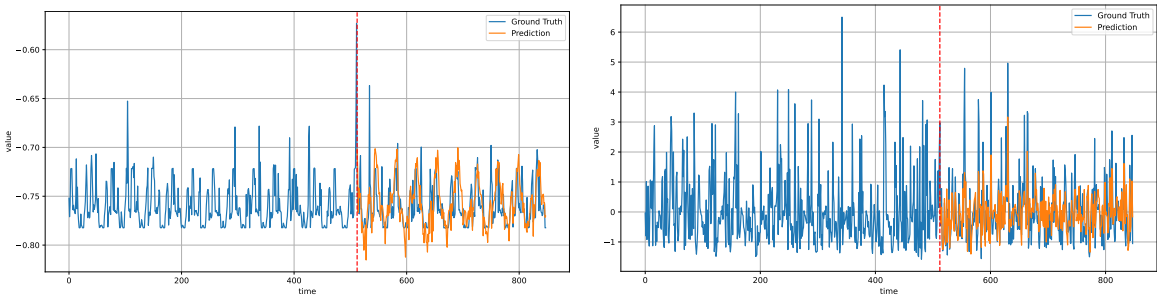
(c) Low Confidence Alerts

Figure 4.3: Visualisation of the forecasting prediction of *UMS-Linear* in the weekly Algerian fire alerts dataset using a forecasting horizon of 60 time steps

proposed *UMS-Linear* can predict the exact locations where fire alerts are increasing at a higher rate. This means that our model can track trends and seasonal patterns better in this real-world application, which will allow us to save more lives and properties by predicting the periods of time when fire rates are higher.

4.6.4 Algerian Rainfall Indicator

The next real-world application that we experimented with is the Algerian rainfall indicator, where the rainfall is an important indicator of the climate and water resources of a region. By forecasting rainfall, we can help farmers, water managers, and other stakeholders make better decisions. The dataset used contains different features, such as 10-day rainfall [mm] and rainfall 3-month anomaly [%]. In our case, we used only the 10-day rainfall [mm] (rfh) and the rainfall anomaly [%] (rfq), where these two features seem to contain no outliers or missing values. This will help the model get more accurate information to use in the prediction process.



(a) 10 day rainfall [mm] (rfh)

(b) rainfall anomaly [%] (rfq)

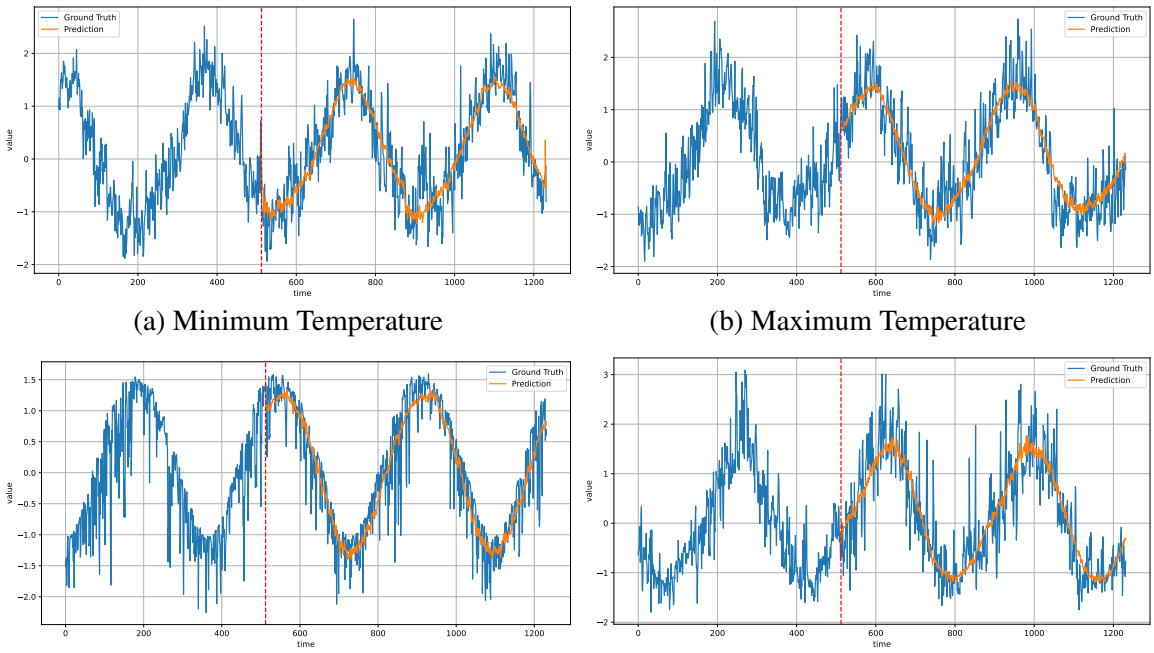
Figure 4.4: Visualisation of the forecasting prediction of *UMS-Linear* in the weekly Algerian Rainfall Indicator dataset using a forecasting horizon of 336 time steps

From Figure 4.4, we can observe that the *UMS-Linear* model is able to capture the seasonal and long-term trends in the rainfall data. The model also performs well in forecasting the rainfall trends of the data during periods of high and low rainfall. In the rainfall anomaly feature, we can clearly see that the data contains much more noise because of its nature. Even though *UMS-Linear* can predict the overall shape of it, it can also predict some of the sudden changes in this data.

4.6.5 Algerian Weather Indicators

Another application of time series forecasting is the weather forecasting task, which aims to predict the different values of the weather indicators, such as the maximum temperature and the apparent temperature. Usually, weather indicators have similar repeated patterns over the year, which are caused by the four seasons. This allows the models to better predict the future values of the values of the indicators.. However, weather forecasting is still an important task that can help people plan their activities and stay safe. By forecasting weather variables, we can help people make better decisions about when to go outside, what to wear, and how to prepare for severe weather events.

In Figure 4.5, we can clearly see that *UMS-Linear* can easily predict the trend of the data, where, as we mentioned before, these weather indicators have similar repeated trend forms, such as the increase in temperature during the summer months and the increase in



(a) Minimum Temperature

(b) Maximum Temperature

(c) Shortwave Radiation Sum

(d) Reference Evapotranspiration (ET_0)

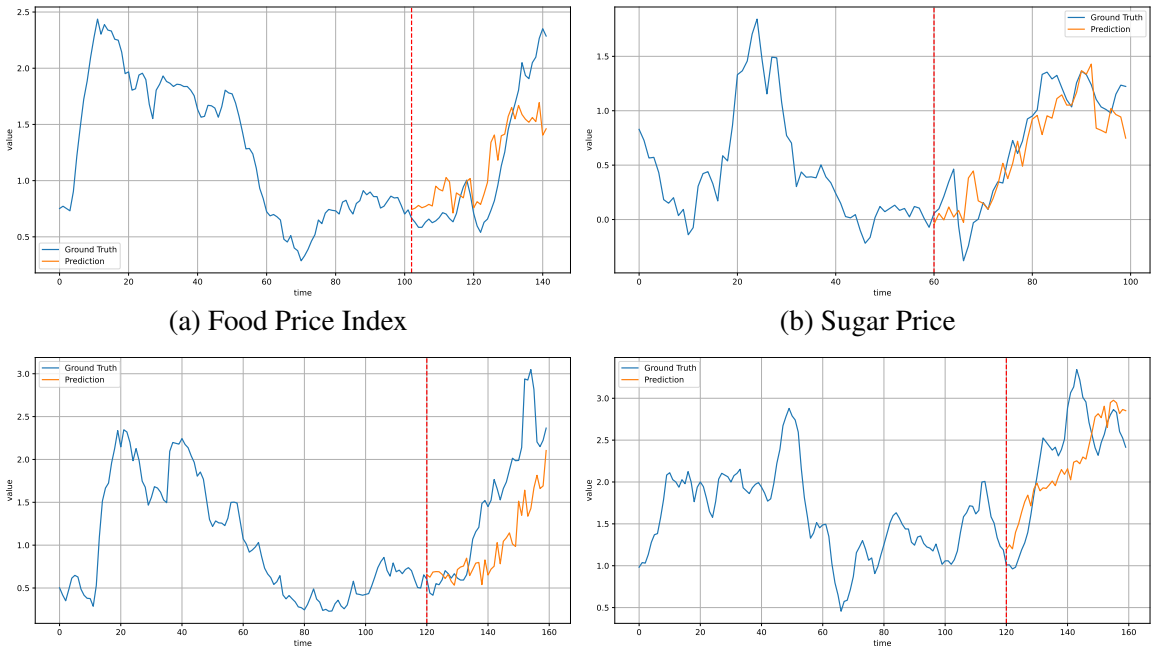
Figure 4.5: Visualisation of the forecasting prediction of *UMS-Linear* in the Algerian Weather Indicators datasets using a forecasting horizon of 720 time steps

precipitation during the winter months. However, the model cannot track the details, which include small fluctuations, which is not an important thing where these fluctuations could be just noise in the data, where identifying the overall shape of the data is more important.

4.6.6 Food Prices

Food prices is one of the many important applications in time series forecasting, where forecasting food prices can help businesses and consumers to make better decisions. This food prices can vary from different categories such as sugar, oils, meat and dairy.

As shown in Figure 4.6, the *UMS-Linear* model is able to capture the overall trend of the food price data well in the multiple food price types. The model also captures the seasonal patterns in the data, such as the increase in prices. However, this dataset is so small that the model cannot learn the different patterns in it, which confirms the limitations of *UMS-Linear* and linear models in small datasets. There is still room for improvement in the accuracy of the forecasts, where we can try using a more complex model or using a



(a) Food Price Index

(b) Sugar Price

(c) Cereals Price

(d) Meat Price

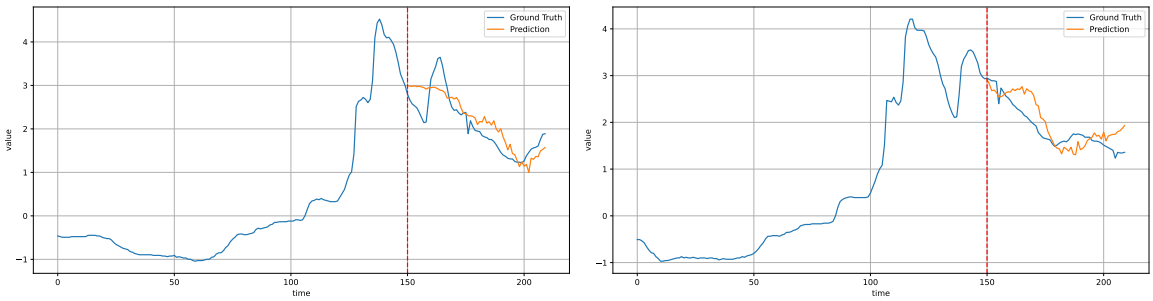
Figure 4.6: Visualisation of the forecasting prediction of *UMS-Linear* in the global food price indexes dataset using a forecasting horizon of 40 time steps

method to increase the dataset size, and we can also try extracting more features from the data.

4.6.7 Diesel Retail Price

Diesel is a type of fuel that is used in a variety of vehicles, including trucks, buses, and cars. The price of diesel is an important factor in the cost of transportation. By forecasting the diesel retail price, we can help businesses and consumers make better decisions.

In our case, we used the average New York City and Utica-Rome diesel prices. As shown in Figure 4.7, our proposed *UMS-Linear* is able to capture the overall trend of the diesel retail price data well for both cities. However, in further experiments, we find that the model struggles to predict the increasing changes in some periods of time; this could be caused by the high values in the data.



