Algerian Democratic and Popular Republic
Ministry of Higher Education and Scientific Research

KASDI MERBAH UNIVERSITY OUARGLA
Faculty of New Information and Communication Technologies
Department of Computer Science and Information Technology

Thesis Submitted in Candidacy for a Master Degree in computer science, option

Artificial Intelligence and Data Science

by Baba Hamou Mouhamed Hamou & Soualah Mouhamed Islam

**Theme**

# Deep Learning for Improved Multi-variate Time Series Forecasting

Jury Members:

| | | | |
|---|---|---|---|
| SUPERVISOR | Dr.Said Bachir | Professor | University of Ouargla |
| CO-SUPERVISOR | Dr.Aiadi Ossama | Professor | University of Ouargla |
| JURY CHAIR | Dr. Youcefa .M | Professor | University of Ouargla |
| EXAMINER | Dr.Khaldi Bilal | Professor | University of Ouargla |

Academic year: 2023/2024

# Acknowledgements

Above all, praise God, and thanks to God Almighty who has enabled us to complete this message.

I would like to express my sincere gratitude to the esteemed faculty, mentors, and staff of Kasdi Merbah Ouargla University for their unwavering support and guidance throughout my academic journey. Your dedication to excellence has played a pivotal role in shaping my intellectual growth and professional development.

I am deeply indebted to my thesis supervisor Dr.Said Bachir and my assistant supervisor Dr.Aiadi Oussama for their valuable insights, encouragement, and constructive comments, which were instrumental in completing my research project. Their experience and guidance have inspired me to push the boundaries of knowledge and pursue academic excellence.

I sincerely thank my colleagues and fellow students for their friendship, cooperation, and companionship. Your diverse perspectives and collective wisdom have enriched my educational experience and fostered a stimulating academic environment.

I am also grateful to the administrative staff and university officials for their administrative support and logistical assistance, which facilitated the smooth progress of my studies.

Last but not least, I would like to acknowledge my family and friends' unwavering support and encouragement. Their love, encouragement, and understanding have been a constant source of strength and motivation throughout my academic endeavors.

As I begin the next chapter of my journey, I carry with me priceless lessons, experiences, and memories from my time at University. I am confident that the skills and knowledge I have acquired during my studies will serve as a solid foundation for future endeavors, and I remain committed to upholding the values of academic excellence and lifelong learning that this esteemed institution has instilled in me.

Thank you again to everyone who played a role in my academic success. Your support and guidance are truly appreciated.

# Abstract

Time series forecasting is crucial across various domains, particularly when dealing with multivariate datasets characterized by complex dynamics and intricate interdependencies among multiple variables. Traditional approaches often employ high-capacity architectures like recurrent neural networks (RNNs) and attention-based models to handle such complexities. However, recent studies have revealed that simpler, univariate linear models can sometimes outperform these sophisticated methods on several benchmarks. This research conducts a comprehensive performance analysis of 10 forecasting models, including transformers, linear models, MLP mixing models, and other deep learning models specialized in time series forecasting. The goal is to identify the most suitable model for different datasets by examining their performance under various conditions and complexities. This analysis provides valuable insights for practitioners, helping them make informed decisions about model selection to achieve optimal forecasting accuracy and efficiency for specific applications. The primary objective is to improve a model TSMixer: An All-MLP Architecture for Time Series Forecasting, that leverages advanced neural network architectures to improve predictive performance. We implemented 5 different versions of modifications to our model including the CNN. This research provides crucial insights into optimizing deep learning models for time series forecasting. The enhanced TSMixer model significantly advances multivariate time series forecasting, establishing a solid foundation for future innovations.

***Key words:*** *Multivariate time series, Forecasting, Deep Learning, Convolutional Neural Networks, Time series mixer, Linear Model, Mixer layer, Multi-layer perceptron layer.*

## ملخص

يعد التنبؤ بالسلاسل الزمنية أمرًا بالغ الأهمية عبر مختلف المجالات، خاصة عند التعامل مع مجموعات البيانات متعددة المتغيرات التي تتميز بديناميكيات معقدة وترابطات معقدة بين متغيرات متعددة. غالبًا ما تستخدم الأساليب التقليدية بنيات عالية السعة مثل الشبكات العصبية المتكررة والنماذج القائمة على الاهتمام للتعامل مع مثل هذه التعقيدات. ومع ذلك، فقد كشفت الدراسات الحديثة أن النماذج الخطية البسيطة وأحادية المتغير يمكن أن تتفوق أحيانًا على هذه الأساليب المعقدة في العديد من المعايير. يقوم هذا البحث بإجراء تحليل أداء شامل لعشرة نماذج للتنبؤ، بما في ذلك المحولات والنماذج الخطية ونماذج خلط طبقة الإدراك الحسي متعدد الطبقات ونماذج التعلم العميق الأخرى المتخصصة في التنبؤ بالسلاسل الزمنية. الهدف هو تحديد النموذج الأنسب لمجموعات البيانات المختلفة من خلال فحص أدائها في ظل ظروف وتعقيدات مختلفة. يوفر هذا التحليل رؤى قيمة للممارسين، مما يساعدهم على اتخاذ قرارات مستنيرة بشأن اختيار النموذج لتحقيق دقة وكفاءة التنبؤ الأمثل لتطبيقات محددة، و الهدف الأساسي هو تحسين نموذج خلاط السلاسل الزمنية: جميع طبقات الإدراك الحسي متعددة الطبقات للتنبؤ بالسلاسل الزمنية، التي تستفيد من بنيات الشبكات العصبية المتقدمة لتحسين الأداء التنبؤي. قمنا بتنفيذ ٥ إصدارات مختلفة من التعديلات على نموذجنا بما في ذلك الشبكات العصبية التلافيفية. يوفر هذا البحث رؤى مهمة لتحسين نماذج التعلم العميق للتنبؤ بالسلاسل الزمنية. يعمل نموذج خلاط السلاسل الزمنية المحسن على تطوير التنبؤ بالسلاسل الزمنية متعددة المتغيرات بشكل كبير، مما ينشئ أساسًا متينًا للابتكارات المستقبلية.

**الكلمات المفتاحية:** السلاسل الزمنية متعددة المتغيرات، التنبؤ، التعلم العميق، الشبكات العصبية التلافيفية، التنبؤ بالسلاسل الزمنية، النموذج الخطي، طبقة الخلاط، طبقة الإدراك الحسي متعدد الطبقات.

# Résumé

La prévision de séries chronologiques est cruciale dans divers domaines, en particulier lorsqu'il s'agit d'ensembles de données multivariés caractérisés par une dynamique complexe et des interdépendances complexes entre plusieurs variables. Les approches traditionnelles utilisent souvent des architectures à haute capacité telles que les réseaux neuronaux récurrents (RNN) et des modèles basés sur l'attention pour gérer de telles complexités. Cependant, des études récentes ont révélé que des modèles linéaires univariés plus simples peuvent parfois surpasser ces méthodes sophistiquées sur plusieurs points de référence. Cette recherche effectue une analyse complète des performances de 10 modèles de prévision, notamment des transformateurs, des modèles linéaires, des modèles de mélange MLP et d'autres modèles d'apprentissage en profondeur spécialisés dans la prévision de séries chronologiques. L'objectif est d'identifier le modèle le plus approprié pour différents ensembles de données en examinant leurs performances dans diverses conditions et complexités. Cette analyse fournit des informations précieuses aux praticiens, les aidant à prendre des décisions éclairées concernant la sélection de modèles afin d'obtenir une précision et une efficacité de prévision optimales pour des applications spécifiques. L'objectif principal est d'améliorer un modèle TSMixer : une architecture entièrement MLP pour la prévision de séries chronologiques, qui exploite des architectures de réseaux neuronaux avancées pour améliorer les performances prédictives. Nous avons implémenté 5 versions différentes de modifications de notre modèle, y compris le CNN. Cette recherche fournit des informations cruciales sur l'optimisation des modèles d'apprentissage profond pour la prévision de séries chronologiques. Le modèle TSMixer amélioré fait progresser considérablement la prévision de séries chronologiques multivariées, établissant ainsi une base solide pour les innovations futures.

***Mots clés:*** *Séries chronologiques multivariées, Prévisions, Deep Learning, Réseaux de neurones convolutifs, Mélangeur de séries chronologiques, Modèle linéaire, Couche de mixage, Couche de perceptron multicouche.*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **TSmixer**: | Time series mixer |
| **CNN-TSmixer**: | CNN Feature Extraction Before Mixer Layer |
| **DL**: | Deep Learning |

# Chapter 1

# General Introduction

## 1.1 Introduction

Artificial Intelligence (AI) has emerged as a transformative force across various domains, including healthcare, finance, transportation, and especially the energy sector. AI's sophisticated algorithms enable the analysis of vast datasets, identifying intricate patterns that traditional methods often overlook. This capability is particularly beneficial in demand forecasting, where AI can enhance accuracy and efficiency, thus supporting better resource management and operational decision-making.

Deep learning, a key component of AI, has demonstrated impressive performance in numerous tasks, including time series forecasting. Models such as recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks excel at capturing temporal dependencies, making them invaluable for applications like stock market prediction, weather forecasting, and electricity demand estimation. These models' ability to automatically learn features from raw data and capture nonlinear relationships sets them apart from conventional statistical methods.

The importance of AI-based forecasting systems cannot be overstated, given their potential to revolutionize various sectors by improving predictive accuracy and operational efficiency. Like electricity forecasting, AI models facilitate the integration of external variables like weather patterns and economic data, offering a comprehensive view of demand drivers. This integration enables utilities to optimize grid operations dynamically, enhancing stability, reducing costs, and benefiting consumers.

