Master of Science
<<Artificial Intelligence and Visual Computing>>

UNIVERSITY OF WEST ATTICA & UNIVERSITY OF LIMOGES

FACULTY OF ENGINEERING

DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

Master Thesis

Deep Learning models for timeseries forecasting using Keras library

Student: Zelios Vasileios
(aivc21003)

Supervisors: Prof. Paris Mastorocostas - George Kandilogiannakis M.Sc.

Athens, February 2023

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου του εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή

Α/Α	ΟΝΟΜΑΤΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Αναστάσιος Κεσίδης	Αναπληρωτής Καθηγητής	Anastasios Kesidis Digitally signed by Anastasios Kesidis Date: 2023.02.17 08:57:27 +02'00'
2	Πάρις Μαστοροκώστας	Καθηγητής	Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2023.02.17 07:37:29 +02'00'
3	Παναγιώτα Τσελέντη	ΕΔΙΠ	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2023.02.17 09:29:05 +02'00'

ABSTRACT

The current thesis aims to conduct a thorough examination of recurrent neural networks for the purpose of forecasting short-term electric load in Greece. The study is motivated by the significant energy crisis that Greece has been experiencing, which is characterized by high electricity costs. As of January 2022, Greece had the highest electricity costs in Europe, reaching 227.3 Euros per megawatt-hour. This dire situation necessitates the development of accurate forecasting methods for electric load demand by experts.

Recurrent neural networks are a type of artificial neural network that have the ability to process sequential data, making them suitable for time series forecasting. The study explores the impact of different network architectures and parameters on the forecasting performance.

We developed deep learning algorithms using Python and trained neural networks with historical data to generate predicted electricity load values and calculate statistical errors. The focus of the study was on using LSTM models, which have been shown to provide highly accurate forecasts for time series data due to their complexity.

In conclusion, the predictions generated by the models developed in the present study were integrated into the Power BI platform, to facilitate the ease and convenience of data visualization for the end-user. Power BI is a business intelligence tool that allows for the creation of interactive visualizations and dashboards, providing a user-friendly interface for data exploration. By integrating the model predictions into Power BI, it becomes possible to present the data in an intuitive and accessible manner, enabling the end-user to gain insights and make informed decisions.

Key words: electric load demand, recurrent neural networks, deep learning, time series forecasting, short-term forecast, algorithms, Python, visualization

Table of Contents

Table of Figures

Table of Tables

This thesis
is
dedicated
to the memory of
my late grandmother
Georgia

Acknowledgements

The present thesis was prepared in the context of my studies in Postgraduate Programme "Artificial Intelligence and Visual Computing" of the Department of Informatics and Computer Engineering of the University West Attica. The research started in July 2022 and was completed in February 2023.

I would like to express my deep appreciation and gratitude to Professor Paris Mastorocostas for his invaluable guidance and support throughout the course of my thesis. His expertise and extensive knowledge in the field proved to be an invaluable asset to my research. Not only did he provide me with a wealth of bibliographic material, but he also offered valuable advice and insights that helped shape the direction and focus of my work. I am truly grateful for his unwavering support and guidance, without which, the completion of my thesis would not have been possible.

I would also like to extend my thanks to MSc, PhD Candidate Georgios Kandilogiannakis for his invaluable contribution to my thesis. His generosity in providing me with the necessary raw data and metadata for analysis was instrumental in the success of my project. The data and information provided by him helped to enrich the graphs and figures in my thesis, making it more comprehensive and informative. I am deeply grateful for his support and assistance throughout the course of my research.

Section 1: Theoretical Part

1.1 Historical Overview of Power Systems

The history of power systems can be traced back to the early 19th century, when the first electrical power systems were developed to provide energy for industrial processes. In 1831, Michael Faraday discovered electromagnetic induction, which led to the development of the first electrical generators. These generators used mechanical energy to produce electricity, and they were used primarily in large industrial plants. In 1879, Thomas Edison developed the first practical incandescent light bulb, which led to the creation of the first electrical power distribution systems. Edison built a direct current (DC) power station, the Pearl Street station in New York City, which supplied electricity to customers within a 1-square-mile radius [16]. However, DC systems had limited transmission capabilities and were not suitable for long-distance power transmission. In 1885, George Westinghouse developed the first alternating current (AC) power systems, which allowed for long-distance power transmission. AC systems quickly replaced DC systems as the dominant power transmission technology.

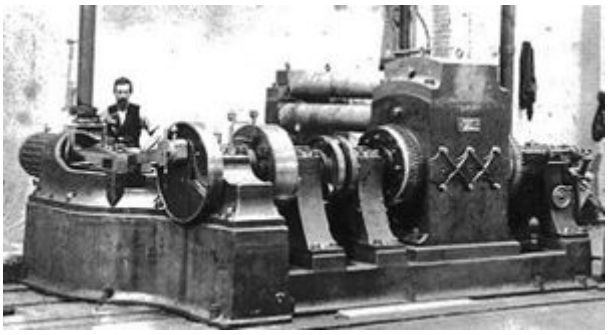

Figure 1 : Pearl Street station

In the early 20th century, large power stations began to be built to generate electricity for entire cities. Hydroelectric power plants were also developed, which harnessed the energy of falling water to generate electricity.

As electricity became more widely available, it transformed many aspects of daily life, from lighting to transportation. The development of power electronics and solid-state devices in the 20th century led to the creation of more efficient and reliable power systems [26].

Over the years the power generation industry has expanded rapidly. Technological advances made in the design of the various components of energy systems were incorporated into each new component installed, resulting in rapid upgrading of equipment [16]. This brings us to modern EPS (Electric Power Systems), which are a complex electricity network that must operate in a way that is safe, reliable, environmentally friendly and provide good quality electricity at the lowest possible price.

1.2 Electric Power System (EPS)

A modern electricity power system (EPS) is the set of all those facilities used to supply electricity to a set of consumers in a safe, reliable and environmentally friendly way. The main functions of such a system are the generation of electricity, transmission via high and medium voltage cables and distribution to consumers. Once the electricity has reached the consumers, it is converted into other forms to be used appropriately to meet the needs of the electrical installation [16].

It is obvious that nowadays the requirements in the electricity industry are constantly changing and the need to create and develop those systems that will meet them is imperative. These requirements stem from the vision of an environmentally friendly economic system that makes maximum use of renewable energy sources. The combination of existing means and the developments observed in the field of computers and communications are leading to the evolution of the existing network into a decentralized local network along the lines of a microgrid or an energy community.

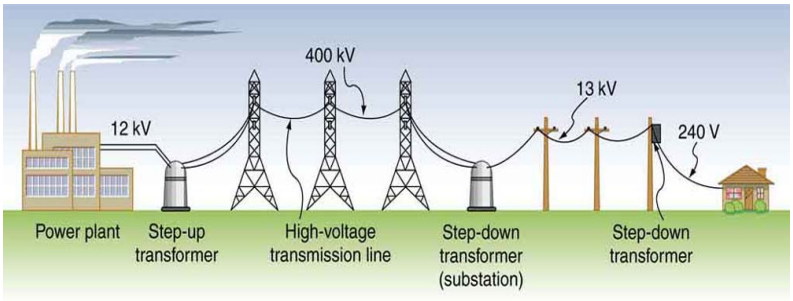

Figure 2 : Electrical Power System

1.3 Load Forecasting Need

Forecasting is a problem that applies to almost every industry in the world. Airlines try to forecast the number of passengers to plan flights, breweries forecast beer consumption to plan production. Retailers forecast the demand for fashion items to decide which discount to offer customers. Brokers predict stock prices to invest their and their clients' money more safely. Clearly, prediction is an essential ingredient for most human activities and the electrical industry is no exception to this.

Electrical load forecasting is a major activity for each country as it contributes to the proper operation and development of the electric power system (EPS). Load forecasting of both demand and generation leads to the determination of a country's energy sufficiency and contributes to planning for the next day, grid planning. Recently, geopolitical turbulence on the European continent that has led to an energy crisis has made load forecasting imperative. According to the European Union Institute for Security Studies, the balance between security and sustainability of the energy sector has led all Member States to take measures to combat climate change and ensure sufficiency. The integration of smart grids and the penetration of ever-increasing renewable energy sources (RES) is a roadmap for tackling the energy crisis [16].

More specifically, load forecasting brings benefits to consumers. Consumers, through an accurate load forecast of their building, have the ability to regulate and reduce the energy consumption they spend. Furthermore, having identified those periods of the day when high consumption occurs, the owner has the ability to adjust the energy behavior of the building by programming the installation's smart appliances to operate at times when energy demand is low, with numerous economic benefits. At the same time, there are also significant advantages for the network. Two of the most important are that the grid can have a real-time reliable estimate of the load that it will have to manage while also being able to identify those times of the day when the highest consumption occurs. This feature allows the grid to produce a sufficient amount of energy without losses. In addition, another important element worth mentioning is that the grid can, where feasible, be used to generate energy from renewable sources [16].

Finally, as can be easily understood from what has already been mentioned above, the need for algorithms that achieve satisfactory electric charge prediction is an open and constantly evolving scientific subject.

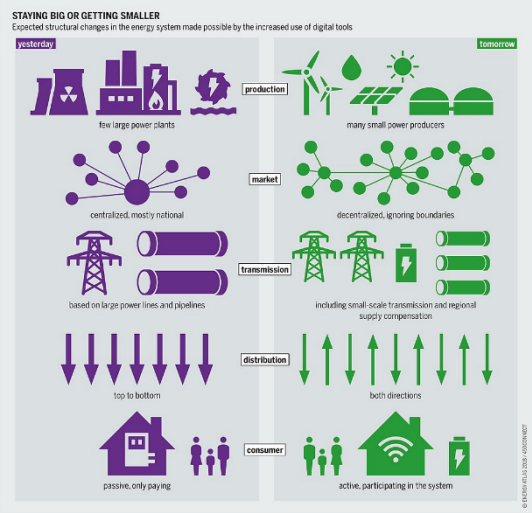

Figure 3 : Smart Grid

1.4 Load Forecasting

Electricity demand forecasting is the process of predicting future electricity consumption. The forecast horizon refers to the length of time into the future that the forecast is being made for.

i. Very-short-term forecasting typically covers a time frame of less than an hour and is used for balancing supply and demand in real-time.
ii. Short-term forecasting covers a time frame of a few hours to a day and is used for scheduling power generation and transmission.
iii. Medium-term forecasting covers a time frame of a few days to a few months and is used for system planning and procurement of resources.
iv. Long-term forecasting covers a time frame of a year or more and is used for long-term planning and infrastructure development.

Each of the forecast horizon has its own unique set of challenges and methodologies are used accordingly [19]. With the increased penetration of renewable energy sources, the forecasting of electricity demand is becoming more complex and accurate forecasting is crucial to ensure a stable and reliable power supply.

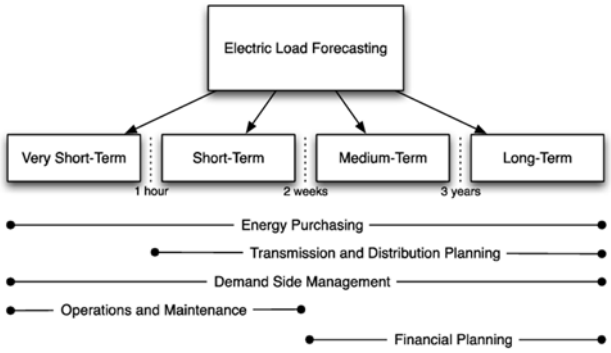

Figure 4 : Load forecasting types

Short-term load forecasting using deep learning (DL) techniques is the purpose of this paper.