(a) New York City (b) Utica-Rome

Figure 4.7: Visualisation of the forecasting prediction of *UMS-Linear* in the Diesel Retail Prices dataset using a forecasting horizon of 40 time steps

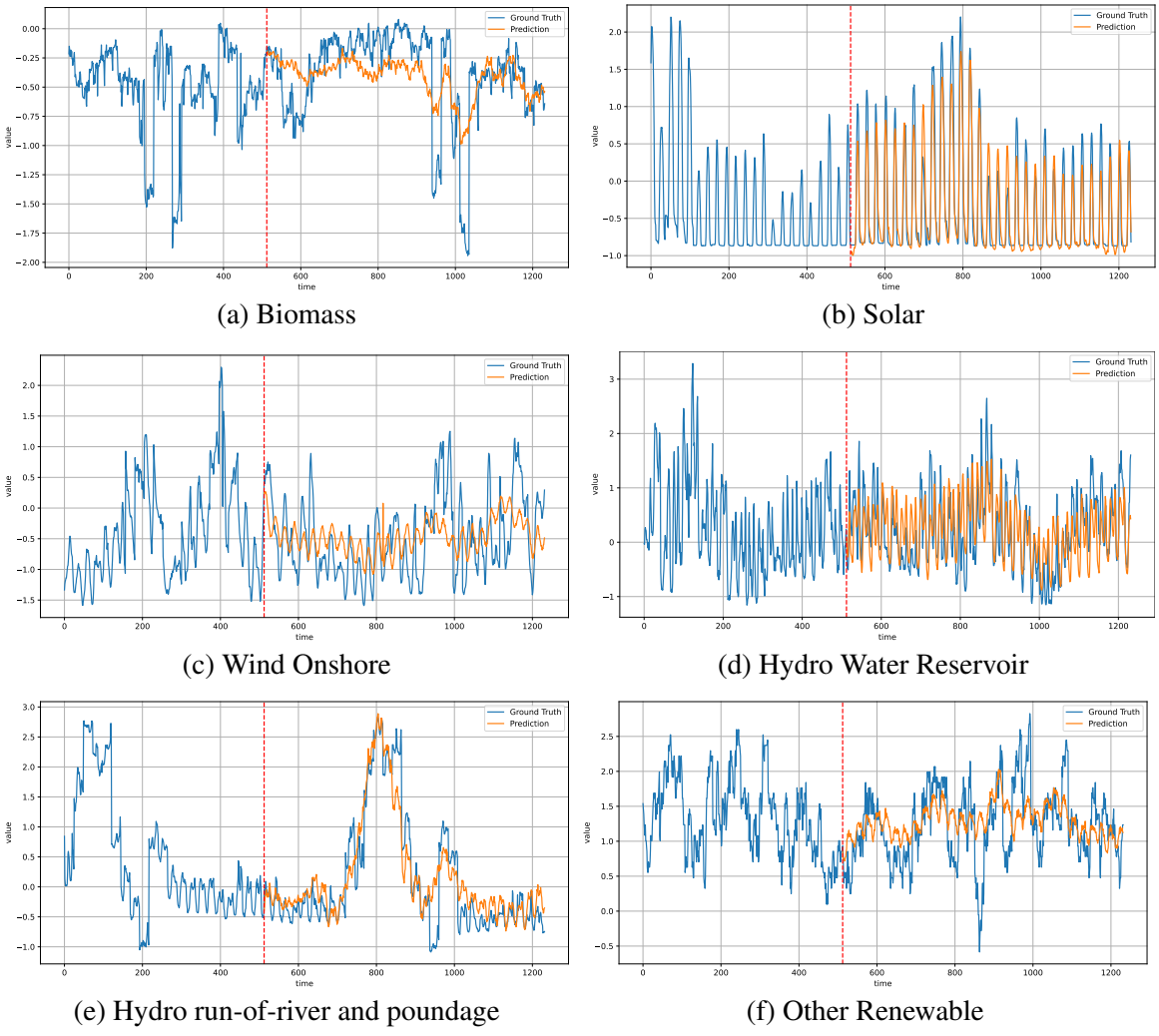
4.6.8 Energy Generation

There are countless other applications in which we can use our model, but in this part we only discussed the important applications that were available during the creation of this research, and to conclude these applications, we chose energy generation forecasting to test our proposed model. Energy generation forecasting is an important task that can help grid operators ensure that there is enough electricity to meet demand. By forecasting energy generation, grid operators can make better decisions about when to dispatch different power plants.

As shown in Figure 4.8, we can use *UMS-Linear* in forecasting renewable energy generation, where we can clearly see that *UMS-Linear* is able to capture the trends in the different datasets. The model also captures the seasonal patterns in the data, which confirms the usability of our model in such datasets.

In addition to forecasting renewable energies, we can also forecast fossil fuel-based energy generations. As shown in Figure 4.9, our proposed model is able to predict trends and the overall shape of the data better. This performance in these types of energy generation datasets shows that *UMS-Linear* is an effective tool in time series forecasting that is energy-related.

And at the end, we will observe two additional features from this promising energy dataset, which are the average prices and the total load of the generated energy. And as



(a) Biomass

(b) Solar

(c) Wind Onshore

(d) Hydro Water Reservoir

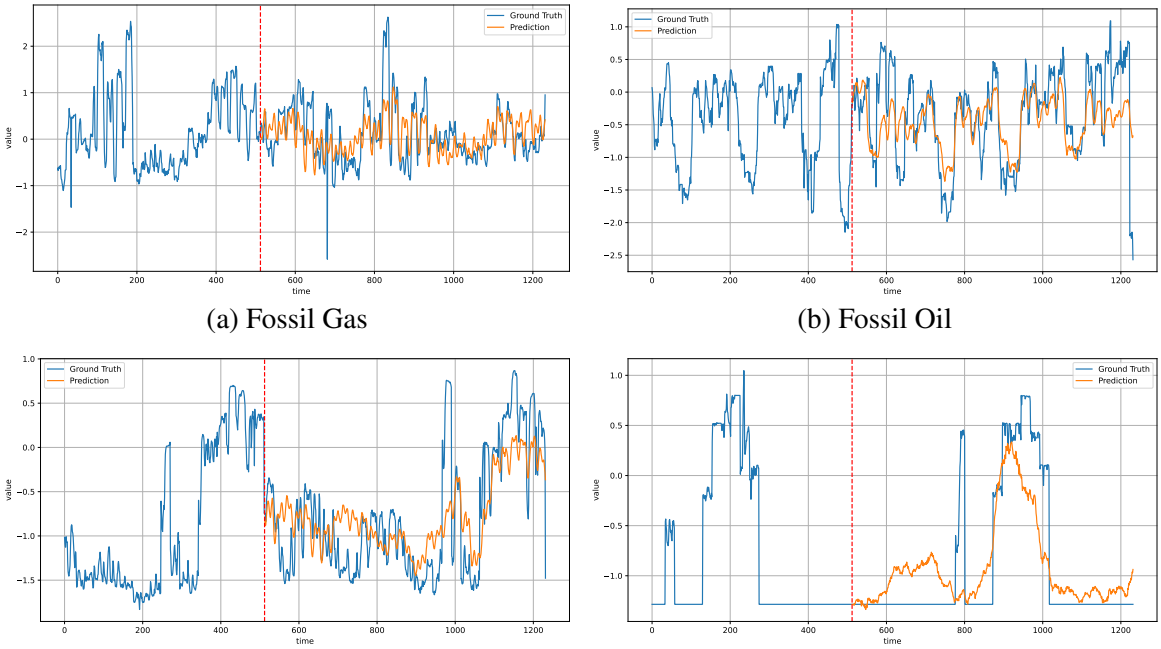
(e) Hydro run-of-river and poundage

(f) Other Renewable

Figure 4.8: Visualisation of the forecasting predictions of *UMS-Linear* in the renewable energy generation dataset using a forecasting horizon of 720 time steps

shown in Figure 4.10, *UMS-Linear* can predict the seasonal patterns and the overall shape of the data, even though it's different from the other variables that were mentioned before. This confirms that our model is able to handle multiple types of data, even if they include fluctuations or noise.

The real-world application presented in this section demonstrates the effectiveness of the *UMS-Linear* model in forecasting real-world time series data. The model is able to capture complex patterns in the data and make accurate forecasts. This makes the model a valuable tool for decision-making in a variety of forecasting problems. We believe that



(a) Fossil Gas

(b) Fossil Oil

(c) Fossil Hard Coal

(d) Fossil Brown Coallignite

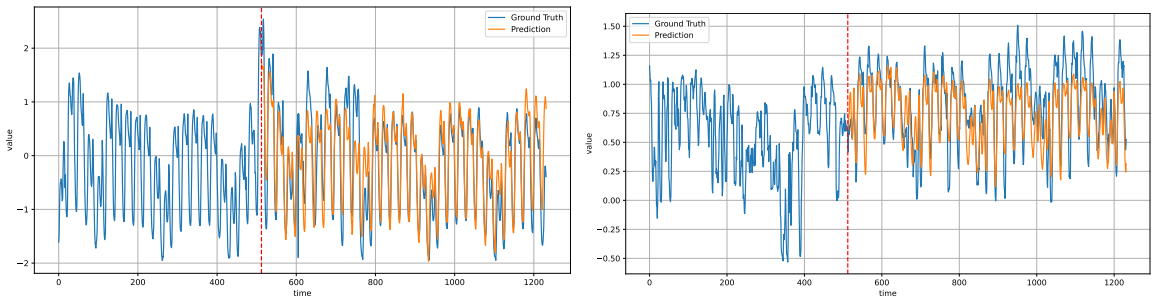
Figure 4.9: Visualisation of the forecasting predictions of *UMS-Linear* in the Fossil fuel-based generation dataset using a forecasting horizon of 720 time steps

the *UMS-Linear* model can be used to forecast a wide range of other real-world time series data. The model is simple to use and can be easily adapted to different forecasting problems.

4.7 Robustness and Sensitivity Analysis

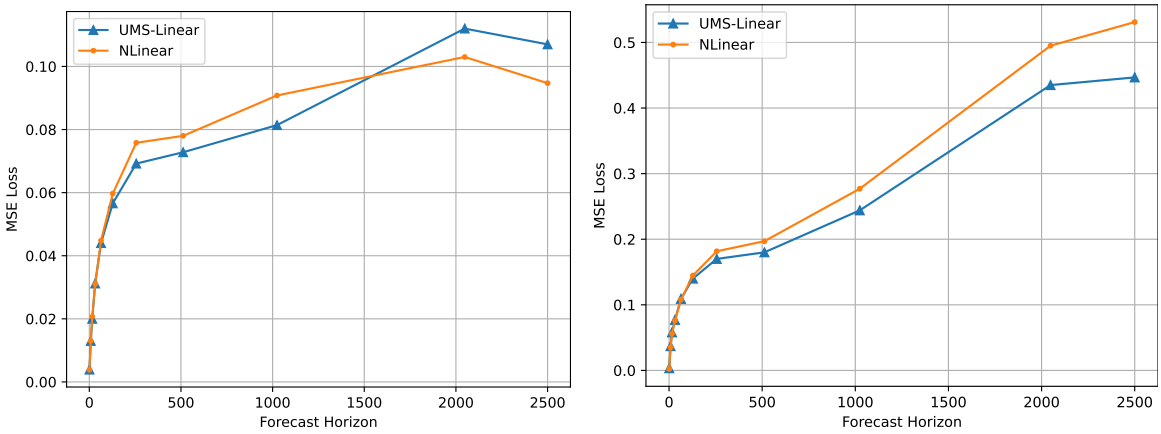
In this section, we conduct a robustness and sensitivity analysis of our proposed *UMS-Linear* model. We first train the model on multiple forecast horizons from 8 to 2500 and visualize the MSE loss. We then analyze the number of parameters in the model and the training time in both the ETTh1 and ETTh2 datasets. This will allow us to better understand how our model handles large scales and types of data, and it will also give us confirmation about the usability of *UMS-Linear* in real-world datasets.