## 1.2 Problematic

Forecasting future demand is crucial for businesses and organizations to anticipate needs and ensure a reliable supply of resources. This task becomes more challenging when dealing with multiple locations or units, each exhibiting unique consumption patterns. Traditional time series analysis and regression techniques often fall short in these scenarios.

Is there any possible way to improve the results of the TSmixer prediction model, so that it can achieve better performance?

## 1.3 Overview on the related techniques

Recent advancements in forecasting have been driven by innovations in machine learning and deep learning, leading to more accurate and versatile models. Traditional models like ARIMA, designed for univariate time series, struggle with complex, multi-dimensional data. In contrast, advanced models such as TSMixer and PatchMixer leverage deep learning to uncover and utilize complex temporal patterns.

Noteworthy works in electricity forecasting have employed sophisticated models tailored to address the unique challenges of predicting in various industries. These models demonstrate significant improvements in capturing intricate dependencies and integrating external factors, leading to more accurate and reliable forecasts.

## 1.4 Work motivation

1. **Limitations of Previous Solutions:** Existing models for electricity demand forecasting exhibit several limitations. Traditional models often fail to handle complex datasets with additional features such as social events and weather data. Even advanced deep learning models, like RNNs and LSTMs, may struggle with issues like overfitting and capturing long-term dependencies.

2. **Motivation for New Approaches:** These limitations highlight the need for further research to develop more effective forecasting models. The goal is to create models that can seamlessly integrate complex, multi-dimensional data and improve resilience to changing conditions. Overcoming these challenges is crucial for advancing the field of electricity forecasting and enhancing the accuracy and efficiency of predictive models.

## 1.5 Work contributions

This research presents significant advancements in the field of deep learning and AI-based forecasting. The key contributions include:

1. Performance Analysis of the Models by Dataset: The model's performance was evaluated across a range of prediction horizons and datasets.

2. Enhancement performance of tsmixer by :

    - Integrating Convolutional Neural Networks (CNNs) to enhance feature extraction.

These contributions significantly improve the accuracy and robustness of the forecasting model Tsmixer, offering valuable insights and potential applications across various domains.

## 1.6 Experimental results

The proposed models were evaluated across multiple datasets and prediction horizons, demonstrating improved accuracy and efficiency. Our various experiments have continually enhanced the model's performance, especially in capturing complex temporal interactions. These results confirm the effectiveness of the proposed improvements in forecasting development in various industries and highlight the potential for broader applications.

## 1.7 Thesis Organization

This thesis is organized as follows:

- Chapter 1 Introduction This chapter provides an introduction to the thesis, outlining the motivation behind the study, the objectives, and the contributions of the research.

- Chapter 2 Understanding Time Series: From Data to Forecasts lays the foundation for the thesis by providing a comprehensive overview of time-series data and forecasting methods.

- Chapter 3 Related Work and Performance Analysis, This chapter provides an overview of time series data, discusses related work in forecasting, and offers an analysis of forecasting models across different datasets.

- Chapter 4 Proposed solutions, This chapter presents the core of the thesis, introducing several innovative approaches to enhance time-series forecasting models. Each proposed solution is systematically examined to highlight its potential benefits and contributions to the field

- Chapter 5 Experiments and Results, This chapter presents experimental results evaluates the performance of the proposed models, and provides a discussion of each experiment across the dataset

# Part I

# Comprehensive Study

# Chapter 2

# Understanding Time Series: From Data to Forecasts

## 2.1 Introduction

Time series data, like stock prices or weather patterns and electricity consumption, is key to predicting future trends. This chapter unlocks this potential by exploring time series data, forecasting techniques, and their applications.

## 2.2 Time Series

A time series is like a collection of snapshots taken at specific moments. Each snapshot captures what's happening at that particular time. There are two main types of time series:

- **Discrete-time**: Imagine taking snapshots at regular intervals, like every hour or every day. This is a discrete-time series. Examples include daily stock prices or hourly temperature readings.
- **Continuous-time**: Imagine recording a video instead of taking snapshots. This is like a continuous-time series, where the observations are captured constantly over some time. An example might be sensor data monitoring a machine's temperature throughout the day, e.g., when $T_0 = [0, 1]$[3].

Illustrate the concept with real-world examples from various domains:

1. **Finance:**Stock prices, interest rates, foreign exchange rates.
2. **Weather:**Temperature, humidity, precipitation levels (recorded at regular intervals).

3. **Sales:**Daily, weekly, or monthly sales figures for a company. **For, e.g. :** Sales of a mature

   pharmaceutical products generally remain stable without changes in marketing or manufacturing. Weekly generic drug sales are consistently around 10.4 million units, showing random variation without clear patterns. Chemical production is monitored for variables like temperature and purity to meet specifications. Due to the continuous process, output properties often show positive autocorrelation, with values above or below the average tending to be followed by similar values.

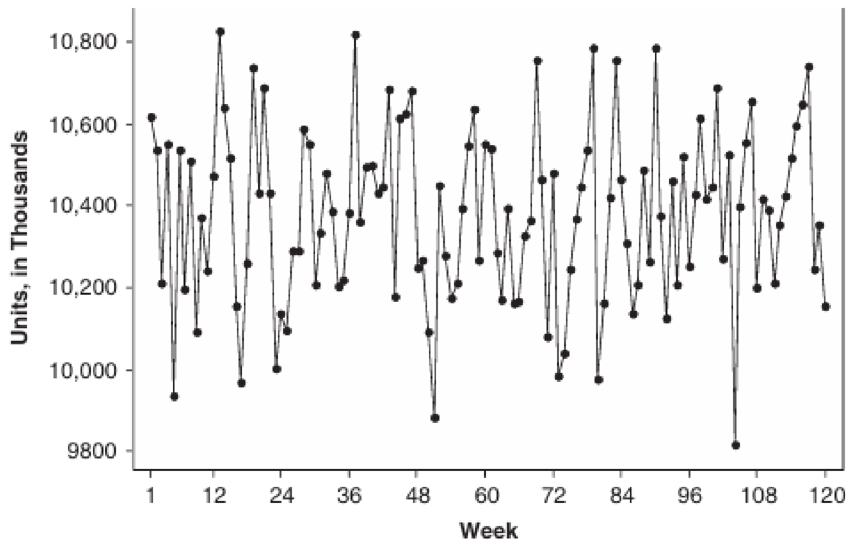

Figure 2.1: Pharmaceutical product sales.[1]

4. **Social Media:** Number of likes, shares, or posts over time.

5. **Healthcare:**Patient vital signs recorded at regular intervals.

### 2.2.1 The key characteristics of time series data

- **Regularity:** Data points are collected at consistent intervals (Hourly, Daily, etc..), like hourly stock prices or daily website traffic. This consistent timing allows for comparisons and trend identification.

- **Order:** The order of data points is crucial, as it captures temporal relationships. For instance, temperature readings throughout the day tell a story of how the temperature changes over time, not just its values.

- **Seasonality:** Patterns that repeat over specific time intervals.[3] For instance, daily temperature fluctuations, weekly sales cycles, or annual electricity demand variations are all examples of seasonality.
- **Trends:** Long-term upward or downward movements in the data[3]. Analyzing stock prices might reveal an overall upward trend over the years, despite short-term fluctuations.
- **Stationarity:** Statistical properties (mean, variance) remain constant over time (not always the case)[3]. For example, population growth data would likely show a non-stationary upward trend.
- **Cycle:** These are recurring patterns that may not have a fixed frequency. Economic boom-and-bust cycles or sunspot activity cycles illustrate this concept. Cycles can be more challenging to predict than regular seasonal patterns[3].
- **Irregularity/Noise:** Unpredictable, random fluctuations can occur within a time series. Measurement errors, external events, or inherent randomness in the data could cause these. Accounting for or filtering out noise can improve the accuracy of analysis.

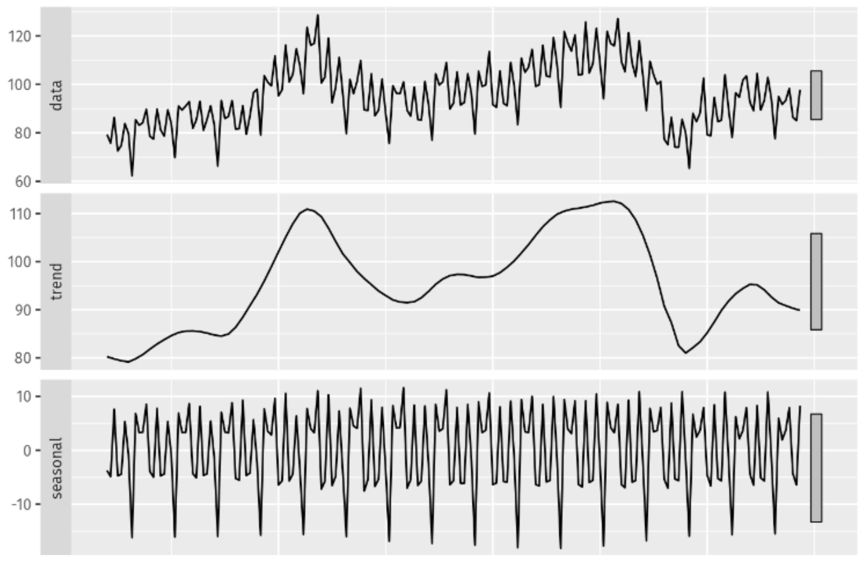

Figure 2.2: Some key characteristics of time series data

### 2.2.2 Objectives of Time Series Analysis

Time series analysis aims to draw inferences from data encountered in various fields like engineering, science, sociology, and economics. The primary steps and objectives include:[3]

1. **Model Setup:** Establishing a hypothetical probability model to represent the data.
2. **Model Selection:** Choosing an appropriate family of models.
3. **Parameter Estimation:** Estimating the parameters of the model.
4. **Goodness of Fit:** Checking the model's fit to the data.
5. **Understanding Mechanisms:** Using the fitted model to understand the data-generating process.
6. **Model Applications:** Utilizing the model for different purposes such as:
   - **Description:** Provide a compact data representation, including trends and seasonal components.
   - **Seasonal Adjustment:** Removing seasonal components to clarify long-term trends.
   - **Noise Filtering:** Separating noise from signals.
   - **Prediction:** Forecasting future values of a series.
   - **Hypothesis Testing:** Testing hypotheses, e.g., global warming using temperature data.
   - **Cross-Prediction:** Predicting one series from another, such as sales from advertising data.
   - **Control:** Adjusting parameters to control future values.
   - **Simulation Studies:** Simulating scenarios, such as estimating the probability of a reservoir running out of water based on modeled inputs.

