SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti"
Master in Control Engineering

Thesis For The Degree Of Master

Wavelet-Driven Recurrent Neural Networks for Short Term Load Forecasting

Thesis Advisor
Prof. Danilo Comminiello

Candidate
Pouya Irani
1937388

Academic Year 2022-2023

Abstract

A short-term load forecast predicts grid loads for one or more hours in a moving window few hours (for example, every one hour), a short-term load forecast will provide predictions regarding grid loads for one or more hours inside a moving window every few hours (for example, every one hour). The forecasting procedure should also estimate good forecast intervals live. This will allow the accuracy of the forecast to be quantified in real time. Accurate forecasting at regular intervals is essential for the allocation of resources and the control of regional output. It also assists energy market operators in making decisions that are in line with common sense. However, this presents a challenge when one takes into account the noisy data collecting process, which may or may not be accompanied by failures of the data acquisition equipment, the various features of load frequency components, and the need for precise derivation and estimates in order to predict the forecast interval in real time.
STLF is an essential component of the energy planning industry. In order to design a time-ahead power market, generation, transmission, and distribution must schedule their demand. STLF assists power system administrators with a variety of power system-related decisions, such as supply planning, generation reserve, system security, dispatching scheduling, demand-side management, and financial planning. While STLF is particularly important for the operation of the time-ahead power market, inaccurate demand forecasting will place a substantial financial burden on the utility.

The purpose of this thesis is to offer a wavelet-based neural network structure known as the multilevel Wavelet Decomposition Network (mWDN) for the purpose of creating frequency-aware deep learning models for time series analysis. The advantages of multilevel discrete wavelet decomposition in frequency learning are maintained by mWDN, which also enables the fine-tuning of all parameters within the context of a deep neural network. With the help of the mWDN, try out a few different deep neural network models, and evaluate the outcomes in light of conventional deep learning models that do not make use of the wavelet decomposition technique.
The models make use of the back propagation technique to learn all of the parameters globally, which makes it possible to integrate wavelet-based frequency analysis seamlessly into existing deep learning frameworks. As inputs, the models either use all of the mWDN decomposed sub-series in all of the different frequencies or just a selection of them. Our time series models that are based on mWDN work remarkably well, as shown by extensive experiments conducted on 10 different datasets as well as one that contains real-world user volume data. To be more specific, the model under consideration is an importance analysis approach for mWDN-based models. This method is designed to efficiently identify those time-series items and mWDN layers that are necessary to the process of time series analysis. This highlights the interpretability benefit of mWDN, and it may be viewed as an all-encompassing examination of interpretable deep learning.

Keywords: Short-term load forecast, Grid loads, Forecast predictions, Accuracy, Resource allocation, Control of regional output, Deep neural network, Wavelet-based neural network, Real-time prediction, Interpretability.

Contents

List of Figures **v**

List of Tables **vi**

1 Introduction **1**
1.1 Research Motivation . . . 1
1.1.1 Challenges in STLF . . . 1
1.2 What is the problem and why it is important . . . 2
1.3 Summary of the contents of the thesis . . . 3

2 Background and literature review **5**
2.1 Short Time Load Forecasting . . . 5
2.1.1 Recurrent neural network(RNN) . . . 6
2.1.2 Long Short Term Memory(LSTM) . . . 8
2.1.3 Gated Recurrent Neural Network(GRU) . . . 9
2.1.4 Recurrent neural networks for forecasting . . . 9
2.2 Transformer model . . . 10
2.2.1 Vanilla Transformer . . . 10
2.2.2 Road Map of Transformers for Time-Series Analysis . . . 14
2.3 Wavelet Transform . . . 18
2.3.1 Wavelet transform literature review . . . 19
2.3.2 Continuous Wavelet Transform (CWT) . . . 20
2.3.3 Discrete wavelet transform . . . 21

3 Methodology **23**
3.1 Time/Frequency domain . . . 24
3.2 Model . . . 25
3.2.1 Multilevel Discrete Wavelet Decomposition . . . 25
3.2.2 Multilevel Wavelet Decomposition Network . . . 26
3.2.3 Multi-frequency Long Short-Term Memory . . . 29
3.2.4 Optimization . . . 30

4 Experiment and Result **32**
4.0.1 Experimental Setup . . . 32
4.0.2 Results and Analysis . . . 33
4.0.3 Experiment . . . 34
4.0.4 Result and analysis . . . 35
4.0.5 Visualization . . . 37
4.0.6 Layer Importance Analysis . . . 42
4.0.7 Result and Analysis . . . 43

5 Conclusion **45**
5.1 Research prospective and future work 47

Bibliography **49**

List of Figures

2.1 RNN architecture [33] . . . 7
2.2 LSTM architecture [20] . . . 8
2.3 Encoder/decoder block [1] . . . 12
2.4 recreation of the original Transformer architecture [1] . . . 15

3.1 The mWDN framework [41] . . . 27
3.2 The mWDN framework [41] . . . 29

4.1 Comparison of prediction performance with varying period lengths (Scenario I) . . . 36
4.2 Comparison of prediction performance with varying prediction sequence length (Scenario II). . . . 36
4.3 LSTM model Comparison . . . 37
4.4 RNN model Comparison . . . 38
4.5 GRU model Comparison . . . 38
4.6 Transformer model Comparison . . . 39
4.7 LSTM model Comparison . . . 40
4.8 RNN model Comparison . . . 41
4.9 GRU model Comparison . . . 41
4.10 Transformer model Comparison . . . 42
4.11 Importance spectra of mLSTM . . . 44

List of Tables

1.1 Availability of climate factors, economics, and land use information for load forecasting [21] 3

2.1 Comparison of Methods 17

4.1 MAPE metrics Comparison 34

Chapter 1

Introduction

1.1 Research Motivation

The short-term load forecasting (STLF) method is an important approach that is used to estimate the future loads in an electrical power network over a period of several hours up to several days, often in intervals of a few hours. These predictions are typically made in intervals of a few hours. This method of forecasting utilises a moving window approach, which involves continuously updating the forecasts in light of the most recent data that is gathered at predetermined time intervals. The regulation of area generation and resource dispatch in the power network is made possible with the use of accurate STLF, which enables efficient management of energy supply and demand.

It is equally as necessary to provide credible prediction intervals (PI) alongside the forecasts as it is to generate accurate load forecasts. A measure of the uncertainty associated with the projections is provided by prediction intervals. This enables decision-makers in the electricity market to evaluate the level of risk and make decisions based on accurate information. The capacity to create precise PIs in real time is absolutely necessary for efficient load management, which is in turn essential for ensuring the stability and dependability of the power grid.

1.1.1 Challenges in STLF

A number of obstacles must be overcome before an effective forecast of short-term loads can be made in conjunction with trustworthy prediction intervals. The presence of several frequency components in the load time series is one of the key obstacles that must be overcome. Each frequency component may display its very own unique pattern, which may include variations on a monthly, weekly, or even hourly basis. Accurate STLF forecasting requires both an effective method of capturing these patterns and an incorporation of those patterns into the forecasting models.

The fact that the process of data collection comes with its own background noise is still another obstacle. There is a possibility that the data collection devices will malfunction, which would result in some data points being absent or incorrect. The inconsistencies and errors that are present in the data might have a substantial impact on the reliability of the projections. To address the situations involving noisy data and to limit the influence these scenarios have on the accuracy of forecasting, dependable strategies and models need to be established.

In addition, accurately deriving prediction intervals in real time is a difficult and complex operation. Incorporating uncertainty measures, taking into account the statistical properties of the

data, and taking into account the potential errors and variances in the forecasting models are all necessary steps in this process. It is essential to develop methods that are dependable and efficient in order to estimate prediction intervals live if one want to maintain the credibility of the forecasts and ensure that they are effective in real-world applications.

It is vital for the proper operation of the power network to address these difficulties and develop effective solutions for STLF with accurate load forecasts and dependable prediction intervals. Doing so will facilitate optimal resource allocation and decision-making by market participants.

1.2 What is the problem and why it is important

Every society's economic development is dependent on the availability of electric power. The continuous existence of electric power and the abundance of sources for its production are of the utmost importance. Today, the electric demand prediction is a crucial process with numerous applications, such as in the production of electricity by an electric company. Within such a company, many departments, such as the marketing department or the trade market, can benefit from electric demand prediction. The following are a summary of the business benefits of using electric capacity prediction [38]:

1. For purchasing and producing electric power.
2. For transmitting, transferring and distributing electric power.
3. For managing and maintaining the electric power sources.
4. For managing the daily electric load demand.
5. For financial and marketing planning.

There is no formal or standard procedure for forecasting electric demand for every type of electric company and for every time period. Forecasting is contingent on a number of variables, including the means and sources of electrical production by each electrical company, the demand for electric capacity, climate factors, economic considerations, and human activity [38].
Climate factors are founded on meteorological characteristics such as temperature, humidity, wind speed, and precipitation. According to scientific literature, temperature plays a significant role in electric load consumption, and numerous electric load forecasting systems use temperature as a factor. Temperature can be used for forecasts with a short horizon, such as 1-2 days, but is unreliable beyond that [21].
Various aspects of the influence of human activity on electric discharge consumption can be analyzed. The day of the week and month of the year are examples of calendar-related properties. Other such characteristics involve economic factors, such as urban economic activity or economic transactions. For long-term forecasting, such as an annual forecast, economic factors play a crucial role in predicting the future. In addition, this type of forecasting can incorporate human activities from rural areas, such as agricultural activities and changes in rural land use [38].

The different types of electric load forecasting based on the data/features available that affect the forecasting and in conjunction with the duration of time for the prediction, can be categorized in the following categories:

1. Very Short Term Load Forecasting (VSTLF): Can be used for small time-window, minutes to one hour at most and it utilizes the amount of previous hourly loads on that current day. It can not use information from economic factors or land because they do not change in this small amount of time.

2. Short Term Load Forecasting (STLF): This type of forecasting can be used from one-hour forecasting up to a day or two. The difference from the previous forecasting method is that here temperature and in general climate/meteorological factors play a more important role for the forecasting. STLF can be used for decision-taking like how much energy should an electric company purchase from other electric companies in the near future.

3. Medium Term Load Forecasting (MTLF): This forecasting method can be used for 1 month to 3 years horizon. Temperature and other climate factors do not be used for forecasting due to their unavailability of their own forecasting for this horizon. Therefore, various methods, such as modelling their behaviour and forecasting using this method, are used in order to make use of MTLF loaf forecasting. MTLF uses economical factors because their impact plays a role in daily life and for the needs of consumers. The same thing applies to the change of land in rural areas.

4. Long Term Load Forecasting (LTLF): This forecasting is used for time-window 3-10 years. Simulation techniques are used to calculate climate factors and the economic transactions/activities.

The following table 1.1 summarizes the availability of climate features, economic factors and land use for load forecasting [21].

Table 1.1: Availability of climate factors, economics, and land use information for load forecasting [21]

factors / forecasting accuracy	Accurate	Inaccurate	Unreliable
climate factors	1 day	2 weeks	>2 weeks
Economics	1 month	3 years	>3 years
Land Use	3 years	5 years	>5 years

1.3 Summary of the contents of the thesis

Thesis composed by the following chapters:

- Chapter 1: **Introduction**
  This chapter presents the Electricity Load Demand Forecasting, trying to describe the issue, why it is important and commercial software on the internet that deals with it.

- Chapter 2: **Background and literature review**
  This chapter describes the theoretical background for four models that were used for this thesis.also a brief background of wavelet transform theory and a road map of it to be handful for time series forecasting

- chapter 3: **Methodology**
  This chapter presents the implementation of machine learning models, their predictions, their accuracy performance and finally the visualizations and comparison between them.

- chapter 4: **Experiment and Result**
  This chapter contains the application of the proposed model to real datasets and evaluate the model and demonstrate the superiority of the proposed model.

- Chapter 5: **Conclusions**
  This chapter summarizes the developed application, the produced models, the results, and the derived conclusions from them.

Chapter 2

Background and literature review

2.1 Short Time Load Forecasting

Forecasting electricity demand has been a significant obstacle for power system scheduling at various energy sector levels. Load forecasting can be conducted using a variety of methods. The selection of a forecasting method is contingent upon a number of factors, such as the relevance and availability of historical data, the forecast horizon, the level of accuracy for weather data, the intended prediction accuracy, etc. Consequently, the time horizon of the prediction largely influences the selection of the most suitable load forecasting method. Based on their specific applications in power system planning, different time horizons are employed for load forecasting. For instance, distribution and transmission planning are involved with STLF, whereas load prediction provides a decision platform for financial or power supply planning for extended durations, ranging from more than a year to a few decades. Similarly, the required level of accuracy in these time horizons is not the same, as long-term decisions are preliminary and may require substantial changes in subsequent planning stages due to highly uncertain input data, whereas a short-term forecast provides information to the day-ahead market, which requires a high level of accuracy. Additionally, various types of predictor variables are considered for each forecast horizon. Long-term load forecasting, for instance, takes population fluctuations, economic development, industrial construction, and technological advancement into consideration, whereas short-term forecasting focuses primarily on calendar variables, weather data, and customer behaviour. In general, both time categories of load forecasts are essential for power system operation and development, particularly with the integration of distributed generators onto the grid, which increases the intermittent nature and vulnerability of power supply. This study investigates the STLF approaches exclusively by reviewing original research papers in the field. Even though some artificial intelligence (AI) techniques used for short-term load forecasting (STLF) may also be applicable for long-term load forecasting (LTLF), they cannot be compared on a methodological basis for the aforementioned reasons.

Traditionally, engineering methods were utilised to painstakingly forecast future demand using charts and tables. These conventional methods primarily considered weather and calendar effects. Currently, these characteristics are still determined for constructing load models using innovative techniques.With the advent of statistical software packages and artificial intelligence techniques, a number of exceptional works of research are devoted to statistical and computational intelligence (CI) approaches to modelling future burden. In the literature, statistical regression-based STLF

approaches such as auto-regressive moving average (ARMA), auto-regressive integrated moving average (ARIMA), and seasonal ARIMA (SARMIA) are examples.Artificial neural network (ANN), support vector machine (SVM), fuzzy logic, etc., are prevalent techniques for CI-based forecasting [14].
Artificial neural networks are able to produce good predictions even in nonlinear functions, without knowing the relationship between the model to be predicted and the data [29].
The experimental results show that long short-term memory (LSTM) and hybrid models work more accurately and significantly reduce error compared to other models, especially in time series power forecasting systems. The effectiveness of neural networks in power generation forecasting has been significantly validated. Artificial neural networks and deep learning (DL) technology have been used in the automation of many industries until today [2]. Compared to neural networks, there are hidden layers in deep learning and the word "deep" in deep learning corresponds to these layers. These layers can simulate neurons in the human brain, allowing the computer perceive problems like human. As a percentage, deep learning has become one of the most popular and attractive technologies for short-term electric power generation forecasting performance with long short-term memory neural network (LSTM) and gated recurrent unit (GRU) algorithms [23],[19].
Neural networks are a popular framework for supervised learning because they are a network-like system of weighted summations and differentiable functions that can learn remarkably complex things. Typically, gradient descent variants in conjunction with back-propagation (chain rule) are employed to determine the optimal network weight values. All of this is straightforward and obvious, yet the resulting networks are typically difficult to comprehend. There are so many weights and connections that we cannot comprehend how the system generated the results. We don't comprehend neural networks, but we like them. The reason for their ever-growing popularity is uncomplicated: they are excellent. They can learn arbitrarily complex functions and frequently provide excellent predictions for quite difficult machine learning issues.
in this section the deep learning models will be fully introduced.