Figure 4.11 shows comparison results between NLinear and *UMS-Linear* in the ETTh1 and ETTh2 datasets in multiple short-term and long-term forecasting horizons. Because



(a) Total Load (b) Actual Price

Figure 4.10: Visualisation of the forecasting predictions of *UMS-Linear* in the additional Demand & Cost features of the energy dataset using a forecasting horizon of 720 time steps



(a) ETTh1 (b) ETTh2

Figure 4.11: A comparision between the performance of *UMS-Linear* against NLinear in both ETTh1 and ETTh2 dataset in multiple forecasting horizons

of the shared nature between these two, the results are similar, where we can clearly see that with each time the forecasting horizon increases, the MSE loss increases at a slow rate. This proportionality forms a curve similar to the log curve. In the ETTh1 dataset we can observe that in the last larger forecasting horizons, the NLinear model manages to get better results, which might be caused by a mismatch in the training parameter. Our model, on the other hand, shows a good performance in larger forecasting horizons in ETTh2. This is because of the fewer fluctuations in the data, where the model could perform better on learning its overall shape.

Most models from different categories (e.g., RNN, CNN, Transformers) contain mil-

Table 4.4: Comparison of performance metrics between *UMS-Linear*, NLinear and DLinear on ETTh1 and ETTh2 datasets with a single NVIDIA T4 GPU. The look-back is 336 and the horizon is 720.

Method		UMS-Linear	Nlinear	Dlinear
Training Time	ETTh1	0,046	0,045	0,041
(s/epoch)	ETTh2	0,048	0,044	0,044
Parameters	ETTh1	2.1	0.36	0.73
(M)	ETTh2	2.1	0.36	0.73

lions of parameters, which results in a long training time of up to minutes. Linear models are more efficient in time series forecasting where training time per epoch is counted by fractions of a second, and mostly these linear models consist of a smaller number of parameters, which makes them better for larger long-term forecasting tasks. The three linear layers in our model result in a higher parameter number, as shown in Table 4.4, but this does not really affect the training time, where the difference between the training model and the one layered NLinear model in the ETTh1 dataset in a forecast horizon of 720 is only $0.001 second$. Overall, the *UMS-Linear* model is a robust and efficient model for time series forecasting.

4.8 Experiments

4.8.1 Multi-Scale decomposition in linear models

Scaling in general is considered a preprocessing technique that is used to transform the values of a dataset so that they are all within a similar range. This can be useful for improving the performance of machine learning models, as many models are sensitive to the scale of the data. One simple way to scale data is to multiply all of the values by a constant factor, as used in *UMS-Linear*. This type of scaling can improve the model performance and reduce overfitting, as it makes it less likely that the model will learn the specific values of the data rather than the underlying patterns. In some cases, scaling the data can also help to improve the interpretability of the model, as it makes it easier to see the underlying

patterns and trends.

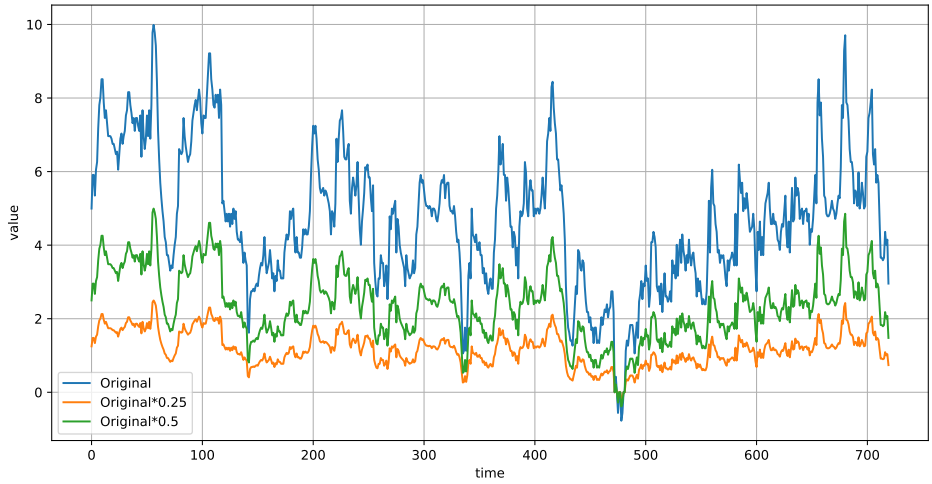
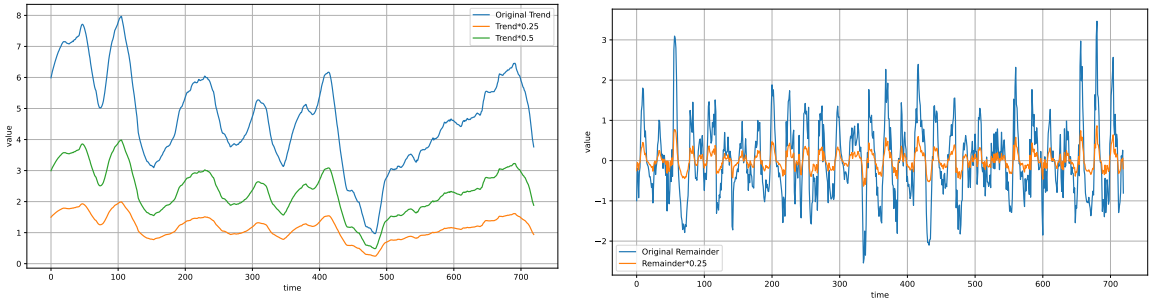


Figure 4.12: The effect of scaling the data by factor multiplication in ETTh1 dataset

As can be seen from Figure 4.12, scaling the original time series has a significant effect on its shape. The scaled time series is more compressed and has a smaller range of values, whereas scaling the original time series can help reduce the impact of outliers and extreme values. Decomposing the data into trend and remainder will give us more interpretability, where, as shown in Figure 4.13, scaling the remainder can help to remove noise from the time series. While scaling the trend can help make the time series more stationary,.



(a) Trend Component (b) Remainder Component

Figure 4.13: The effect of scaling the data components (left: Trend, right: Remainder) by factor multiplication in ETTh1 dataset

In order to prove that combining different data components on multiple scales will improve the forecasts in some scenarios, we replaced the decomposition in DLinear by the

multi-scale decomposition phase of *UMS-Linear*, but for a fair comparison, we used the standard moving average that DLinear used. Using the standard moving average will reduce the effect of the multi-scaling process, where the model will not pick the important changes that occur in the data; instead, the moving average will smooth out the sharp changes in the data. However, in special cases where the forecast contains change points, the modified DLinear model shows better ability than the original DLinear model in tracking data. Figure 4.14 shows one of these special cases in the ETTm1 dataset where the multi-scale decomposition model improved the original DLinear model prediction by 0.2214 in terms of the MSE loss. If we take a deep look at the new multi-scale model forecast, we can clearly observe that the model could literally predict the general shape of the forecast even though the amplitude of prediction is still quite far from the truth values.

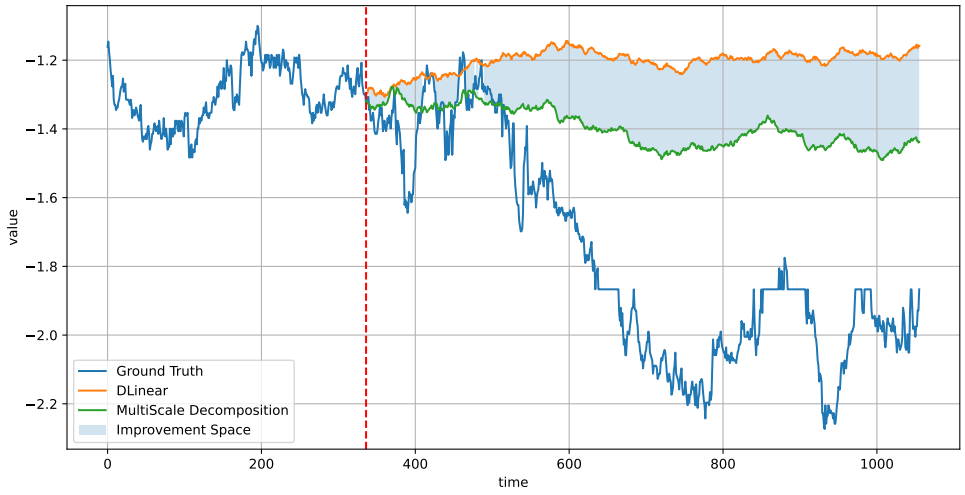


Figure 4.14: The improvement between using the Multi-Scale Decomposition and the original DLinear model in ETTm1 dataset at the forecast horizon of 720

Another observation of the effect of using this technique in special cases can be seen in the ETTm2 dataset in Figure 4.15, where the modified model could improve the forecast prediction MSE loss of the DLinear model by 0.2427. Specifically, the linear model with multiscale decomposition is able to capture the sharp decrease in the time series at around

time step 500. The linear DLinear, on the other hand, is not able to capture this decrease as well. This confirms that multiscale decomposition can be particularly helpful for time series that have sudden changes or sharp increases or decreases.

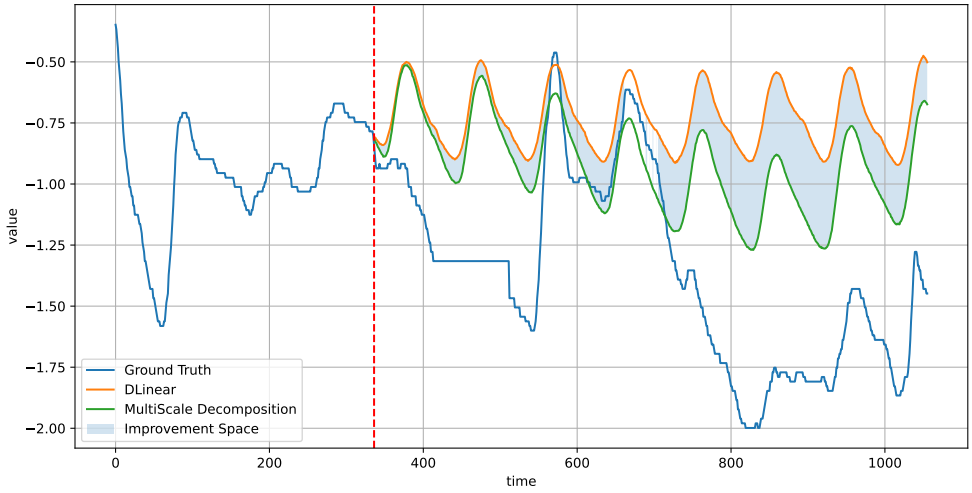


Figure 4.15: The improvement between using the Multi-Scale Decomposition and the original DLinear model in ETTm2 dataset at the forecast horizon of 720

4.8.2 Merging timestamps with linear models

In long-term time series forecasting, models struggle to capture the patterns produced by similar input data, where over time the data may produce new or different patterns from similar known patterns. The use of timestamps will result in additional time information that helps the model identify the time period of the current forecast. This will allow the model to distinguish similar patterns to create the required forecasts for a specific period of time. To prove this assumption, we combined the timestamp embedding with the input data, similar to the process when using positional encoding, where if we denote the input data at time t as X and the timestamp embedding for this data point as E_t, then we could

combine this embedding with the data as a simple addition as follows:

$$\hat{X} = X_t + E_t$$

This will produce a new series that we could use to predict the forecast values. In order to show the effect of using the timestamps, we will provide a long-term time series comparison with a simple one-layer linear model. It is worth noting that we tried to use the positional encoding technique as a comparison, but it achieved bad results.