### 2.2.3 Challenges in Time Series Analysis

Real-world time series data is messy and presents a unique set of challenges for forecasting models[4]:

1. **Generalizability:** Ensuring that the prediction model generalizes well when using synthetic data generated to preserve temporal dynamics.
2. **Data Efficiency:** Develop a new and data-efficient descriptive algorithm to augment data that helps in MTS (Multivariate Time Series) forecasting.
3. **Defects in Generative parameters:** Identifying and addressing defects associated with increased MTS in generative settings.

4. **Drawbacks of MTS Augmentation:** Identifying and addressing the drawbacks of time series augmentation in generative settings is essential. This involves understanding potential limitations or biases the augmentation process introduces and mitigating them effectively.

5. **Qualitative and Quantitative Analysis of Synthetic Data:** Conducting thorough qualitative and quantitative analysis of synthetic MTS data is necessary to evaluate its suitability for long-term forecasting. This involves assessing the augmented data's accuracy, reliability, and usefulness.

## 2.3 Forecasting

A forecast predicts future events, as Niels Bohr once said. Predictions, and educated guesses about what will happen, are important in fields as diverse as business, economics, and even medicine. These forecasts come in different forms: short-term (thinking about day-to-day operations), medium-term (setting next year's budget), and long-term (overall strategic planning). To make these predictions, especially over shorter periods, analysts rely heavily on historical data (time series) and statistical methods to identify patterns and trends. The trend may be correct in the future. This data can be from daily stock prices to temperature indices and is essentially a series of observations over time of a particular variable.[1]

### 2.3.1 The Forecasting Process:

Forecasting can be visualized as a step-by-step journey, like a recipe. Here's how it works:

1. **Define the problem:** What exactly are you trying to predict?

2. **Data collection:** Collect relevant information to fuel your forecast.

3. **Analyze the data:** Uncover patterns and trends hidden within the information.

4. **Choose the right recipe (model):** Select a forecasting method that best suits your data and problem.

5. **Test the recipe (model validation):** See how well your chosen method performs on past data.

6. **Put the recipe to use (deployment):** Implement the chosen method to make actual forecasts.

7. **Monitoring forecasting model performance:** Track your forecasts' accuracy and adjust the recipe if needed.

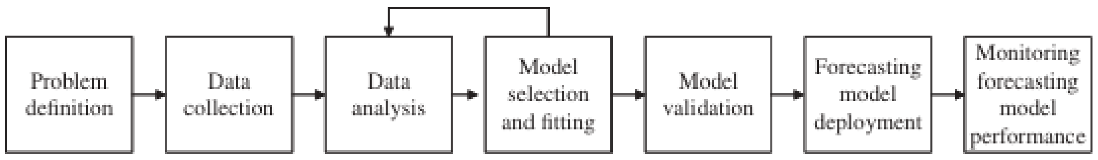
Problem definition
Data collection
Data analysis
Model selection and fitting
Model validation
Forecasting model deployment
Monitoring forecasting model performance

Figure 2.3: The forecasting process.[1]

### 2.3.2 Applications of Forecasting

Forecasting is applied in various fields to make informed decisions and strategic plans[1]:

1. **Operations Management:** Businesses use forecasts to schedule production, manage inventories, optimize the supply chain, plan capacity, and determine staffing needs.
2. **Marketing:** Forecasting helps assess the effectiveness of advertising, promotions, and pricing policies, enabling adjustments to meet sales goals.
3. **Finance and Risk Management:** Investors forecast returns on assets like stocks and bonds. Financial forecasts aid in risk evaluation, portfolio management, and pricing derivatives.
4. **Economics:** Forecasts of economic indicators such as GDP, unemployment, and inflation guide government policies, budgeting, and strategic business decisions.
5. **Industrial Process Control:** Production quality forecasts help adjust processes and ensure optimal operation through feedback and feedforward control schemes.
6. **Demography:** Population forecasts, including breakdowns by age, gender, and race, inform government policies and business strategies for product and service development.

## 2.4 Conclusion

This chapter has introduced time series data and forecasting, emphasizing their importance across various fields like finance, weather, sales, and healthcare. Key characteristics of time series data include regularity, order, seasonality, trends, stationarity, cycles, and noise. The chapter also covered the objectives and challenges of time series analysis, the forecasting process, and its diverse applications. Understanding and applying these concepts is essential for making accurate predictions and informed decisions.

# Chapter 3

# Related Work and Performance Analysis

## 3.1 Introduction

Time series forecasting is crucial in various fields, involving the prediction of future trends based on sequential data points. High-quality datasets, such as the ETT (Electricity Transformer Temperature), Electricity, Traffic, and Weather datasets, are essential for accurate forecasting. These datasets capture fundamental dynamics, enabling the study of complex temporal patterns and informing decision-making processes.

## 3.2 Dataset

Time series forecasting is a vital task across many fields, including forecasting sequential data points to anticipate future trends and behaviors. Central to this endeavor is the availability of high-quality datasets that capture the fundamental dynamics of the phenomena under study. One dataset of interest in forecasting is ETT (Electricity Transformer Temperature), Electricity, Traffic, and Weather, which provide insight into time series forecasting.

### 3.2.1 ETT (Electricity Transformer Temperature):

Electricity distribution in different areas depends on sequential usage, varying depending on factors such as the day of the week, holidays, season, weather, and temperature. Predicting future demand for a particular region is difficult due to these fluctuations. Currently, there is no method capable of making long-term forecasts based on rich real-world data with high accuracy. Inaccurate forecasts can damage power transformers, causing managers to make decisions based on empirical estimates that are often much higher than actual demand. This leads to wasted electricity and unnecessary equipment depreciation. However, oil temperature can reflect the condition of the power transformer. Predicting the safe operating temperature of transformer oil is an effective strategy to avoid unnecessary waste. To solve this problem, a practical platform was created, collecting data for two years to predict the oil temperature of power transformers and study their extreme load capacity[5].

### ETT-small Dataset

Two years of data were collected, with each data point recorded every minute (denoted "m"), in two regions of a province in China, labeled ETT-small-m1 and ETT-small-m2. Each dataset contains 70,080 data points (2 years × 365 days × 24 hours × 4 data points per hour). Additionally, hourly-level variants are provided for faster development (denoted by “h”), i.e., ETT-small-h1 and ETT-small-h2. Each data point includes 8 characteristics, including date, “oil temperature” predicted value, and 6 different types of external electrical load characteristics[6].

Table 3.1: Description for each column

| **Field** | **date** | **HUFL** | **HULL** | **MUFL** | **MULL** | **LUFL** | **LULL** | **OT** |
|---|---|---|---|---|---|---|---|---|
| **Description** | The recorded date | High UseFul Load | High UseLess Load | Middle UseFul Load | Middle UseLess Load | Low UseFul Load | Low UseLess Load | Oil Temperature (target) |

Below is a summary of statistics for various versions of the ETT (Electricity Transformer Temperature) dataset. This table provides key insights into the data time index, granularity, number of data columns, data types, memory usage, standard deviation, and mean for each version of the dataset.

Table 3.2: Statistics of ETT datasets.

| | **ETTh1** | **ETTh2** | **ETTm1** | **ETTm2** |
|---|---|---|---|---|
| **Data time index** | 17,420 | 17,420 | 699,680 | 699,680 |
| **Granularity** | 1 hour | 1 hour | 15 minutes | 15 minutes |
| **Data columns (features)** | 7 | 7 | 7 | 7 |
| **dtypes** | Float64(7) | Float64(7) | Float64(7) | Float64(7) |
| **memory usage** | 1.1 MB | 1.1 MB | 4.3 MB | 4.3 MB |
| **Std** | 6.5329 | 19.6167 | 6.5341 | 19.6287 |
| **Mean** | 4.5781 | 16.9976 | 4.5936 | 17.0099 |

### 3.2.2 Weather Dataset

The Weather dataset documented by Olaf Kolle from the Max-Planck-Institute for Biogeochemistry is a valuable resource for environmental and climatic research. This dataset, collected from a weather station located on top of the Institute building, is integral for understanding atmospheric conditions and their impact on biogeochemical cycles and their impacts on different domains such as agriculture, energy management, and environmental monitoring.

The documentation[2], dated 12 August 2008, provides detailed insights into the setup and data collection processes of this weather station.