1.5 Factors significantly affecting the load

The exploitation of the electrical load has many components. A significant part of the electricity is consumed by industry. It is also used by individuals to meet basic needs such as heating, lighting, etc. Another important part is used by public utilities for street lighting, public transport, etc. We can distinguish the factors that influence the load in the following categories:

A) Financial factors

B) Temporal factors

C) Weather factors

D) Random factors

The influence of all the above factors should be considered when creating a load forecasting model [2]. The economy as an influencing factor has an important role in shaping electricity demand. The prosperity or decline of the industrial sector in a country, the growth or saturation of the appliance market, the primary sector and changes in it and the Economy in general, have a serious impact on the rate of increase or decrease of the load on the System. Time is another factor that plays an important role as seasonal changes affect the load, as do changes in time during the day or the change from winter to summer. The weekly and daily periodicity of load is due to people's work and their holidays from it. For example, on the days corresponding to Saturday and

Sunday the demand for the load is reduced. In addition, the weather causes significant changes as the temperature has a significant impact on the load shape. Humidity, rainfall and wind are also factors that have an influence. Finally, factors such as random disturbances in the operation of the system, generated by the whole range of consumers, whether they belong to the small (households, popular TV programs, etc.) or large (industry) category, affect the operation of the system and add uncertainty to the forecasts [20].

1.6 Neural Networks

The term Neural Networks describes a number of different mathematical models inspired by biological models, i.e., models that try to mimic the behavior of neurons in the human brain.

Since the 19th century, scientists have admitted that the brain is made up of discrete elements, called neurons, that communicate with each other. Neurons are the basic building block of the human brain. It is estimated that the brain contains approximately 10 billion neurons arranged in groups, each of which constitutes a physical neural network. The human brain contains hundreds of physical neural networks, each containing thousands of interconnected neurons with an average number of connections per neuron of 1000 to 10,000 [10].

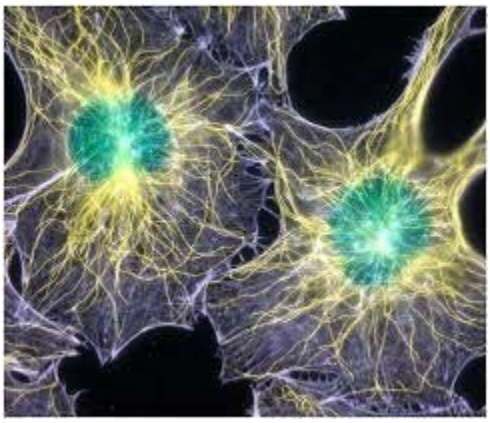

Figure 5 : Human neurons

A neuron is separated from other cells by a membrane and has the ability to carry electrical signals from that neuron to other neurons which it communicates

Each neuron consists of 3 main parts:

- the dendrites, which act as input channels for the neuron,
- the main cell body ,
- the cell-neuroaxis which connects a neuron to other neurons.

The axon of one neuron carries signals to the dendrites of neighboring neurons through the junction called the neuroaxon terminal or synapse. A neuron can receive signals from one set of neighboring neurons through the dendrites, process them and feed its output through the axon to another set of neighboring neurons. The signals coming through the dendrites are "weighed" and the results are added up. When the sum exceeds the threshold level (threshold value), the neuron generates an output (in the form of a nerve impulse or electrical signal) on its axon, which is then transferred through the synapses to the neighboring neurons [10].

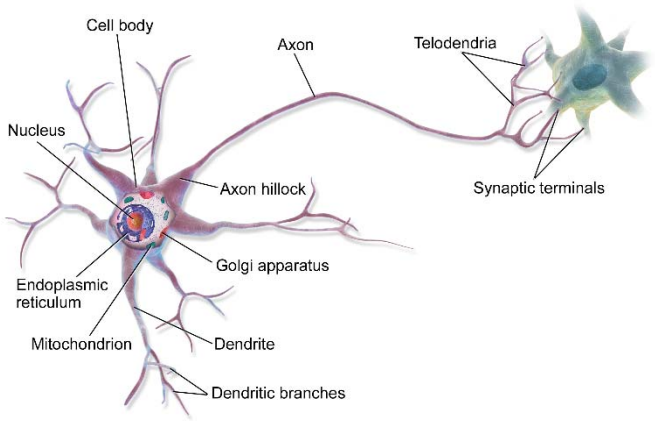

Figure 6 : Schematic diagram of a typical neuron

To generate a signal, the neuron receives input signals that affect its potential by varying it. When the cumulative potential exceeds a certain threshold (varying from cell class to cell class between - 40 mV and - 75 mV), the neuron is excited and produces the electrical signal. The neuron always

carries the electrical signal in a predictable and stable direction. There are two distinct states of signals:

i. Resting potential
ii. Energy potential

Signals received by a neuron are altered by the electrical characteristics of the contacts of the synapses, so that some are blocked and others are allowed to propagate. The electrical characteristics of the synapses constitute some kind of information unique to each neuron. In this way the information held by a network is distributed to its neurons [10]. The transmission of information is based on an energy potential determined not by the type of signal, but by the pathway of the brain through discrete communicating neurons through the signal passes.

With all the knowledge and tools that we have available to us today, we can't replace a brain with a computer. Or can we?! Elon Musk, the world's greatest visionary and richest man, has made it known that his company Neuralink, which specializes in interfacing the human brain with a computer, is only six months away from its first human trials!

1.6.1 Simulation of natural neurons with artificial neural networks

The mathematical models of artificial neural networks, in full correspondence with biological ones, consist of several simple and highly interconnected processing units, organized in layers [5]. Artificial Neural Networks (ANNs) process information dynamically in response to external stimuli (inputs). Each artificial neuron consists of several inputs x_i and a single output y. Each input x_i is 'weighted' with a weight w_i and the results are summed up by means of a summation function F:

$$F = \sum_{i}^{n} x_i w_i$$

The artificial neuron gives output via the transfer function only when the weighted sum of the inputs is greater than a certain threshold value θ when:

$$\sum_{i}^{n} x_i w_i - \theta > 0$$

An artificial neuron is a simplified model of the physical neuron in that the interconnection weights form the electrical characteristics of the synapse contact and the threshold value simulates the saturation behavior of the physical neuron [10].

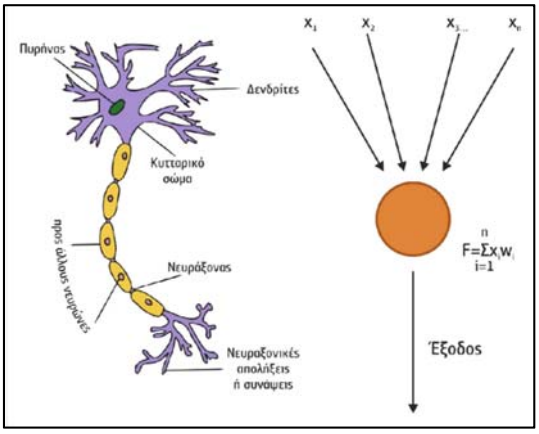

Figure 7 : Natural neuron in relation to the elementary artificial neuron

One of the simplest ANNs that simulate the physical neuron is the basic Perceptron, i.e. an ANNs consisting of a single neuron [9]. The output α of the Perceptron for an input vector x= (x_1, x_2,.., x_n) is given by the transition function g as follows:

$$a = g\left(\sum_{i=1}^{n} x_i w_i\right)$$

Brief description of an ANN:

- ANN are usually organized in layers. The intermediate layers are called hidden.
- Layers consist of a number of units or nodes that are interconnected in such a way that one unit has links to many other units at the same or another level.

- Units affect other units by stimulating or inhibiting their activation. To achieve this, the unit receives the weighted sum of all inputs through the links leading to it and produces a single output via the transition function if the sum exceeds a threshold value.
- Inputs are presented to the network through the input layer which communicates with one or more hidden layers the hidden layers are connected to the output layer from which the response is extracted.

Key elements of the architecture of the ANNs that need to be defined during their creation [10]:

- the number of intermediate hidden layers,
- the number of units (or nodes) per layer,
- the way the modules are connected to each other,
- the activation value (threshold value),
- the form of the transition function,
- the values of the initial weights between units,
- the algorithms (training rules) used to strengthen the links between units during the training process.

1.6.2 Historical Review

The milestones in the evolution of the field of ANNs are the following:

1943 - The First Concept of a Neural Network: In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts published a paper titled “A Logical Calculus of the Ideas Immanent in Nervous Activity” which introduced the first concept of a neural network.

1949 - The First Artificial Neuron: Donald Hebb proposed the concept of the artificial neuron in 1949.

1957 - The First Neural Network Computer: In 1957, Frank Rosenblatt developed the first neural network computer called the Perceptron.

1965 - Backpropagation: In 1965, Paul Werbos proposed the idea of backpropagation, which is a method of training a neural network [21].

1980 - The First Convolutional Neural Network: In 1980, Kunihiko Fukushima developed the first convolutional neural network (CNN) for recognizing handwritten characters.

1998 - Long Short-Term Memory: In 1998, Sepp Hochreiter and Jürgen Schmidhuber introduced the long short-term memory (LSTM) recurrent neural network.

2006 - Deep Learning: In 2006, Geoffrey Hinton and Ruslan Salakhutdinov introduced the concept of deep learning.

2012 - Geoffrey Hinton and his team demonstrate the power of deep learning with AlexNet.

2015 - Google's AlphaGo AI defeats world champion Lee Sedol at the game of Go.

2017 - Google's AI system AlphaZero defeats the world's best chess engine.

2022 - ChatGPT is the most advanced language models available and has been able to achieve state-of-the-art results on a wide range of NLP tasks.

1.6.3 Architecture of ANNs

The most common type of ANN is the feedforward neural network, in which information flows in one direction from input layers, through hidden layers, to output layers. The nodes in the input layer represent the input features, while the nodes in the output layer represent the predictions or decisions made by the network. The hidden layers are responsible for extracting complex features or patterns from the input data [9].

The strength of the connections between neurons, known as weights, are adjusted during training to optimize the performance of the network. This process is known as backpropagation, which uses gradient descent to update the weights in such a way that the network's predictions are as close as possible to the true values.

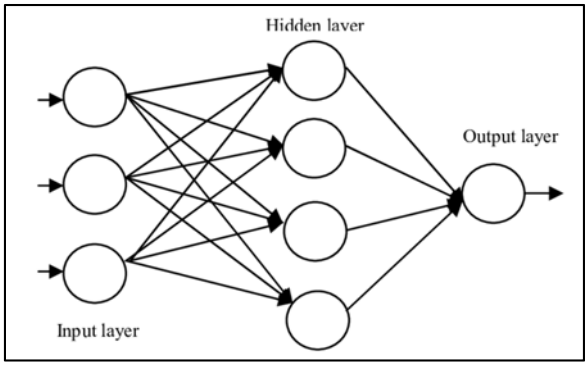

Figure 8 : Feed forward ANN architecture

Another popular type of ANN is the recurrent neural network (RNN), which is designed to process sequential data such as time series or natural language. RNNs have feedback connections, which allow information to flow in a loop and the network to maintain a "memory" of past inputs. The architecture of RNN typically consists of a hidden state that is passed from one time step to the next, along with the input at that time step, and the output is generated based on the current input and hidden state. One of the most popular variations of RNN is LSTM (Long Short-Term Memory) which uses gates to control the flow of information through the network and addresses the problem of vanishing gradients. This makes RNNs well suited for tasks such as language translation and speech recognition.

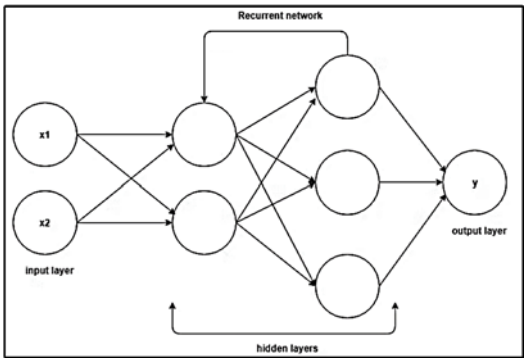

Figure 9 : Recurrent NN architecture

There are many other types of ANNs, such as convolutional neural networks (CNNs) which are designed to process images and are commonly used in computer vision [17]. They are designed to process data with a grid-like topology, such as an image, where the spatial relationship between the pixels is important. The architecture of a CNN typically consists of multiple layers of convolutional and pooling layers, followed by one or more fully connected layers. In the convolutional layers, filters are applied to the input data to extract

features, and the pooling layers are used to reduce the dimensionality of the data. These layers are designed to automatically and adaptively learn spatial hierarchies of features from input data [14].