2.1.1 Recurrent neural network(RNN)

RNNs are neural networks designed for sequential data; we adapt them here to time series. In order to make the current prediction, recurrent neural networks utilise not only the input data, but also the previous outputs. This concept makes a great deal of sense; we could create neural networks that pass values forward in time. However, such straightforward solutions rarely perform as anticipated. They are difficult to train and neglectful. Rather, we require a system with memory of some kind [35].
The RNN model architecture consists of varying numbers and varieties of layers and units per layer. The primary distinction between RNN and Fully Connected Neural Network is that each RNN unit receives both current and previous input data simultaneously. The output of the RNN model is dependent on the previous data. RNNs process input sequences sequentially at all times during operation. The units in the concealed layer keep track of the input's history in the "state vector". An RNN is converted to a Deep Multilayer Perceptron (DMLP) when the output of the units in the hidden layer is partitioned into discrete time increments [26].
fig 2.1 illustrates that In the RNN's hidden layer, the information flow is divided into discrete phases. The status of the node S at distinct times of t is represented by s_t, the input value x is

represented by x_t, and the output value o is represented by o_t. In each stage, parameter values (U, W, V) are always utilised.

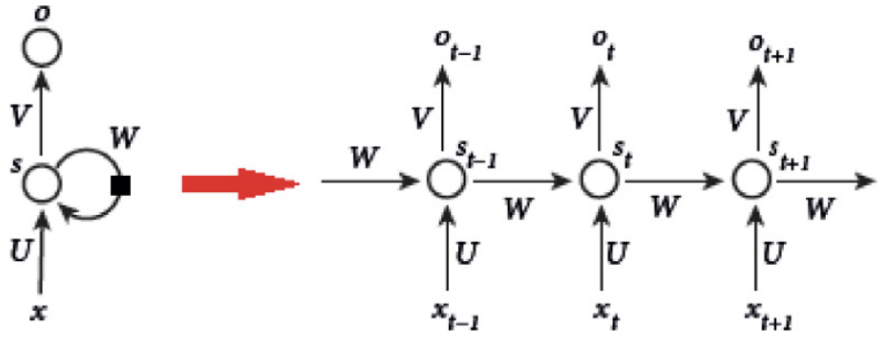


Figure 2.1: RNN architecture [33]

RNN structure has a temporal dependence in reverse. Consequently, RNNs become more complicated as the learning period increases. Although the primary purpose of an RNN is to learn long-term dependencies, research indicates that when knowledge is stored for extended periods of time, it is difficult to learn with an RNN [33].

As in the DMLP case, the hyperparameters of the RNN also define the network architecture, and the performance of the network is influenced by the parameter choices. RNN hyperparameters consist of the number of hidden layers, the number of units in each layer, regularisation techniques, network weight initialization, activation functions, learning rate, momentum values, number of epochs, batch size (mini-batch size), decay rate, optimisation algorithms, model (Vanilla RNN, Gated- Recurrent Unit (GRU)), and sequence length. Finding the optimal network hyperparameters is a significant challenge. Different methods exist in the literature for determining the optimal hyperparameters: MS, GS, RS, in addition to Bayesian Methods [4],[5].

2.1.2 Long Short Term Memory(LSTM)

Long-term short-term memory [20] is a gated neural network memory unit. It has three gates that govern the memory's contents. These gates are simple logistic functions of weighted sums, the weights of which may be learned through back propagation. This indicates that, despite its apparent complexity, the LSTM fits flawlessly into the neural network and its training process. Without special training or optimisation, it can learn what it needs to learn, retain what it needs to remember, and recall what it needs to recall.
Figure 2.2 demonstrates the the LSTM architecture visually.

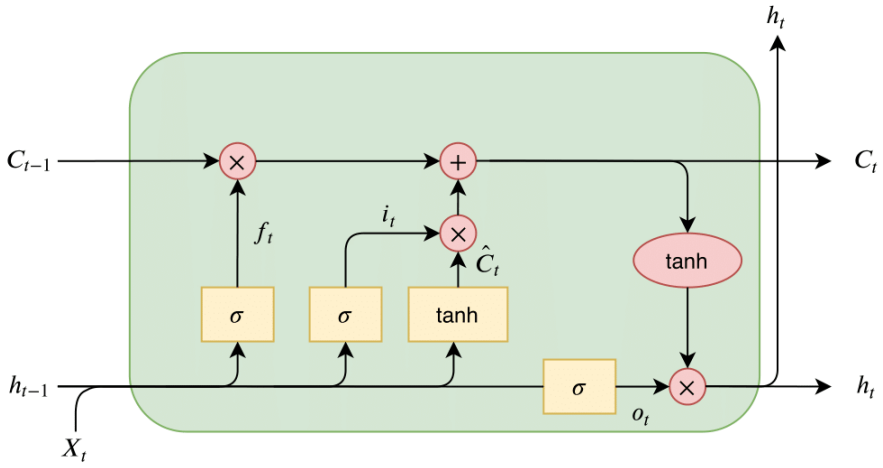


Figure 2.2: LSTM architecture [20]

Input gate i_t and forget gate f_t control the state of the cell c_t, which is the long-term memory. The output gate $o_t$generates the output vector or hidden state h_t, which is used to focus memory. This memory system enables the network to retain information for an extended period of time, which was severely lacking in traditional recurrent neural networks.

$$\begin{aligned} i_t &= \text{sigmoid}\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ f_t &= \text{sigmoid}\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ o_t &= \text{sigmoid}\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \\ h_t &= o_t \odot \tanh\left(c_t\right) \end{aligned} \tag{2.1}$$

2.1.3 Gated Recurrent Neural Network(GRU)

In essence, a gated recurrent unit is a simplified LSTM. It serves the same function in the network. The primary distinction lies in the number of gates and weights; GRU is somewhat less complex. There are 2 gates. Since there is no output gate, the memory content cannot be controlled. The reset gate r_t is inserted into the candidate activation while the update gate z_t regulates the flow of information from the previous activation and the addition of new information h_t. Overall, it is comparable to LSTM. From these differences alone, it is difficult to determine which option is superior for a given problem [9].

$$\begin{aligned} z_t &= \text{sigmoid}\left(W_z x_t + U_z h_{t-1} + b_z\right) \\ r_t &= \text{sigmoid}\left(W_r x_t + U_r h_{t-1} + b_r\right) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tanh\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \end{aligned} \tag{2.2}$$

2.1.4 Recurrent neural networks for forecasting

Both LSTM and GRU networks are frequently utilised for time series forecasting, despite the fact that this is most likely not their primary purpose. For the purpose of learning temporal distances, in [15] utilised LSTMs with peephole connections. LSTM networks that are layered were utilised by [30] in order to identify anomalies in time data. An adaptive gradient learning approach was proposed by [18] for RNNs. This method gives RNNs the ability to produce accurate predictions for time series that contain outliers as well as change points. [22] implemented autoencoder into LSTM in order to improve the model's performance in terms of forecasting. An extended attention mechanism was introduced by [10] in order to simulate missing values and capture intervals that occur in time series. Researchers [3] applied LSTMs to groupings of comparable time series that were discovered through the application of clustering algorithms. When it comes to forecasting unusual events, [25] used RNNs and discovered that neural networks might be a better option than traditional time series approaches when the number of time series, the length of the time series, and the correlation between the time series are all high. To account for informative gaps in multivariate time series, [8] developed a GRU-based model that included a decay mechanism. Several attempts have been made on better comprehending RNNs.[24] investigated the source of the performance of recurrent neural networks by conducting three performance comparisons and an error analysis on character-level language models. In the study from 2017, [16] employed a variety of datasets in order to visualise the activities of LSTMs. [17] evaluated the effectiveness of a number of different LSTM variants and discovered that the forget gate and the output activation function are the two components of an LSTM block that are most critical to its operation. A approach for ranking features was suggested by [7], and it makes use of variational dropout.
Several studies looked into the possibility of determining how to quantify the degree of uncertainty connected to the time series forecasts produced by recurrent neural networks. In [48] created uncertainty estimates by utilising Monte Carlo dropout in conjunction with an encoder-decoder structure comprised of LSTM units. The authors of the [6]study used bootstrapping to create prediction intervals for convolutional LSTMs. In this section, we are going to investigate several facets of RNN-based time series forecasting and present a framework for making meaningful forecasts that covers all components of the process from beginning to conclusion.

2.2 Transformer model

Vaswani first presented the transformer model in 2017 [40], and it has since garnered a lot of attention in the field of deep learning as a result of its outstanding successes in natural language processing, computer vision, and speech processing applications. Because of its forward-thinking architecture, significant progress has been made in a variety of fields, and a large number of transformer variants have been proposed to improve the performance of state-of-the-art devices in a variety of contexts.

Because transformers are able to accurately depict long-range dependencies and interactions in sequential data, this modelling technique has become increasingly popular in the field of time series modelling. In order to solve certain issues that are associated with time series modelling, such as forecasting, anomaly detection, and classification, researchers have investigated several iterations of the transformer technique. Time series data are characterised by a critical feature known as seasonality or periodicity. However, the topic of how to incorporate seasonality while simultaneously capturing long-range and short-range temporal correlations remains an unresolved research question.

The use of deep learning techniques for time series analysis has been the subject of a significant amount of research, with a particular emphasis on forecasting, classification, anomaly detection, and data augmentation. These research intend to construct models that are capable of effectively extracting patterns from time series data and making accurate predictions based on such patterns.

In this section, we will describe the architecture of the transformer model for time series forecasting and provide a quick introduction to the model itself. In this section, we will investigate how the transformer makes use of self-attention processes in order to capture dependencies that span several time steps, and we will analyse the introduction of extra components in order to solve the special issues that are presented by time series analysis. The purpose of this project is to develop a comprehensive understanding of the application of the transformer model in time series forecasting as well as its potential implications for other activities that are related.

2.2.1 Vanilla Transformer

The transformer model, in its fundamental form, adopts the encoder-decoder architecture commonly used in many neural sequence models. This architecture consists of multiple identical building blocks that are repeated in both the encoder and decoder components. Each building block comprises distinct modules that play a crucial role in capturing and processing the input sequence.

Within the decoder, each block incorporates cross-attention models, which enable the model to attend to different parts of the input sequence when generating each output token. This cross-attention mechanism allows the decoder to focus on relevant information from the input during the decoding process. The cross-attention module operates between the multi-head self-attention module and the position-wise feed-forward network, enhancing the model's ability to capture meaningful relationships and dependencies in the sequence.

Conversely, the encoder block consists of a multi-head self-attention module and a position-wise feed-forward network. The self-attention module is a key component that enables the model to analyze and understand the dependencies between different elements within the input sequence. It allows the model to assign varying levels of importance to different parts of the sequence during the encoding process, capturing both local and global contextual information. The position-wise

feed-forward network, on the other hand, applies a non-linear transformation to each position in the sequence independently, further enhancing the model's representation capabilities.

By utilizing the multi-head mechanism, the transformer model is able to perform attention operations in parallel across multiple heads, allowing it to capture different types of dependencies and patterns within the data. This parallel processing enhances the model's ability to handle complex and varied relationships within the input sequence.

Overall, the encoder-decoder structure, with its combination of self-attention and position-wise feed-forward networks, provides the transformer model with a powerful framework for capturing intricate dependencies and generating high-quality predictions. This architecture has proven to be highly effective in a wide range of applications, including natural language processing, computer vision, and now, time series forecasting. **Input Encoding and Positional Encoding**
The LSTM and RNN models feature recurrence, but the vanilla Transformer does not. Instead, it models the sequence information by utilising the positional encoding that was added in the embeddings of the input data.
Absolute Positional Encoding In vanilla Transformer, for each position index t, encoding vector is given by

$$PE(t)_i = \begin{cases} \sin(\omega_i t) & \text{if } i\%2 = 0 \\ \cos(\omega_i t) & \text{if } i\%2 = 1 \end{cases} \tag{2.3}$$

where ω_i is the hand-crafted frequency for each dimension. Another way is to learn a set of positional embeddings for each position which is more flexible [13]

Multi-head Attention
When utilizing the Query-Key-Value (QKV) paradigm, the scaled dot-product attention that Transformer requires is provided by

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{D_k}}\right)\mathbf{V} \tag{2.4}$$

where queries $\mathbf{Q} \in \mathcal{R}^{N \times D_k}$, keys $\mathbf{K} \in \mathcal{R}^{M \times D_k}$, values $\mathbf{V} \in \mathcal{R}^{M \times D_v}, N, M$ denote the lengths of queries and keys (or values), and D_k, D_v denote the dimensions of keys (or queries) and values. Transformer uses multi-head attention with H different sets of learned projections instead of a single attention function as:

$$\text{Multi-Head-Attn}\ (\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\ (\ \text{head}\ _1, \cdots, \text{head}\ _H)\, \mathbf{W}^O,$$
$$\text{where}\ _{\text{head}\ _i} = \text{Attention}\ \left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right).$$

Feed-forward and Residual Network
The feed-forward network is a fully connected module as:

$$FFN(\mathbf{H}') = \text{ReLU}(\mathbf{H}'\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2 \tag{2.5}$$

where $\mathbf{H}'$ is outputs of previous layer, $\mathbf{W}^1 \in \mathcal{R}^{D_m \times D_f}$, $\mathbf{W}^2 \in \mathcal{R}^{D_f \times D_m}, \mathbf{b}^1 \in \mathcal{R}^{D_f}, \mathbf{b}^2 \in \mathcal{R}^{D_m}$ are trainable parameters. In a deeper module, a residual connection module followed by a layer

normalization module is inserted around each module. That is,

$$\begin{aligned} \mathbf{H}' &= \text{LayerNorm}(\text{SelfAttn}(\mathbf{X}) + \mathbf{X}), \\ \mathbf{H} &= \text{LayerNorm}(FFN(\mathbf{H}') + \mathbf{H}'), \end{aligned} \tag{2.6}$$

where SelfAttn(.) denotes self-attention module and Layer Norm(.) denotes the layer normalization operation.

Encoder

The encoder is made up of a stack of four encoder layers that are all the same, in addition to an input layer and a layer for positional encoding. Through the use of a fully-connected network, the input layer converts the time series data that is provided to a vector of dimension of the model. In order for the model to make use of a multi-head attention mechanism, this step is absolutely necessary. By performing element-wise addition of the input vector with a positional encoding vector, positional encoding using sine and cosine functions is used to encode sequential information in the time series data. This is accomplished by using sine and cosine functions. The generated vector is then inputted into four different levels of encoders. Each encoder layer has two sub-layers: a self-attention sub-layer and a fully-connected feed-forward sub-layer. These sub-layers work together to form the encoder layer. A normalisation layer comes after each sub-layer that's been added. The decoder receives the output of the encoder, which is a vector of the dimension of the model.

Decoder

We utilise a decoder design that is conceptually comparable to the architecture of the first Transformer. In addition, the decoder has an input layer, four identical decoder layers, and an output layer in its construction. The last data point of the encoder input is carried over to the beginning of the decoder input. The decoder input is mapped by the input layer to a vector that is in the model dimension. The decoder adds a third sub-layer to the encoder output in order to apply self-attention techniques on top of the encoder output. This is in addition to the two sub-layers that are already present in each encoder layer. In the end, there is a layer known as the output layer that is responsible for mapping the output of the most recent decoder layer to the desired time sequence. We assure that the prediction of a time series data point will only depend on the data points that came before it by using a technique called look-ahead masking and an offset of one position between the decoder input and the target output in the decoder.