Table 3.3: Variables, their symbols, and units of the Dataset weather[2].

| Symbol | Unit | Variable |
|---|---|---|
| Date Time | DD.MM.YYYY HH:MM (MEZ) | Date and time of data record (**end**) |
| P | mbar | air pressure |
| T | °C | air temperature |
| Tpot | K | potential temperature |
| Tdew | °C | dew point temperature |
| rh | % | relative humidity |
| VPmax | mbar | saturation water vapor pressure |
| VPact | mbar | actual water vapor pressure |
| VPdef | mbar | water vapor pressure deficit |
| sh | $g\ kg^{-1}$ | specific humidity |
| H2OC | $mmol\ mol^{-1}$ | water vapor concentration |
| rho | $g\ m^{-3}$ | air density |
| wv | $m\ s^{-1}$ | wind velocity |
| max. wv | $m\ s^{-1}$ | maximum wind velocity |
| wd | ° | wind direction |
| rain | mm | precipitation |
| raining | s | duration of precipitation |
| SWDR | $W\ m^{-2}$ | short wave downward radiation |
| PAR | $\mu mol\ m^{-2}\ s^{-1}$ | photosynthetically active radiation |
| max. PAR | $\mu mol\ m^{-2}\ s^{-1}$ | maximum photosynthetically active radiation |
| Tlog | °C | internal logger temperature |
| CO2 | ppm | $CO_2$-concentration of ambient air |

The following table provides statistics summarizing the Weather dataset. This dataset includes a comprehensive range of weather-related features recorded at a granularity of 10 minutes. It offers insights into various aspects.

Table 3.4: Statistics of Weather dataset.

| | **Weather** |
|---|---|
| **Data time index** | 52,696 |
| **Granularity** | 10minutes |
| **Data columns (features)** | 21 |
| **dtypes** | Float64(21) |
| **memory usage** | 8.8 MB |
| **Std** | 368.2373 |
| **Mean** | 188.8933 |

### 3.2.3 Electricity dataset

The dataset consists of electricity usage data in kW recorded every 15 minutes for 321 clients. We aggregate blocks of 4 columns to reflect hourly consumption, resulting in 26,304 hourly measurements. The coefficient of variation for the electricity data is 60,341[7].

The dataset is complete with no missing values. Each column corresponds to a client; for those who joined after 2011, their consumption before joining is recorded as zero. All timestamps are in Portuguese time. Each day includes 96 measurements (24 hours * 4 measurements per hour). On the March time change day, which has only 23 hours, the values between 1:00 am and 2:00 am are set to zero for all clients. On the October time change day, which has 25 hours, the values between 1:00 am and 2:00 am aggregate the consumption of both hours[8].

Table 3.5: Statistics of Electricity dataset.

| | **Electricity** |
|---|---|
| **Data time index** | 26304 |
| **Granularity** | 1 hour |
| **Data columns (features)** | 321 |
| **dtypes** | Float64(321) |
| **memory usage** | 64.6 MB |
| **Std** | 15027.5701 |
| **Mean** | 2538.7916 |

### 3.2.4 Traffic dataset

Traffic comprises hourly data spanning 48 months (2015-2016) sourced from the California Department of Transportation. This dataset delineates road occupancy rates, ranging between 0 and 1, as observed by various sensors installed along the freeways in the San Francisco Bay area[9].

Table 3.6: Statistics of Traffic dataset.

| | **Traffic** |
|---|---|
| **Data time index** | 17,544 |
| **Granularity** | 1 hour |
| **Data columns (features)** | 862 |
| **dtypes** | Float64(862) |
| **memory usage** | 115.5 MB |
| **Std** | 0.05403 |
| **Mean** | 0.05671 |

## 3.3 Data Preprocessing

Data preprocessing is a crucial step in any machine learning or deep learning pipeline. It involves transforming raw data into a format that is suitable for training models. In this section, we describe the various steps involved in preprocessing time-series data for various tasks it's involve as follow:

1. **Data Reading and Initialization:** define a set of classes to handle different types of datasets, indicating the granularity of time intervals (e.g., hourly or minutely).

2. **Data Loading:** The dataset is loaded from a CSV file located at a specified root path. This file typically contains timestamped data along with other relevant features.

3. **Data Scaling:** Standard scaling is applied to the dataset if specified by the scale parameter. This preprocessing step ensures that all features have a mean of 0 and a standard deviation of 1, which helps in training machine learning models effectively.

4. **Temporal Encoding:** Depending on the chosen encoding method (timeenc), temporal features such as month, day, weekday, hour, and minute are extracted from the timestamp data. This encoding converts temporal information into a format that can be understood by machine learning algorithms.

5. **Sequence Generation:** Sequences are generated for both input and output data, along with corresponding temporal marks. The sequence length (seq-len), label length (label-len), and prediction length (pred-len) are specified to define the length of sequences.

6. **Data Splitting:** The dataset is split into training, validation, and test sets based on specified proportions. This splitting ensures that the model is trained on a portion of the data, validated on another portion to tune hyperparameters, and finally evaluated on a separate portion to assess its performance.

7. **Inverse Transformation (Optional):** An inverse transformation is provided to revert scaled data back to its original scale if needed. This step is useful when predictions made by the model need to be interpreted in the original data space.

## 3.4 The different types of forecasting models

### 3.4.1 MLP-Mixer models

Recent research has shown a renewed interest in using multi-layer perceptrons (MLPs) for various tasks[10][11][12], including image analysis and time series forecasting. In image analysis, one approach involves arranging images through rearrangements, allowing the model to learn relationships between different image parts (patches) without relying on the complex self-attention mechanisms in Transformers. This method, called MLP-Mixer, achieves performance comparable to Convolutional Neural Networks (CNNs) and Transformers. Building on this concept, gMLP incorporates a special Spatial Gating Unit (SGU) within the MLP-Mixer architecture, while ResMLP introduces residual connections to improve the model's capabilities.

Regarding time series forecasting, MLP mixers have also shown promise. TSMixer (Time Series Mixer) is a model that leverages MLPs to capture complex relationships within time series data. Introduced in the paper **"TSMixer: An All-MLP Architecture for Time Series Forecasting"**, TSMixer[13] breaks down the data into segments and utilizes MLPs to perform mixing operations. These operations capture information exchange between different time steps and features, allowing TSMixer to handle complex dynamics effectively. **"TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting"** remains the primary research focusing on applying MLP mixers to this specific domain. This model[14], introduced in the paper, demonstrates that MLPs can achieve competitive results in multivariate time series forecasting compared to traditional methods like RNNs or attention-based models.

It's important to note that MLP mixers aren't the only innovative architecture for time series forecasting. PatchMixer is another approach that utilizes a convolutional neural network (CNN) architecture focusing on ”patch-mixing.” This method involves segmenting the data into temporal patches and employing a novel technique to integrate information within and between these patches. PatchMixer, described in the paper[15], is particularly efficient for long-term forecasting tasks.

### 3.4.2 Transformer-based time series models

The remarkable success of Transformers in natural language processing (NLP) has inspired researchers to explore their application in time series analysis. However, a significant challenge arises - standard Transformers struggle with the much longer sequences typical of time series data. This limitation stems from their inherent computational complexity, which grows dramatically with sequence length[14].

To address this bottleneck, researchers have proposed several time series-specific Transformer models. These models share the core Transformer architecture but introduce modifications to the critical attention mechanism. These modifications aim to achieve significantly faster processing while maintaining forecasting accuracy, allowing them to scale efficiently even for very long time series.

These time series-specific Transformer models can be broadly categorized based on their approach to tackling the long sequence challenge:

- **Modified Attention Mechanisms:** Several models, including Informer[6], FEDformer[16], and Triformer[17], focus on modifying the standard self-attention mechanism within the Transformer. These modifications aim to make the attention mechanism more efficient by strategically focusing on relevant parts of the time series data. This can involve techniques like probabilistic pruning (Informer) or introducing frequency-awareness (FEDformer) to prioritize informative sections of the sequence[18].

- **Hierarchical Attention:** Another approach, employed by Pyraformer, involves incorporating a hierarchical pyramid structure within the self-attention mechanism[19]. This allows the model to capture long-range dependencies at various granularities within the time series data, improving efficiency and accuracy.

- **Segmentation and Attention:** Patch TST tackles the long sequence problem by dividing the time series into smaller segments (patches)[20]. The model then processes these patches individually before feeding them into the Transformer architecture. This approach allows the model to capture local dependencies within each segment while maintaining overall forecasting accuracy.

- **Temporal Fusion:** The TFT specifically addresses the challenge of multi-horizon forecasting, a common task in time series analysis[21]. It achieves this by incorporating various time-varying inputs, along with past target values, into the forecasting process. This allows the model to effectively utilize both historical data and known future information for accurate predictions across multiple horizons.

### 3.4.3 Deep learning models specialized in time series forecasting

Deep learning has significantly advanced time series forecasting by offering powerful models capable of capturing complex temporal patterns. Recent breakthroughs in this field have led to the developing of specialized models tailored to address specific challenges.

N-BEATS, introduced by Boris Oreshkin et al.[22], is a notable contribution in this domain. This model showcases the superiority of pure deep learning architectures over traditional statistical approaches, delivering exceptional performance across various datasets. Moreover, N-BEATS offers interpretable outputs, enhancing its practical utility for real-world applications.

On the other hand, N-HITS[23], developed by another research team, addresses specific weaknesses present in N-BEATS. By incorporating innovative techniques such as multi-rate data sampling and hierarchical interpolation, N-HITS significantly improves the efficiency and accuracy of forecasting, particularly for hierarchical time series data. Together, these advancements highlight the remarkable progress made in deep learning-based time series forecasting, offering both improved accuracy and interpretability in forecasting models.

### 3.4.4 Linear models in time series forecasting

Linear models such as autoregressive (AR), moving average (MA), and their combination ARMA/ARIMA have been fundamental in time series analysis. These models effectively capture linear dependencies and trends, making them ideal for short- to medium-term forecasting tasks. Their simplicity facilitates rapid implementation and easy interpretation. However, recent advancements have introduced more complex models like Transformer models, as prominent tools to address time series forecasting challenges. These models are celebrated for capturing complex, nonlinear relationships and long-term dependencies in data. Despite their potential, these models have increased computational requirements and often need large datasets for effective training.