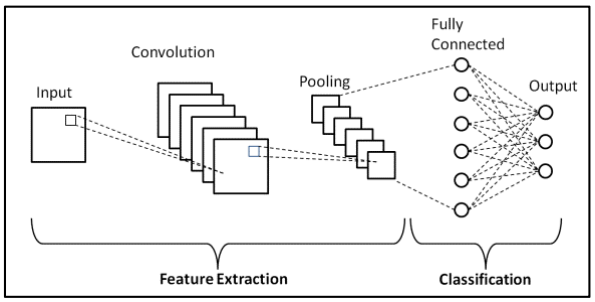

Figure 10 : Convolutional NN architecture

1.6.4 Multi-layer ANNs

A common feature of the structure of multi-level ANNs is that they have at least one hidden layer. The nodes of the different layers can be fully connected, i.e. each node of one layer is connected to all nodes of the next layer or partially connected [8]. ANNs are further characterized by the way in which their nodes are connected, as mentioned in the previous section on ANNs Architecture. In the majority of applications, single-hidden-layer feed-forward networks with fully connected nodes are used.

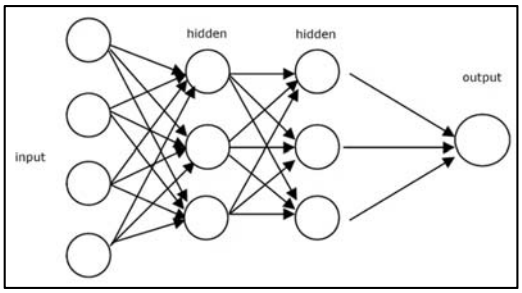

Figure 11 : Multi-layer ANN

1.6.5 Transfer functions

In artificial neural networks (ANNs), transfer functions, also known as activation functions, are used to introduce non-linearity into the network, allowing it to learn and represent more complex patterns and relationships in

the input data. There are two main types of transfer functions: linear and non-linear [10].

Linear transfer functions are functions that have a linear relationship between the input and the output, meaning that the output is proportional to the input. Linear transfer functions include the identity function, hard limiter, step function, signum function and piecewise linear functions. These functions are simple and computationally efficient, but they can't introduce non-linearity and can't model complex data distributions.

i. Identity function: It is a linear transfer function that simply returns the input without modification.
ii. Hard Limiter: It is a linear transfer function that returns a fixed value if the input is above a certain threshold, and a different fixed value if the input is below the threshold.
iii. Step function: It is a linear transfer function that returns a fixed value if the input is above a certain threshold, and a different fixed value if the input is below the threshold.
iv. Signum function: It is a linear transfer function that returns -1 if the input is negative, 0 if the input is zero, and 1 if the input is positive.

Non-linear transfer functions are functions that have a non-linear relationship between the input and the output, meaning that the output is not proportional to the input. Non-linear transfer functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) functions. These functions are computationally more expensive but can introduce non-linearity and can model complex data distributions.

i. Sigmoid: A sigmoid function maps any input value to a value between 0 and 1, making it useful for output layers that represent probability or likelihood.
ii. Tanh (hyperbolic tangent): A tanh function maps any input value to a value between -1 and 1, it is similar to sigmoid function but the output range is symmetric around the origin.
iii. ReLU (Rectified Linear Unit): ReLU is a simple but effective transfer function that maps negative values to 0 and positive values to themselves. It is widely used in hidden layers of neural networks to introduce non-linearity.

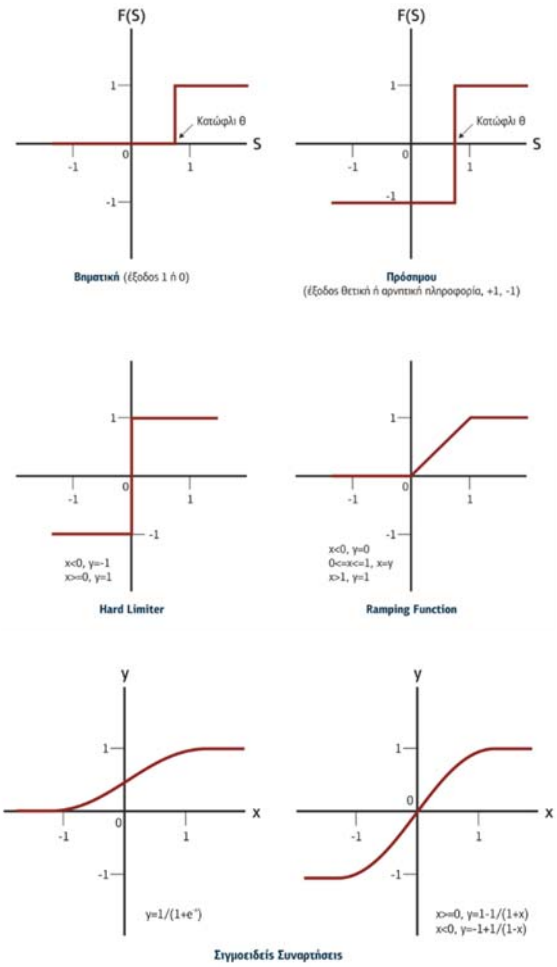

Figure 12 : Transfer functions

1.6.6 RNN

A recurrent neural network (RNN) is a type of artificial neural network where the connections between nodes form a directed graph along a time sequence. This allows it to exhibit temporal dynamic behavior. Unlike other networks, RNNs can use their internal state (memory) to process sequences of inputs. The logic of this kind of network is that people do not start thinking from scratch every second as persistent thoughts have a continuum. Traditional neural networks cannot do this and essentially look like a big vacuum. Recurrent neural networks have loops that allow them to retain information. They are essentially networks where the iterative loops that occur allow them to have this memory-preserving property.

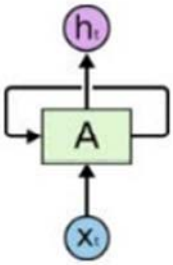

Figure 13 : Recurrent NN architecture

Figure 13 shows an example of such a network architecture. An iterative process (loop) allows information to flow from one layer to another. An iterative neural network can be considered a multiple copy of the above network where each layer will pass information to the next step following this iterative process before it is transferred.

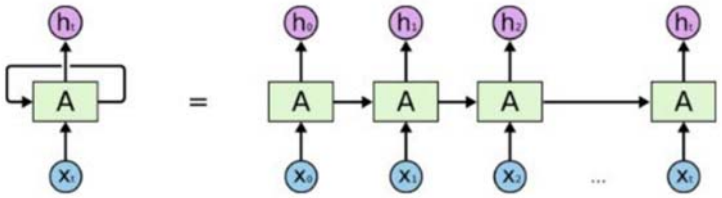

Figure 14 : Recurrent NN architecture

This chain form (Figure 14) reveals the association of these networks with lists and plausibilities. It is essentially the natural architecture of the RNN. In recent years there has been a great deal of success in applying recurrent neural networks to various kinds of problems such as speech recognition, language modelling, translation, image projection, etc [2].

RNNs have the potential to connect previous information to the current task, such as using previous video frames to understand the current frame, making them useful for various applications. However, the effectiveness of this capability can vary depending on the specific task and implementation.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in “the clouds are in the *sky*,” we don’t need any further context – it’s obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it’s needed is small, RNNs can learn to use the past information.

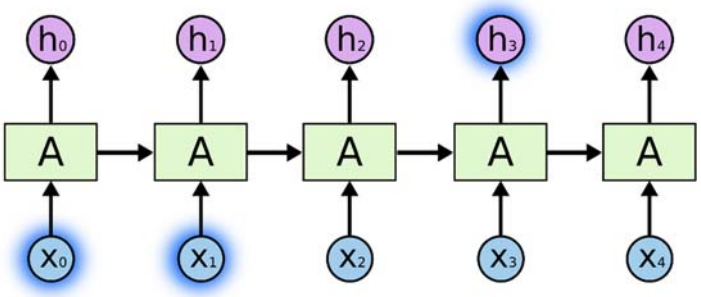

Figure 15 : RNN (small gap)

But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent ***French***." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

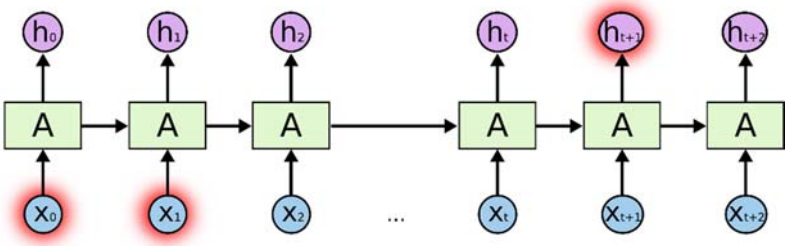

Figure 16 :RNN (large gap)

1.6.7 LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are designed to process sequential data, such as time series or natural language. LSTMs are able to handle the problem of vanishing gradients, which is a common issue in traditional RNNs, by introducing a memory cell and three gates: the input gate, forget gate, and output gate.

The memory cell is a unit that stores information and allows it to persist over long periods of time. It is represented by a horizontal line in LSTM diagrams and its value is updated at each time step.

The input gate controls the amount of information that is allowed to enter the memory cell. It is represented by the sigmoid function and it takes into account the current input and the previous hidden state to decide which information should be stored in the memory cell.

The forget gate controls the amount of information that is discarded from the memory cell. It is represented by the sigmoid function and it considers the current input and the previous hidden state to decide which information should be forgotten.

The output gate controls the amount of information that is read from the memory cell. It is represented by the sigmoid function and it takes into account the current input and the previous hidden state to decide which information should be used to update the hidden state.

At each time step, the values of the input, forget, and output gates are calculated and used to update the memory cell and the hidden state. The hidden state is then passed on to the next time step and used as input for the next LSTM unit in the network [2].

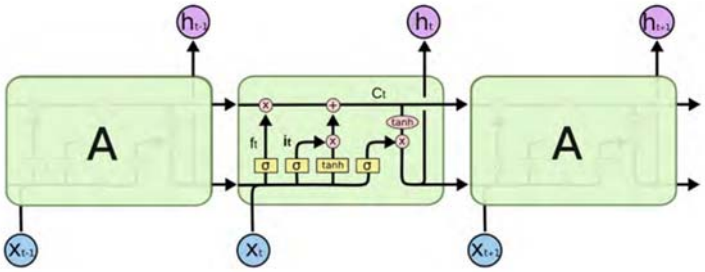

Figure 17 : LSTM architecture

1.6.8 GRU

The gated recurrent units were created in 2014 by Kyunghyun Cho. GRUs are based on the same logic as LSTMs except that they use fewer parameters and do not provide an output gate. The accuracy of GRU networks appears to be very close to that of LSTM networks in many problems. Also in specific cases it has been observed that GRU networks achieve better times than their LSTM counterparts.

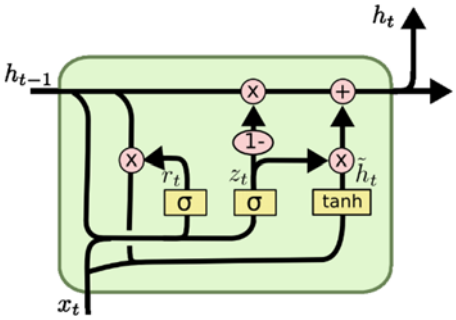

Figure 18 : GRU architecture

1.6.9 Operation of ANNs

The neural network consists of two stages of operation. The first is the training of the neural network, i.e. the process in which the network learns the data (training set), where a set of observation values are entered at the input of the network and at the end of the training process results are extracted. Using the learning set and the appropriate algorithm, the neural network is trained by calculating the weights and polarizations if they exist. The purpose of the procedure is to minimize the error in the prediction. The second stage is the prediction process. At this point a test set is created. To calculate the values in the output, the neural network considers the values from the weights and possible biases calculated during the learning process. Finally, to calculate the statistical prediction error, the predicted values at the output shall be considered by comparing the predicted values with the desired values from the control set [1].