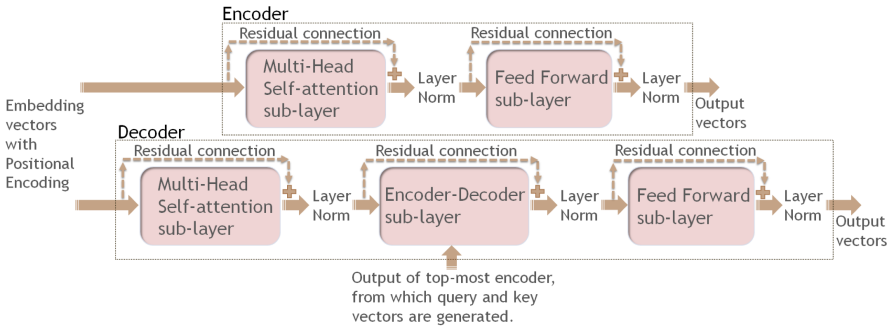


Figure 2.3: Encoder/decoder block [1]

In Figure 2.3 each encoder block uses multi-head self-attention and feed-forward layers with

residual connections around each layer and layer normalisation. Decoder blocks are encoder blocks with an encoder-decoder attention layer. The uppermost encoder block generates the key and value vectors for this encoder-decoder attention layer. The multi-head self-attention layer before the encoder-decoder layer generates query vectors.

2.2.2 Road Map of Transformers for Time-Series Analysis

Since its inception in 2017, the Transformer model has undergone significant advancements and adaptations to address the unique challenges posed by time-series analysis. While the original Transformer architecture was primarily designed for natural language processing tasks, researchers have explored its potential in various domains, including time-series analysis, leading to tailored versions of the model.

Initially, the Transformer model did not incorporate specific components for handling time-series data. However, as the demand for effective time-series analysis grew, researchers began customizing and enhancing the Transformer architecture to address the unique characteristics of time-series data. This evolution has resulted in notable improvements in performance and the development of a road map for applying Transformers to time-series tasks.

One important application of Transformers in time-series analysis is classification. Researchers have explored different strategies to adapt the Transformer architecture for accurate classification of time-series data. By incorporating appropriate modifications, such as utilizing positional encoding and self-attention mechanisms, Transformers have shown promising results in classifying time-series sequences into different categories [44].

Another significant area where Transformers have been applied is time-series analysis itself. Researchers have investigated the suitability of the Transformer architecture for tasks such as forecasting, anomaly detection, and semantic segmentation of time-series data. Various modifications and enhancements have been proposed to enable Transformers to capture temporal dependencies and patterns effectively in time-series sequences [39, 46].

As illustrated in Figure 2.4, the original architecture of the Transformer model can serve as a starting point for time-series analysis. However, specific modifications are required to adapt the model for time-series forecasting. These modifications include adjustments to the input representation, incorporation of temporal context, and optimization of loss functions to account for the sequential nature of time-series data.

To achieve accurate time-series forecasting, it is crucial to design the Transformer architecture in a way that preserves temporal information and captures long-term dependencies. Techniques such as positional encoding, attention mechanisms, and hierarchical modeling have been explored to address these requirements. By integrating these modifications into the Transformer model, researchers have achieved significant advancements in time-series forecasting accuracy.

In summary, the road map of Transformers for time-series analysis demonstrates the continuous evolution and customization of the Transformer architecture to tackle the unique challenges posed by time-series data. By adapting the original model and incorporating domain-specific modifications, researchers have made significant progress in applying Transformers to various time-series tasks. The subsequent sections will delve into the specific changes and design considerations necessary to leverage the power of the Transformer model for time-series forecasting.

Positional Encoding

Encoding the placements of time series in the input to transforms is of the utmost necessity due to the fact that the order in which time series are given is crucial. To begin, positional information is typically encoded in the form of vectors before being afterwards injected into the model as an additional input alongside the input time series. This is a popular design. When modelling time series with Transformers, the process of obtaining these vectors can be broken down into three

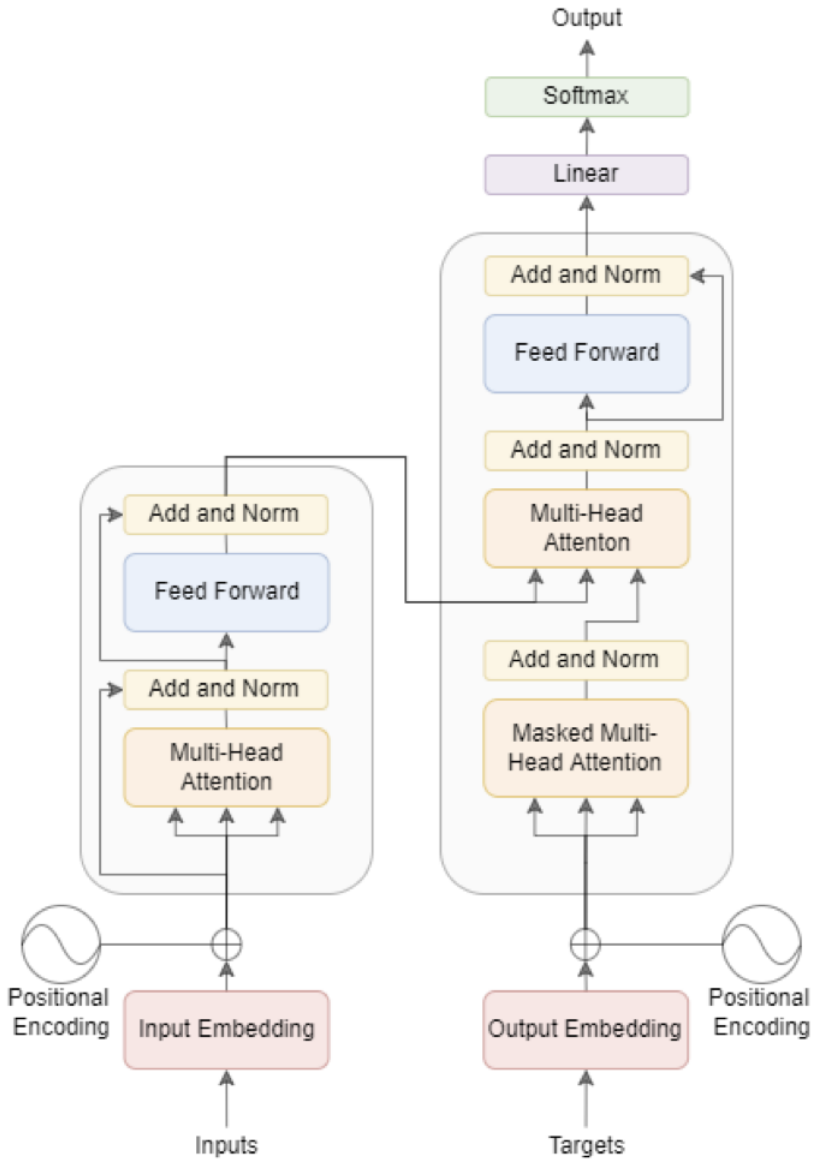


Figure 2.4: recreation of the original Transformer architecture [1]

primary categories depending on which approach is taken.

- **Vanilla Positional Encoding.**
  A select number of the works Simply by providing the vanilla positional encoding that was applied in [27], which is afterwards appended to the input time series embeddings and delivered to Transformer, is the first step in the process. They were unable to completely leverage the crucial properties of the time series data and, as a result, this method, while it can extract some positional information from time series, was not successful overall.

- **Learnable Positional Encoding**
  Several studies demonstrated that learning appropriate positional embeddings from time series data can be far more effective than using the vanilla positional encoding. This is due to the fact that the vanilla positional encoding is hand-crafted and is less expressive and adaptive.

Learned embeddings are more adaptable and flexible than fixed vanilla positional encoding, which means they can be used for a wider variety of purposes. In [45] adds an embedding layer to Transformer, which discovers embedding vectors for each position index in conjunction with the other parameters of the model. Also it is possible to encodes positional embeddings using an LSTM network, which is able to better leverage sequential ordering information in time series [28].

- **Timestamp Encoding**
  When modelling time series in real-world scenarios, the timestamp information is typically accessible. This includes calendar timestamps (for example, seconds, minutes, hours, weeks, months, and years) as well as special timestamps (for example, holidays and events). These timestamps can be quite important in real-world applications, but in Transformers' default implementation, they are rarely used. Informer advocated employing learn able embedding layers in order to encode timestamps as additional positional encoding as a means of finding a solution to the problem [47].

Attention Module

The self-attention module plays a crucial part in Transformer. It is possible to think of it as a completely connected layer with weights that are dynamically created depending on the pairwise similarity of input patterns. This interpretation is possible since it is possible to perceive it as a fully connected layer. As a consequence of this, it has the same maximum path length as completely linked layers, but it has a significantly lower number of parameters, which enables it to be utilised for the modelling of long-term dependencies. Since the self-attention module in the vanilla Transformer has a time and memory complexity of $\mathcal{O}\left(N^2\right)$ (where N is the length of the input time series), this module becomes the computational bottleneck when working with lengthy sequences, as we explain in the preceding section. Many efficient Transformers were proposed to reduce the quadratic complexity that can be classified into two main categories: (1) explicitly introducing a sparsity bias into the attention mechanism like LogTrans and Pyraformer, (2) exploring the low-rank property of the self-attention matrix to speed up the computation, e.g. Informer and FEDformer . In Table 2.1, we compare the time and memory requirements of several well-known transformers when they are used for modelling time series [42].

Table 2.1: Comparison of Methods

Methods	Training		Testing
	Time	Memory	Steps
Transformer	$\mathcal{O}(N^2)$	$\mathcal{O}(N^2)$	N
LogTrans	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Informer	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Autoformer	$\mathcal{O}(N \log N)$	$\mathcal{O}(N \log N)$	1
Pyraformer	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Quatformer	$\mathcal{O}(2cN)$	$\mathcal{O}(2cN)$	1
FEDformer	$\mathcal{O}(N)$	$\mathcal{O}(N)$	1
Crossformer	$\mathcal{O}\left(\frac{D}{L_{seg}^2} N^2\right)$	$\mathcal{O}(N)$	1

Architecture-based Attention Innovation

Several studies aim to remodel Transformers on the architectural level in order to make room for specific modules inside the framework of the Transformers model while modelling time series. In recent work, hierarchical architecture has been incorporated into Transformer in order to take into consideration the multi-resolution property of time series. The algorithm Informer uses to down-sample series into its half slice inserts max-pooling layers with stride 2 between attention blocks. Pyraformer constructs a C-ary tree-based attention mechanism. In this attention mechanism, nodes at the finest scale correspond to the original time series, while nodes at the coarser scales reflect series at lower resolutions. In order to more accurately reflect temperature dependence over a range of resolutions, Pyraformer established both intra-scale and inter-scale attentions. A hierarchical architecture has many benefits, including the capacity to integrate information at multiple multi-resolutions, as well as the benefits of efficient computation, particularly for long-time series. Several studies aim to remodel Transformers on the architectural level in order to make room for specific modules inside the framework of the Transformers model while modelling time series. In recent work, hierarchical architecture has been incorporated into Transformer in order to take into consideration the multi-resolution property of time series. The algorithm Informer uses to down-

sample series into its half slice inserts max-pooling layers with stride 2 between attention blocks. Pyraformer constructs a C-ary tree-based attention mechanism. In this attention mechanism, nodes at the finest scale correspond to the original time series, while nodes at the coarser scales reflect series at lower resolutions. In order to more accurately reflect temperature dependence over a range of resolutions, Pyraformer established both intra-scale and inter-scale attentions. A hierarchical architecture has many benefits, including the capacity to integrate information at multiple multi-resolutions, as well as the benefits of efficient computation, particularly for long-time series [42].

2.3 Wavelet Transform

In recent decades, there has been a discernible uptick in interest, across the board, in the study of non-stationary time series across a variety of different scientific subfields. The capacity to extract distinct components from non-stationary time series, such as seasonal, trend, and abrupt components, using one of several decomposition methods that have been developed has led to an improved understanding of the temporal variability. The wavelet transform is one of these methods, and because it has been shown to be successful in applications across a wide variety of research domains, it has become an essential instrument for decomposing non-stationary time series into the time-frequency domain.

The wavelet transform makes it possible to investigate how frequency shifts occurred over the course of time, as opposed to the conventional Fourier analysis, which only provides information on frequency without any temporal localization. Due to this feature, it is very well-suited for the analysis of non-stationary data. The utilisation of wavelets, which are tiny functions that may be scaled and translated to analyse different components of a time series, is the fundamental principle that underpins the wavelet transform. These wavelets have qualities that allow them to localise in time and frequency, which enables them to provide a more accurate representation of signals that exhibit time-varying characteristics.

When the wavelet transform is applied to a time series, it is feasible to extract different frequency components at varying scales. This is made possible by the wavelet transform. This decomposition gives a multi-resolution representation, with high-frequency components capturing minute details and low-frequency components capturing coarse trends or patterns respectively.

The wavelet transform is useful in a wide variety of contexts, including signal processing and image analysis, as well as finance, geophysics, and biomedical engineering, amongst others. It has proven to be very useful in circumstances in which the data display non-stationary behaviour or contain elements that are fleeting.

In this chapter, our goal is to present a condensed overview of the wavelet transform method, which will then be followed by an in-depth investigation of the approach's applications and consequences in relation to time series analysis. In this lesson, we are going to dig into several aspects of wavelet decomposition, such as the selection of wavelet functions, the calculation of decomposition levels, and the interpretation of components that are produced.

In addition, we will talk about the benefits and drawbacks of using the wavelet transform to capture the time-frequency features of a non-stationary time series. In order to evaluate its efficacy in relation to the performance of alternative decomposition strategies, a comparative study will be

carried out. In addition, we will highlight essential issues to keep in mind when applying the wavelet transform to datasets derived from the real world.

In general, the wavelet transform is a useful method that can be applied to the analysis of non-stationary time series as well as the extraction of relevant information from complicated data. Researchers and practitioners are able to improve their understanding of the underlying dynamics of time-varying signals thanks to this capability's capacity to provide a thorough representation of the time-frequency domain.

2.3.1 Wavelet transform literature review

The wavelet transform is a recently developed mathematical tool for signal analysis. It has been applied successfully in astronomy, data compression, signal and image processing, earthquake prediction and so on.

In wavelet analysis, the first step is to choose a suitable wavelet to act as the "mother wavelet," after which an analysis is carried out making use of the wavelet's translated and dilated forms. The term "mother wavelet" can refer to a number of different types of wavelets, including the Harr wavelet, the Meyer wavelet, the Coiflet wavelet, the Daubechies wavelet, the Morlet wavelet, and many others. These wavelets have different specificities. In this thesis, the Daubechies wavelet is applied.

Similar to the Fourier transform, there are different wavelet transforms, called continuous wavelet transform (CWT), also known as integral wavelet transform (IWT), discrete wavelet transform (DWT), and fast discrete wavelet transform (FWT). The fast wavelet transform is known as multiresolution analysis (MRA) or Mallat pyramidal algorithm.