The paper *"Are Transformers Effective for Time Series Forecasting?"*[18] introduces a new, very simple single-layer linear model (LTSF-Linear) as a robust baseline, which surprisingly outperforms transformer-based methods. This work makes several key contributions:

1. It challenges the effectiveness of transformers in forecasting long-term time series, making it the first work, according to the authors, to critically examine this currently popular approach. The authors present LTSF-Linear, a suite of simple one-layer linear models, which supports their argument by outperforming transformer-based solutions on nine benchmark datasets. This suggests that complex transformer models may not always be necessary and that LTSF-Linear can serve as a strong baseline for future research.

2. The paper [18] conducts a comprehensive evaluation of transformer-based solutions. It explores various aspects including the ability to model long inputs, sensitivity to the order of data points in a time series, the effect of positional coding and subseries embedding, and efficiency comparisons. These in-depth studies provide deeper insights into the strengths and weaknesses of transformer models in the context of time series forecasting, providing valuable guidance for future research and model development in this area.

## 3.5 Analyze the Performance of Forecasting Models

1. **TFT (Temporal Fusion Transformer)** The best results for the TFT[21] model are obtained with the Electricity and ETTm2 datasets. Regardless of the prediction length $T$ (ranging from 96 to 720), the Electricity dataset has close convergence in the results. This performance is attributed to the large feature set (321 columns) and high standard deviation (15027.57). For the ETTm2 dataset, the best results are for prediction lengths $T = 96$ and $T = 336$. The high standard deviation (19.62) in ETTm2 contributes to better performance than ETTm1, which has a lower standard deviation and yields inferior results despite having the same feature size.

2. **Informer**

   The Weather, Electricity, and ETTm2 datasets show the best results for the Informer model[[6]]. For the Weather dataset, the model is suitable for short forecast lengths $T = 96$ and $T = 192$, with a feature size of 21 columns. This performance surpasses ETTh1 dataset, which has a higher standard deviation (368.23). For the ETTm2 dataset, the model performs well for short prediction lengths $T = 96$ and $T = 192$, benefitting from the higher standard deviation (19.62) of ETTm2 dataset, outperforming ETTm1 dataset. The Electricity dataset shows consistent results across different prediction lengths $T$ (ranging from 96 to 720) due to the large feature set (321 columns) and substantial standard deviation (15027.57).

3. **Autoformer**

   The Autoformer[[24]] model performs well across almost all datasets (ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity) except Traffic. The Traffic dataset's ineffectiveness is due to its minimal standard deviation (0.054) and mean (0.054), which hinder accurate predictions. In contrast, other datasets with larger sizes and numerous features positively impact the results, making Autoformer suitable for both short and long-term forecasting.

4. **FEDformer**

   The FEDformer[[16]] model shows a convergence in results similar to the Autoformer model. It performs well across almost all datasets except Traffic. The Traffic dataset's small standard deviation (0.054) and mean (0.054) do not support accurate predictions. Other datasets (ETTh1, ETTh2, ETTm1, ETTm2, Weather, Electricity) with larger sizes and more features positively influence the results.

5. **Pyraformer**

   The best results for the Pyraformer[[19]] model are obtained with the Electricity dataset. Regardless of the prediction length $T$ (ranging from 96 to 720), there is a close convergence in the results. This performance is driven by the large feature set (322 columns) and high standard deviation (15027.57). For ETTm1 and ETTm2 datasets, the model is suitable for short forecast lengths $T = 96$ and $T = 192$ with a feature size of 8 columns. ETTm1 has a standard deviation of 6.53, while ETTm2 has a standard deviation of 19.62.

6. **Triformer**

   The best results for the Triformer[[17]] model are seen with the Electricity and ETTm1 datasets. For the ETTm1 dataset, the model is suitable for short forecast lengths $T = 24$ and $T = 48$ with 7 features and a standard deviation of 6.53. The Electricity dataset performs well for various prediction lengths $T$ (ranging from 48 to 960) due to the large standard deviation (15027.57).

7. **N-BEATS**

   The N-BEATS model[22] focuses on solving the univariate time series point forecasting problem using a deep neural architecture. N-BEATS exhibited exceptional performance across multiple datasets. The M4 dataset, which includes a 100,000-time series, surpassed the competition winner without requiring any specialized components or feature engineering. Similarly, for the M3 dataset, consisting of 3,003-time series, N-BEATS matched or outperformed traditional statistical models like the Theta method. It also achieved state-of-the-art results on the TOURISM dataset, significantly improving forecast accuracy over existing methods.

The model's success is attributed to its ability to handle diverse time series without needing domain-specific adjustments. It effectively captures complex patterns and reduces the need for extensive feature engineering, resulting in enhanced accuracy and efficiency.

8. **N-HITS**

   The N-HITS model[23] (Neural Hierarchical Interpolation for Time Series Forecasting) introduces innovative techniques to address the challenges in long-horizon forecasting, and computational complexity.

   N-HiTS has demonstrated excellent performance in various forecasting tasks, particularly excelling in long-horizon forecasting. Its hierarchical interpolation and multi-rate sampling techniques help maintain accuracy over extended prediction windows. This is evident in its superior performance on weather datasets with a large number of features (21) and the ECL dataset, which has 321 features, where it achieves the best results for long-term forecasting.

   Moreover, N-HiTS significantly reduces computational costs compared to N-BEATS and other state-of-the-art models. This reduction is achieved through its innovative hierarchical approach and efficient use of computational resources.

9. **PATCH TST**

   The length of the time series is critical for long-term forecasting as it determines the amount of historical data available for model training. Longer time series provide more context and historical trends essential for accurate forecasting. In the Weather dataset, with time series lengths extending up to 52,696 data points and 21 features, the Patch TST's[20] model excels by capturing long-term dependencies and patterns necessary for precise weather predictions over extended periods. This model's performance demonstrates its ability to effectively utilize extensive historical data for accurate forecasts.

   In contrast, the Electricity dataset, with time series lengths of about 26,304 points and a large standard deviation, presents a different challenge. Here, the Patch TST's ability to capture seasonality and long-term trends is crucial for achieving accurate forecasts, allowing it to identify and leverage seasonal patterns and long-term dependencies effectively.

10. **LTSF-Linear**

    The paper 's[18] comparative analysis reveals that linear models often match or surpass the performance of Transformer models across various datasets. This is especially evident in scenarios where the datasets exhibit high standard deviation, high dimensionality, and varying lengths, which present significant challenges. Linear models achieve the best performance for long and short forecast horizons in datasets such as electricity, weather, ETTh1, ETTh2, ETTm1, and ETTm2.

    The simplicity and efficiency of linear models allow them to adapt flexibly to different data structures, providing reliable forecasts without the complexity and computational overhead of Transformer models. While Transformer models are theoretically powerful, their complexity can lead to issues like overfitting and computational inefficiencies, which undermine their practical performance in time series forecasting. Linear models, with their straightforward approach and focus on fundamental patterns and trends, consistently deliver competitive and sometimes superior results, making them a robust choice for diverse forecasting tasks.

## 3.6 Conclusion

This chapter has outlined the importance of time series forecasting and highlighted various high-quality datasets. we detailed the steps in data preprocessing and we discussed different forecasting models, including MLP-Mixer, Transformer-based, deep learning models, and linear models. Analyzing the performance of these models on various datasets demonstrated their strengths and limitations, emphasizing the significance of selecting the right model for accurate predictions.

# Part II

# Proposed Solutions

# Chapter 4

# Proposed solutions

## 4.1 Introduction

This chapter presents a series of innovative approaches to enhance time-series forecasting models. We explore advanced techniques to improve predictive accuracy and computational efficiency. Each proposed solution is systematically examined to highlight its potential benefits and contributions to the field.

## 4.2 TSMixer Model Overview

### 4.2.1 Overview

The TS-Mixer model is introduced as a novel approach for time series forecasting, leveraging a multi-layer perceptron (MLP) architecture. It is designed to address the limitations of complex sequential models like RNNs and Transformers by simplifying the model architecture while maintaining performance on par with state-of-the-art models. The primary objective of TS-Mixer is to capture temporal and cross-variate information in multivariate time series efficiently data[13].

### 4.2.2 Model Architecture

In the context of multivariate time series forecasting where only historical data is accessible, TSMixer employs MLPs alternately in time and feature domains. The architecture, illustrated in Figure 3.1, comprises the following key components:

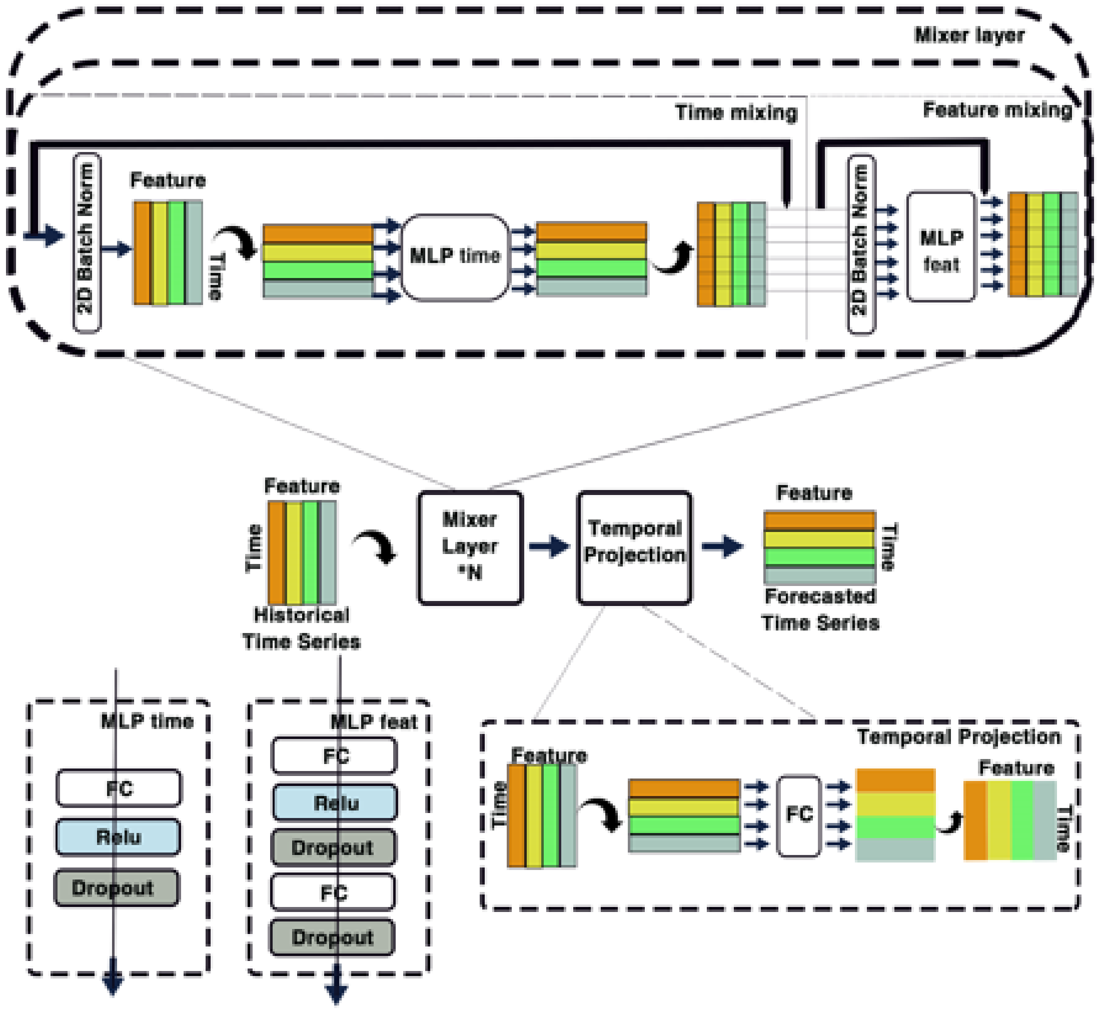

Figure 4.1: Model Architecture (TSmixer).

1. **Time-Mixing MLP:**These MLPs model temporal patterns across time series. Each consists of a fully connected layer followed by an activation function and dropout. The input is transposed to apply the fully connected layers along the time domain, shared by features. where a simple linear model proves effective for learning complex temporal patterns.

2. **Feature-Mixing MLP:** These MLPs applied consistently across time steps, incorporate covariate information. Like Transformer-based models, they utilize two-layer MLPs to understand complex feature transformations and capture the interactions among different features at each time step.

3. **Temporal Projection:** This component, comparable to the linear models, is a fully connected layer operating within the time domain. It learns temporal patterns and reconfigures the time series from the original input length (L) to the target forecast

length (T), thereby bolstering the model's ability to capture long-term dependencies and enhancing forecasting accuracy.

4. **Residual connections:**Residual connections are incorporated between each time-mixing and feature-mixing layer. These connections facilitate the learning of deeper architectures by allowing gradients to flow efficiently through the model. They enable the model to bypass unnecessary operations, thus enhancing learning efficiency and model performance.

5. **Normalization:**Normalization is a critical technique for improving the training of deep learning models. While the choice between batch normalization and layer normalization depends on the specific task, Nie et al. (2023)[20] highlight the benefits of batch normalization on common time series datasets. Instead of applying normalization along the feature dimension alone, TSMixer employs 2D normalization across both time and feature dimensions, accommodating the unique time-mixing and feature-mixing operations involved.

## 4.3 Proposed Improvements for TSMixer

### 4.3.1 Version 1.0: CNN Feature Extraction Before Mixer Layer

To enhance our time-series forecasting model's temporal pattern recognition and feature extraction capabilities, we integrated Convolutional Neural Networks (CNNs) introduced by Keiron O'Shea et al(2015)[25], before the Mixer layer. This setup aims to capture local patterns and dependencies within the data more effectively, thereby improving the model's performance on forecasting tasks.

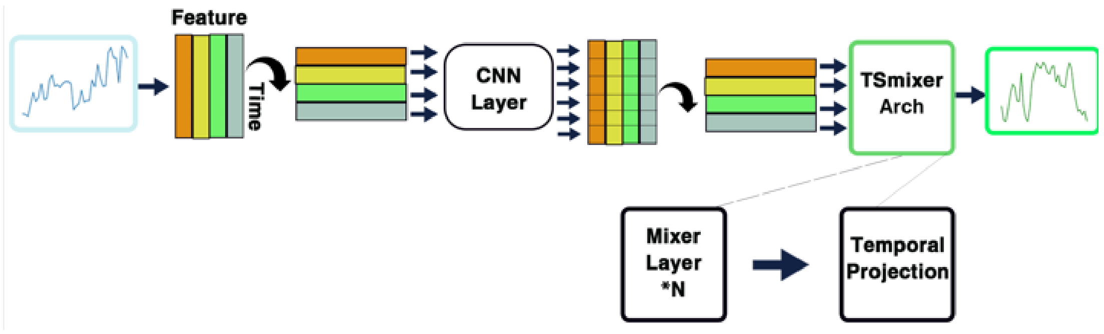

Figure 4.2: Architecture CNN-TSmixer.

The core architecture of our model is processing input time-series data to extract features for prediction, Including these crucial building blocks

1. **Convolutional Layer:** The CNN layer was implemented to process the input data sequence by applying convolution along the time axis. This layer detects important local dependencies and patterns within the data, such as trends and periodicity, which are critical for accurate forecasting. The parameters used in the CNN layer are:

    - **Input channels:**The number of input channels (features) of the time-series data.
    - **Output channels:**The number of output channels, is set to be the same as the number of input channels to maintain the feature dimensionality.
    - **kernel size:** Set to 3, which specifies the size of the filter that moves along the sequence to capture local information.
    - **Padding:**Set to 1, to ensure that the output length remains the same as the input length, preserving the sequence's temporal structure.

2. **TSmixer Layer:** Following the CNN layer, we integrate the TSMixer layer, which has been extensively discussed in Section 4.2. The TSMixer architecture is designed to facilitate both local and global temporal dependencies and feature interactions.

## 4.4 Conclusion

In summary, the proposed methodologies demonstrate improvements in time-series forecasting accuracy and efficiency. The experiments validate the effectiveness of novel architectural modifications, paving the way for future advancements in forecasting models.

# Chapter 5

# Experimental Results

## 5.1 Introduction

This chapter explores various experimental setups to evaluate the effectiveness of different deep-learning models for time series forecasting. It examines datasets with distinct temporal characteristics, detailing hyperparameters for consistent comparisons. Our experiments compare its performance with the original model TSmixer. Results are presented with performance metrics MSE and MAE across forecast horizons and the effectiveness and limitations of each modification. The analysis provides insight into how architectural changes affect forecast accuracy.

## 5.2 Materials

To evaluate the effectiveness of TSMixer for long-term time series forecasting, we conducted five experiments on Google Colaboratory (Colab). This cloud-based platform provided a flexible and accessible environment for training and testing various TSMixer configurations. Colab's integration with a Tesla T4 GPU significantly accelerated the computational processes. This powerful GPU was essential for handling the demanding tasks involved in training and evaluating the different TSMixer models we explored throughout the experiments. We also used GPU L4 and A100 for the dataset electricity to ensure robust performance across different TSmixer configurations.

## 5.3 The Used Hyperparameters

### 5.3.1 Hyperparameters Used

We trained our five experiments according to the hyperparameters with which the TSMixer model was trained according to the table below [13]:

Table 5.1: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset ETTh1 benchmark.

| | | ETTh1 | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.0001 | 6 | 0.9 | 512 | relu |
| | 192 | 0.001 | 4 | 0.9 | 256 | relu |
| | 336 | 0.001 | 4 | 0.9 | 256 | relu |
| | 720 | 0.001 | 2 | 0.9 | 64 | relu |

Table 5.2: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset ETTh2 benchmark

| | | ETTh2 | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.0001 | 4 | 0.9 | 8 | relu |
| | 192 | 0.001 | 1 | 0.9 | 8 | relu |
| | 336 | 0.0001 | 1 | 0.9 | 16 | relu |
| | 720 | 0.0001 | 2 | 0.9 | 64 | relu |

Table 5.3: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset ETTm1 benchmark

| | | ETTm1 | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.0001 | 6 | 0.9 | 16 | relu |
| | 192 | 0.0001 | 4 | 0.9 | 32 | relu |
| | 336 | 0.0001 | 4 | 0.9 | 64 | relu |
| | 720 | 0.0001 | 4 | 0.9 | 16 | relu |

Table 5.4: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset ETTm2 benchmark

| | | ETTm2 | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.001 | 8 | 0.9 | 256 | relu |
| | 192 | 0.0001 | 1 | 0.9 | 256 | relu |
| | 336 | 0.0001 | 8 | 0.9 | 512 | relu |
| | 720 | 0.0001 | 8 | 0.1 | 256 | relu |

Table 5.5: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset Electricity benchmark

| | | Electricity | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.0001 | 6 | 0.7 | 32 | relu |
| | 192 | 0.0001 | 8 | 0.7 | 16 | relu |
| | 336 | 0.0001 | 6 | 0.7 | 64 | relu |
| | 720 | 0.001 | 6 | 0.7 | 64 | relu |

Table 5.6: Hyperparameter tuning spaces, best configurations for TSMixer, and our experiments on dataset Weather benchmark

| | | Weather | | | | |
|---|---|---|---|---|---|---|
| Search space | | Learning rate | Blocks | Dropout | Hidden size | Heads |
| Model | T | 0.01, 0.0001 | 1, 2, 4, 6, 8 | 0.1, 0.3, 0.5, 0.7, 0.9 | 8, 16, 32, 64 | 4, 8 |
| All models | 96 | 0.0001 | 4 | 0.3 | 64 | relu |
| | 192 | 0.0001 | 8 | 0.7 | 32 | relu |
| | 336 | 0.0001 | 2 | 0.7 | 8 | relu |
| | 720 | 0.0001 | 8 | 0.7 | 16 | relu |