Table 4.5: A comparison between using timestamps and the original simple linear model in ETT dataset at long-term forecasts horizon $T \in \{720, 1500\}$

Methods		IMP	TimeStamps Linear		Linear	
Metric		MSE	MSE	MAE	MSE	MAE
ETTh1	720	34,60%	**0,115**	**0,267**	0,176	0,345
	1500	75,30%	**0,141**	**0,3**	0,572	0,662
ETTh2	720	22,40%	**0,225**	**0,379**	0,29	0,438
	1500	57%	**0,218**	**0,375**	0,507	0,59
ETTm1	720	0%	**0,085**	**0,218**	0,081	0,212
	1500	11,30%	**0,102**	**0,24**	0,115	0,264
ETTm2	720	6,80%	**0,164**	**0,308**	0,176	0,321
	1500	12,30%	**0,213**	**0,357**	0,243	0,39

Table 4.5 shows the result of this comparison, where the timestamps model manages to improve the linear model's predictions, especially in ETTh1 and ETTh2 datasets, where these improvements are up to 75% in terms of MSE loss. The improvements in the ETTm dataset are not that big, but they're still a remarkable improvement, especially because of the nature of this dataset.

It is clear that the forecasting results of the vanilla linear model are improved when we combine the timestamp embedding with this simple model. Figure 4.16 can give us confirmation of this result, where the timestamps model is able to handle the sudden decrease in the middle of the forecast, while on the other hand, the standard linear model could not give us any information about the trend changes in the forecast. In a special location of this visualization, we can clearly see that the timestamps model is managed to perfectly predict the future values (e.g., between 1100 and 1200). This shows that we could use timestamps

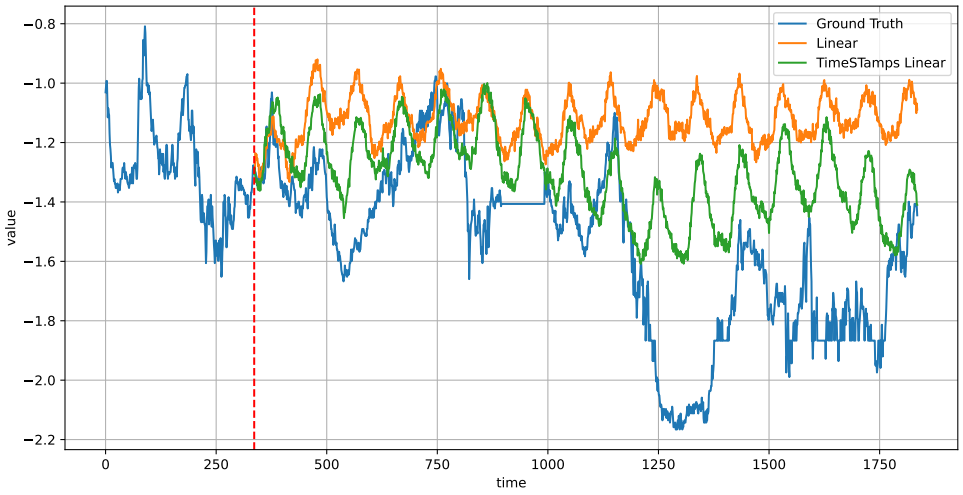


Figure 4.16: Visualisation of the impact of using timestamps in a vanilla linear model in ETTm1 dataset at a forecast horizon of 1500

in a different, enhanced way to predict more realistic and accurate future values.

Also, in Figure 4.17, which represents a visualization of the standard linear model with and without the timestamp embedding, we can see that the standard linear model without the timestamps produced a repeatedly forecast where we cannot benefit from it because it is devoid of trends. Using timestamps allows the model to learn more underlying patterns and trends from small inputs. We can conclude that this experiment shows that incorporating timestamps into our simple linear model improves its accuracy. This is because timestamps provide the model with additional information about the temporal relationships between the data points.

4.8.3 Impact of the new Loss function

The loss function is an important component in training models in deep learning tasks, where training a model in a special task with a specific loss function will produce different results than training it with another loss function. This is also the case in time series forecasting, where models could perform better when using specific loss functions that cor-

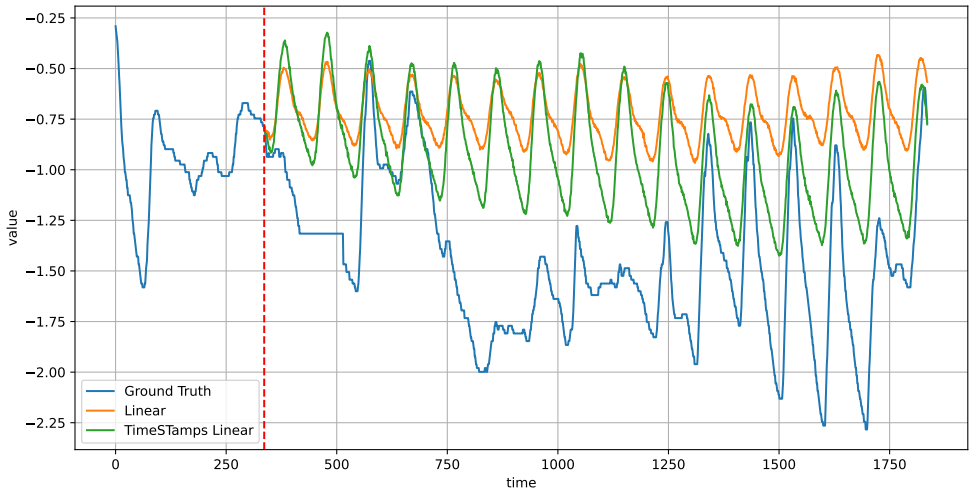


Figure 4.17: Visualisation of the impact of using timestamps in a vanilla linear model in ETTm2 dataset at a forecast horizon of 1500

respond to the nature of the used model. In this experiment, we will prove and study the impact of using the different loss functions that we mentioned in section 3.7, where we will train *UMS-Linear*, NLinear, and DLinear using these loss functions.

Starting with the univariate results, Table 4.6 shows the comparison we mentioned before, where we compare the results of each model separately when training with different loss functions. Our proposed loss function is more effective for time series forecasting because it takes into account incremental changes. This loss function achieved the best results in total of 72 out of 96 metrics, which is 75%. The difference that is used in our proposed loss function is important for time series forecasting, as it can help the model learn the underlying patterns and trends in the data. The MSE loss function only takes into account the squared error between the predicted and actual values. This can lead to the model learning to predict the mean of the data rather than the underlying patterns and trends. The combined MSE/MAE loss function takes into account both the squared error and the absolute error between the predicted and actual values. This can help the model learn to predict the mean of the data as well as the underlying patterns and trends. However, it does not take

Table 4.6: A separately comparison between training *UMS-Linear*, NLinear and DLinear [17] using different loss function where loss 1 is the default MSE loss, loss 2 is the combined MSE/MAE loss and finally loss3 is our presented MSE/MAE/Difference loss, results are in bold and the second best results are in underlined.

Methods		UMS-Linear						NLinear						DLinear					
Train Loss		Loss 3		Loss 2		Loss 1		Loss 3		Loss 2		Loss 1		Loss 3		Loss 2		Loss 1	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0,049**	**0,17**	0,05	0,173	0,052	0,174	**0,053**	**0,177**	**0,053**	**0,177**	**0,053**	**0,177**	0,059	0,183	0,059	0,182	**0,056**	**0,18**
	192	**0,062**	**0,194**	0,067	0,204	0,064	0,196	**0,068**	**0,203**	**0,068**	**0,203**	0,069	0,204	0,072	0,205	0,08	0,214	**0,071**	**0,204**
	336	**0,069**	**0,207**	0,07	0,209	0,072	0,212	**0,08**	**0,225**	**0,08**	0,226	0,081	0,226	**0,086**	**0,232**	0,092	0,238	0,098	0,244
	720	0,074	**0,214**	0,077	0,218	**0,073**	0,215	0,081	0,227	0,081	0,227	**0,08**	**0,226**	**0,18**	**0,349**	0,181	0,351	0,189	0,359
ETTh2	96	**0,127**	**0,28**	0,132	0,284	0,13	0,284	**0,128**	**0,276**	**0,128**	**0,276**	0,129	0,278	**0,13**	**0,277**	**0,13**	**0,277**	0,131	0,279
	192	0,16	0,318	0,163	0,32	**0,157**	**0,315**	0,168	**0,322**	**0,167**	**0,322**	0,169	0,324	**0,171**	**0,325**	**0,171**	**0,325**	0,176	0,329
	336	**0,178**	**0,339**	0,182	0,341	0,187	0,346	0,193	**0,353**	**0,192**	**0,353**	0,194	0,355	**0,204**	**0,363**	**0,204**	**0,363**	0,209	0,367
	720	**0,194**	**0,354**	0,198	0,356	0,201	0,359	**0,221**	**0,377**	**0,221**	**0,377**	0,225	0,381	0,284	0,433	0,282	0,431	**0,276**	**0,426**
ETTm1	96	0,026	**0,121**	0,026	0,122	**0,025**	**0,121**	**0,026**	**0,121**	**0,026**	**0,121**	0,026	0,122	**0,028**	0,125	0,029	0,125	**0,028**	**0,123**
	192	**0,039**	**0,151**	0,04	0,152	0,04	0,152	**0,039**	**0,148**	**0,039**	**0,148**	**0,039**	0,149	0,046	0,159	0,056	0,175	**0,045**	**0,156**
	336	0,053	**0,175**	**0,052**	**0,175**	**0,052**	0,176	**0,052**	**0,171**	**0,052**	0,172	**0,052**	0,172	0,064	0,187	0,064	0,188	**0,061**	**0,182**
	720	**0,07**	**0,205**	0,071	**0,205**	**0,07**	**0,205**	**0,072**	**0,205**	**0,072**	**0,205**	0,073	0,207	0,082	0,213	0,081	0,212	**0,08**	**0,21**
ETTm2	96	**0,06**	**0,176**	**0,06**	**0,176**	0,062	0,18	**0,063**	**0,181**	**0,063**	**0,181**	0,063	0,182	**0,063**	**0,181**	**0,063**	**0,181**	**0,063**	0,183
	192	**0,089**	**0,221**	**0,089**	**0,221**	0,092	0,228	**0,09**	**0,223**	**0,09**	**0,223**	**0,09**	**0,223**	**0,091**	**0,225**	**0,091**	**0,225**	0,092	0,227
	336	**0,122**	**0,265**	**0,122**	0,266	0,123	0,268	**0,116**	**0,257**	0,117	0,259	0,117	0,259	**0,118**	**0,259**	**0,118**	**0,259**	0,119	0,261
	720	**0,165**	0,316	0,166	0,317	0,169	**0,311**	**0,169**	0,318	**0,169**	**0,317**	0,17	0,318	**0,172**	**0,317**	**0,172**	**0,317**	0,175	0,32
Count		26		8		9		27		26		8		19		14		15	

into account the difference between the predicted and actual values. Our proposed loss function takes into account the squared error, the absolute error, and the difference in the incremental changes. This allows the model to learn to predict the mean of the data, the underlying patterns and trends, and the incremental changes in patterns.

Table 4.7: A comparison between training NLinear [17] using different loss functions in multivariate in ETT dataset using a forecasting horizon of $T \in \{96, 720\}$, results are in bold and the second best results are in underlined.