For a given square integrable (or finite energy) function (or signal) $f(t)$, its continuous wavelet transform is defined by,

$$(\mathrm{Wf})(a,b) = a^{-1/2} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-b}{a}\right) dt \tag{2.7}$$

where $\psi(t)$ is the mother wavelet. The value of the wavelet transform $(\mathrm{Wf})(a,b)$ is called the wavelet coefficient, which stands for the similar degree between the signal and the wavelet at the translation b and the dilation a. Translation means the time shift, and dilation means the time scale. In other words, it means how many components of the wavelet at dilation a are included in the original signal at translation b. Through a simple mathematical map, formulation (1) can be re-written as a convolution of signal $f(t)$ and the wavelet at scale a. So the wavelet plays a role of band-pass filter (the band corresponds to the scale). For computer implementation, the discrete wavelet transform (DWT) is the most commonly used. Similar to the fast Fourier transform (FFT), there is a fast algorithm for the DWT, known as the fast DWT or Mallat and Daubechies' pyramidal algorithm. First, an original discrete signal $c_0[n]$ is decomposed into two components, $c_1[n]$ and $d_1[n]$ by a low-pass filter $h[n]$ and a high-pass filter $g[n]$, respectively. The transform is an orthogonal decomposition of the signal. The $c_1[n]$, named the approximation of the signal, contains the low frequency components of the signal $c_0[n]$, and the $d_1[n]$, named the detail of the signal, is associated with the high frequency components of the $c_0[n]$. Then, the approximation $c_1[n]$ is again decomposed into a new approximation, $c_2[n]$ and a detail $d_2[n]$ by a bigger scale and continuing to a third scale, fourth scale and so on, according to the application. The original signal, $c_0[n]$, can also

be reconstructed by these approximations and details. Fig. 1(a) and (b) illustrate the decomposing and reconstructing processes, respectively.

Both the Fourier transforms and wavelet transforms are domain transform functions, but one of the disadvantages of Fourier transforms is that frequency information can only be extracted for the complete duration of a signal $f(t)$. If, at some point in the life time of $f(t)$, there is a local oscillation representing a particular feature, this will contribute to the calculated Fourier transform $f(w)$, but its location on the time axis will be lost. There is no way of judging whether the value of $f(w)$ at a particular w is derived from a frequency present throughout the life of $f(t)$ or during just one or a few selected periods. Another disadvantage is the phase shift.

These disadvantages can be overcome by the wavelet transform. For the windowed Fourier transform, the window width is always ®xed, and the window is a square shape. An important advantage of the wavelet transform is that the window widths of the wavelet transform can be adjusted automatically. At low frequency, the window widths are longer, while at high frequency, the window widths are shorter. The shorter the window width, the better is the resolution. This means that the wavelet transform can provide better time resolution for high frequency components of a signal and better frequency resolution for low frequency components of the signal. Meanwhile, because of the shorter time window, the phase shift is hardly observed. These are the reasons why the wavelet transform is applied in this paper.

A signal can be decomposed by the WT on a variety of timescales. It is defined as an ensemble of basic functions known as $\psi_{a,b}(t)$, which can be produced by translating and scaling the so-called mother wavelet in the manner described below [12].

$$\psi_{a,b} = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \quad \text{where} \quad a > 0, \quad -\infty < b < \infty \tag{2.8}$$

where a represents the scale parameter and b determines the wavelet's position. This paper describes the most prevalent WT in the scientific literature. In practise, there are various WT varieties that employ numerous mother wavelets. These are summarised below.

2.3.2 Continuous Wavelet Transform (CWT)

In the realm of non-stationary signal analysis, the Continuous Wavelet Transform (CWT) has emerged as a powerful tool for capturing the dynamic characteristics of signals. Unlike the Fourier transform, which provides a static frequency representation of a signal, the CWT offers a time-frequency representation that allows for the detection of transient features and localized variations in frequency content.

The CWT can be mathematically defined as follows:

$$W_f(a,b) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,b}(t)dt = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} \psi^*_{a,b}\left(\frac{t-b}{a}\right) f(t)dt, \tag{2.9}$$

where $W_f(a,b)$ represents the wavelet coefficient at scale a and position b for the signal $f(t)$. The wavelet mother conjugate $\psi^*_{a,b}(t)$ serves as the analyzing function that is scaled and shifted to capture different features of the signal. The CWT generates a time-frequency representation by continuously adjusting the scale parameter a and the position parameter b, enabling the extraction of wavelet coefficients for different segments of the signal [31, 34].

The wavelet coefficients obtained from the CWT can be thought of as the building blocks of the signal, as they represent the signal's decomposition into its component parts. Each coefficient is multiplied by the corresponding wavelet function, which has been scaled and shifted, resulting in the reconstruction of the original signal. By analyzing the magnitude and phase information of the wavelet coefficients, one can gain insights into the localized frequency content and time-varying features of the signal.

In practice, the "Morlet" wavelet function is widely used as the mother wavelet for the CWT. The Morlet wavelet exhibits equal variance in both time and frequency domains, making it suitable for capturing features with comparable time and frequency scales. Its complex exponential form allows for the simultaneous analysis of both the amplitude and phase information of the signal, providing a more comprehensive representation of the signal's time-frequency characteristics.

By employing the CWT, researchers and practitioners can effectively analyze non-stationary signals and explore their dynamic properties. The CWT's ability to capture localized variations in frequency content and transient features makes it valuable in various fields, including biomedical signal processing, audio and speech analysis, image processing, and geophysical data analysis. Moreover, the CWT's flexibility in choosing different wavelet functions and adjusting the scale and position parameters allows for customized analyses tailored to specific signal characteristics.

2.3.3 Discrete wavelet transform

The Discrete Wavelet Transform (DWT) is a variant of the wavelet transform that is specifically designed to operate on discrete or sampled signals [12]. Unlike the Continuous Wavelet Transform (CWT), which operates on continuous signals, the DWT decomposes a signal into a set of discrete wavelet scales and translations. The key distinction between the DWT and the CWT lies in the level of detail achieved in the decomposition. In the DWT, the signal is decomposed into a group of mutually orthogonal wavelets, whereas the CWT provides a continuous time-frequency representation.

The DWT operates on a dyadic grid, where the mother wavelet is scaled by a factor of two ($a = 2^j$) and translated by an integer ($b = k2^j$). Here, k is a location index ranging from 1 to $2^{-j}N$ (where N is the total number of observations), and j ranges from 0 to J (representing the total number of scales) [34].

Mathematically, the DWT is expressed as follows:

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k) \tag{2.10}$$

where $\psi_{j,k}(t)$ represents the wavelet function at scale j and translation k. The DWT coefficients, denoted as $W_{j,k}$, are obtained by evaluating the inner product of the signal $f(t)$ with the scaled and translated wavelet functions:

$$W_{j,k} = W(2^j, k2^j) = 2^{-j/2} \int_{-\infty}^{\infty} f(t)\overline{\psi(2^{-j}t - k)}dt \tag{2.11}$$

To reconstruct the original signal (or its parts) from the DWT coefficients $W_{j,k}$, an Inverse

Discrete Wavelet Transform (IDWT) is applied:

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} W_{j,k} \psi_{j,k}(t) \tag{2.12}$$

The DWT can be employed with various mother wavelets, including the Haar, Daubechies, Biorthogonal, Symlet, Meyer, and Coiflets wavelets. The Haar wavelet is the simplest and earliest known mother wavelet, while the Daubechies wavelet is a family of orthogonal wavelets, with different members labeled by the parameter N. The Symlet and Coiflet wavelets can be considered as modified versions of the Daubechies wavelet, with the Symlet wavelet exhibiting greater symmetry. The Coiflet wavelet, on the other hand, introduces six scaling and wavelet function coefficients, resulting in a smoother analysis. Lastly, the biorthogonal wavelet family encompasses semi-orthogonal, biorthogonal, and non-orthogonal wavelet bases. It employs two distinct wavelet functions and two scaling functions, enabling symmetric wavelets and supporting unique multi-resolution analyses.

By utilizing the DWT, researchers and practitioners can efficiently decompose signals into their constituent wavelet components, facilitating the analysis of localized features and capturing signal characteristics at different scales. The choice of the mother wavelet depends on the specific requirements of the signal and the desired properties of the wavelet transform. The DWT has found extensive applications in areas such as signal and image processing, data compression, denoising, and feature extraction.

in this section, the theory, properties, and applications of the Discrete Wavelet Transform (DWT) were explored in detail. Different mother wavelets, including the Haar, Daubechies, Symlet, Coiflet, and Biorthogonal wavelets, were investigated to analyze their strengths and weaknesses in handling various types of signals. The practical aspects of implementing the DWT, including the utilization of fast algorithms such as the Mallat and Daubechies' pyramidal algorithms, were examined.

Furthermore, advanced topics related to the DWT, such as wavelet packet decomposition, wavelet thresholding, and time-frequency analysis, were delved into. The effectiveness and versatility of the DWT in various signal processing applications were demonstrated through case studies and experiments.

By deepening the understanding of the DWT, contributions were made to the existing knowledge in the field of wavelet analysis, and valuable insights were provided to researchers and practitioners. The aim of this exploration was to inspire new developments, applications, and innovations in signal processing and to promote the utilization of the DWT as a powerful tool in different domains.

Chapter 3

Methodology

In recent years, there has been a tremendous increase in the quantity of time series data that is available across academic and industrial fields. Because of this widespread pattern, there is a pressing want for reliable and efficient models that are able to do time series analysis. Although a number of different deep neural network models have been used in time series analysis, there is still a significant problem in addressing the frequency information contained within the data in an appropriate manner. This chapter addresses this shortcoming by presenting a wavelet-based neural network structure called the multilevel Wavelet Decomposition Network (mWDN). The purpose of the mWDN is to develop deep learning models with the ability to recognise the frequency characteristics that are present in time series data. The mWDN keeps the benefit of frequency learning while enabling the fine-tuning of all parameters through the integration of multilevel discrete wavelet decomposition within the framework of a deep neural network.

This chapter creates additional multi-frequency deep learning models based on the basis of mWDN in order to make time series forecasting easier. These models are built on top of mWDN. These models take as input either the complete mWDN deconstructed sub-series over all frequencies or only a subset of it, and then use the backpropagation approach to learn all parameters globally. The models are given additional capabilities for analysing time series data as a result of this smooth integration of wavelet-based frequency analysis into deep learning frameworks.

In order to test the effectiveness of our time series models that are based on mWDN, a large number of experiments have been run on 10 different datasets, some of which include data on the volume of real-world users. The findings provide evidence of their remarkable performance in a variety of contexts. In addition to that, this chapter presents a technique to importance analysis that is capable of being utilised with models that are based on mWDN. The most relevant time-series elements and mWDN layers may be effectively identified using this method, which is useful for time series analysis. The addition of this important analysis approach constitutes a full investigation into interpretable deep learning and demonstrates the interpretability advantage afforded by mWDN.

The capacity of this study to fill the void in frequency-aware deep learning models for time series analysis is what gives it its prominence as a body of work. The mWDN models offer a more thorough understanding of the frequency dynamics contained within time series data by integrating wavelet decomposition and deep learning. This has significant repercussions for fields such as banking, climate research, and the processing of sensor data, all of which are examples of areas where frequency plays an essential part. The fact that mWDN may be interpreted in a variety of

ways adds even more value to the algorithm by illuminating the underlying components and layers that are responsible for generating the model's predictions.

We are going to go into the methodology, experimental setting, and outcomes of our investigation in the following parts. We will exhibit the interpretability produced by the significance analysis methodology and highlight the performance of the mWDN models in comparison to other existing methods. The findings of this research point in promising avenues for the advancement of time series analysis, for overcoming the frequency information gap, and for unleashing the potential of interpretable deep learning in a variety of different areas.

3.1 Time/Frequency domain

A time series is a set of data points that are ordered according to the passage of time. Time-domain methods and frequency-domain methods are the two categories that can be used to describe the various approaches to analysing time series. Methods that operate in the time domain examine the correlations between the points in a time series by viewing it as a series of ordered points. In frequency-domain approaches, transform algorithms like the discrete Fourier transform and the Z-transform are used to convert a time series into a frequency spectrum. This frequency spectrum may then be used to analyse the original time series by employing the frequency spectrum's properties. In recent years, with the rise of the deep learning idea, several types of deep neural network models have been introduced to time series analysis and have reached state-of-the-art results in a wide variety of real-life applications [36]. This has been accomplished in a number of different ways. Recurrent Neural Networks (RNN) [43] and Long Short-Term Memory (LSTM) [20] in addition to transformer model are three well-known models that use memory nodes to represent correlations of series points. These models are examples of well-known models. The majority of these models are classified as time-domain approaches; however, they do not utilise the frequency information of a time series in any way, despite the fact that some begin to consider this information in indirect ways [11].

Wavelet decomposition is a well-known method that may be used to capture characteristics of time series in both the frequency and time domains.. These approaches are discussed in chapter 4, "Wavelet Decomposition." Intuitively, we are able to make use of them as tools for feature engineering for the data pre-processing that comes before deep modelling. The performance of raw neural network models could be improved by using this loose coupling method; nevertheless, these models are not globally optimised with independent parameter inference procedures. There is still a lot of effort to be done to figure out how to incorporate wavelet transforms into the overall structure of deep learning models.
To construct frequency-aware deep learning models for time series analysis, this strategy will propose a wavelet-based neural network structure, which will be given the term multilevel Wavelet Decomposition Network (mWDN). mWDN can deconstruct a time series into a group of sub-series with frequencies ordered from high to low, which is essential for collecting frequency factors for deep learning and is similar to the conventional Multilevel Discrete Wavelet Decomposition (MDWD) model [32]. mWDN can also breakdown a time series into a group of sub-series with frequencies ranked from high to low. Unlike MDWD, which has set parameters, mWDN's parameters can all

be fine-tuned to fit the training data of a variety of learning tasks. This is a key difference between the two. To put it another way, mWDN has the potential to benefit from both the wavelet-based time series decomposition and the capability of deep neural networks to learn. For the purpose of time series forecasting, two deep learning models, also known as multi-frequency deep learning models, have been created using mWDN. In time series forecasting, the most important challenge is determining how to predict the future states of a time series based on patterns that are buried in the data at different frequencies. As a result, the models put all of the mWDN deconstructed sub-series at high frequencies through an individual deep learning model, and then they ensemble all of the outputs for the final predicting. It is important to keep in mind that the back propagation technique is used to train all parameters, including the ones in the mWDN, in an end-to-end fashion. In this way, the wavelet-based frequency analysis can be integrated into the frameworks of deep learning without any problems. the models were assessed using real-world user volume time series datasets in order to do time series forecasting. The results show that mWDNs with trainable parameters have benefits over state-of-the-art baselines and illustrate the superiority of these networks over such baselines. We further propose a significance analysis technique to mWDN-based models, which successfully identifies those time-series elements and mWDN layers that are crucially significant to the success of time series analysis. This is a nice try for interpretable deep learning, and it is one of the things that we present as part of our effort to make deep learning more accessible. This demonstrates that mWDN has an advantage in terms of interpretability because it integrates wavelet decomposition for frequency factors.

3.2 Model

Throughout this section, lowercase letters, such as a and b, are used to signify scalars; bold lowercase letters, such as a and b, are used to denote vectors; bold uppercase letters, such as A and B, are used to denote matrices; and uppercase letters, such as A and B, are used to denote constants.