### 5.3.2 ReLU Activation Function

ReLU is an activation function used in neural networks, introduced with strong biological and mathematical underpinning. It is widely used to improve the training of deep neural

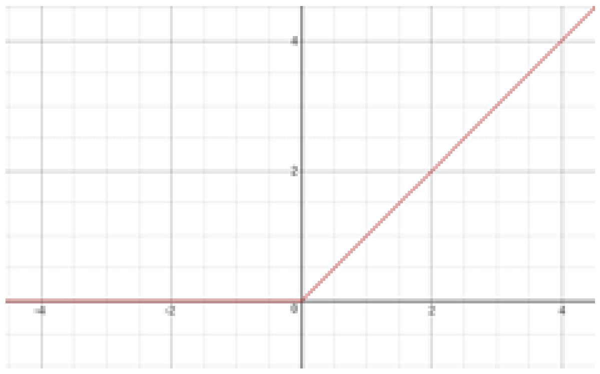

Figure 5.1: The Rectified Linear Unit (ReLU) activation function produces 0 as an output when x < 0, and then produces a linear output with a slope of 1 when x > 0.

networks. the input value itself.[26] The ReLU function works by thresholding the input values at 0. If the input is less than 0, the output is 0; if the input is greater than or equal to 0, the output is

### 5.3.3 GELU Activation Function

Building on ReLU's success, GELU (Gaussian Error Linear Unit) emerges as a smoother and more statistically sound activation function for neural networks. It surpasses the limitations of sigmoid's slow training and ReLU's lack of theoretical grounding by incorporating a Gaussian distribution, offering a balance of smoothness and the ability to handle negative values. This paves the way for a potentially better approximation of complex functions. The Gaussian Error Linear Unit (GELU) is defined as:

$$\mathrm{GELU}(x) = x\Phi(x)$$

where $\Phi(x)$ is the standard Gaussian cumulative distribution function.

The formula for the GELU is given as:

$$\mathrm{GELU}(x) = x \cdot \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

where $\mathrm{erf}(x)$ is the Gaussian error function.

GLEU also provides two approximations for faster computation:

$$\mathrm{GELU}(x) \approx 0.5x\left(1 + \tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]\right)$$

$$\mathrm{GELU}(x) \approx x\sigma(1.702x)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.[27]

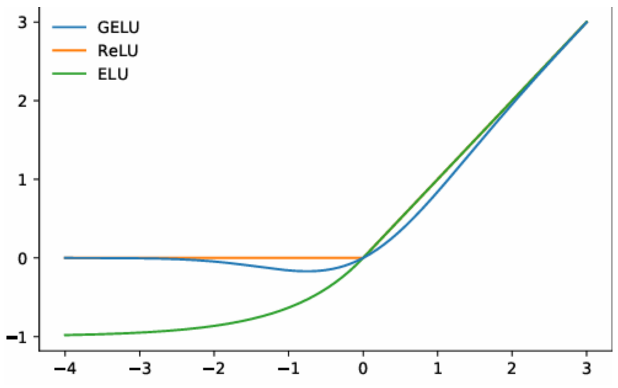

Figure 5.2: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).

### 5.3.4 Loss Function

The loss function is crucial in numerous deep-learning algorithms and is vital for training and optimizing models. Our approach utilized the Mean Absolute Error (MAE) and Mean Square Error (MSE).

#### Mean Absolute Error (MAE)

The Mean Absolute Error is a metric used to measure the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5.1}$$

#### Mean Square Error (MSE)

The Mean Squared Error measures the average of the squares of the errors. It is the second moment (about the origin) of the error and thus incorporates both the variance of the estimator and its bias.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5.2}$$

### 5.3.5 ADAM Optimizer

Adam (short for Adaptive Moment Estimation) is an algorithm for first-order gradient-based optimization of stochastic objective functions. It is designed to combine the advantages of two other popular optimization methods: AdaGrad and RMSProp. Adam is particularly well-suited for problems with large data and parameter spaces, non-stationary objectives, and noisy or sparse gradients.[28]

**Role of Adam:**

- Computes individual adaptive learning rates for different parameters.
- Efficient with little memory requirement.
- Invariant to diagonal rescaling of the gradients.
- Suitable for online and non-stationary settings.

**Algorithm Formula:**

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

**Where:**

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\ m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \end{aligned}$$

This formula shows how Adam adjusts the learning rates for each parameter based on the estimates of the first and second moments of the gradients, resulting in a robust and adaptive optimization process.

## 5.4 Experimental Results

### 5.4.1 Discussion of Exp 1:

This experiment added a CNN feature extraction layer to the TSMixer architecture to improve its ability to capture local patterns in time-series data. However, the original TSMixer consistently outperformed the CNN-enhanced version across various datasets and forecasting horizons, especially for simpler datasets with straightforward temporal patterns. For instance, in the ETTh1 dataset, the original TSMixer had a lower MSE of **0.441** compared to CNN-TSMixer's **0.713** for a **96**-time step forecast.

The CNN-TSMixer showed some advantages in complex datasets like Electricity and Weather, where it performed better in shorter-term forecasting. For example, in the Electricity dataset, the CNN-TSMixer achieved an MSE of **0.291** compared to the original's **0.335** for a **96**-time step forecast. However, these benefits were inconsistent across all forecasting horizons, and the original TSMixer generally performed better for longer forecasts.

The additional complexity of CNN layers did not significantly improve performance, particularly on simpler datasets. This discrepancy suggests that convolutional layers may not integrate well with the TSMixer architecture. Alternatively, the model may not utilize these layers effectively, or the layers might introduce noise, negatively impacting performance in some cases. The results are shown in the table as follows 5.7:

| **Models** | | **Tsmixer** | | **CNN-TSmixer** | |
|---|---|---|---|---|---|
| **Metrics** | **T** | **MSE** | **MAE** | **MSE** | **MAE** |
| **ETTh1** | 96 | **0.441** | **0.441** | 0.713 | 0.565 |
| | 192 | **0.436** | **0.442** | 0.568 | 0.517 |
| | 336 | **0.506** | **0.496** | 1.119 | 0.727 |
| | 720 | **0.525** | **0.515** | 0.845 | 0.683 |
| **ETTh2** | 96 | **0.331** | 0.392 | 0.353 | **0.391** |
| | 192 | **0.357** | **0.390** | 0.432 | 0.440 |
| | 336 | **0.342** | **0.398** | 0.411 | 0.452 |
| | 720 | **0.402** | **0.439** | 0.441 | 0.476 |
| **ETTm1** | 96 | **0.392** | **0.406** | 0.505 | 0.475 |
| | 192 | **0.340** | **0.375** | 0.371 | 0.400 |
| | 336 | **0.383** | **0.406** | 0.433 | 0.442 |
| | 720 | 0.505 | 0.490 | **0.502** | **0.489** |
| **ETTm2** | 96 | **0.179** | **0.260** | 0.196 | 0.279 |
| | 192 | **0.231** | **0.300** | 0.244 | 0.312 |
| | 336 | **0.320** | **0.369** | 0.354 | 0.390 |
| | 720 | **0.416** | **0.429** | 0.423 | 0.434 |
| **Electricity** | 96 | 0.335 | 0.416 | **0.291** | **0.375** |
| | 192 | 0.280 | 0.387 | **0.265** | **0.365** |
| | 336 | **0.235** | **0.345** | 0.239 | 0.348 |
| | 720 | **0.242** | **0.342** | 0.266 | 0.369 |
| **Weather** | 96 | 0.227 | 0.270 | **0.203** | **0.257** |
| | 192 | **0.263** | **0.306** | 0.266 | 0.306 |
| | 336 | 0.276 | 0.320 | **0.260** | **0.296** |
| | 720 | **0.341** | **0.364** | 0.357 | 0.370 |

Table 5.7: Performance comparison of **TSmixer** with **CNN-TSmixer** experiments on various datasets, The bold red is the best results

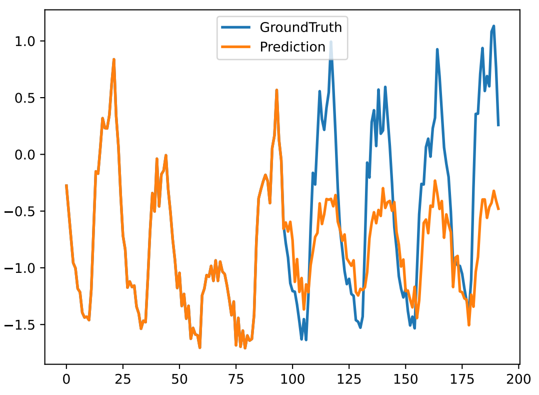

Figure 5.3: Seq(96) vs Pred(96) Length in CNN-TSMIXER: [Electricity]

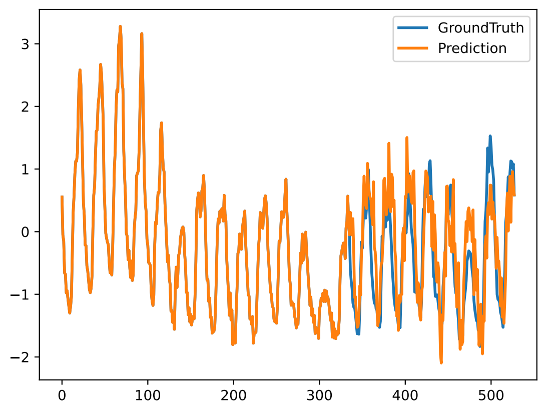

Figure 5.4: Seq(336) vs Pred(192) Length in CNN-TSMIXER: [Electricity]