Methods		NLinear					
Train Loss		Loss 3		Loss 2		Loss 1	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0,368**	**0,389**	0,368	**0,389**	0,373	0,394
	720	**0,43**	**0,447**	0,431	**0,447**	0.441	0,453
ETTh2	96	**0,276**	**0,333**	0,278	0,335	0,283	0,342
	720	**0,392**	**0,428**	0,467	0,473	0,398	0,437
ETTm1	96	**0,299**	**0,341**	0,301	0,343	0,305	0,347
	720	**0,427**	**0,415**	0,431	0,418	0,433	0,421
ETTm2	96	**0,162**	**0,249**	0,163	**0,249**	0,164	0,254
	720	**0,366**	**0,381**	0,367	**0,381**	0,368	0,384
Count		16		4		0	

Using our proposed loss function is clearly better in training the linear models, but for a more robust experiment, we trained the NLinear [17] model in the multivariate time series forecasting task, where we can clearly see from Table 4.7 that our proposed loss function (Loss 3) outperforms the MSE loss function (Loss 1) and the combined MSE/MAE loss

function (Loss 2) in terms of both MSE and MAE on all four datasets. This shows that our proposed loss function is more effective for multivariate time series forecasting.

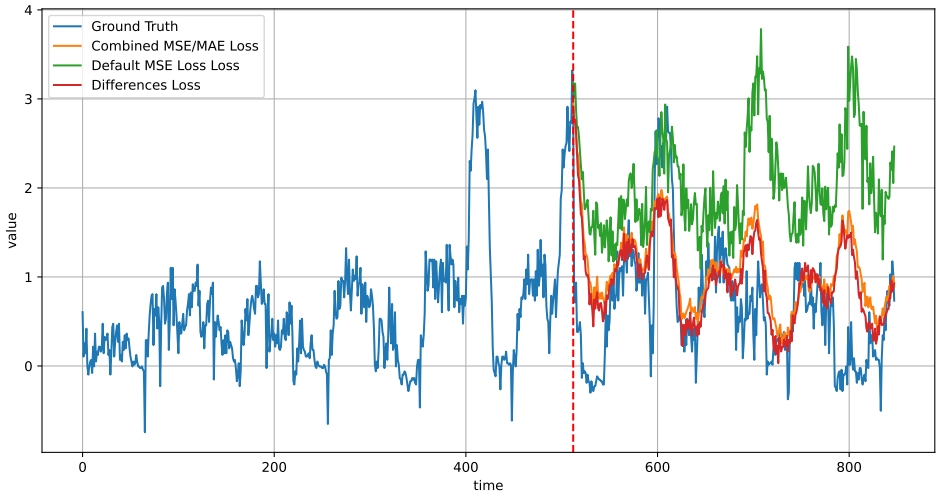


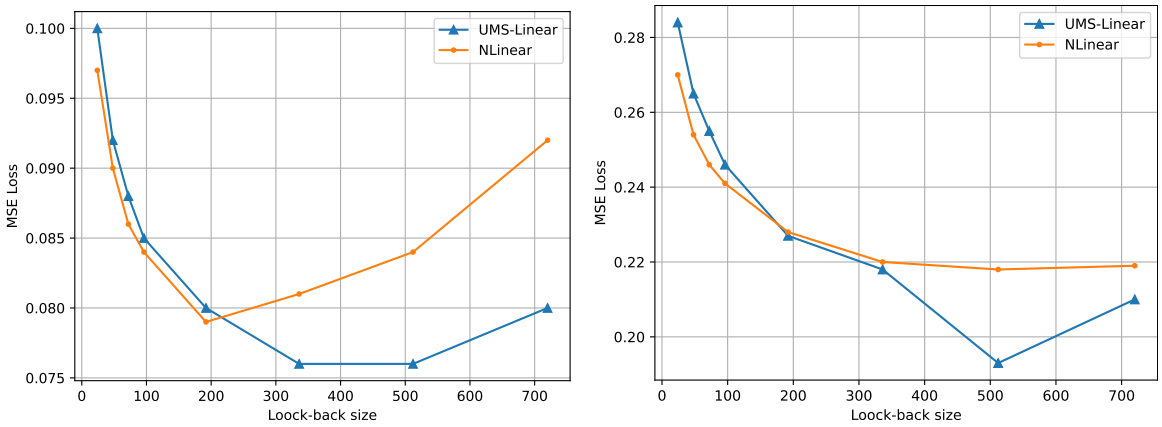
Figure 4.18: Visualisation of the impact of using different loss functions in *UMS-Linear* in ETTm1 dataset at a forecast horizon of 336 in MULL channel

In Figure 4.18 we can see a simple comparison between training *UMS-Linear* in ETTm1 dataset in MULL channel, where we can clearly see that the prediction of *UMS-Linear* that is trained with the default MSE loss function does not handle the sudden changes, which result in a less accurate prediction. On the other hand, the combined MSE/MAE loss and our proposed loss show a good capability in predicting the overall shape of the data while handling the change points.

4.8.4 Impact of look-back length

To verify that our linear-based model could handle large look-backs and doesn't suffer from the same problems with long-term information as transformer-based models [17], we trained our model with different look-backs and compared it to the NLinear model on the ETTh1 and ETTh2 datasets. As illustrated in Figure Figure 4.19, our model performs better when increasing the look-back window size, leading to more accurate predictions. This is

in contrast to transformer-based models, which can suffer from performance degradation when the look-back window size is too large. This shows that our linear-based model is able to effectively handle long-term information in time series data and is not as sensitive to the look-back window size as transformer-based models.



(a) ETTh1 (b) ETTh2

Figure 4.19: The forecast MSE error of *UMS-Linear* with different look-back length on ETTh1 and ETTh2 datasets. The horizon is 720.

Also, if we compare the predictions of *UMS-Linear* in different look-backs, we can clearly see the difference, where, as seen in Figure 4.20, the predictions of our model become more accurate as the look-back window size increases. This is because a larger look-back window allows the model to gain more information about the data, whereas predictions with small window sizes contain fewer trends. In a bigger look-back, it is clear that the model can predict better trends.

One possible explanation for the superior performance of our linear-based model on large look-back windows is that it is able to learn the underlying patterns and trends in the data more effectively. This is because of the different components that *UMS-Linear* used, which help it learn the underlying dynamics of the data. Transformer-based models, on the other hand, may have difficulty learning the underlying patterns and trends in data with large look-back windows. This is because transformer-based models rely on attention mechanisms, which can become less effective when the look-back window size is too large.

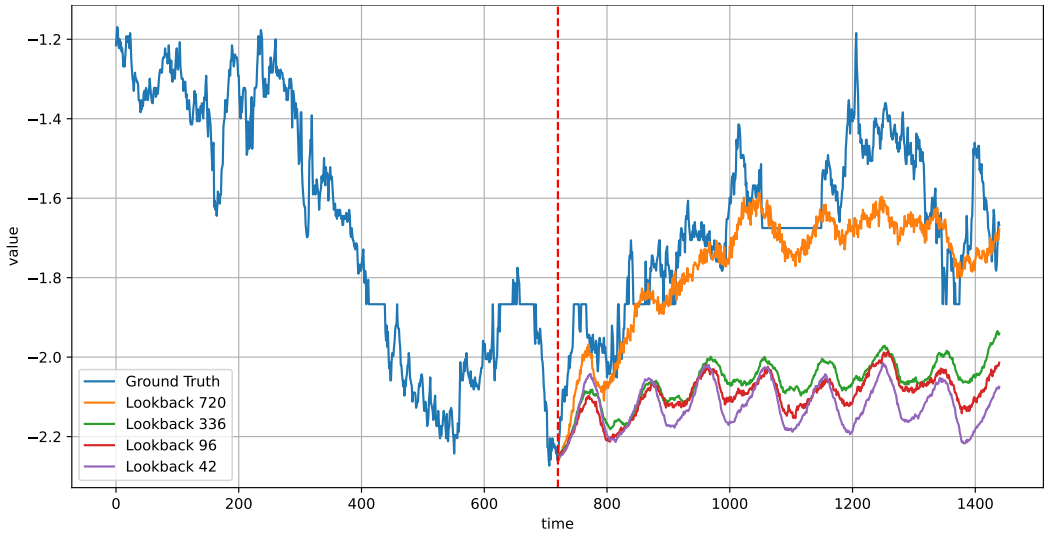


Figure 4.20: Visualisation of the impact of using different look-back window sizes $T \in \{48, 94, 336, 720\}$ in *UMS-Linear* in ETTm1 dataset at a forecast horizon of 336

Overall, our results show that our linear-based model is able to effectively handle large look-back windows and is not as sensitive to the look-back window size as transformer-based models. This makes our model a good choice for time-series forecasting tasks with long-term dependencies. However, it is important to note that there is a trade-off between look-back length and computational cost. A longer look-back length will require more computational resources to train and use the model. Therefore, it is important to choose a look-back length that is long enough to capture the important patterns and trends in the data but not so long that the computational cost becomes prohibitive.

4.8.5 Impact of Training parameters

We experimented with different training parameters to find the best settings for our model. We used a grid search approach to find the best learning rates and batch sizes; this will allow us to see how our model performs under different parameters.

Learning Rate. We experimented with different learning rates, ranging from 0.005 to 0.09. This will give us an overview of whether the model needs a high learning rate or a low learning rate. As can be seen from Figure 4.21, the MSE loss decreases as the learning rate

decreases. This is because a high learning rate can cause the model to overfit the training data. Also, the nature of our model, which contains two separate phases, may cause one phase to converge before the other, which results in bad results caused by random peaks in the predictions. Using small learning rates will ensure equality in learning among the two phases.

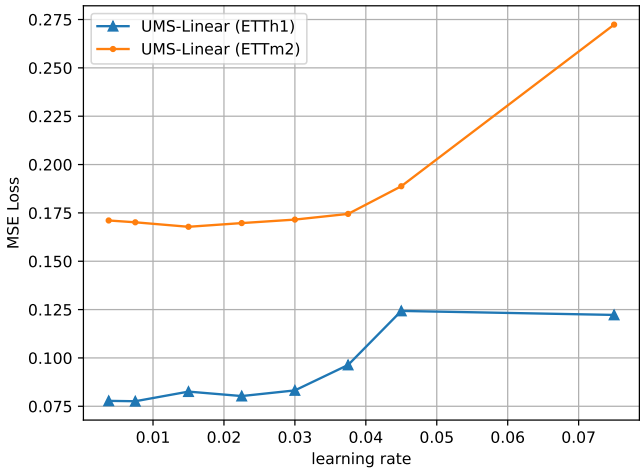


Figure 4.21: Illustration of the impact of using different learning rates in training *UMS-Linear* in ETTh1 and ETTm2 datasets at a forecast horizon of 720

It is important to note that the optimal learning rate may vary depending on the dataset and the model. Therefore, it is important to experiment with different learning rates to find the best settings for your specific application. However, our results suggest that a learning rate in the range of 0.015 is a good starting point for training linear models for time series forecasting. prevent the model from overfitting or underfitting the data.

Batch Size. We experimented with different batch sizes ranging from 8 to 512 to find the optimal batch size for our model, where in general the batch size is an important hyper-parameter that can affect the performance of any model. A too small batch size can lead to overfitting, while a too large batch size can lead to underfitting.

As can be seen from Figure 4.22, the MSE loss decreases as the batch size increases. However, the MSE loss starts to increase again when the batch size is too large. This is a sign that a too large batch size may cause the model to underfit the training data. In general,

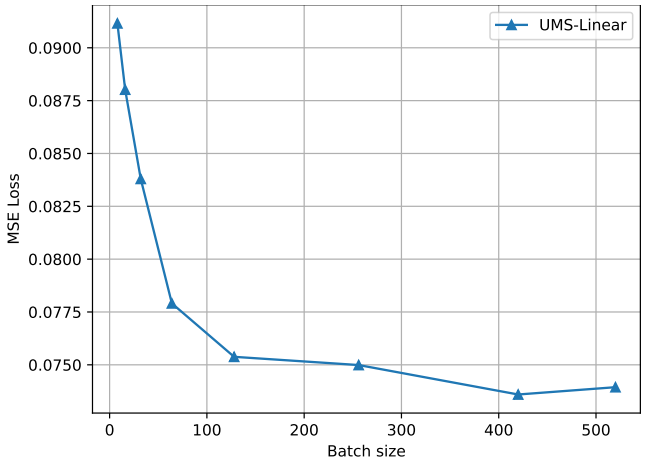


Figure 4.22: Illustration of the impact of using different batch sizes in training *UMS-Linear* in ETTh1 datasets at a forecast horizon of 720

we found that our model requires a batch size of 128 or higher, which gave the best results on the most datasets. Also, a larger batch size will lead to faster training times. However, a larger batch size can also lead to higher memory usage. Therefore, it is important to choose a batch size that is large enough to improve the performance of your model without causing any memory issues.