3.2.1 Multilevel Discrete Wavelet Decomposition

A wavelet-based discrete signal analysis method known as multilevel Discrete Wavelet Decomposition (MDWD) [32] allows for the extraction of multilevel time-frequency features from a given time series. This technique achieves this by decomposing the time series into low and high frequency sub-series at each level.

Let us denote the input time series as $\mathbf{x} = \{x_1, \ldots, x_t, \ldots, x_T\}$, and the low and high sub-series generated at the i-th level as $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$, respectively. At each subsequent level, MDWD utilizes a low pass filter $\mathbf{l} = \{l_1, \ldots, l_k, \ldots, l_K\}$ and a high pass filter $\mathbf{h} = \{h_1, \ldots, h_k, \ldots, h_K\}$, where $K \ll T$. These filters are used to convolve the low-frequency sub-series from the previous level, resulting in:

$$\begin{aligned} a_n^l(i+1) &= \sum_{k=1}^{K} x_{n+k-1}^l(i) \cdot l_k, \\ a_n^h(i+1) &= \sum_{k=1}^{K} x_{n+k-1}^l(i) \cdot h_k, \end{aligned} \tag{3.1}$$

where $x_n^l(i)$ represents the n-th element of the low-frequency sub-series at the i-th level, and $\mathbf{x}^l(0)$ is set as the input series. The resulting low and high frequency sub-series, denoted as $\mathbf{x}^l(i)$ and $\mathbf{x}^h(i)$, respectively, are obtained through down-sampling by a factor of 1/2 from the intermediate variable sequences $\mathbf{a}^l(i) = \{a_1^l(i), a_2^l(i), \ldots\}$ and $\mathbf{a}^h(i) = \{a_1^h(i), a_2^h(i), \ldots\}$.

The set of sub-series, $\mathcal{X}(i) = \{\mathbf{x}^h(1), \mathbf{x}^h(2), \ldots, \mathbf{x}^h(i), \mathbf{x}^l(i)\}$, represents the decomposed results at the i-th level of $\mathbf{x}$. It satisfies the following properties:

1. $\mathbf{x}$ can be fully reconstructed from $\mathcal{X}(i)$.
2. The frequency decreases as we move from high to low, starting with $\mathbf{x}^h(1)$ and ending with $\mathbf{x}^l(i)$.
3. The time and frequency resolutions of $\mathcal{X}(i)$ vary depending on the level of decomposition. As i increases, the frequency resolution improves, while the time resolution decreases, particularly for the low-frequency sub-series.

MDWD is widely regarded as a time-frequency decomposition technique because the sub-series in $\mathcal{X}$, with varying frequencies, preserve the same order information as the original series $\mathbf{x}$, enabling the analysis of time-frequency characteristics within the data.

3.2.2 Multilevel Wavelet Decomposition Network

In this section, we propose the Multilevel Wavelet Decomposition Network (mWDN), which aims to approximate the implementation of Multilevel Discrete Wavelet Decomposition (MDWD) within the framework of a deep neural network.

The structure of the mWDN is illustrated in Figure 3.1. This model leverages two functions to perform a hierarchical decomposition of a given time series, as depicted in the diagram.

$$\begin{aligned} \mathbf{a}^l(i) &= \sigma\left(\mathbf{W}^l(i)\mathbf{x}^l(i-1) + \mathbf{b}^l(i)\right), \\ \mathbf{a}^h(i) &= \sigma\left(\mathbf{W}^h(i)\mathbf{x}^l(i-1) + \mathbf{b}^h(i)\right), \end{aligned} \tag{3.2}$$

where $\sigma(\cdot)$ represents a sigmoid activation function. The trainable parameters $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$ are weight matrices, while $\mathbf{b}^l(i)$ and $\mathbf{b}^h(i)$ are trainable bias vectors, which are initialized as close-to-zero random values.

The first function computes the low-frequency sub-series $\mathbf{a}^l(i)$ at the i-th level by applying the weight matrix $\mathbf{W}^l(i)$ to the previous low-frequency sub-series $\mathbf{x}^l(i-1)$, and then adding the bias vector $\mathbf{b}^l(i)$. The sigmoid activation function introduces non-linearity into the process.

Similarly, the second function calculates the high-frequency sub-series $\mathbf{a}^h(i)$ at the i-th level. It employs the weight matrix $\mathbf{W}^h(i)$ to transform the previous low-frequency sub-series $\mathbf{x}^l(i-1)$, and then incorporates the bias vector $\mathbf{b}^h(i)$. Again, the sigmoid activation function adds non-linearity to the computation.

By iteratively applying these functions, the mWDN conducts a hierarchical decomposition of the time series, capturing both low and high-frequency components at each level. This allows the model to learn and extract multi-level time-frequency features from the input data, mimicking the essence of MDWD within a deep neural network framework.

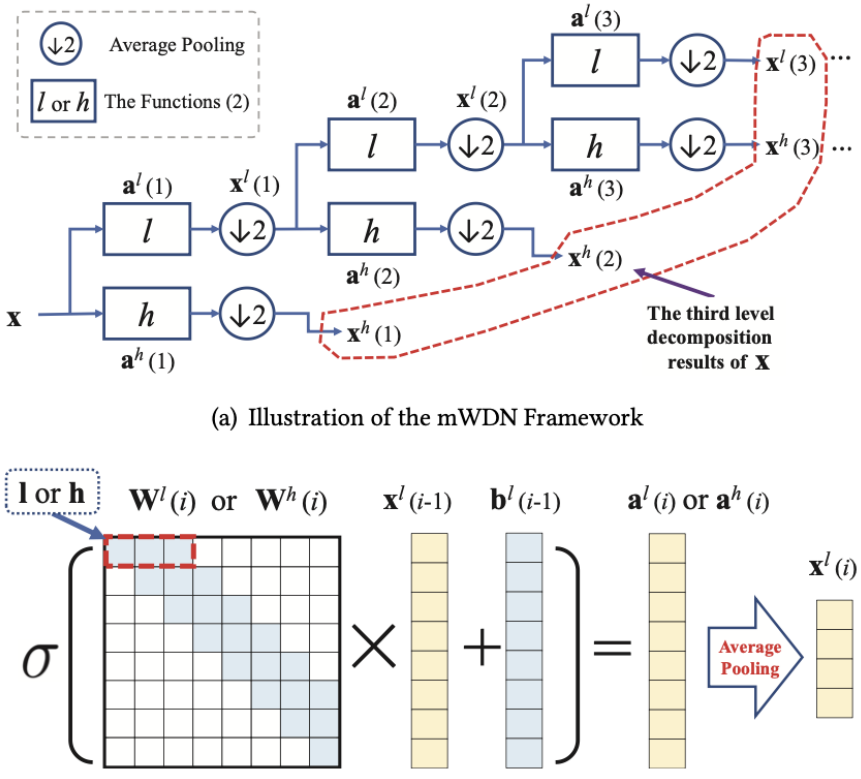


Figure 3.1: The mWDN framework [41]

The low-frequency sub-series $\mathbf{x}^l(i)$ and high-frequency sub-series $\mathbf{x}^h(i)$ can also represent the down-sampled versions of the intermediate variables $\mathbf{a}^l(i)$ and $\mathbf{a}^h(i)$, respectively. This downsampling is typically achieved using an average pooling layer, where each element $x_j^l(i)$ of $\mathbf{x}^l(i)$ is computed as the average of adjacent elements $a_{2j}^l(i)$ and $a_{2j-1}^l(i)$:

$$x_j^l(i) = \frac{a_{2j}^l(i) + a_{2j-1}^l(i)}{2}. \tag{3.3}$$

This downsampling process reduces the dimensionality of the sub-series while preserving the essential frequency information.

To implement the convolution operation described earlier, the weight matrices $\mathbf{W}^l$ and $\mathbf{W}^h$ can be initialized as follows:

$$\mathbf{W}^l(i) = \begin{bmatrix} l_1 & l_2 & l_3 & \cdots & l_K & \epsilon & \cdots & \epsilon \\ \epsilon & l_1 & l_2 & \cdots & l_{K-1} & l_K & \cdots & \epsilon \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & l_1 & \cdots & l_{K-1} & l_K \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & l_1 & l_2 \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & \epsilon & l_1 \end{bmatrix},$$
$$\mathbf{W}^h(i) = \begin{bmatrix} h_1 & h_2 & h_3 & \cdots & h_K & \epsilon & \cdots & \epsilon \\ \epsilon & h_1 & h_2 & \cdots & h_{K-1} & h_K & \cdots & \epsilon \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & h_1 & \cdots & h_{K-1} & h_K \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & h_1 & h_2 \\ \epsilon & \epsilon & \epsilon & \cdots & \cdots & \cdots & \epsilon & h_1 \end{bmatrix}. \tag{3.4}$$

Obviously, The weight matrices $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$ are both of size $\mathbb{R}^{P\times P}$, where P represents the dimensionality of $\mathbf{x}^l(i-1)$. To initialize these weight matrices, random values ϵ are generated such that $|\epsilon| \ll |l|$ for all $l \in \mathbf{l}$, and $|\epsilon| \ll |h|$ for all $h \in \mathbf{h}$. It is important to ensure that these random values are small compared to the filter coefficients.

In our implementation, we utilize the Daubechies 4 Wavelet [37], which has specific filter coefficients:

$$\begin{aligned} \mathbf{l} =& -0.0106, 0.0329, 0.0308, -0.187, -0.028, 0.6309, 0.7148, 0.2304, \\ \mathbf{h} =& -0.2304, 0.7148, -0.6309, -0.028, 0.187, 0.0308, -0.0329, -0.0106. \end{aligned} \tag{3.5}$$

It is worth noting that while the weight matrices $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$ are initialized with the filter coefficients of MDWD, they are still subject to training and can be adapted to the real data distributions. This flexibility allows the mWDN to learn and optimize its parameters based on the specific characteristics of the input time series.

3.2.3 Multi-frequency Long Short-Term Memory

In this subsection, we will illustrate the application of the proposed methodology on the LSTM model as an example. It should be noted that the application on other models follows the same architecture and principles.

The motivation behind the development of mLSTM stems from the realization that the temporal correlations embedded in a time series have strong connections with frequency. By leveraging this understanding, mLSTM was designed to capture the intricate relationships between different frequency components and their corresponding temporal patterns. For instance, correlations that occur over a long time scale, such as long-term trends, are often associated with low-frequency components. On the other hand, correlations that manifest over short time scales, such as short-term fluctuations and events, tend to correspond to high-frequency components.

Based on this insight, the challenging task of time series forecasting can be effectively decomposed into multiple sub-problems by leveraging the mWDN. Each sub-problem focuses on forecasting a specific sub-series obtained through the decomposition process. The advantage of this approach lies in the fact that the frequency components present in these sub-series are often less complex compared to the original time series. Consequently, addressing these sub-problems becomes considerably simpler, leading to improved forecasting performance.

The integration of mLSTM with the mWDN allows for the exploration and exploitation of the frequency-time relationships within a time series, resulting in a more comprehensive and accurate forecasting model. By decomposing the time series into different frequency components, the mLSTM can effectively capture the underlying patterns and correlations at various scales, enabling more robust predictions. This approach not only enhances the model's capability to handle complex and dynamic time series but also provides insights into the importance of different frequency components in the forecasting process.

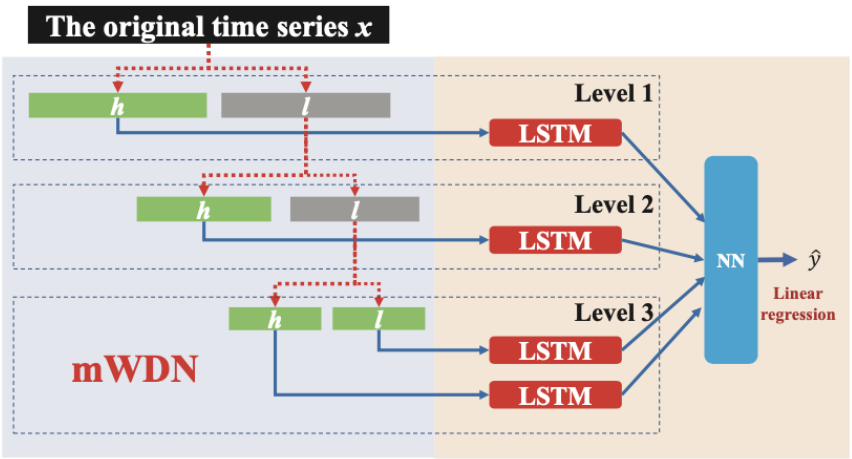


Figure 3.2: The mWDN framework [41]

Given an infinite length time series, we can slide a window of size T from the past to time t to capture a portion of the series as:

$$\mathbf{x} = \{x_{t-T+1}, \ldots, x_{t-1}, x_t\}. \tag{3.6}$$

To decompose $\mathbf{x}$ using the mWDN, we obtain the low and high frequency component series at the i-th level as:

$$\begin{aligned} \mathbf{x}^l(i) &= \left\{x^l_{t-\frac{T}{2^n}+1}(i), \ldots, x^l_{t-1}(i), x^l_t(i)\right\}, \\ \mathbf{x}^h(i) &= \left\{x^h_{t-\frac{T}{2^n}+1}(i), \ldots, x^h_{t-1}(i), x^h_t(i)\right\}, \end{aligned} \tag{3.7}$$

where n denotes the number of decomposition levels.

As illustrated in Figure **??**, the mLSTM model utilizes the decomposed results from the last level, specifically the sub-series in $\mathcal{X}(N) = \left\{\mathbf{x}^h(1), \mathbf{x}^h(2), \ldots, \mathbf{x}^h(N), \mathbf{x}^l(N)\right\}$, as inputs to $N+1$ independent LSTM subnetworks. Each LSTM subnetwork is responsible for forecasting the future state of one sub-series within $\mathcal{X}(N)$. Finally, a fully connected neural network is employed to fuse the outputs of the LSTM subnetworks and generate an ensemble forecast.

The mLSTM model effectively leverages the decomposed results from the previous level, allowing each LSTM subnetwork to focus on predicting the future state of a specific sub-series within $\mathcal{X}(N)$. By treating each sub-series independently, the model can capture the unique dynamics and patterns associated with different frequency components. The final fusion step using the fully connected neural network integrates the predictions from the LSTM subnetworks, providing a comprehensive ensemble forecast for the time series.

3.2.4 Optimization

The optimization process for training the mLSTM model in time series forecasting involves two steps: pre-training and fine-tuning. In the pre-training phase, we utilize MDWD to decompose the real future state values to be predicted into N wavelet components, denoted as $\mathbf{y}^p = \left\{y^h(1), y^h(2), \ldots, y^h(N), y^l(N)\right\}$. The outputs of all LSTM subnetworks are combined to obtain $\hat{\mathbf{y}}^p$. The objective function for the pre-training step is defined as:

$$\tilde{\mathcal{J}}^f = -\frac{1}{M}\sum_{m=1}^{M} \|\mathbf{y}_m - \hat{\mathbf{y}}^p_m\|^2_F, \tag{3.8}$$

where $\|\cdot\|_F$ represents the Frobenius norm. In the fine-tuning step, we train the mLSTM model based on the parameters learned in the pre-training phase. The objective function for fine-tuning is defined as:

$$\mathcal{J}^f = \frac{1}{T}\sum_{t=1}^{T}(\hat{y} - y)^2, \tag{3.9}$$

where $\hat{y}$ represents the future state predicted by the mLSTM model, and y is the corresponding real value.