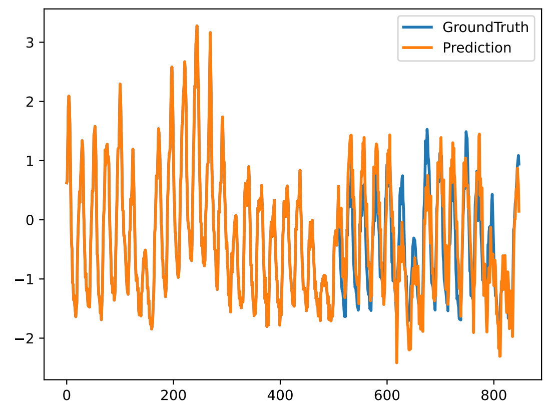

Figure 5.5: Seq(512) vs Pred(336) Length in CNN-TSMIXER: [Electricity]

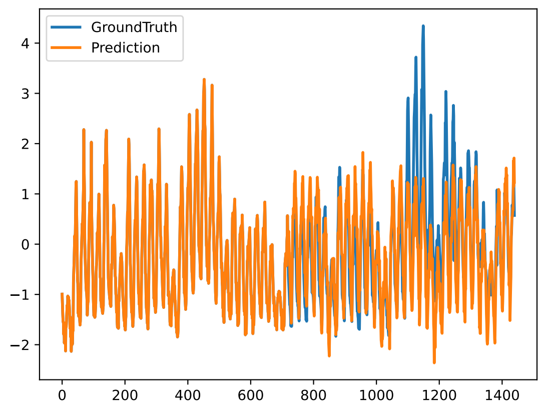

Figure 5.6: Seq(720) vs Pred(720) Length in CNN-TSMIXER: [Electricity]

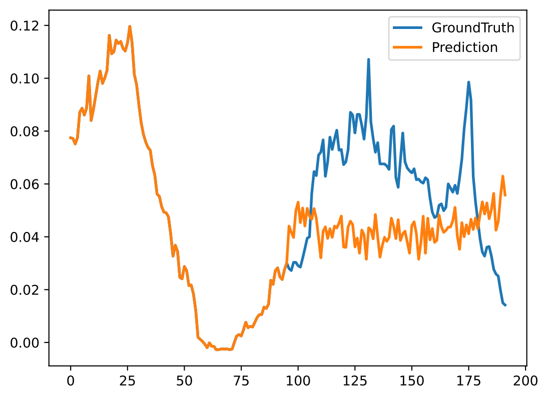

Figure 5.7: Seq(96) vs Pred(96) Length in CNN-TSMIXER: [Weather]

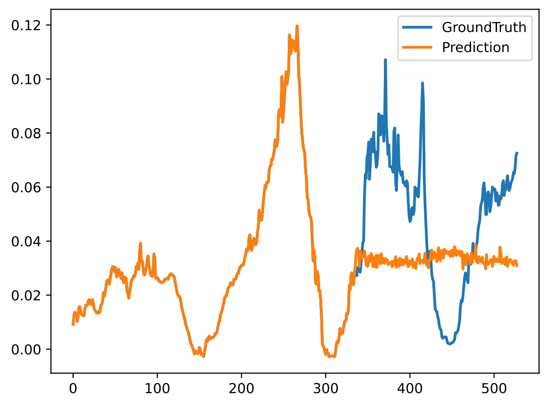

Figure 5.8: Seq(336) vs Pred(192) Length in CNN-TSMIXER: [Weather]

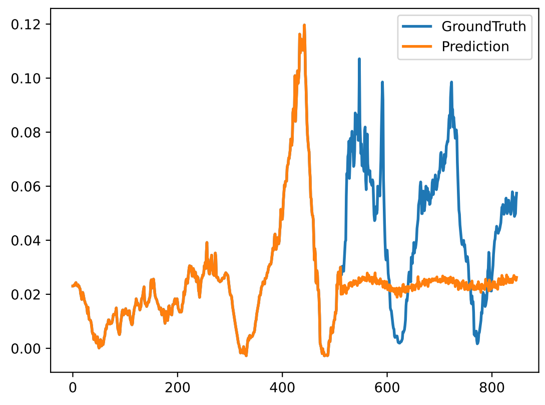

Figure 5.9: Seq(512) vs Pred(336) Length in CNN-TSMIXER: [Weather]

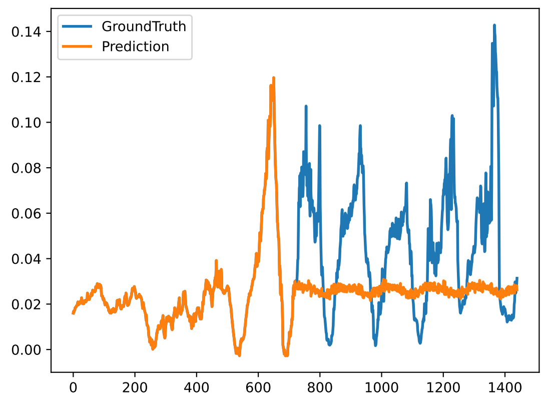

Figure 5.10: Seq(720) vs Pred(720) Length in CNN-TSMIXER: [Weather]

## 5.5 Conclusion

This chapter explored several architectural modifications to the TSmixer model to enhance its time series forecasting performance. The key findings of these experiments highlighted the impact of integrating convolutional technique and adjusting feature mixing strategies.

# General Conclusion

In this thesis, we focused on deep learning (DL) to enhance multi-variate time series forecasting, which involves unique challenges due to the intricate interdependencies and varying temporal patterns across multiple time series. Deep learning's capability to model complex, non-linear relationships makes it suitable for addressing these challenges. Our research aimed to evaluate and optimize the performance of DL models, specifically improving the TSMixer architecture.

1. **Performance Analysis of the Models by Dataset :** pivotal part of our research involved detailed performance analysis of various models across multiple datasets and prediction horizons to understand their strengths and limitations.
2. **Architectural Enhancements for Improving TSMixer Performance**

In conclusion, this thesis has demonstrated significant advancements in multi-variate time series forecasting through the performance analysis and architectural enhancement of DL models, particularly the TSMixer. The detailed performance analysis across various datasets and prediction horizons highlighted the models' robustness and versatility. Future work will focus on refining these architectural enhancements and exploring their applicability to even broader types of time series data. Additionally, integrating more advanced optimization techniques and exploring ensemble methods could further enhance the models' performance and generalization capabilities.

# Bibliography

[1] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. *Introduction to Time Series Analysis and Forecasting*. Wiley Series in Probability and Statistics. Wiley, 2nd edition, 2021.

[2] Olaf Kolle. Documentation of the weather station on top of the roof of the institute building of the max-planck-institute for biogeochemistry. Technical report, Max-Planck-Institute for Biogeochemistry, August 2008.

[3] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer, New York, second edition, 2002.

[4] Ankur Debnath, Govind Waghmare, Siddhartha Asthana, Ankur Arora, and Hardik Wadhwa. Exploring generative data augmentation in multivariate time series forecasting: Opportunities and challenges. 2024.

[5] Haoyi Zhou. Ett dataset. https://github.com/zhouhaoyi/ETDataset, 2021. Accessed: 2024-05-29.

[6] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 11106–11115. AAAI Press, 2021.

[7] Hsiang-Fu Yu, Inderjit S. Dhillon, and Nikhil Rao. Temporal regularized matrix factorization for high-dimensional time series prediction. *arXiv preprint arXiv:1610.01024*, 2016.

[8] UCI Machine Learning Repository. Electricity load diagrams 2011-2014 data set. https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014, 2014. Accessed: 2024-05-29.

[9] Guokun Lai, Wei-Cheng Chang, Hanxiao Liu, and Yiming Yang. Modeling long- and short-term temporal patterns with deep neural networks. *arXiv preprint arXiv:1703.07015*, cs.LG(v3), April 2018. Accessed: 2024-05-29.

[10] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[11] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. *Advances in Neural Information Processing Systems*, 34:9204–9215, 2021.

[12] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

[13] Si-An Chen, Chun-Liang Li, Nathanael C. Yoder, Sercan Ö. Arık, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, Sep 2023. Reviewed on OpenReview: `https://openreview.net/forum?id=wbpxTuXgm0`.

[14] NamNguyen Phanwadee Sinthong Jayant Kalagnanam Vijay Ekambaram, Arindam Jati. Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. *PubMed*, 2023. `https://arxiv.org/abs/2306.09364`.

[15] Zeying Gong, Yujin Tang, and Junwei Liang. Patchmixer: A patch-mixing architecture for long-term time series forecasting. October 2023.

[16] Tian* Zhou, Ziqing* Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. June 2022.

[17] Razvan-Gabriel Cirstea, Chenjuan Guo, Bin* Yang, Tung Kieu, Xuanyi Dong, and Shirui Pan. Triformer: Triangular, variable-specific attentions for long sequence multivariate time series forecasting. 2023.

[18] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Github repo: Are transformers effective for time series forecasting? *arXiv preprint*, 2022. `https://github.com/cure-lab/LTSF-Linear`.

[19] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramid attention for long-range time series modeling and forecasting. 2023. † Corresponding author.

[20] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[21] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[22] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[23] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint*, 2022.

[24] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 22419–22430, 2021.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, December 2015.

[26] Abien Fred M. Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2019. https://arxiv.org/abs/1803.08375v2.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). June 2023.

[28] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

[29] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. Stl: A seasonal trend decomposition procedure based on loess (with discussion). *Journal of Official Statistics*, 6:3–73, 1990.

[30] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? arguments against avoiding rmse in the literature. *Geosci. Model Dev.*, 7:1247–1250, 2014. ©Author(s) 2014. CC Attribution 3.0 License.