4.8.6 Contribution of Trend & Remainder to the error

The series decomposition technique using the moving average approach allows us to extract the trend and residual components from the time series. This could give us a view of the component that the model could learn and the one that it struggles with. Furthermore, to apply this idea, we trained the original DLinear model on the full univariate ETT dataset. The results of this experiment are shown in Table Table 4.8, where, as expected, the trend component is causing 80% of the full loss, while the remainder causes only 20% of the full loss. This is caused by the fact that the trend component contains an overview of the data movements, which includes the change points that most models struggle with, while the remainder contains only the fluctuations that the series have, which allows the model to easily predict them. These results could tell us the new direction that future research

should aim towards.

Table 4.8: Contribution of Trend & Remainder to the error in ETT dataset using the original DLinear [17].

		Loss Values				Contributing to the error		Average	Contributing to the error		Average
Component		Trend		Remainder		Trend			Remainder		
Metric		MSE	MAE	MSE	MAE	MSE	MAE		MSE	MAE	
ETTh1	96	0.0454	0.1582	0.0068	0.0637	86%	71%	76.25%	13%	28%	22.75%
	192	0.0644	0.1885	0.0071	0.0649	90%	74%		9%	25%	
	336	0.0803	0.2212	0.0068	0.0635	92%	77%		7%	22%	
	720	0.1574	0.3263	0.007	0.0642	95%	83%		4%	16%	
ETTh2	96	0.0918	0.2277	0.0319	0.1382	74%	62%	67%	25%	37%	32%
	192	0.1332	0.2815	0.0355	0.1472	78%	65%		21%	34%	
	336	0.1683	0.3282	0.0362	0.1487	82%	68%		17%	31%	
	720	0.2537	0.4113	0.0346	0.1464	87%	73%		12%	26%	
ETTm1	96	0.0249	0.1154	0.0019	0.0326	92%	77%	81.75	7%	22%	17.25%
	192	0.049	0.1605	0.0021	0.0345	95%	82%		4%	17%	
	336	0.0585	0.1764	0.002	0.033	96%	84%		3%	15%	
	720	0.0737	0.2014	0.0023	0.0357	96%	84%		3%	15%	
ETTm2	96	0.0562	0.1725	0.0017	0.0297	97%	85%	87.50%	2%	14%	11.50%
	192	0.083	0.2149	0.0019	0.0311	97%	87%		2%	12%	
	336	0.1114	0.2523	0.0021	0.0323	98%	88%		1%	11%	
	720	0.1603	0.3051	0.0023	0.0337	98%	90%		1%	9%	

4.8.7 The Effect of Decomposition in Learning the series components

The time series decomposition technique used in DLinear [17] allows the model to learn the series components separately. This results in more robust results, as the model is able to focus on learning each component individually. This can be simply shown by removing the decomposition from DLinear and changing the two layers input into the original $\mathcal{X}$ series. The results in Table 4.9 show that using two separated linear layers with the same input series is not as good as using decomposition. This means that if we could possibly apply another type of complex decomposition, we may reach more accurate results.

One possible explanation for this is that the decomposition technique used in DLinear is able to capture the different characteristics of the trend and remainder components. The trend component is typically smooth and slowly varying, while the remainder component is typically more volatile and unpredictable. By learning these components separately, the model is able to better capture the overall dynamics of the time series. Another possible explanation is that the decomposition technique helps to reduce the complexity of the input data. This can make it easier for the model to learn the underlying patterns and trends in

Table 4.9: A comparation between the original DLinear [17] and the Non decomposition version in univariate ETT dataset.

Methods		Dublicated				DLinear			
Component		Trend		Remainder		Trend		Remainder	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.0462	0.1631	0.0081	0.0697	**0.0454**	**0.1582**	**0.0068**	**0.0637**
	192	0.0675	0.1948	0.0128	0.089	**0.0644**	**0.1885**	**0.0071**	**0.0649**
	336	**0.0754**	**0.2162**	0.0077	0.0679	0.0803	0.2212	**0.0068**	**0.0635**
	720	**0.143**	**0.3076**	0.0083	0.0709	0.1574	0.3263	**0.007**	**0.0642**
ETTh2	96	0.0927	0.2294	0.0327	0.14	**0.0918**	**0.2277**	**0.0319**	**0.1382**
	192	0.1341	0.2828	0.0362	0.1486	**0.1332**	**0.2815**	**0.0355**	**0.1472**
	336	0.1694	0.3292	0.0369	0.1502	**0.1683**	**0.3282**	**0.0362**	**0.1487**
	720	0.2551	0.4128	0.0353	0.148	**0.2537**	**0.4113**	**0.0346**	**0.1464**
ETTm1	96	0.0256	0.117	0.0021	0.0343	**0.0249**	**0.1154**	**0.0019**	**0.0326**
	192	**0.0399**	**0.1454**	**0.0021**	0.0346	0.049	0.1605	**0.0021**	**0.0345**
	336	**0.0568**	**0.1735**	0.0022	0.0349	0.0585	0.1764	**0.002**	**0.033**
	720	0.0782	0.2066	**0.0022**	**0.035**	**0.0737**	**0.2014**	0.0023	0.0357
ETTm2	96	**0.0558**	0.1728	0.0018	0.031	0.0562	**0.1725**	**0.0017**	**0.0297**
	192	**0.0828**	**0.2148**	0.002	0.0321	0.083	0.2149	**0.0019**	**0.0311**
	336	0.1185	0.261	0.0024	0.0357	**0.1114**	**0.2523**	**0.0021**	**0.0323**
	720	0.1604	0.3055	0.0024	0.0347	**0.1603**	**0.3051**	**0.0023**	**0.0337**

the data. Overall, the results in Table Table 4.9 confirm that the decomposition technique used in DLinear is an important factor in its success. By learning the series components separately, the model is able to achieve more accurate and robust results. One possible direction for future work is to explore the use of different decomposition techniques. There are many different decomposition techniques available, each with its own strengths and weaknesses. It is possible that a different decomposition technique could lead to even better results.

4.8.8 Series Decomposition Kernels

In the DLinear model [17], the series decomposition kernel is used to extract the trend and remainder components of the time series data. The default kernel is a moving average kernel, which computes the average of the past k values in the time series. However, we found that changing the series decomposition kernel can lead to significant improvements in performance. This is because different pooling kernels are better suited for different

types of time series data. Figure 4.23 shows the different methods that we used in this experiment, where Max pooling is a type of pooling that takes the maximum value from a window of data. This makes it a good choice for time series data with multiple upper change points. This is because max pooling is able to capture the highest values in the time series, which are often associated with change points. Min pooling is a type of pooling that takes the minimum value from a window of data. This makes it a good choice for time series data with low change points. This is because min pooling is able to capture the lowest values in the time series, which are often associated with low change points. We also used a combined version that combines max pooling and average pooling. This kernel takes the maximum value from a window of data and then computes the average of the remaining values. This makes it a good choice for time series data with a mix of change points.

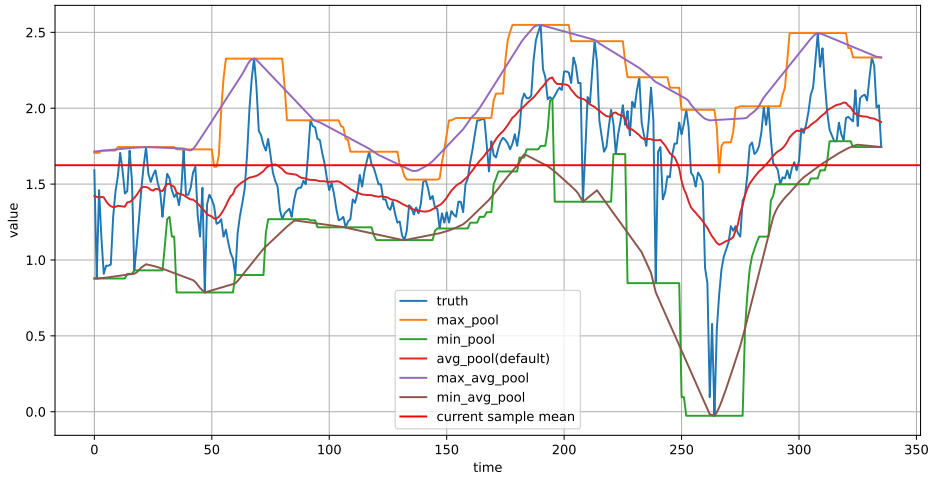


Figure 4.23: Series Decomposition using different kernel techniques in ETTh1 dataset.

In terms of results, Table 4.10 shows that max pooling generally outperforms the other pooling techniques. This is especially true for the ETTh1 and ETTm1 datasets, which have multiple upper change points. On the other hand, the min pooling kernels show good performance in the ETTh2 and ETTm2 datasets, which, in contrast, have multiple under change points. The choice of series decomposition kernel is an important factor in the

performance of the model. By choosing the right kernel, we can improve the model's ability to capture the important features of the time series data and make more accurate predictions.

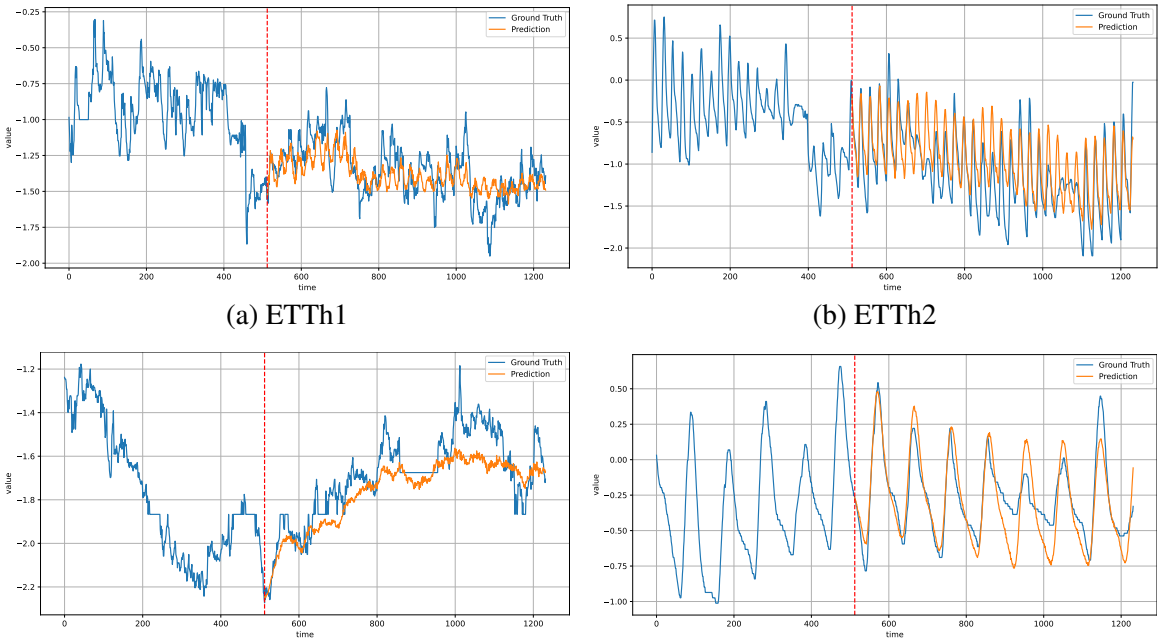
Table 4.10: Results of different pooling technique in the original DLinear [17]