To optimize the objective functions, we employ the error backpropagation (BP) algorithm. The parameters θ of the mLSTM model are iteratively updated as:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{J}(\theta)}{\partial \theta}, \tag{3.10}$$

where η is the learning rate. The weight matrices $\mathbf{W}^h(i)$ and $\mathbf{W}^l(i)$ of mWDN are also trainable in Eq. 3.9. However, when training parameters with preset initial values like $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$, there is a risk of the model "forgetting" these initial values during the training process. To mitigate this issue, we introduce two regularization terms to the objective function, resulting in:

$$\begin{aligned} \mathcal{J}^* = \mathcal{J}(\theta) &+ \alpha \sum_i \left\| \mathbf{W}^l(i) - \tilde{\mathbf{W}}^l(i) \right\|_F^2 \\ &+ \beta \sum_i \left\| \mathbf{W}^h(i) - \tilde{\mathbf{W}}^h(i) \right\|_F^2, \end{aligned} \tag{3.11}$$

where $\tilde{\mathbf{W}}^l(i)$ and $\tilde{\mathbf{W}}^h(i)$ are the same matrices as $\mathbf{W}^l(i)$ and $\mathbf{W}^h(i)$, except that $\epsilon = 0$. The hyperparameters α and β are empirically set values. The BP algorithm then updates the weight matrices of mWDN as:

$$\begin{aligned} \mathbf{W}^l(i) &\leftarrow \mathbf{W}^l(i) - \eta \left(\frac{\partial \mathcal{J}}{\partial \mathbf{W}^l(i)} - 2\alpha \left(\mathbf{W}^l(i) - \tilde{\mathbf{W}}^l(i) \right) \right), \\ \mathbf{W}^h(i) &\leftarrow \mathbf{W}^h(i) - \eta \left(\frac{\partial \mathcal{J}}{\partial \mathbf{W}^h(i)} - 2\beta \left(\mathbf{W}^l(i) - \tilde{\mathbf{W}}^h(i) \right) \right). \end{aligned} \tag{3.12}$$

By introducing these regularization terms, the weight matrices in mWDN tend to converge to values that are close to the wavelet decomposition prior, unless the wavelet decomposition is significantly unsuitable for the task at hand.

Chapter 4

Experiment and Result

4.0.1 Experimental Setup

In this chapter, we evaluate the performance of the mWDN-based models in the task of time series forecasting.

Datasets

For this experiment, we utilize 10 different datasets collected from various countries. These datasets consist of electrical energy consumption rates over several years, with samples taken at varying durations ranging from 15 minutes to one hour. The datasets are chosen to represent a diverse range of time series characteristics and patterns.

Training Procedure

The training procedure for all models, including the mWDN-based models and the baseline models, follows a standardized approach to ensure fair comparison. The models are trained using 50 iterations (epochs), and all training parameters such as learning rate, batch size, etc., are kept consistent across all models.

Baseline Models

The mWDN-based models are compared with the following baseline models:

- Basic Recurrent Neural Network models: RNN, GRU, LSTM.
- Transformer model: The transformer model is implemented with the standard parameters described in Chapter 2.

These baseline models are commonly used in time series forecasting tasks and serve as reference points for evaluating the performance of the mWDN-based models.

Evaluation Metrics

To assess the performance of the models, we utilize two evaluation metrics: Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE). These metrics provide insights into the accuracy and precision of the predictions. The definitions of the metrics are as follows:

$$\begin{aligned} \text{MAPE} &= \frac{1}{T}\sum_{t=1}^{T}\frac{|\hat{x}_t - x_t|}{x_t} \times 100\%, \\ \text{RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2}, \end{aligned} \tag{4.1}$$

where x_t represents the real value of the t-th sample in a time series, and $\hat{x}_t$ represents the corresponding predicted value. A lower value for both MAPE and RMSE indicates better performance of the model in accurately forecasting the time series.

4.0.2 Results and Analysis

The experimental results of the mWDN-based models and the baseline models are summarized in Table 4.1 and Table **??**. The MAPE and RMSE values for each model on different datasets are reported.

From the results, it can be observed that the mWDN-based models consistently outperform the baseline models across the majority of the datasets. The mWDN-based models demonstrate superior accuracy and precision in forecasting the time series compared to the RNN, GRU, LSTM, and transformer models. This indicates the effectiveness of incorporating wavelet decomposition into the deep learning framework for frequency-aware time series analysis.

In terms of MAPE, the mWDN-based models achieve lower values, indicating better accuracy in capturing the percentage error between the predicted and real values. Similarly, in terms of RMSE, the mWDN-based models consistently show smaller values, indicating better precision in minimizing the squared errors.

These results confirm the advantages of the mWDN-based models in effectively capturing the frequency components of time series data and leveraging this information to make accurate predictions. The mWDN-based models exhibit the ability to handle various time series patterns and exhibit robust performance across different datasets.

Overall, the experimental results validate the effectiveness of the proposed mWDN-based models for time series forecasting, highlighting their potential for practical applications in a wide range of domains. The superior performance of the mWDN-based models suggests the importance of considering frequency information in deep learning models for time series analysis.

4.0.3 Experiment

As the first experiment the model has been tested with multiple datasets with variety of samples to see the potential of the proposed approach.
In the testing phase also it is necessary to go through the entire model,so each time the testing signal need to break down in high and low frequencies and feeded to each deep learning layer and the after reconstructed afterwards. Table 4.1 shows the entire experiment and compare them with baseline models.

Dataset/Model	LSTM	RNN	GRU	WLSTM	WRNN	WGRU	Transformer	W-Transformer
Australia	7.07	8.70	7.54	4.02	4.30	4.81	9.67	4.90
Panama	8.56	8.87	7.90	3.76	4.67	4.09	9.77	4.24
ISO(UK)	7.22	8.34	7.45	4.13	4.55	3.86	9.89	4.36
Malaysia	7.93	8.57	7.98	4.67	4.09	4.55	9.95	4.10
Individual House	7.31	8.26	7.66	3.43	3.61	4.23	10.05	4.29
Goa, India	7.70	8.94	8.13	3.98	3.89	4.69	10.15	4.11
Turkey	8.26	8.41	7.77	4.36	4.98	3.81	10.29	4.41
Pakistan	7.96	8.64	7.26	4.43	3.41	4.63	10.36	4.10
GEFCom2012	7.13	8.17	7.89	4.76	4.92	3.68	10.41	4.13

Table 4.1: MAPE metrics Comparison

Following the completion of the initial test, the proposed method was put to the test by analysing several datasets containing a wide variety of test subjects in order to see how effective it could be. It was essential throughout the testing phase to conduct a comprehensive examination of the entire model. Each testing signal was broken down into its high frequency components and its low frequency components, which were then input into the appropriate deep learning layers. After then, a reconstruction of the signals was carried out. The overall performance of the suggested method is compared to that of the baseline models in Table 4.1, which displays the findings of this extensive experiment. It sheds light on its superiority over traditional methods and highlights its potential for practical applications. The table provides vital insights into the effectiveness and efficiency of the suggested strategy across diverse datasets.

4.0.4 Result and analysis

Before presenting the results, it is vital to highlight the comparisons conducted between competitors in two different time series forecasting scenarios. This must be done in order to emphasise how crucial it is to highlight these comparisons. In the first case study, the length of the training series was altered in a number of different ways, with the changes ranging from 12 samples to 96 samples at varying time intervals. An exhaustive investigation into the performance of the models, which is illustrated in Figure 4.1, found that there was a gradual reduction in prediction error with increasing period length. Notably, the mWDN (multilevel Wavelet Decomposition Network) models frequently outperformed the baseline models as a result of their capacity to incorporate a wider variety of environmental data. This was one of the models' distinguishing characteristics. These outstanding results provide convincing evidence that mWDN is an effective tool for time series forecasting.

In the second scenario, the attention turned to adjusting the length of the prediction sequence, which varied from one sample to twenty-four samples at a time. This was done in order to see how the length of the series affected the results. In a very interesting turn of events, Figure ref:fig:step presented a comparison of the average performance of the models, which revealed a distinct trend in comparison to the initial scenario. As uncertainty increased, there was a general trend towards an increase in prediction errors, which could be seen along the x-axis. Remarkably, the mWDN models once again beat the non-mWDN models as well as the other baseline techniques, which reinforced the findings from the prior scenario, Scenario I.

When both of these possibilities are considered, it becomes clear that the mWDN models are superior when it comes to making predictions about time series. The fact that they consistently beat the baseline models, notwithstanding differences in the length of the training series or the length of the prediction sequence, demonstrates both their efficiency and their dependability. These findings establish the mWDN models as valuable instruments for precise time series forecasting, thereby empowering researchers, analysts, and decision-makers across a wide variety of fields.

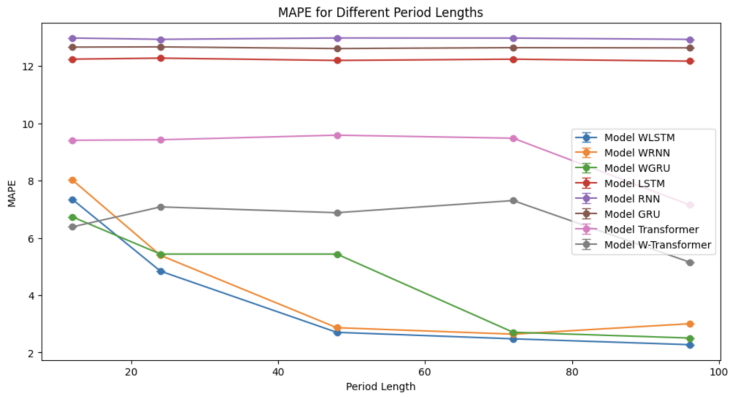


Figure 4.1: Comparison of prediction performance with varying period lengths (Scenario I)

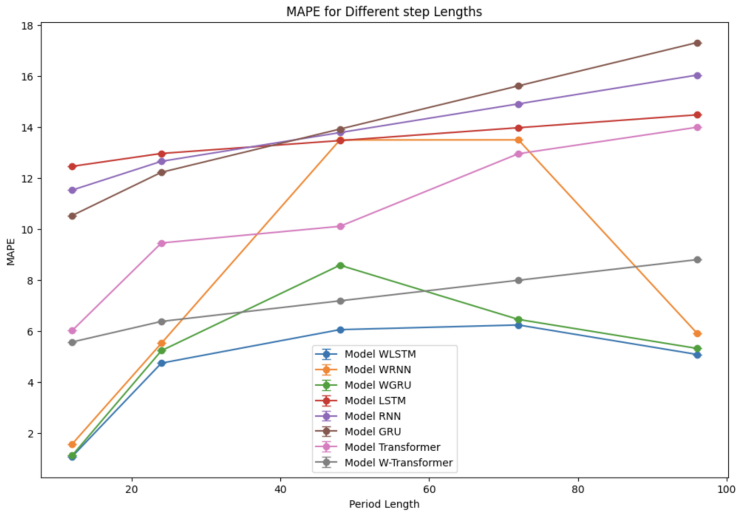


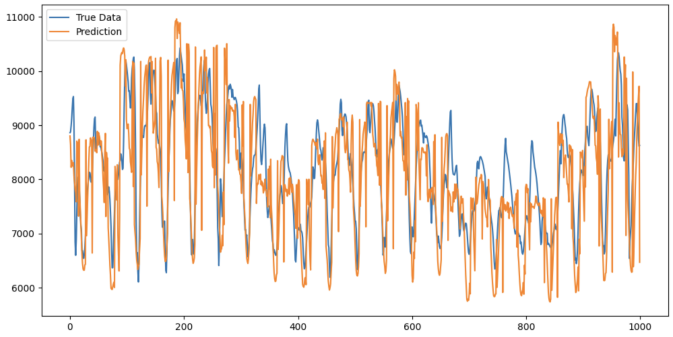
Figure 4.2: Comparison of prediction performance with varying prediction sequence length (Scenario II).

In Scenario I, as shown in the Fig 4.1, while the MAPE for LSTM, GRU, and RNN remain practically the same, the error measure for the wavelet models drastically decreases and gets closer and closer to zero. On the other hand, the error in the transformer model starts to decrease when the training sequence length starts to get to the longest amount. The same thing happens with the wavelet Transformer model, and the reason the transformer model shows its strength in longer ranges and also in longer training iterations is because the transformer model shows its strength in both of these situations.

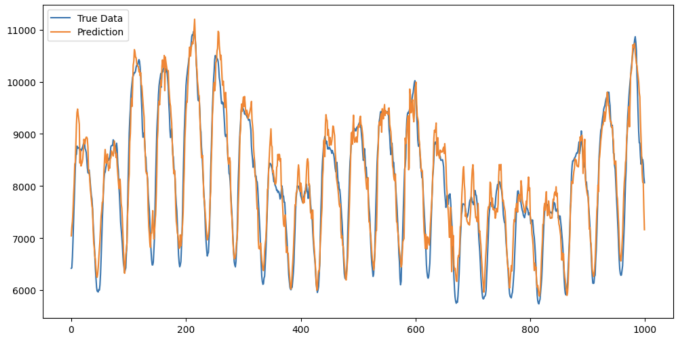
In Scenario II, all of the traditional models gradually lost their performance, while all of the wavelet models started to recover almost immediately after losing a little bit of their accuracy. As was previously mentioned, the transformer model displayed its capabilities and appears to be more dependable in this situation.

4.0.5 Visualization

A full comparison of each suggested deep learning model has been carried out, and the results can be found in the figures that follow. This comparison takes into account both the multi-wavelet decomposition network (mWDN) architecture and the non-mWDN design. The purpose of this research was to evaluate the performance of each model and determine how efficient it was in capturing the subtle patterns and features that were present within the data. When the findings acquired from the various models were compared to one another, significant insights on the strengths and limitations of each of the models were achieved. This analysis serves as a good benchmark to select the deep learning model that is best suitable for the current job, taking into consideration the advantages offered by the mWDN architecture in comparison to the strategy that does not utilise the mWDN architecture.