For the neural network to perform optimally, the following factors should be considered, which are of great importance for its proper operation.

i. The definition of its architecture considering the number of hidden layers, the number of neurons in each layer, the activation function, the algorithm for training and the number of iterations.
ii. The determination of the percentage, from the data sets available, to be used as a training set and as a test set. It is usual to use either 75% of the data for training and 15% for testing, or 80% for training and 20% for testing.
iii. The ability of the neural network to predict with low statistical prediction error both during training and testing. The neural network training

process is iterative and as a result it is a process which requires a large amount of time, in particular when there is a large amount of data or when the number of neurons in the network layers is large. The number of neurons in the hidden layer is determined by trial and error as there is no specific procedure by which the number of neurons can be decided. In the case where the number of neurons in the hidden layer is small the neural network cannot learn efficiently due to the complexity of the relationships. If the number of neurons is too large, then we face other kinds of problems such as overfitting. The number of neurons as we can see is not constant but changes in every different problem case but also depends on the volume of data.

1.6.10 Training of ANNs

Neural networks have the ability to learn from input data, and through the internal processes, produce accurate results. There are two main types of learning methods used in neural networks: supervised and unsupervised learning.

A. **Supervised Learning**

This type is used either in regression to produce output data for some input data. The most common algorithms are logistic regression, naive bayes, support vector machines, artificial neural networks, and random forests.

B. **Unsupervised Learning**

The basic function of this type of learning is clustering and is primarily used in photo recognition.

1.6.11 Accuracy check of ANNs forecasting

The accuracy check of the predictions extracted for a neural network is done by means of statistical errors. Error is defined as the uncertainty that exists in the measurement of a physical quantity: value ± uncertainty. There are many types of error, the most important of which are listed below.

A. **Mean Squared Error (MSE):** In statistics, the mean squared error is defined as the average squared difference between a measured value and the true value. The mean squared error is a function of risk and is mathematically defined as:

$$\mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\mathrm{A_i} - \hat{\mathrm{A}}_i)^2$$

B. **Root Mean Squared Error (RMSE):** it is the root of mean square error and its mathematical expression is:

$$\mathbf{RMSE} = \sqrt{MSE}$$

C. **Mean Absolute Error (MAE):** in statistics, MAE is a measure of the difference between two values.

$$\mathbf{MAE} = |\mathrm{A_i} - \hat{\mathrm{A}}_i|$$

D. Mean Absolute Percentage Error (MAPE): The mean absolute percentage error is the average absolute error expressed as a percentage of the mean absolute error [15].

1.7 Time series

A time series is a sequence of data points dependent on time. That means that each data point has a timestamp assigned to it. Ideally, these data points are measured at constant intervals (e.g., every day) and in chronological order (e.g., Monday, Tuesday, Wednesday, etc.). Time series are usually numerical values, but they can also be categorical. Time series data usually comes in tabular format (e.g., CSV files) with a column for the timestamps and at least one for the time series values.

Time series is basically sequentially ordered data indexed over time. Here time is the independent variable while the dependent variable might be

- Stock market data
- Sales data of companies
- Data from the sensors of smart devices

- The measure of electrical energy load

To gain some useful insights from time-series data, we have to decompose the time series and look for some basic components such as **trend, seasonality, cyclic behavior, and irregular fluctuations.** Based on some of these behaviors, we are deciding on which model to choose for time series modelling [4]. Assume that we are having the time-series data of an airline passenger company, if we do an initial analysis on the data, we can find that in each year during particular periods of time, a particular pattern may be found (a seasonal pattern). Further investigating we may find that it was because vacations were happening in those months because of which families were traveling. Also, we may be able to find other insights like an increase/decrease in the passenger count (upward/downward trend), which may be related to some other factors that were affecting the airlines at that time.

Seasonality in time series refers to repeating patterns that occur at regular intervals, such as daily, weekly, monthly, or yearly. It is a common characteristic in many types of time series data, including sales, weather, and transportation data. Identifying seasonality in time series is important for accurate forecasting and understanding underlying patterns in the data. To remove seasonality from time series data, techniques such as seasonal decomposition, differencing and exponential smoothing with the Holt-Winters method can be used. It is important to note that not all time series have seasonality and it depends on the characteristics and context of the data [3].

Trend in time series refers to the general direction or movement of the data over time. It can be upward, downward, or flat. It is a long-term pattern in the data and it can be influenced by factors such as economic and demographic changes. Identifying trend in time series data is important for understanding the underlying patterns and making accurate predictions and forecasts. There are different methods to identify trend such as linear regression, moving averages, and exponential smoothing. In addition, it is important to note that in some cases, a trend can change over time, this is called a change point, and it's important to be aware of it when analyzing the data [3].

Cyclic behavior in time series refers to patterns that repeat over a period longer than a season, such as several years. It can be caused by

external factors such as economic cycles, or other factors that affect the data. For example, in an economic context, the cyclic behavior can be observed in gross domestic product (GDP), employment, and production, among others. Identifying cyclic behavior in time series data is important for understanding the underlying patterns and making accurate predictions and forecasts. There are different methods to identify cyclic behavior such as decomposition, spectral analysis, and wavelet analysis.

Irregular fluctuations in time series refer to random variations in the data that cannot be explained by the trend, seasonality or cyclic behavior. These fluctuations can be caused by unpredictable events such as natural disasters, unexpected changes in demand, or other unforeseen factors. Identifying irregular fluctuations in time series data is important for understanding the underlying patterns and making accurate predictions and forecasts.

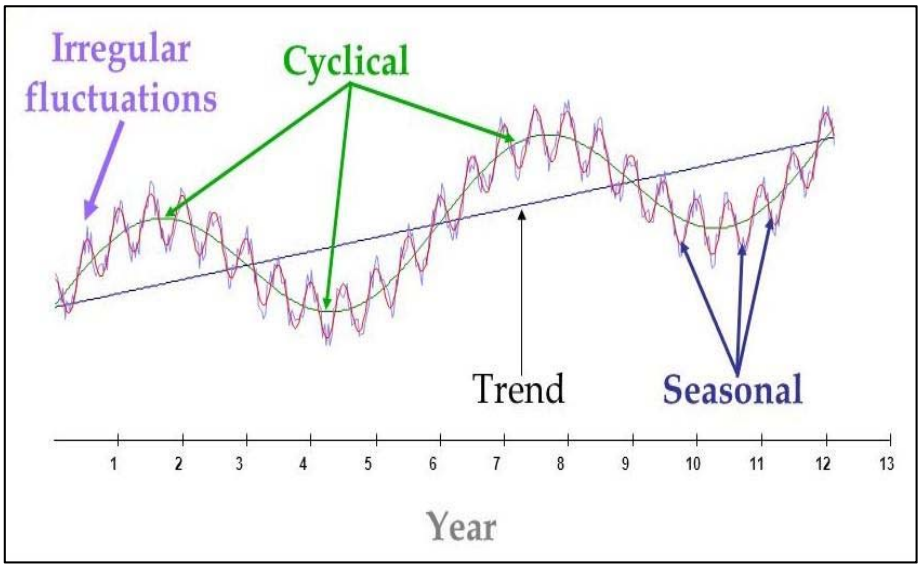

Figure 19 : Trend, Seasonality, Cyclic behavior, and Irregular fluctuations

The two main aspects of time series forecasting are:

a) Number of observed time series to predict: Time series forecasting can be performed on univariate data, which is a single time series, or multivariate data, which is multiple time series. In univariate forecasting, the focus is on a single variable, such as the sales of a specific product, while in

multivariate forecasting [12], multiple variables are considered, such as sales and weather.

b) Prediction time frame: Time series forecasting can be done for short-term or long-term predictions. Short-term forecasting is used to predict values in the near future, such as next month or next quarter. Long-term forecasting is used to predict values in the distant future, such as several years. The prediction time frame can influence the choice of forecasting model and the amount of data needed.

A time series is considered to be stationary if its statistical properties such as mean, variance and autocorrelation do not change over time. In other words, a stationary time series has a constant mean and variance, and the relationship between the observations and the time at which they occur is consistent. Stationarity is an important assumption for many time series models, as it simplifies the analysis and makes it easier to forecast future values. Non-stationary time series, on the other hand, can be more difficult to model and forecast [6].

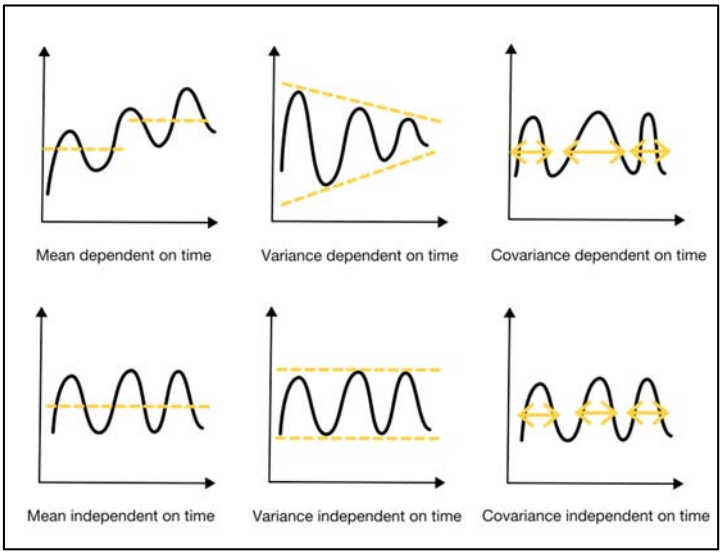

Figure 20 : Stationarity

Time series **analysis** and time series **forecasting** are related but distinct fields. Time series analysis is the process of understanding and modeling the underlying patterns and structures in a time series data. It involves techniques such as decomposition, trend analysis, and spectral analysis to identify and describe the characteristics of the time series. Time series forecasting, on the

other hand, is the process of using the knowledge gained from the analysis to make predictions about future values of the time series.

Both time series analysis and forecasting use similar techniques such as moving averages, exponential smoothing, and ARIMA models [7]. Both fields also rely on the assumption that the underlying patterns in the data are consistent over time. However, the main difference between the two is the goal of the analysis. Time series analysis aims to understand the underlying patterns in the data, while time series forecasting aims to make predictions about future values [5].

Section 2: Implementation

2.1 Clarification of important terms

For the implementation and training of the prediction algorithms, the TensorFlow library and specifically the Keras High-Level API was used.

2.1.1 TensorFlow

TensorFlow is an incredibly powerful tool from the most powerful Internet company in the world, Google, and is based on machine learning and neural networks. Essentially TensorFlow is an open-source Google neural network library, developed by the Google Brain team for many uses [11]. TensorFlow removes the need to create a neural network from scratch. So, since the foundation is already there, we can train TensorFlow with our own data and use the results we want.

2.1.2 Keras

Keras is an Open-Source Neural Network library written in Python that runs on top of TensorFlow [30]. It is designed to be modular, fast and easy to use. It was developed by a Google engineer. It is a useful library to construct any deep learning algorithm.

2.1.3 Google Colaboratory

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs. During the preparation of this thesis, only the free version of Google Colab was used. Colab sessions initialize with a K80 GPU and 12GB of RAM.

2.1.4 Power Bi

Power BI is an interactive data visualization software product developed by Microsoft with a primary focus on business intelligence. It is part of the Microsoft Power Platform [23]. Power BI is a collection of software services, apps, and connectors that work together to turn unrelated sources of data into coherent, visually immersive, and interactive insights. Data may be input by reading directly from a database, webpage, or structured files such as spreadsheets, CSV, XML, and JSON [27].