Type		Separated						Combination			
Methods		Average Pooling		Max Pooling		Min Pooling		Max / Average		Min / Average	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.056	0.178	**0.051**	**0.174**	0.075	0.219	**0.051**	**0.174**	0.067	0.207
	192	0.076	0.208	**0.067**	**0.201**	0.104	0.261	**0.067**	**0.201**	0.109	0.266
	336	0.096	0.242	0.083	0.225	0.087	0.237	**0.081**	**0.223**	0.085	0.236
	720	0.185	0.356	0.156	0.318	0.099	0.251	0.154	0.316	**0.098**	**0.249**
ETTh2	96	0.13	0.279	0.123	0.276	**0.118**	**0.27**	0.123	0.276	0.119	0.271
	192	0.175	0.328	0.163	0.321	**0.162**	**0.32**	0.165	0.323	0.168	0.325
	336	0.21	**0.368**	0.209	0.37	0.21	0.371	**0.207**	**0.368**	0.211	0.372
	720	0.294	0.441	0.305	0.451	**0.236**	**0.393**	0.309	0.454	0.304	0.446
ETTm1	96	0.028	0.124	**0.025**	**0.12**	0.028	0.126	0.026	0.121	0.026	0.12
	192	0.044	0.156	0.041	0.152	**0.039**	0.149	**0.039**	0.15	0.039	**0.148**
	336	0.059	0.179	0.065	0.191	0.05	0.168	**0.048**	**0.167**	0.05	0.169
	720	0.081	0.212	**0.064**	**0.197**	0.07	0.207	0.066	0.199	0.071	0.207
ETTm2	96	**0.063**	0.183	**0.063**	0.182	0.064	0.184	**0.063**	**0.181**	**0.063**	**0.181**
	192	0.093	0.228	0.09	0.224	0.091	0.224	**0.089**	**0.22**	**0.089**	0.222
	336	0.119	0.261	**0.113**	**0.253**	0.115	0.256	0.115	0.256	0.115	0.256
	720	0.174	0.319	0.167	0.314	**0.164**	**0.313**	0.167	0.314	0.167	0.314

4.9 Visualization of Forecasting Results

In this section, we visualize the forecasting results of our proposed *UMS-Linear* model on the ETTh1, ETTh2, ETTm1, and ETTm2 datasets. We visualize the forecasting results with a forecasting horizon of 720. This visualization will allow us to gain a better understanding of the performance of the model and identify any potential problems. This will help us improve the model's accuracy and make it more useful for forecasting time series data.

From Figure 4.24, we can observe that the *UMS-Linear* model can capture the overall trend of the data well. However, there are still some errors in the details. For example, in ETTh1 and ETTm1 datasets, the model underestimates the values in some time steps, while in ETTh2 and ETTm2, the model overestimates the values in multiple time steps. This problem could be due to the nature of the data. In ETTh1 and ETTm1, we can see that



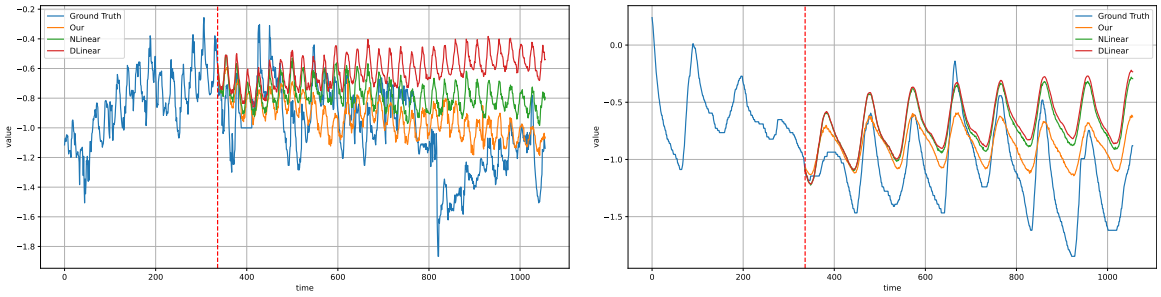
(a) ETTh1 (b) ETTh2

(c) ETTm1 (d) ETTm2

Figure 4.24: Visualisations of the forecasting results of *UMS-Linear* in the different ETT datasets in a forecasting horizon of 720

the data contains a lot of fluctuations that include some change points, which will make predicting the exact values harder. For ETTh2 and ETTm2, we can see the different nature of the data, which consists of repeated waves with different amplitudes. This will cause limitations for the model to find the exact amplitude of the wave at a certain time step. However, in all the datasets, we can see that the model can predict the exact small, wavy increases and decreases. Overall, the visualization of the forecasting results shows that the *UMS-Linear* model can capture the overall trend of the data well. However, there is still room for improvement in future work.

In addition, we can also compare our model to other linear models, where the shared linearity nature of our model and the LTSF-Linear models results in similar forecast shapes. However, our model manages to track better the trend of the prediction, which can be a great benefit in some real-life serious cases. Figure Figure 4.25 shows a comparison plot between ETTh1 and ETTm2, where *UMS-Linear* generates more accurate predictions.



(a) ETTh1 (b) ETTh1

Figure 4.25: A comparison between the forecast of *UMS-Linear*, NLinear and DLinear models in ETTh1 and ETTm2 for a forecast horizon of 720

4.10 Discussion on Interpretability and Explainability

One benefit of using linear models is their interpretability, which consists of dense layers that are mathematically noted as matrix multiplication between the input and the weights of a specific layer. This weight can be visualized to reveal characteristics and patterns from the data that the used model learns. In our case, *UMS-Linear* contains 3 different dense layers, which are the timestamp embedding layer, the decomposition phase layer, and the base wave layer. We visualize the weights in this layer for ETTh1 and ETTm1 datasets in different forecasting horizons from 96 to 720 using a look-back window of 96, We also follow [17] by initializing the weights with $1/L$ rather than random initialization to smooth weights and produce clear visualizations.

Figure 4.26 visualizes the weights of each linear layer in the UMS-Linear model trained on the ETTH1 dataset. The weights are visualized as a heatmap, where the x-axis represents the time steps in the input sequence and the y-axis represents the features in the input data. The color of each cell in the heatmap represents the magnitude of the weight, with darker colors indicating larger weights. From this figure, we can observe that the model learns to focus on specific time steps and features in the input sequence. For example, in the base wave phase layer, the model assigns larger weights to the time steps in the pattern $0, 23, 47, 71, 95...719$. Where the model has learned that these time steps are important for predicting future values, the ETTh1 dataset is an hourly dataset, which means that 24 time

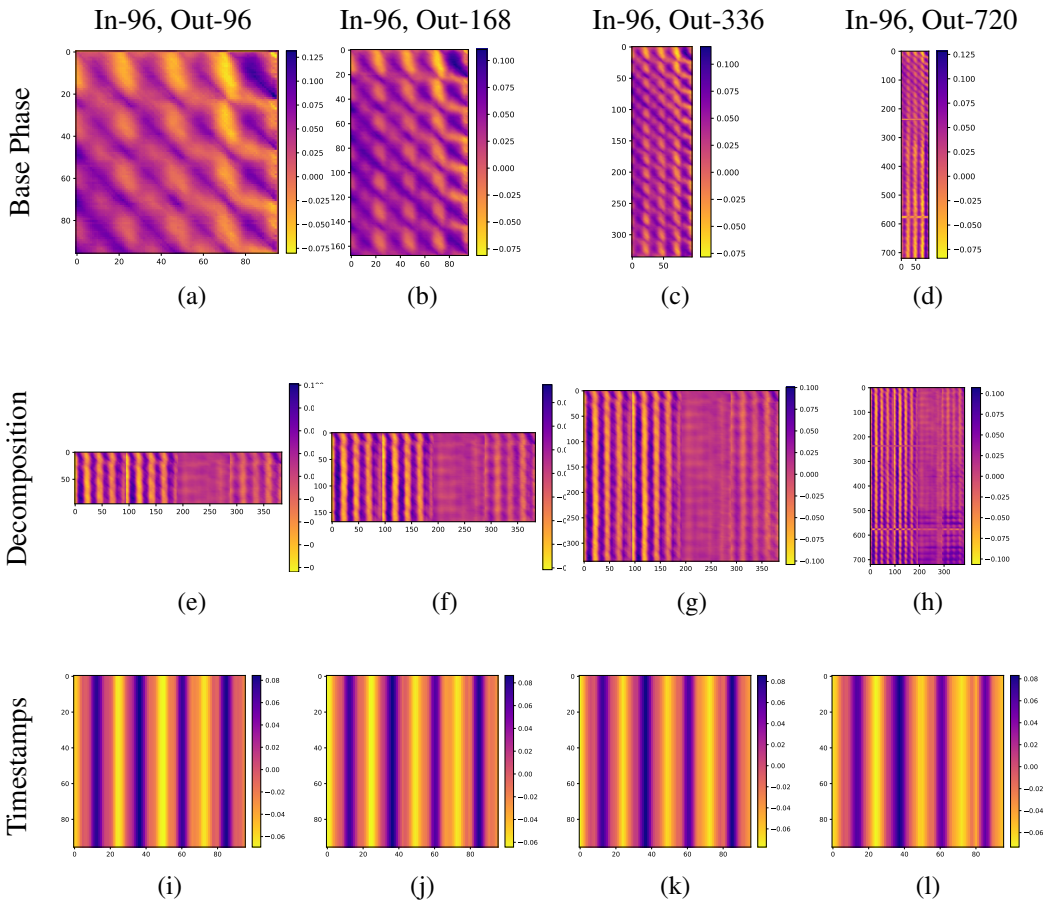


Figure 4.26: Visualization of the weights(T*L) of each linear layer in *UMS-Linear* on ETTh1 dataset. Models are trained with a look-back window L (X-axis) and different forecasting time steps T (Y-axis).

steps represent days. This indicates that ETTh1 has a daily periodicity. Also, there are some forecasts around 225 and around 550; these two require negative inputs. In terms of time steps, 225 represents nearly 10 days and 550 nearly represents 24 days; these two do not have a specific meaning, but they might refer to holidays. The weights of the timestamp embedding layer reveal the pattern that the model used to indicate the current forecasting periods in time.