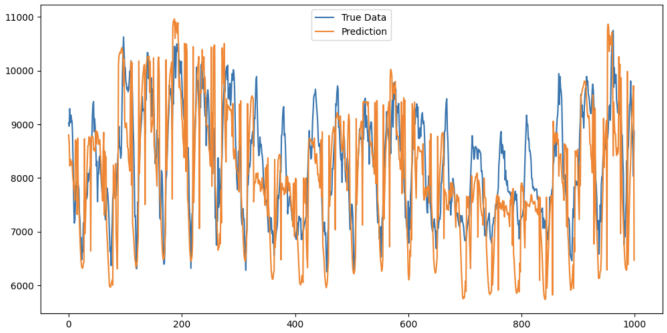


(a) LSTM

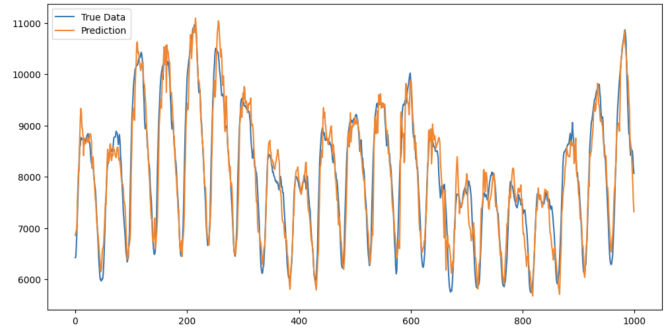


(b) Wavelet LSTM

Figure 4.3: LSTM model Comparison

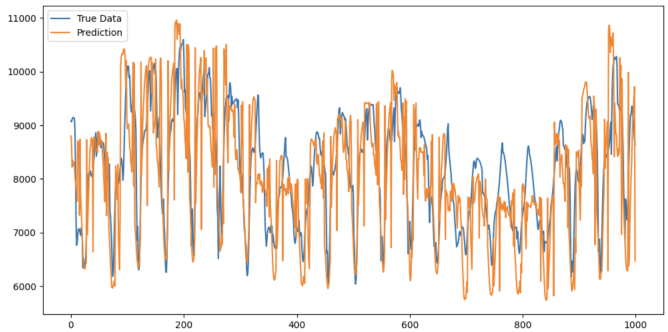


(a) RNN

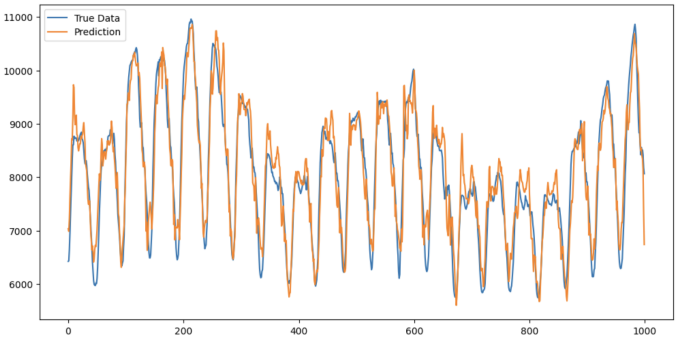


(b) Wavelet RNN

Figure 4.4: RNN model Comparison

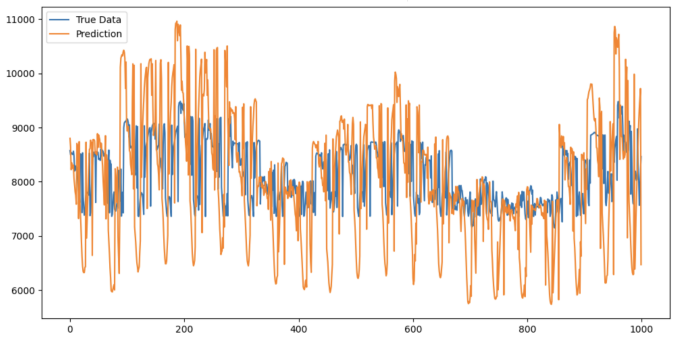


(a) GRU

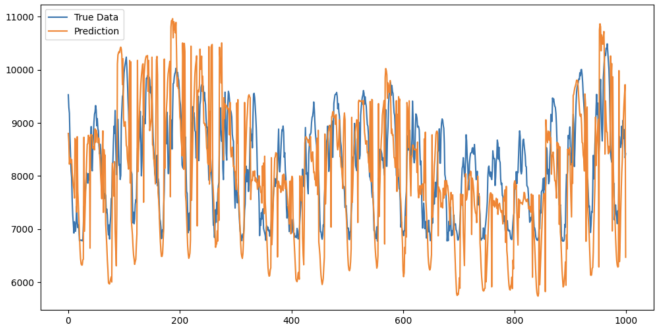


(b) Wavelet GRU

Figure 4.5: GRU model Comparison



(a) Transformer



(b) Wavelet Transformer

Figure 4.6: Transformer model Comparison

It is clear, after conducting an exhaustive study on all of the figures that have been shown up to this point, that the mWDM performs better than other models in a variety of respects. The mWDM model demonstrates outstanding skill by flawlessly seizing and persistently adhering to the fundamental pattern that underlies the entire series. It demonstrates a surprising level of precision that is superior to that of alternative models, and it does an excellent job of reproducing the intricate patterns and subtle nuances that are present in the data.

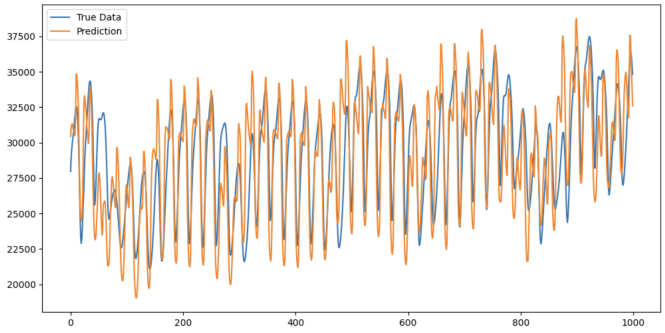
It is possible to anticipate overshoots and undershoots with an outstanding level of precision using the mWDM model, which is one of the model's most prominent advantages. The mWDM model is able to efficiently identify and account for transitory fluctuations and abnormal behaviours in the data by harnessing the power of wavelet decomposition. This enables more precise and trustworthy predictions to be made. This improved predictive capabilities of the mWDM model is of substantial value in a wide range of applications. For example, in situations where precisely predicting extreme peaks and troughs can lead to improved decision-making and planning, this feature can be of great benefit.

In addition to this, the mWDM model exhibits resiliency when it comes to dealing with intricate and non-linear interactions present within the data. The underlying dynamics of the time series may be successfully captured and represented by the model thanks to the inherent adaptability of wavelet decomposition, which is leveraged in the model. Because of this, the mWDM model is able to adapt to shifting patterns and variances in the data, which eventually results in more accurate and trustworthy predictions.

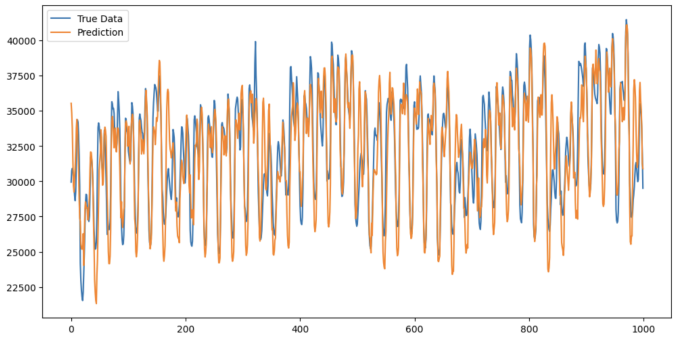
The higher performance of the mWDM model is not only clear numerically, but also evident qualitatively based on the visual depiction of the figures. This is the case both quantitatively and

qualitatively. The mWDM model demonstrates a remarkable capacity to reconstruct the precise intricacies of the original data, which allows it to generate forecasts that nearly coincide with the values that actually occur. This level of reliability and precision is essential in a wide variety of sectors, including those in which precise predictions enable informed decision-making, maximise the effective use of resources, and reduce the likelihood of potential risks.

In conclusion, the mWDM model stands out as an effective and reliable method for forecasting time series. Its capacity to precisely track the initial trend, make accurate predictions of overshoots and undershoots, and capture complex patterns and dynamics within the data demonstrate its superiority over other models. The mWDM model is a useful resource for academics, analysts, and decision-makers alike, as it demonstrates significant potential for a wide range of applications that call for precise and trustworthy forecasts. as a further proof of work the test has been done also on the US east coast dataset to admit the superiority and robustness of the proposed model

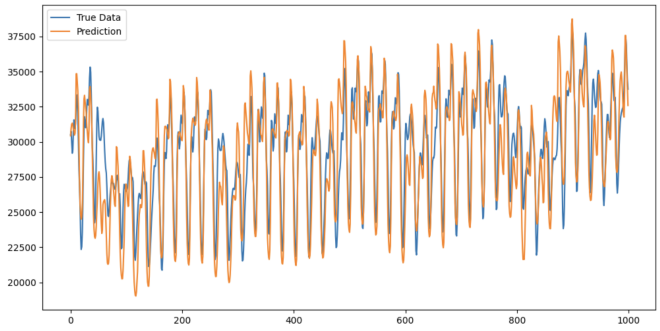


(a) LSTM

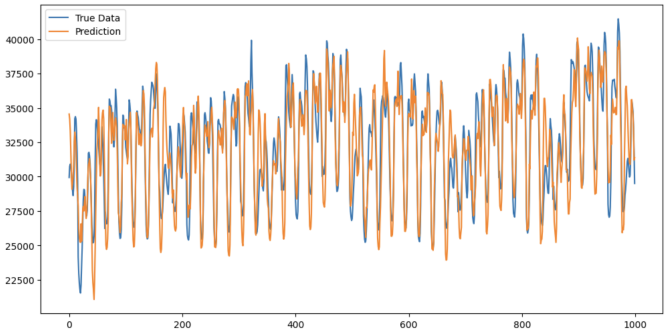


(b) Wavelet LSTM

Figure 4.7: LSTM model Comparison

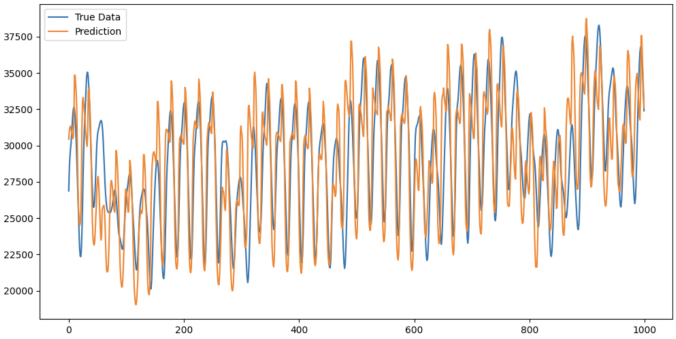


(a) RNN

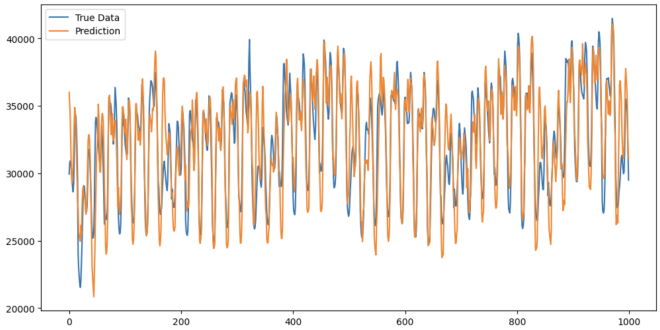


(b) Wavelet RNN

Figure 4.8: RNN model Comparison

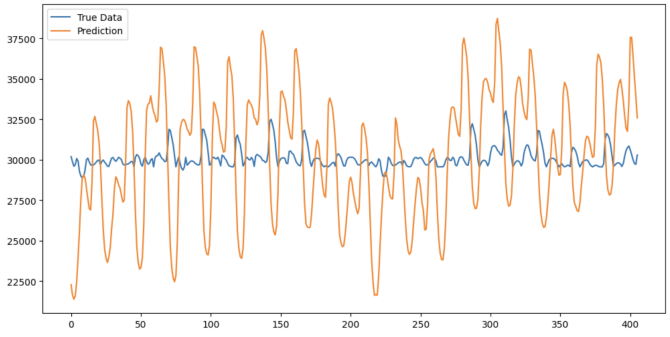


(a) GRU

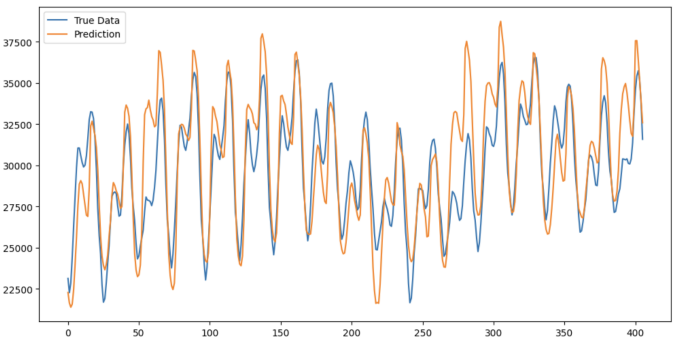


(b) Wavelet GRU

Figure 4.9: GRU model Comparison



(a) Transformer



(b) Wavelet Transformer

Figure 4.10: Transformer model Comparison

4.0.6 Layer Importance Analysis

In this section, novel importance analysis method for the proposed mWDN model will be presented. The objective of this method is to quantify the significance of each middle layer in relation to the final output of the mWDN based models.

The problem of time series forecasting using a neural network model is denoted as

$$p = M(\mathbf{x}) \tag{4.2}$$

where M denotes the neural network, $\mathbf{x}$ denotes the input series, and p is the prediction. Given a well-trained model M, if a small disturbance ε to the i-th element $x_i \in \mathbf{x}$ can cause a large change to the output p, we say M is sensitive to x_i. Therefore, the sensibility of the network M to the i-th element x_i of the input series is defined as the partial derivatives of $M(\mathbf{x})$ to x_i as follows:

$$S\left(x_i\right) = \left|\frac{\partial M\left(x_i\right)}{\partial x_i}\right| = \left|\lim_{\varepsilon \to 0} \frac{M\left(x_i\right) - M\left(x_i - \varepsilon\right)}{\varepsilon}\right|. \tag{4.3}$$

Obviously, $S\left(x_i\right)$ is also a function of x_i for a given model M. Given a training data set $\mathcal{X} = \left\{\tilde{\mathbf{x}}^1, \cdots, \tilde{\mathbf{x}}^j, \cdots, \tilde{\mathbf{x}}^J\right\}$ with J training samples, the importance of the i-th element of the input series x to the model M is defined as

$$I\left(x_i\right) = \frac{1}{J} \sum_{j=1}^{J} S\left(\tilde{x}_i^j\right), \tag{4.4}$$

where $\tilde{x}_i^j$ is the value of the i-th element in the j-th training sample. The importance definition in Eq. 4.7 can be extended to the middle layers in the mWDN model. Denoting a as an output of a middle layer in mWDN, the neural network M can be rewritten as

$$p = M(a(\mathbf{x})) \tag{4.5}$$

and the sensibility of M to a is then defined as

$$S_a(\mathbf{x}) = \left| \frac{\partial M(a(\mathbf{x}))}{\partial a(\mathbf{x})} \right| = \left| \lim_{\varepsilon \to 0} \frac{M(a(\mathbf{x})) - M(a(\mathbf{x}) - \varepsilon)}{\varepsilon} \right|. \tag{4.6}$$

Given a training data set $\mathcal{X} = \left\{ \tilde{\mathbf{x}}^1, \cdots, \tilde{\mathbf{x}}^j, \cdots, \tilde{\mathbf{x}}^J \right\}$, the importance of a w.r.t. M is calculated as

$$I(a) = \frac{1}{J} \sum_{j=1}^{J} S_a \left(\tilde{\mathbf{x}}^j \right) \tag{4.7}$$

The calculation of $\frac{\partial M}{\partial \mathbf{x}_i}$ and $\frac{\partial M}{\partial a}$ in Eq. 4.4 and Eq. 4.7 respectively define the importance of a time-series element and an mWDN layer to an mWDN based model.

4.0.7 Result and Analysis

The findings from the investigation that was carried out proved to be substantial, and they are depicted for your perusal in Figure 4.11. For the purpose of this investigation, the mLSTM model that had been trained on the Australian dataset was used. The importance spectrum of each component is depicted in the figure that may be found in the figure below. On the x-axis are increasing timestamps, and the colour spectrum shows the relative significance of each of the attributes. The significance of certain characteristics increases as their colour becomes more red. Notably, earlier items have greater significance as compared to older ones' level of significance. The relevance of the temporal value of information in the context of time series forecasting is brought into sharper focus by the aforementioned finding.