Many of the visualizations of this thesis have been produced by Power Bi. Chapter 3 gives an extensive description of this tool. As for the interactivity provided by this tool you can take a look at this report.

2.2 Data preparation

First, we import the python libraries pandas, numpy and matplotlib into Colab. Then we import the dataset. For the purpose of this project, load consumption data were first used from the ENTSO-E. ENTSO-E is the European Network of Transmission System Operators for Electricity and collects load consumption data from most European countries. The Greek transmission system operator (ΑΔΜΗΕ) publishes every day the load consumption in hourly intervals in MW for the entire Greek power system which are reported to ENTSO-E. However, because there were missing values and at the suggestion of the MSc supervisor, Ph.D. candidate Kandilogiannakis G., another dataset was used for the years 2013 to 2016 which did not contain any missing values. The dataset used for the training is shown below in code.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

dat="1in_1out_2013data=2015trn_2016tst_3"
data=pd.read_csv(dat, header=None)

data
```

	0
0	5525 5925
1	5040 5427
2	5011 5387
3	4817 5207
4	4685 5146
...	...
26083	7889 8937
26084	7711 8527

26088 rows × 1 columns

This is a dataset with hourly (index) load values for the years 2013 to 2015. We renamed the load column to LOAD.

```
#Rename the single column
 data.columns =['LOAD']
 data
```

	LOAD
0	5525 5925
1	5040 5427
2	5011 5387
3	4817 5207
4	4685 5146
...	...
26083	7889 8937
26084	7711 8527
26085	7291 7695
26086	6690 6712
26087	6179 6211

26088 rows × 1 columns

As shown by the dataset name (***1in_1out...***) we can see above that the LOAD column is duplicated so we deleted one of the 2 vectors.

```
#Split the single column into two new columns

DATA=data['LOAD'].str.split(' ', expand=True)

DATA.shape

    (26088, 2)

#Names of the new columns

DATA.columns =[ 'To Delete' , 'Actual Load – TRAINING' ]

DATA
```

	To Delete	Actual Load - TRAINING
0	5525	5925
1	5040	5427
2	5011	5387
3	4817	5207
4	4685	5146
...	...	...
26083	7889	8937
26084	7711	8527
26085	7291	7695
26086	6690	6712
26087	6179	6211

26088 rows × 2 columns

Now we have a table with 26088 lines and a single column, that of Actual LOAD for training. Recurrent neural networks as already mentioned above do NOT need to model the input vector x to predict an output y as time series analysis works. They model the vector y to predict an output y. This is the most powerful weapon of RNNs and that is why they are the most popular of their kind.

```
#Delete the one column

del DATA['To Delete']
DATA
```

	Actual Load - TRAINING
0	5925
1	5427
2	5387
3	5207
4	5146
...	...
26083	8937
26084	8527
26085	7695
26086	6712
26087	6211

```
26088 rows × 1 columns


DATA.isnull().sum()
    Actual Load – TRAINING:0

DATA.dtypes
    Actual Load - TRAINING   object dtype:object
#String to integer
df=DATA['Actual Load - TRAINING'].astype(str).astype(int)
```

As we can see above there is no null value. Furthermore, it is useful to convert the column of interest from string to integer. A snapshot of all the load values for the 3 years of training is shown below.

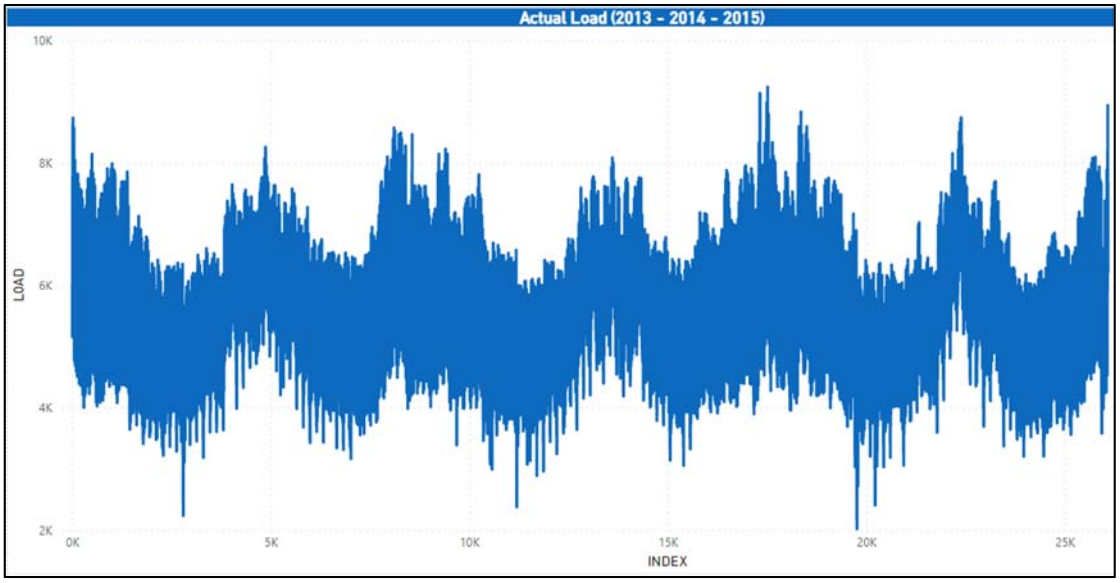

Figure 21 : ACTUAL LOAD - TRAINING DATA

We detect a cyclical behavior in the data which is due to seasonality. We observe that approximately every 9000 index values the pattern repeats. Specifically for 365 days x 24 hours = ~9000 hours. Therefore, every year the pattern repeats itself. We then plotted the data for all 3 years using bar charts.

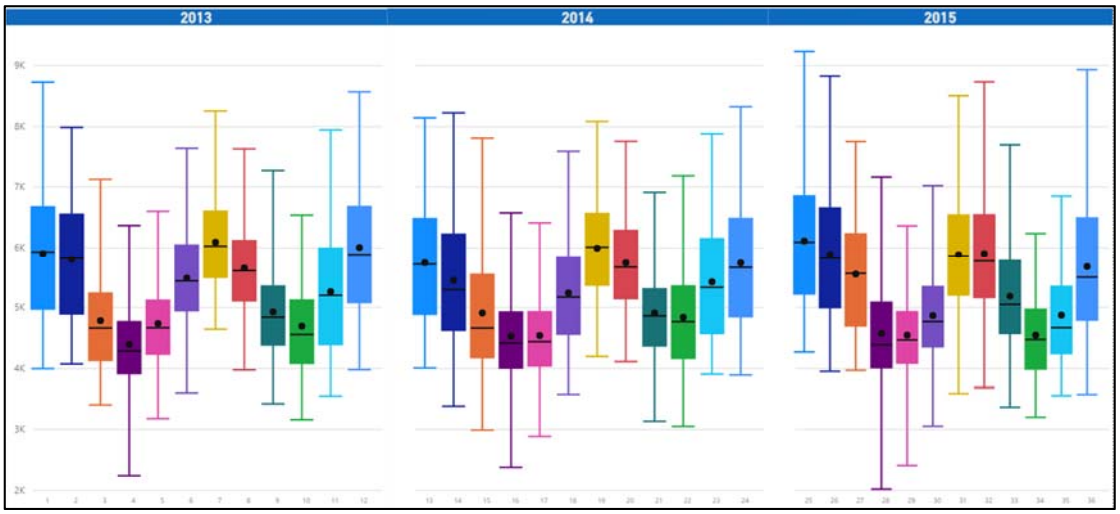

Figure 22 : Boxplots

In agreement with what we mentioned above, with the use of the boxplots we see a repeating pattern for the 3 years (2013 - 2014 - 2015). Remarkable here is that in the winter months the range of the load is greater in relation to the remaining 9 months of the year. The size of the boxplot on the winter months (number of months: 12-1-2) is larger in relation to all the rest. We also notice that within a calendar year we distinguish 2 seasons with peaks

(winter & summer) and 2 seasons with troughs (spring & autumn). This is called seasonality. Then we set aside one year separately for study.

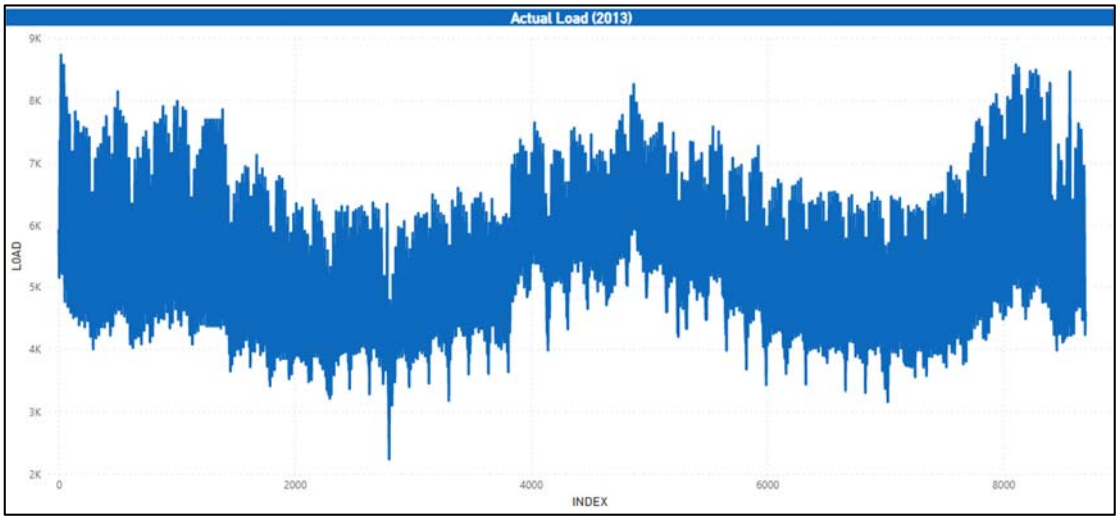

Figure 23 : ACTUAL LOAD - 2013

Seasonality here is obvious. For the year 2013, the summer (index ~4000-6000) and winter (index ~0-1500 & ~8000-9000) load demand was high in contrast to spring and autumn. Let's focus on a random day in the spring.

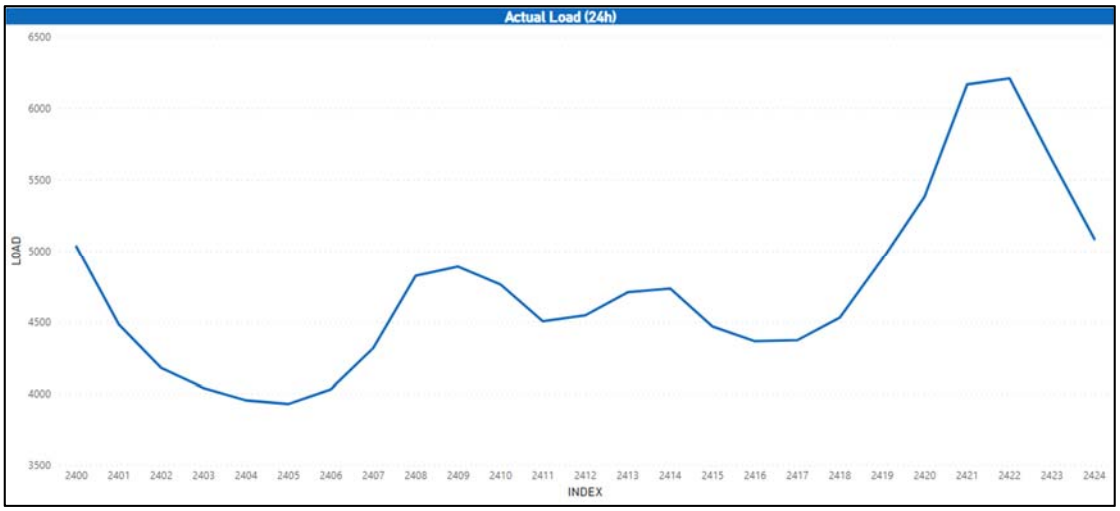

Figure 24 : ACTUAL LOAD - 24h

We notice that in the evening (9pm-10pm) demand is at its peak. To generalize these thoughts, we use the statsmodels library in python [13]. We used the following code.