We also visualized the weights in the ETTm1 dataset, as shown in Figure 4.27. We can observe that the base phase layer of the model learns to focus on specific periodic time steps. This high weight is repeated in a pattern of $0, 100, 200, .., 700$; each 100 time step refers to 1 hour and 40 minutes. This indicates that there is periodicity in the oil temperature

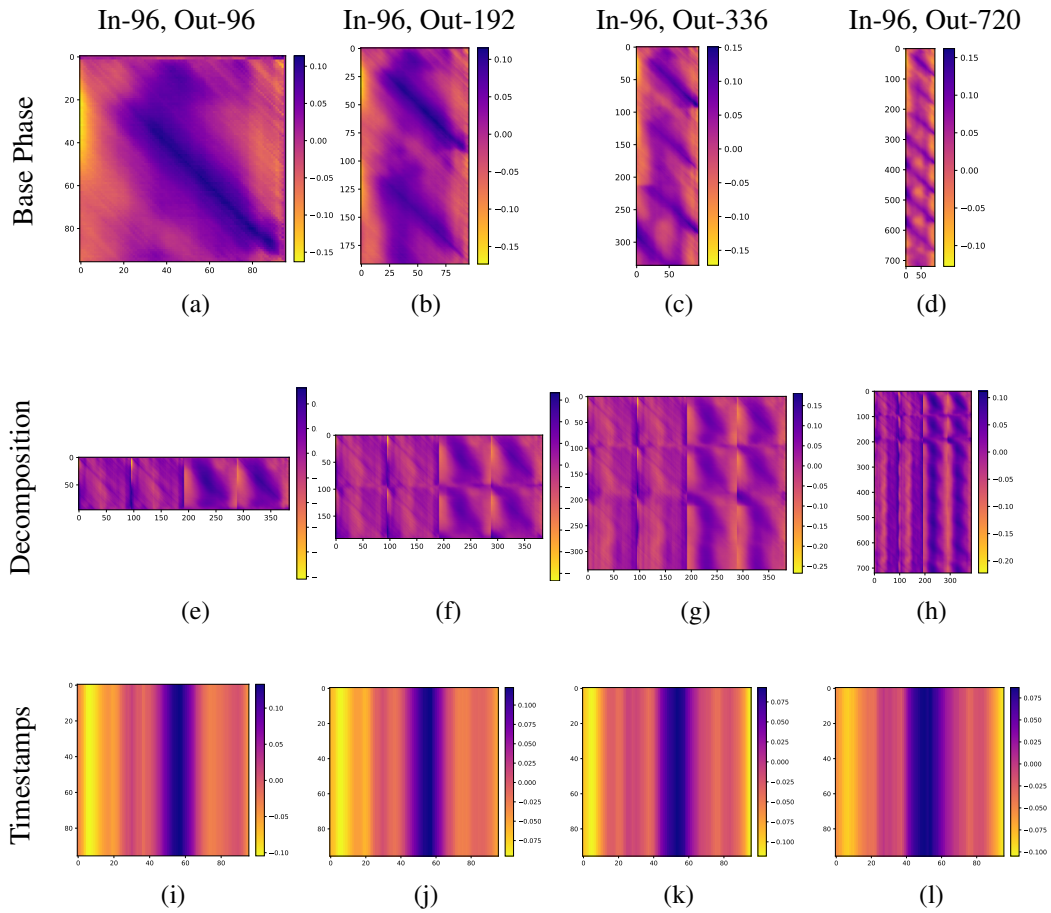


Figure 4.27: Visualization of the weights(T*L) of each linear layer in *UMS-Linear* on ETTm1 dataset. Models are trained with a look-back window L (X-axis) and different forecasting time steps T (Y-axis).

in ETTm1. We also observe that the model used a new pattern of timestamps to indicate the current forecasting time period in the timestamp embedding layer. For the decomposition phase layer, we can't observe a clear pattern because of the combination between different components. However, the 100 time step periodicity is still visible in this layer.

4.11 Conclusion

To conclude this chapter, we presented different experiments where this experimental evaluation on a wide range of datasets demonstrated the effectiveness of *UMS-Linear*, outperforming strong baselines across various forecasting horizons. Moreover, *UMS-Linear*

exhibits robustness and sensitivity to different hyperparameters, making it a practical and reliable choice for time-series forecasting tasks. The interpretability and explainability of *UMS-Linear* provide valuable insights into the underlying time series dynamics. enhancing its usefulness in real-world applications. Overall, *UMS-Linear* represents a significant contribution to the field of time series forecasting, offering a powerful and effective solution for both short-term and long-term forecasting tasks.

CHAPTER 5
CONCLUSION

The appearance of time-related data allows for the introduction of the time series forecasting task, where learning the related patterns within this data and predicting new future data points is the main objective of this task. However, real-world data and applications include complex patterns and trends that create limitations in learning these kinds of patterns and data components. As mentioned in the literature review, several methods and techniques have been presented over the years to solve this problem; each of them consists of a different type of artificial intelligence method (e.g., MLP-based, CNNs, RNNs).

In this thesis, we have deeply studied the MLP-based models that are presented in the time series forecasting task. This simple model outperforms the existing transformers and other types of methods for this task, which makes us focus our attention on studying and analyzing these models. Studying these models reveals that the decomposition and the training loss function are two important things to enhance the performance of the MLP-based models.

Because of this, we have proposed a novel approach for time series forecasting tasks called *UMS-Linear*. We have used several techniques and methods to enhance the simple LTSF-linear models. This proposed model inherits the same linear nature as the LTSF-Linear models while combining two new techniques called multi-scale decomposition and timestamp embedding. Moreover, the proposed approach presents multiple benefits, where *UMS-Linear* is an interpretable, non-complex, and fast-to-use model. Also, we have introduced a new training loss function that combines the mean squared error (MSE), the mean absolute error (MAE), and the MAE of the differences between the prediction and the truth values. Using this loss function enhances the balancing and robustness of linear models in time series forecasting.

To prove the efficiency of the proposed approach, we have conducted multiple real-world experiments for time series forecasting. We also conducted experiments on the different components of the model and the effects and impacts of using them separately in linear models. From these experiments, we can make the following observations:

- *UMS-Linear* outperforms existing linear models in univariate time series forecasting in all metrics.
- Our proposed model inherits and improves the performance of linear models in learning various types of data.
- *UMS-Linear* is able to learn the different trends and seasonality patterns in clear datasets.
- Our proposed model shows remarkable performance in dealing with real-world applications that vary from weather and exchange rates to energy demand generation and others.
- Each technique used in *UMS-Linear* is a useful component in enhancing the accuracy of the forecasting results in linear models.

The advancements made in this thesis contribute to the field of time series forecasting and provide a solid foundation for further research and practical applications. This thesis opens up several avenues for future research. Some promising directions include exploration of different decomposition techniques, extension to multivariate time series, and improving forecasting accuracy in small datasets.

REFERENCES

[1] F. Dama and C. Sinoquet, *Time series analysis and modeling to forecast: A survey*, 2021. arXiv: 2104.00164 [cs.LG].

[2] G. J. J. van den Burg and C. K. I. Williams, *An evaluation of change point detection algorithms*, 2022. arXiv: 2003.06222 [stat.ML].

[3] A. Vaswani *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL].

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL].

[5] E. M. B. Nagoudi, A. Elmadany, and M. Abdul-Mageed, *Arat5: Text-to-text transformers for arabic language generation*, 2022. arXiv: 2109.12068 [cs.CL].

[6] U. Khandelwal, K. Clark, D. Jurafsky, and L. Kaiser, *Sample efficient text summarization using a single pre-trained transformer*, 2019. arXiv: 1905.08836 [cs.CL].

[7] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5884–5888.

[8] Z. Liu *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, 2021. arXiv: 2103.14030 [cs.CV].

[9] A. Zeng *et al.*, "Deciwatch: A simple baseline for 10x efficient 2d and 3d pose estimation," in *European Conference on Computer Vision*, Springer, 2022.

[10] N. Parmar *et al.*, "Image transformer," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Jul. 2018, pp. 4055–4064.

[11] M. Chen *et al.*, "Generative pretraining from pixels," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jul. 2020, pp. 1691–1703.

[12] A. Muhamed *et al.*, "Symbolic music generation with transformer-gans," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, pp. 408–417, May 2021.

[13] Q. Wen *et al.*, *Transformers in time series: A survey*, 2023. arXiv: 2202.07125 [cs.LG].

[14] H. Zhou *et al.*, *Informer: Beyond efficient transformer for long sequence time-series forecasting*, 2021. arXiv: 2012.07436 [cs.LG].

[15] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting*, 2022. arXiv: 2106.13008 [cs.LG].

[16] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, *Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting*, 2022. arXiv: 2201.12740 [cs.LG].

[17] A. Zeng, M. Chen, L. Zhang, and Q. Xu, *Are transformers effective for time series forecasting?* 2022. arXiv: 2205.13504 [cs.AI].

[18] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, 2014, pp. 106–112.

[19] A. E. Permanasari, I. Hidayah, and I. A. 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)*, 2013, pp. 203–207.

[20] R. M. Schmidt, "Recurrent neural networks (rnns): A gentle introduction and overview," *CoRR*, vol. abs/1912.05911, 2019. arXiv: 1912.05911.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997.

[22] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, and H. Zhang, *Segrnn: Segment recurrent neural network for long-term time series forecasting*, 2023. arXiv: 2308.11200 [cs.LG].

[23] S. Bai, J. Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, 2018. arXiv: 1803.01271 [cs.LG].

[24] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, and Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[25] M. Liu *et al.*, *Scinet: Time series modeling and forecasting with sample convolution and interaction*, 2022. arXiv: 2106.09305 [cs.LG].

[26] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, *Timesnet: Temporal 2d-variation modeling for general time series analysis*, 2023. arXiv: 2210.02186 [cs.LG].

[27] Z. Gong, Y. Tang, and J. Liang, *Patchmixer: A patch-mixing architecture for long-term time series forecasting*, 2023. arXiv: 2310.00655 [cs.LG].

[28] Y. Liu and M. Lapata, *Text summarization with pretrained encoders*, 2019. arXiv: 1908.08345 [cs.CL].

[29] J. Howard and S. Ruder, *Universal language model fine-tuning for text classification*, 2018. arXiv: 1801.06146 [cs.CL].

[30] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, *Behavior sequence transformer for e-commerce recommendation in alibaba*, 2019. arXiv: 1905.06874 [cs.IR].

[31] S. Liu *et al.*, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *International Conference on Learning Representations*, 2022.

[32] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, *A time series is worth 64 words: Long-term forecasting with transformers*, 2023. arXiv: 2211.14730 [cs.LG].

[33] W. Xue, T. Zhou, Q. Wen, J. Gao, B. Ding, and R. Jin, *Card: Channel aligned robust blend transformer for time series forecasting*, 2024. arXiv: 2305.12095 [cs.LG].

[34] Y. Zhang and J. Yan, "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting," in *The Eleventh International Conference on Learning Representations*, 2023.

[35] Z. Li, Z. Rao, L. Pan, and Z. Xu, *Mts-mixers: Multivariate time series forecasting via factorized temporal and channel mixing*, 2023. arXiv: 2302.04501 [cs.LG].

[36] S.-A. Chen, C.-L. Li, N. Yoder, S. O. Arik, and T. Pfister, *Tsmixer: An all-mlp architecture for time series forecasting*, 2023. arXiv: 2303.06053 [cs.LG].

[37] A. Das, W. Kong, A. Leach, S. Mathur, R. Sen, and R. Yu, *Long-term forecasting with tide: Time-series dense encoder*, 2023. arXiv: 2304.08424 [stat.ML].

[38] R. Ni, Z. Lin, S. Wang, and G. Fanti, *Mixture-of-linear-experts for long-term time series forecasting*, 2024. arXiv: 2312.06786 [cs.LG].

[39] Q. Ma *et al.*, *A survey on time-series pre-trained models*, 2023. arXiv: 2305.10716 [cs.LG].

[40] A. Koochali, P. Schichtel, A. Dengel, and S. Ahmed, "Probabilistic forecasting of sensory data with generative adversarial networks – forgan," *IEEE Access*, vol. 7, pp. 63 868–63 880, 2019.

[41] M. Jin *et al.*, *Time-llm: Time series forecasting by reprogramming large language models*, 2024. arXiv: 2310.01728 [cs.LG].

[42] T. Zhou, P. Niu, X. Wang, L. Sun, and R. Jin, *One fits all:power general time series analysis by pretrained lm*, 2023. arXiv: 2302.11939 [cs.LG].

[43] G. F. Watch. "{Global forest watch} fires in algeria." (2024), (visited on 03/26/2024).

[44] D. Rolnick *et al.*, *Tackling climate change with machine learning*, 2019. arXiv: 1906.05433 [cs.CY].