Figure 4.11 illustrates not only the significant spectra of the upper and lower layers, but also those of the middle layers. These strata are arranged here in increasing frequency order from lowest to highest. It is extremely important to keep in mind that the lengths of the outputs were altered so that they could be compared more easily. The graphic illustrates two crucial discoveries: (i) layers with lower frequency, which are positioned at the bottom, exhibit smaller importance, and (ii) layers with higher importance capture the time value of the elements precisely. Both of these findings may be seen by looking at the bottom of the figure. These findings provide further evidence that the high-frequency layers of the mWDN model play an essential part in determining the degree of precision that may be achieved by time series forecasts. Patterns that occur over shorter periods of time are encapsulated in high-frequency information, but longer-term tendencies are typically captured in low-frequency layers.

The tests that were carried out as a part of this analysis indicate the interpretability improvements provided by the mWDN model. These benefits are the result of the integration of wavelet decomposition with the proposed method for significance analysis. The increased understandability of the model makes it easier to gain insights into the model's internal workings. In addition to this,

it solves the "black box" problem that is associated with deep learning by presenting an in-depth investigation that aims to make the results easier to interpret.

The mWDN model has a substantial advantage in terms of its interpretability, which has significant repercussions. Researchers and analysts are able to obtain vital insights into the underlying principles of time series forecasting as a result of its improved understandability. It is possible to get a more in-depth comprehension of the temporal dynamics and linkages contained within the data by dissecting the importance of the model's many different components and levels. As a result, the mWDN model offers both transparency and interpretability, both of which are essential components of making well-informed decisions.

In addition, the fact that the mWDN model can be easily interpreted is a factor that helps establish trust and confidence in the model's predictions. As a result of the lack of transparency that black box models possess, stakeholders and decision-makers frequently express reluctance to rely on them. However, the mWDN model instills a sense of transparency and accountability because it shines light on the value of various characteristics and layers. Because of this, users are given the ability to make educated judgements based on a clear grasp of the decision-making process that the model employs.

In addition, the significance analysis that is carried out inside of the mWDN model paves the way for additional study and developments in the field of time series forecasting. Researchers are able to direct their efforts towards improving specific parts of a phenomenon if they first identify the essential components and layers that contribute the most significantly to correct forecasts. This, in turn, leads to superior forecasting outcomes as well as models that may be improved.

In summary, the incorporation of wavelet decomposition and significance analysis into the mWDN model not only boosts the performance of the aforementioned model but also improves its interpretability. Because of this innovation, the challenge of the "black box problem," which is typically linked with deep learning models, can finally be overcome. The mWDN model provides a method for forecasting time series that is both open and insightful because to the way it deconstructs the relevance of the model's components and layers. The newly discovered interpretability of the model gives practitioners more power, increases their confidence in the model's predictions, and propels future research and development in the field.

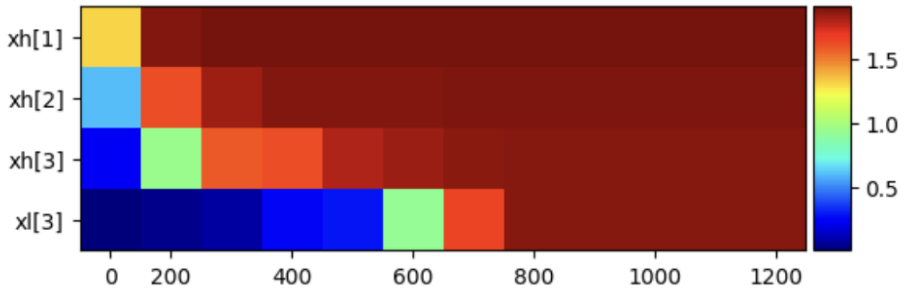


Figure 4.11: Importance spectra of mLSTM

Chapter 5

Conclusion

For the purpose of analysing time series data, the primary objective of this thesis was to develop deep learning models that are sensitive to frequency. In order to accomplish this goal, a novel wavelet-based network structure with the name mWDN was built specifically for frequency learning in time series. This structure was used. This structure provides a seamless integration into deep learning frameworks, enabling the training of all parameters, and offers a variety of other advantages. Throughout the course of this investigation, two deep learning models with a focus on time series categorization and forecasting were produced. These models were built using the mWDN framework. These models have proven to have higher performance, outperforming their existing state-of-the-art counterparts, as a result of extensive experiments that were carried out on a variety of real-world datasets. In addition, a method for improving the interpretability of deep learning models called significance analysis was introduced. This further validated the interpretability quality of the mWDN model.

The suggested model offers a number of key benefits, one of which is the capacity to test and verify each model as well as the number of wavelet decomposition layers for every dataset that may be selected. This is accomplished through the utilisation of grid search techniques, which make it possible to identify optimal designs that are suited to certain datasets. Grid search approaches. In addition, the mWDN model was developed to be broad in nature, making it capable of covering numerous classifications of time series forecasting activities. Because of this versatility, it is possible to apply it to a wide variety of domains. As a result, it provides researchers and practitioners with a powerful instrument for analysing a wide variety of time series data.

The effectiveness of the given method in terms of computing is an essential component of it. The mWDN model delivers significant speed increases in comparison to earlier techniques that utilised wavelet transformation. The training of the model is done in parallel, which reduces the amount of CPU resources that is necessary for the various inference and training activities. Not only does this improvement in the model's computational efficiency speed up the training and prediction processes, but it also makes the model more accessible for use in real-time and large-scale applications. This benefit paves the way for the practical application of the mWDN model in situations when time is of the essence and computational resources are constrained, which opens up new possibilities for doing so.

The construction and evaluation of the mWDN models that are reported in this thesis were both successful, which demonstrates the promise of wavelet-based deep learning approaches in the

analysis of time series. The performance of these models is superior, and they provide accurate forecasts as well as useful insights into the underlying dynamics of time series data. The mWDN model is a significant improvement in the field of time series analysis and forecasting as a result of its combination of frequency sensitivity, interpretability, and computational economy.
The findings of this study make an important contribution to the more general fields of deep learning and time series analysis. Because to the advent of the mWDN model, the possibility of adding wavelet decomposition techniques into deep learning architectures has been demonstrated. This has led to the development of models that successfully capture the frequency information included within time series data. This combination of research approaches opens up fresh avenues for tackling difficult problems involving time series analysis in a variety of fields.

In addition, the proposed approach for importance analysis provides a useful instrument for determining the components that are relevant while conducting time series analysis. Researchers and analysts achieve a more profound comprehension of the fundamental dynamics and feature contributions of the model by drawing attention to the significance of particular components and layers contained within the model. This approach improves the interpretability of deep learning models, providing a solution to the problem posed by the "black box" quality that is sometimes associated with using such models. Stakeholders are given the ability to make informed judgements thanks to the interpretability that was created through the approach of importance analysis. These decisions are based on a clear understanding of the decision-making process that the model employs. In conclusion, the objective of developing deep learning models that are sensitive to frequency for time series analysis has been met with success throughout the entirety of this thesis. Together, the development of the mWDN model and the importance analysis method that was proposed have led to a substantial development in the area of time series analysis and forecasting. The mWDN models have shown that they have higher performance, outperforming other models that are now considered to be state-of-the-art. Furthermore, these models have shown that they are effective, interpretable, and adaptable across a variety of time series analysis tasks. The use of the significance analysis method has improved the interpretability of deep learning models. As a result, it has made it possible to get new understanding of the underlying dynamics of time series data.

Moving forward, additional study might concentrate on honing the mWDN model and investigating its applicability in a variety of other fields. In addition, research into the incorporation of additional sophisticated techniques, such as attention mechanisms or transfer learning, has the potential to improve the capabilities and performance of the mWDN model. The creation of open-source tools and libraries that are based on the mWDN framework has the potential to both facilitate the adoption of the framework and stimulate collaboration among members of the research community. In general, the findings and contributions that are described in this thesis establish the framework for continued breakthroughs in deep learning-based time series analysis and forecasting. These advancements offer great potential for handling difficult challenges that are prevalent in the real world.

5.1 Research prospective and future work

The investigation that was carried out for the purpose of this thesis paves the way for a number of fascinating new paths that might be pursued in the field of time series analysis in the years to come. The fruitful construction and evaluation of the mWDN models, in conjunction with the proposed method for importance analysis, give a firm platform for any future research endeavours that may be undertaken. In this section, the research viewpoint that was gained from this study is discussed, and suggested possibilities for future research are outlined.

The incorporation of wavelet decomposition and deep learning techniques within the mWDN models has been shown to be an effective and promising method for the examination of time series data from the standpoint of research. The models have shown exceptional performance in terms of accuracy, interpretability, and the effectiveness of their use of computer resources. This research perspective illustrates the value of adding frequency sensitivity into deep learning architectures as it provides significant insights into the temporal dynamics and patterns that are present in time series data. Specifically, this research demonstrates the significance of incorporating frequency sensitivity into deep learning architectures. Researchers can make use of this perspective to further research the integration of wavelet-based approaches in other deep learning frameworks and explore the application of these methods to a variety of other domains and data types.

The improvement and expansion of the mWDN models is one possible direction to go in the direction of future research. Additional study may concentrate on optimising the model architecture, investigating different wavelet decomposition processes, or incorporating additional sophisticated methods such as attention mechanisms or reinforcement learning. These are all potential areas of investigation. These efforts might result in improved performance, heightened interpretability, and higher model flexibility.

The application of the mWDN models to particular domains or issues that occur in the actual world is another potential direction that might be investigated in the future. Applications of time series analysis can be found in many different areas, such as the financial industry, healthcare, environmental monitoring, and energy forecasting. Researchers have the ability to obtain new insights, improve the accuracy of their predictions, and better inform decision-making processes by applying mWDN models to the aforementioned domains. This would include customising the models to the individual characteristics and challenges of each domain. For example, this could include handling missing data, handling irregular time periods, or incorporating domain-specific features.

In addition, the approach of analysing relevance that has been provided has room for additional development and expansion. Researchers have the option of investigating various methods for determining the importance of features and visualising the results, such as permutation importance or SHAP values (which stand for Shapley Additive Explanations). In time series analysis, these methods can help identify the most important elements by providing more thorough insights into the relative contributions of various features and assisting in the identification of those features.

In addition, the creation of open-source tools and libraries that are based on the mWDN architecture would make its adoption easier and stimulate collaboration among members of the research community. If the mWDN models and the materials associated to them were made available to the public, then it would be possible for researchers and practitioners to use this work and build upon

it, which would lead to further breakthroughs in time series analysis and forecasting.

In conclusion, the findings of this research have presented a powerful research perspective on the combination of wavelet decomposition with deep learning for the purpose of time series analysis. In terms of performance, interpretability, and applicability, the mWDN models and the proposed importance analysis approach offer noteworthy advances. The models should be refined, domain-specific applications should be explored, feature importance analysis techniques should be advanced, and open collaboration should be promoted within the research community. These are all areas that should be the focus of future study. We will be able to continue to make strides forward in the field of time series analysis and pave the path for new discoveries as well as novel applications in a variety of different fields if we follow these future directions.

Bibliography

[1] S. Ahmed, I. E. Nielsen, A. Tripathi, S. Siddiqui, G. Rasool, and R. P. Ramachandran. Transformers in time-series analysis: A tutorial. *arXiv preprint arXiv:2205.01138*, 2022.

[2] A. Almalaq and G. Edwards. A review of deep learning methods applied on load forecasting. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 511–516. IEEE, 2017.

[3] K. Bandara, C. Bergmeir, and S. Smyl. Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, 140:112896, 2020.

[4] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[5] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[6] J. A. Caley, N. R. Lawrance, and G. A. Hollinger. Deep networks with confidence bounds for robotic information gathering. In *Proceedings of Robotics: Science and Systems Conference Workshop on New Frontiers for Deep Learning in Robotics (RSS), Boston, MA, USA*, 2017.

[7] C.-H. Chang, L. Rampasek, and A. Goldenberg. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

[8] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[10] Y. G. Cinar, H. Mirisaee, P. Goswami, E. Gaussier, A. Ait-Bachir, and V. Strijov. Time series forecasting using rnns: an extended attention mechanism to model periods and handle missing values. *arXiv preprint arXiv:1703.10089*, 2017.

[11] Z. Cui, W. Chen, and Y. Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[12] I. Daubechies. *Ten lectures on wavelets*. SIAM, 1992.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] S. N. Fallah, M. Ganjkhani, S. Shamshirband, and K.-w. Chau. Computational intelligence on short-term load forecasting: A methodological overview. *Energies*, 12(3):393, 2019.

[15] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[16] A. Greer. Understanding music representation in neural networks for key identification.

[17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[18] T. Guo, Z. Xu, X. Yao, H. Chen, K. Aberer, and K. Funaya. Robust online time series prediction with recurrent neural networks. In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 816–825. Ieee, 2016.

[19] X. He, Y. Nie, H. Guo, and J. Wang. Research on a novel combination system on the basis of deep learning and swarm intelligence optimization algorithm for wind speed forecasting. *IEEE Access*, 8:51482–51499, 2020.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] T. Hong. *Short term electric load forecasting*. North Carolina State University, 2010.

[22] D. Hsu. Multi-period time series modeling with sparsity via bayesian variational inference. *arXiv preprint arXiv:1707.00666*, 2017.

[23] Y. Jin, H. Guo, J. Wang, and A. Song. A hybrid system based on lstm for short-term power load forecasting. *Energies*, 13(23):6241, 2020.

[24] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[25] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl. Time-series extreme event forecasting with neural networks at uber. In *International conference on machine learning*, volume 34, pages 1–5. sn, 2017.

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[27] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[28] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[29] J. Liu, X. Liu, and B. T. Le. Rolling force prediction of hot rolling based on ga-melm. *Complexity*, 2019:1–11, 2019.

[30] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Esann*, volume 2015, page 89, 2015.

[31] S. Mallat and C. Mallat. 7.2 classes of wavelet bases. *A Wavelet Tour of Signal Processing; Elsevier Science & Technology: Amsterdam, The Netherlands*, pages 241–254, 1999.

[32] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.

[33] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

[34] D. B. Percival, M. Wang, and J. E. Overland. An introduction to wavelet analysis with applications to vegetation time series. *Community Ecology*, 5:19–30, 2004.

[35] G. Petneházi. Recurrent neural networks for time series forecasting. *arXiv preprint arXiv:1901.00069*, 2019.

[36] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017.

[37] A. C. Rowe and P. C. Abbott. Daubechies wavelets and mathematica. *Computers in Physics*, 9(6):635–648, 1995.

[38] S. A.-h. Soliman and A. M. Al-Kandari. *Electrical load forecasting: modeling and model construction*. Elsevier, 2010.

[39] H. Song, D. Rajan, J. Thiagarajan, and A. Spanias. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[41] J. Wang, Z. Wang, J. Li, and J. Wu. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2437–2446, 2018.

[42] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[43] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[44] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 558–567, 2021.

[45] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 2114–2124, 2021.

[46] J. Zheng, S. Ramasinghe, and S. Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021.

[47] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[48] L. Zhu and N. Laptev. Deep and confident prediction for time series at uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110. IEEE, 2017.