```
#Extract and plot trend, seasonal and residuals
from statsmodels.tsa.seasonal import seasonal_decompose
decomposed = seasonal_decompose(df["Actual Load - TRAINING"],
freq=8760)
```

```
trend = decomposed.trend
seasonal= decomposed.seasonal
residual = decomposed.resid

plt.figure(figsize=(12,8))
plt.subplot(411)
plt.plot(df,label='Original',color='red')
plt.legend(loc='upper left')
plt.subplot(412)
plt.plot(trend,label='Trend',color='red')
plt.legend(loc='upper left')
plt.subplot(413)
plt.plot(seasonal,label='Seasonal',color='red')
plt.legend(loc='upper left')
plt.subplot(414)
plt.plot(residual,label='Residual',color='red')
plt.legend(loc='upper left')
plt.show()
```

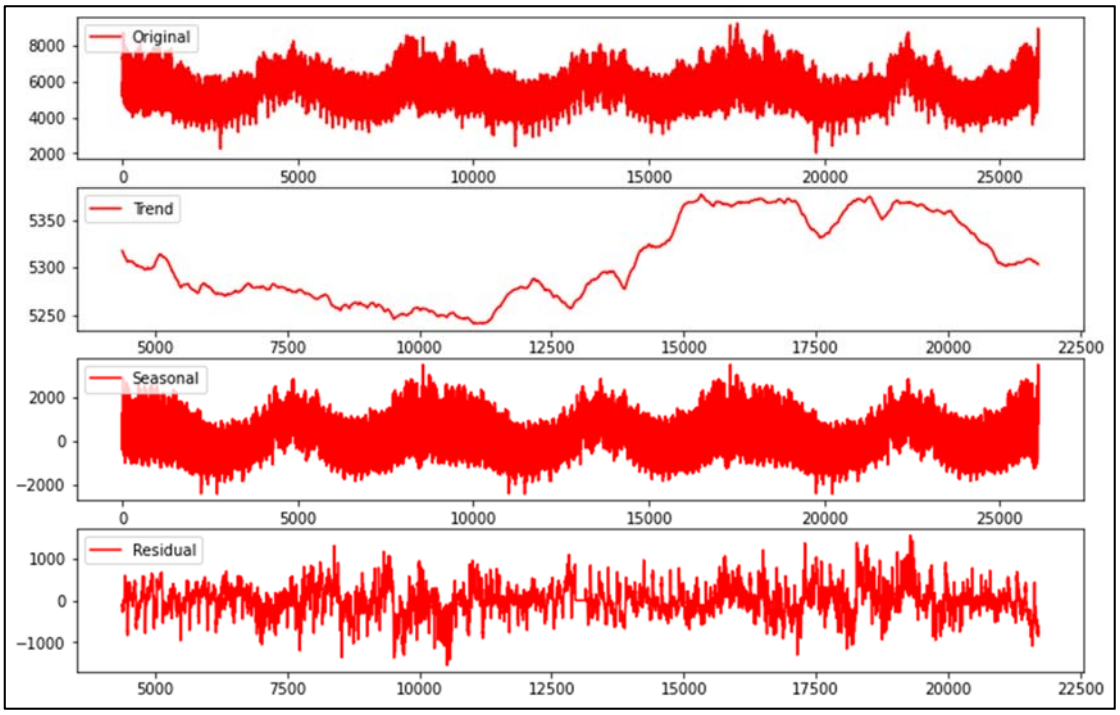

Figure 25 : Trend / Seasonal / Residual

As we can see, in the last two years of time-series (index >10.000) there is an upward trend over time. As for the seasonality, we can see that peaks & troughs which indicate summers/winters [peaks] and autumn/spring [troughs]. Regarding the residual part of the graph is what we have already mentioned in the previous chapter about irregular fluctuations on timeseries.

We did the same procedure for the testing dataset. The test dataset is for the year 2016 and is shown in the graph below. Then we use again the statsmodels library.

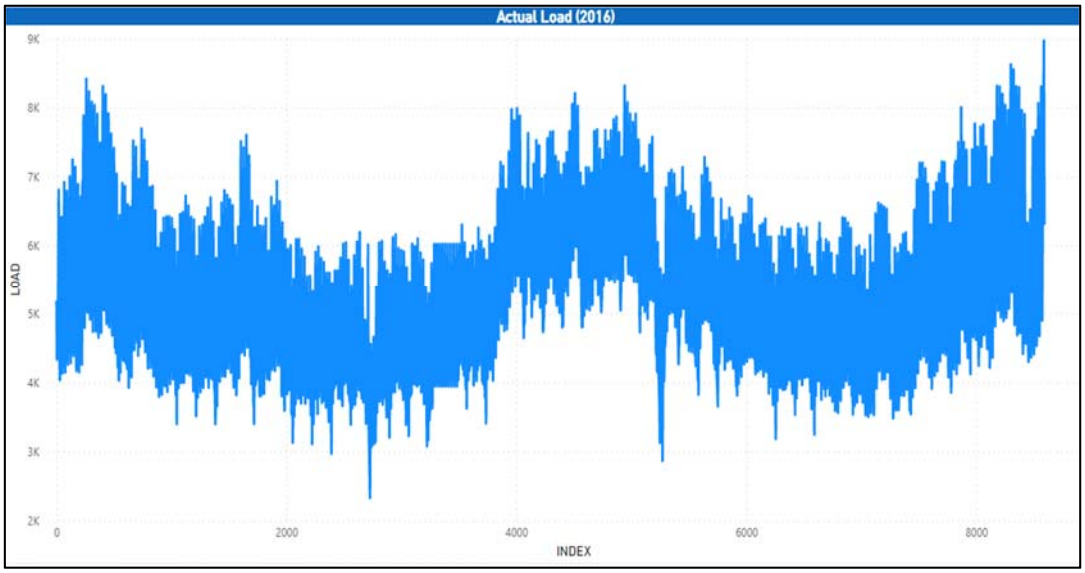

Figure 26 : ACTUAL LOAD - 2016

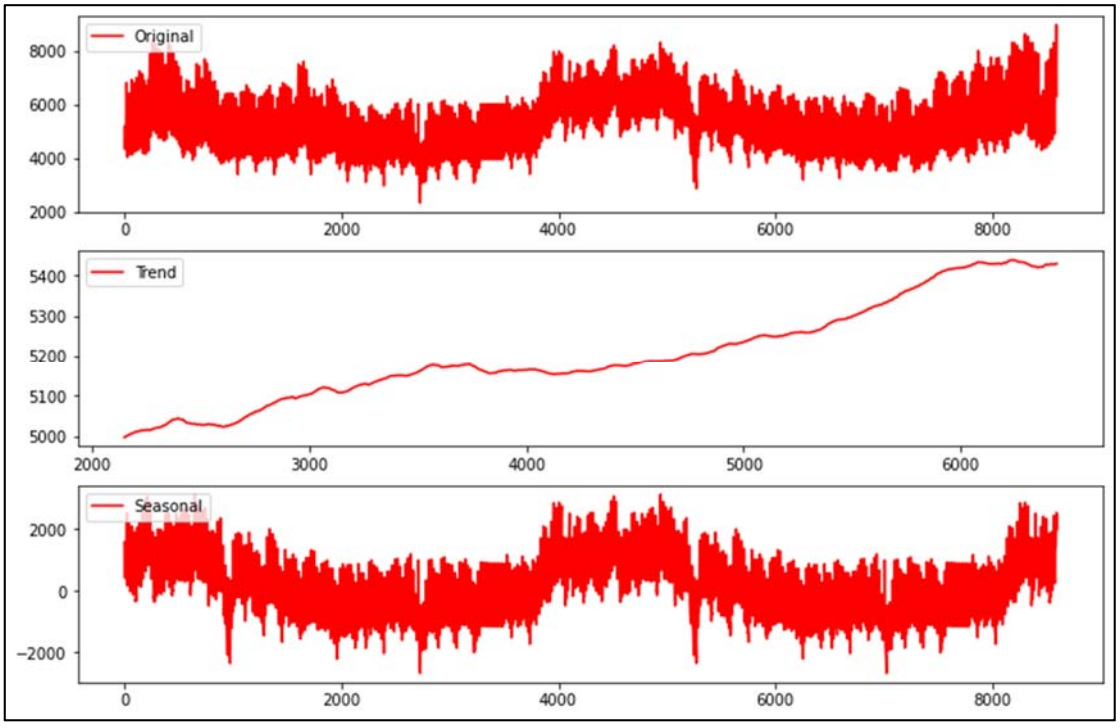

Figure 27 : Trend / Seasonal (2016)

The year 2016 recorded a steady increase trend in energy demand. Seasonality continues to be evident as in the training data. Peaks & troughs are detected in summer/winter and spring/autumn respectively.

2.3 Algorithms / Architectures

Initially we had to model the forecasting problem as a supervised learning problem in which given some input features x, our task would be to predict some output (target) variable y. In our case the target features or variable y are the future values of electrical load, while the input features are the n past values of electrical load. To this end, we need to define at this stage three key concepts that we used when building our models. The first is the output or ***forecasting length***, which is the number of time steps we need our model to be able to predict. Since we are interested in short-term forecasting this output length is set to 24 values each corresponding to an hour of the day, hence our models will only be able to predict load values in MW for the day ahead. The second concept is the input or ***past values length***, which is defined as the number of past timesteps our model will use as input to provide the forecasts. During the model's development stage, we applied different input length sizes ranging from 6 to 168 past values. Lastly, we use the ***sliding / rolling window*** method to produce forecasts. This method describes the way our data are used when training the model. Specifically in a sliding window forecast, we progressively use fixed length input arrays of n-past values of our input features to predict the next 24 future values. Since we are using neural network algorithms, we had to transform our data to [observations, input features], [observations, output features] format. In a sliding window forecast, each observation in the 2nd array holds the future 24 values of the corresponding observation in the 1st array. Each observation differs from the previous one by a fixed window size called the **sliding window size**. We applied two categories of models using 24h sliding window size and 1h sliding window size. The figure below depicts this process.

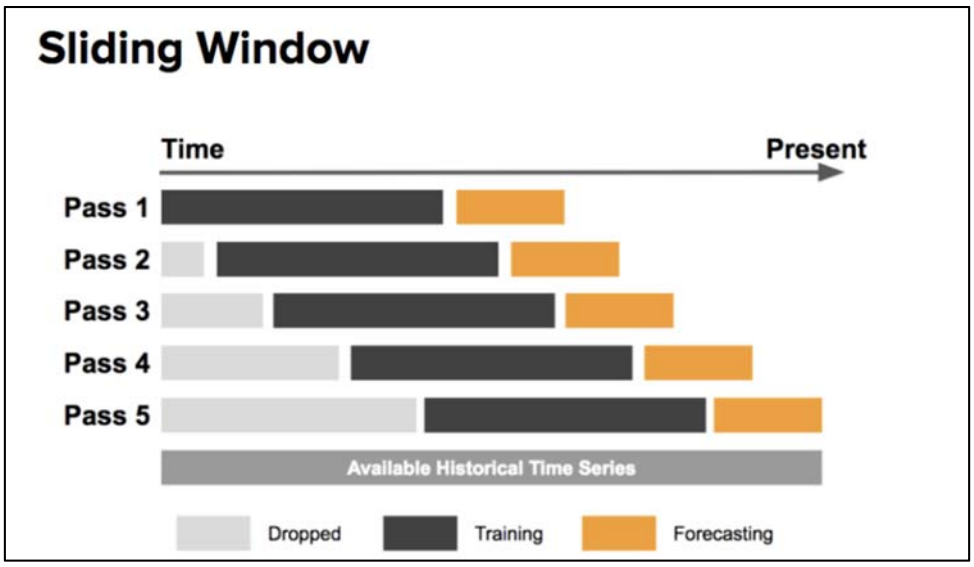

Figure 28 : Sliding window process

Before we feed our data to the neural network, we need to scale them to [0,1] range because, generally, it speeds up learning and leads to faster convergence. We did that via the MinMax Scaler function, which uses the following mathematical formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.3.1 Feed Forward Models

The first category of models we built were based on feed forward models using a sliding window of size 1. This means that every observation we have in our x and y arrays is shifted 1 hour ahead of the previous observation. The first model we created under this category uses 24h & 12h as the past values size, while the output length is 1h ahead.

Building the model step-by-step

As always at the beginning we introduce the main libraries we will use

```
import pandas as pd
import numpy as np
import math

import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import mean_squared_error,
mean_absolute_percentage_error
from keras.callbacks import ModelCheckpoint
from sklearn.preprocessing import MinMaxScaler


train = pd.read_excel("Actual Load - TRAINING.xlsx")

train
```

	Actual Load - TRAINING
0	5925
1	5427
2	5387
3	5207
4	5146
...	...
26083	8937
26084	8527
26085	7695
26086	6712
26087	6211

26088 rows × 1 columns

Since we're going to use neural networks for forecasting, we should convert our data in [samples, input_features] , [samples, output] format. In our case the output is the future values of electrical load, while the input will be all the past electrical load values.

```
features_request = {"window": [23]} #input size
#23+timeZero=24
build_df = build_features(train['Actual Load - TRAINING']
,features_request, target_lag=1,
include_tzero=True) # tzero (time_zero) refers to the current
time
build_df.index.names = ['Index']
build_df
# target lag shows how many hours ahead we want to predict.
In our case  choose to predict 1 value 1 hour
into the future
```

We normalized the data and then followed the same pipeline with the test data. We split the data into train & test and build the first model.

```
scaler_features2 =
MinMaxScaler().fit(build_df2[build_df2.colum
ns.values[:-1]]) # input features scaler
scaled_features2 =
scaler_features.transform(build_df2[build_df
.columns.values[:-1]])

scaler_label2 =
MinMaxScaler().fit(np.array(build_df2[build_df2.columns.v
alues[-1]]).reshape(-1, 1)) #output feature scaler
scaled_label2 =
scaler_label.transform(np.array(build_df2[build_df2.colum
ns.values[-1]]).reshape(-1, 1))

# Split data
x_train, y_train =scaled_features, scaled_label
x_test, y_test = scaled_features2, scaled_label2

#At this stage we can build a simple feed forward network
model that only uses the previous electrical load values to
predict 1h into the future.
model = Sequential()
model.add(Dense(100, activation='relu',
input_dim=x_train.shape[1]))
model.add(Dense(50, activation='relu'))
model.add(Dense(y_train.shape[1],activation='sigmoid'))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_24 (Dense)            (None, 100)               2500
 dense_25 (Dense)            (None, 50)                5050
 dense_26 (Dense)            (None, 1)                 51

=================================================================
Total params: 7,601
Trainable params: 7,601
Non-trainable params: 0
_________________________________________________________________
```

We then fitted the model based on some key parameters:

- epochs = we use earling stopping method
- batch size=360 (15 days)
- optimizer=Adam
- loss= mse

```
n_epochs = 1000 # don't care much about epochs since we use
early stopping
batch=360
model.compile(optimizer='adam', loss='mse')
checkpointer=
ModelCheckpoint(filepath="load_model_1h_pred.h5",
verbose=1,save_best_only=True)
es_callback=keras.callbacks.EarlyStopping(monitor=
'val_loss',patience=3) # early stopping
history = model.fit(x_train, y_train,
epochs=n_epochs, batch_size=batch, shuffle=True,
validation_split=0.20, verbose=0,
callbacks=[checkpointer,es_callback])
df_loss = pd.DataFrame(history.history)
```

```
df_loss[['loss','val_loss']].plot()
```

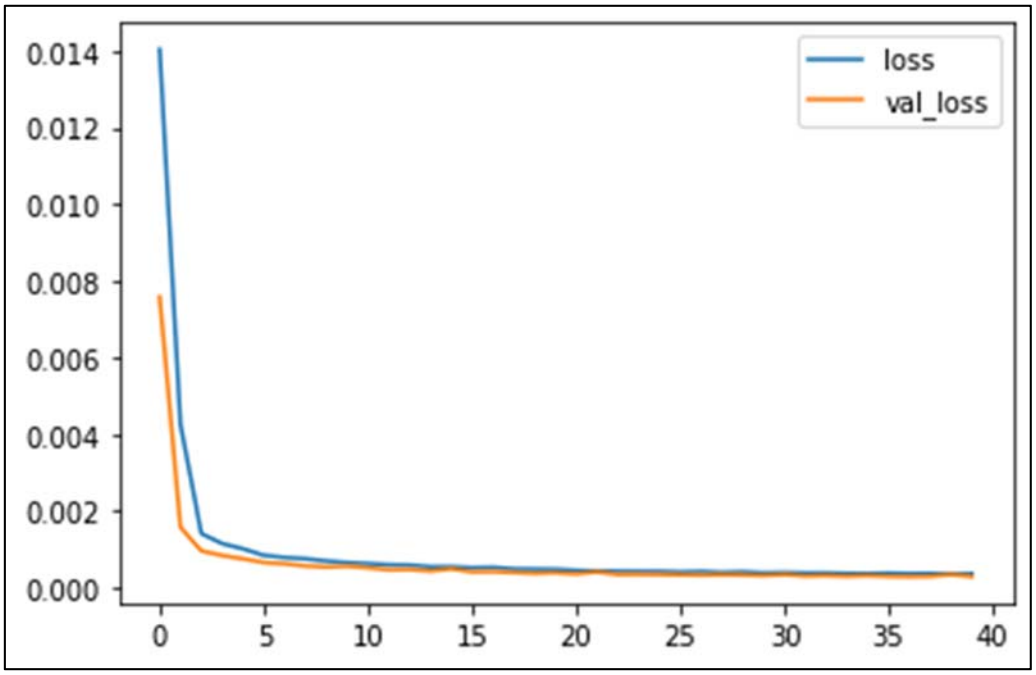

Figure 29 : Loss curve

We can see that the validation and the training loss follow the same curve. This is an indicator that the model is probably not overfitted.

We then make predictions on the testing data set. The results are shown below.

```
predictions_train=model.predict(x_train)
predictions_test = model.predict(x_test)

predictions_train=scaler_label.inverse_transform(predictions_
train) y_train=scaler_label.inverse_transform(y_train)

predictions_test=scaler_label.inverse_transform(predictions_t
est)
y_test=scaler_label.inverse_transform(y_test)

trainScore=math.sqrt(mean_squared_error(y_train,predictions_t
rain))
print('Train Score: %.2f RMSE' % (trainScore))
```

```
testScore=math.sqrt(mean_squared_error(y_test,predictions_tes
t))
print('Test Score: %.2f RMSE' % (testScore))
APE=mean_absolute_percentage_error(y_test,
predictions_test,sample_weight=None,multioutput='uniform_aver
age')
print('APE Score: %.2f' % (APE*100),"%")

  Train Score: 133.56 RMSE
  Test Score: 130.91 RMSE
  APE Score: 1.91 %
```

As shown in the code above, our FFN Network with an input vector of 24h values got a satisfactory score (MAPE=1.9%, 131MWh RMSE). Two Actual-Prediction comparison graphs are shown below.

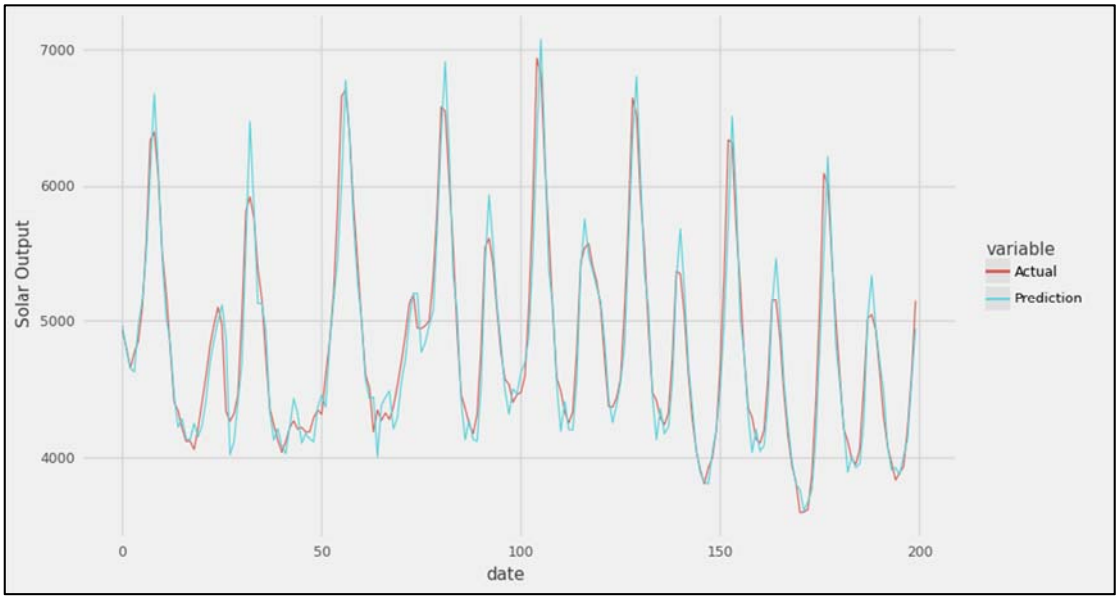

Figure 30 : FFNN Prediction

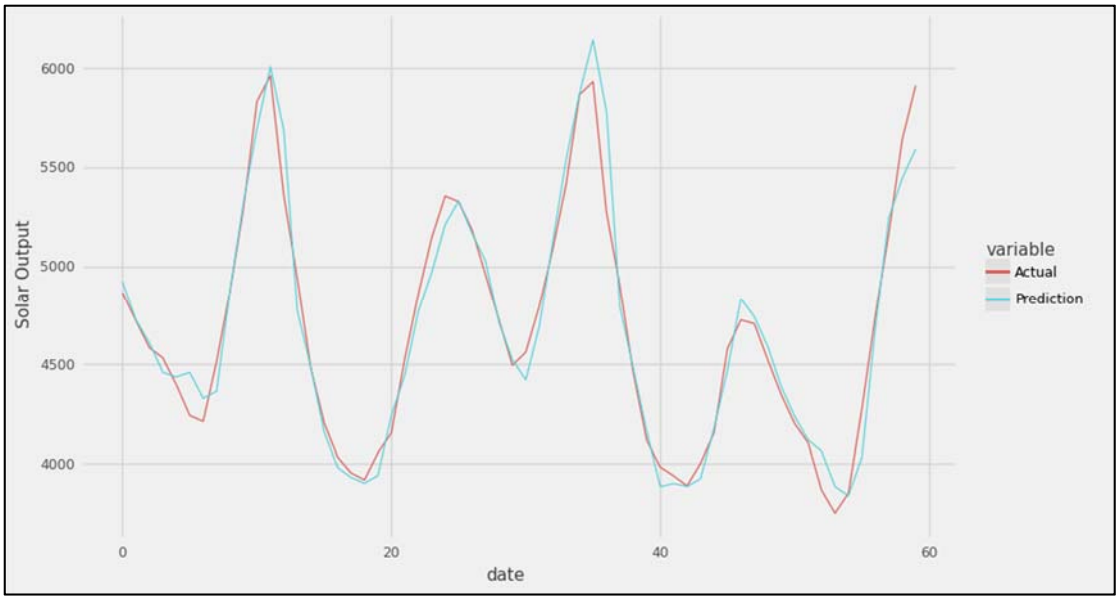

Figure 31 : FFNN Prediction

We note that the model has captured the "peaks" and "troughs" of the time series satisfactorily. Then we implemented with the same architecture the same network, but this time with a different input vector (12h values).

```
features_request = {"window": [11]} #input size
#11+timeZero=12
build_df = build_features(train['Actual Load - TRAINING']
,features_request, target_lag=1,
include_tzero=True) # tzero (time_zero) refers to the current
time
build_df.index.names = ['Index']
build_df
# target lag shows how many hours ahead we want to predict. In
our case  choose to predict 1 value 1 hour
into the future
```

The prediction errors for the test data are shown in below.

```
Train Score: 171.29 RMSE
Test Score: 162.62 RMSE
APE Score: 2.28 %
```

We notice that the accuracy in our predictions is acceptable again but using a shorter input vector length the error prediction increases. In this model

though it is necessary to have data of the last 24 hours (or the last 12 hours) so that we can predict a single value into the future. After that, we built the same model, but try a longer horizon of 48h. We follow the same pipeline:

```
features_request = {"window": [23]} #input size
#23+timeZero=24
build_df_tst_48   =   build_features(train['Actual   Load   -
TESTINGING'] ,features_request, target_lag=48,
include_tzero=True) # tzero (time_zero) refers to the current
time
build_df_tst_48.index.names = ['Index']
build_df_tst_48
```

The results of this forecast are shown in below.

```
  Train Score: 499.73 RMSE
  Test Score: 524.90 RMSE
  APE Score: 7.84 %
```

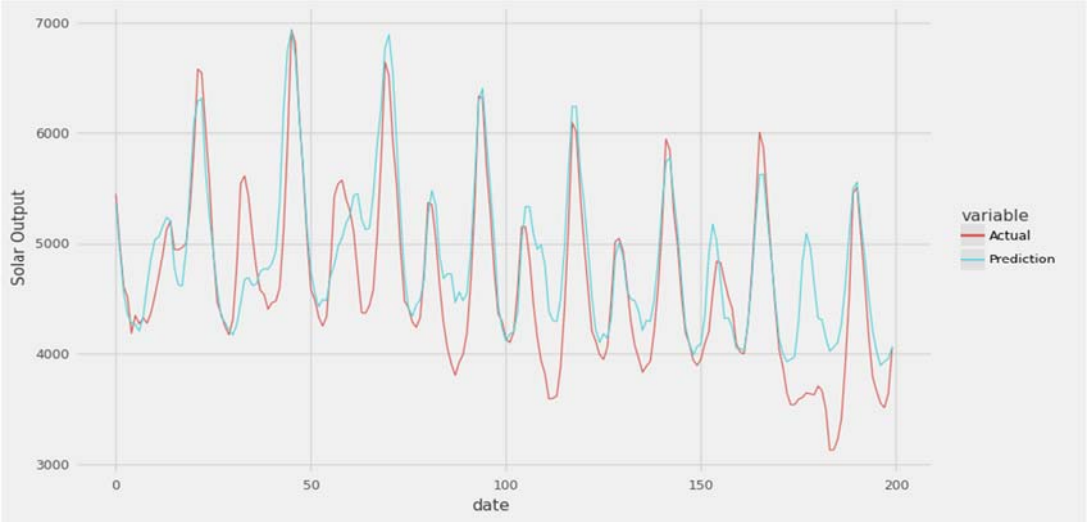

Figure 32 : 48h horizon FFNN Prediction

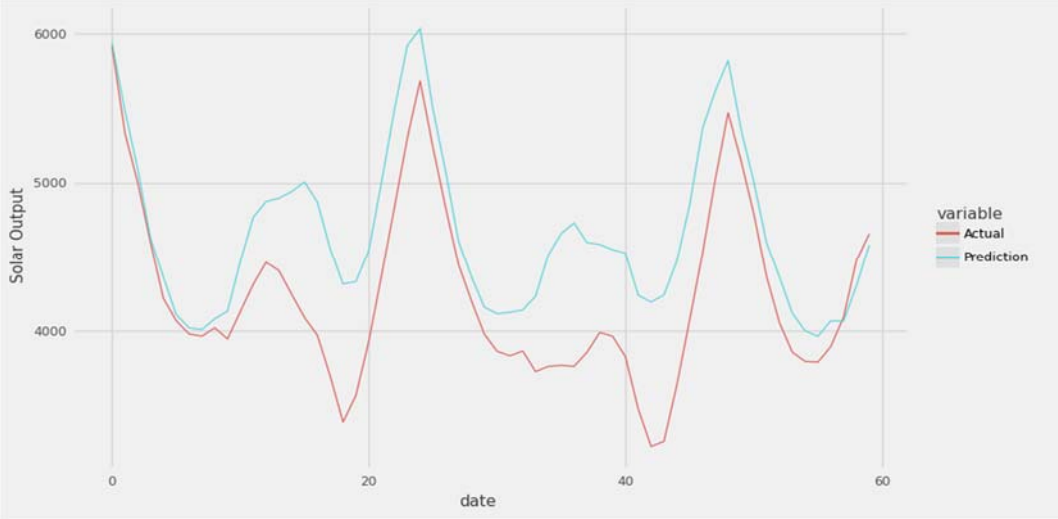

Figure 33 : 48h horizon FFNN Prediction

As expected, this model's accuracy is much less than the previous model. We can still see that the model has captured the electrical load seasonality, but it is not able to estimate all the peaks and troughs.

2.3.2 Recurrent Models

In this category of models, we used the sliding window method as described previously but instead of Dense layers, we used RNN, LSTM and GRU layers. Using these layers requires our data to be in a three-dimensional shape [samples, timesteps, features]. The samples are the total observations we are going to use, the number of timesteps is the past values length and the features is the number of variables we are using as input features. In this category we tried different input size lengths and sliding window lengths. The output length was set to 1h, 2h and 24h hence this category of models uses n past values of all the available input features we described in chapter 2.3 to predict load values into the future. Below is script of these model's structure.

```
#RNN
model = Sequential()
model.add(SimpleRNN(40,return_sequences=True,
input_shape=(96, 1)))
model.add(Dropout(0.35))
model.add(SimpleRNN(40, return _sequences=False))
```

```
model.add(Dense(1,activation="sigmoid"))
model.summary()
```

```
#LSTM
model = Sequential()
model.add(LSTM(40,return_sequences=True,      input_shape=(96,
1)))
model.add(Dropout(0.35))
model.add(LSTM(40, return _sequences=False))
model.add(Dense(1,activation="sigmoid"))
model.summary()
```

```
#GRU
model = Sequential()
model.add(GRU(40,return_sequences=True,
input_shape=(96, 1)))
model.add(Dropout(0.35))
model.add(GRU(40, return _sequences=False))
model.add(Dense(1,activation="sigmoid"))
model.summary()
```

We further investigated this category of models by creating a function which grid searches all combinations of parameters such as input size, batch size type of layer and rolling window size.

2.4 Prediction Errors

Like other neural networks, RNNs are deterministic, meaning that they will produce the same output given the same input and the same set of parameters. However, the predictions made by RNNs may not be completely deterministic due to the same reasons as in the case of ANNs, as the training process of RNNs is typically based on stochastic gradient descent which involves random sampling of training data and the initialization of the network's weights is random. This means that the predictions made by an RNN may vary slightly each time it is trained and evaluated, even if the input and parameters

are the same. Therefore, we have performed 5 runs for each set of parameters. The results from all the runs are shown below.

2.4.1 Output = 1h & 2h

Table 1 : RNN (1h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	120,801	24	6	1	1	68,8	188,87	2,83	143,6	198,43	3	83,8	202,31	2,96	82,3	198,89	3,07	83,9	199,54	3,06	92,48	197,61	2,984
2	200x200	no	120,801	24	12	1	1	143,9	171,11	2,52	394,4	140,74	2,01	143,5	175,32	2,59	203,3	161,99	2,37	203	167,62	2,4	217,62	163,36	2,378
2	200x200	no	120,801	24	24	1	1	203,67	154,73	2,44	203,7	157,26	2,4	263,2	151,76	2,25	563,5	115,33	1,71	322,9	144,32	2,14	311,39	144,68	2,188
2	200x200	no	120,801	80	48	1	1	279,3	117,37	1,8	413,3	106,42	1,6	442,6	102,74	1,51	348,3	106,4	1,56	167,7	142,42	2,11	330,24	115,07	1,716
2	200x200	no	120,801	120	96	1	1	742,7	121,11	1,81	503,5	113,74	1,66	387,5	132,4	2,04	322,5	143,76	2,08	563,2	104,34	1,52	503,88	123,07	1,822
2	200x200	no	120,801	240	168	1	1	322,8	179,72	2,6	175,6	209,69	3,04	417,2	120,51	1,72	263,3	153,85	2,2	388,4	127,75	1,85	313,46	158,3	2,282

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	4,961	24	6	1	1	203,5	177,25	2,65	145,5	182,89	2,71	203,5	174,32	2,55	144,1	184,7	2,84	143,6	183,93	2,71	168,04	180,62	2,692
2	40x40	no	4,961	24	12	1	1	155,5	167,09	2,45	240,1	159,32	2,37	443,7	145,81	2,15	203,5	168,5	2,4	323,2	149,88	2,16	273,2	158,12	2,306
2	40x40	no	4,961	24	24	1	1	263,6	156,15	2,33	453,3	127,7	1,88	623,4	124,36	1,78	528,2	126,03	1,88	468,4	135,75	2,08	467,38	134	1,99
2	40x40	no	4,961	24	48	1	1	666,8	116,99	1,75	749,8	118,57	1,78	863,1	116,51	1,74	1042,5	111,64	1,72	718,3	113,3	1,65	808,1	115,4	1,728
2	40x40	no	4,961	24	96	1	1	1042,9	123,03	1,8	1163	122,97	1,77	1702,1	119,65	1,82	1513,4	111,88	1,64	1522,1	113,42	1,68	1388,7	118,19	1,742
2	40x40	no	4,961	240	168	1	1	431	103,74	1,52	376,8	119,6	1,74	503,6	158,05	2,29	563,7	139,95	2,05	646,8	131,07	2,01	504,38	130,48	1,922

Table 2 : LSTM (1h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	482,601	24	6	1	1	85,3	177,7	2,59	145,2	167,12	2,47	85	173,39	2,52	144,6	166,47	2,43	82,8	174,03	2,54	108,58	171,74	2,51
2	200x200	no	482,601	24	12	1	1	84,8	158,17	2,27	44,2	168,27	2,44	145,2	155,08	2,24	65,9	162,7	2,33	100,9	170,64	2,52	88,2	162,97	2,36
2	200x200	no	482,601	24	24	1	1	145,3	144,13	2,16	85	132,78	1,98	204,6	130,37	1,94	85,2	132,65	1,93	145,4	128,64	1,87	133,1	133,71	1,976
2	200x200	no	482,601	24	48	1	1	144,3	125,4	1,86	130	138,61	2,07	85	138,88	2,06	204,6	129,4	1,92	204,4	104,64	1,52	153,66	127,39	1,886
2	200x200	no	482,601	24	96	1	1	106,8	116,15	1,68	145,1	131,68	1,93	145	128,99	1,85	128,5	115,76	1,69	239,9	101,24	1,48	153,06	118,76	1,726
2	200x200	no	482,601	24	168	1	1	384,9	107,11	1,57	145	115,28	1,65	178	126,16	1,86	128,7	110,12	1,57	444,6	96,64	1,39	256,24	111,06	1,608

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN (Dropout No)			4th RUN			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	1,285,201	24	6	1	1	207,7	167,53	2,42	100,4	174,02	2,58	147,2	170,26	2,48	98,3	172,5	2,57	98,4	169,52	2,51	130,4	170,77	2,512
2	200x200	yes	1,285,201	24	12	1	1	88,5	168,52	2,46	89,4	160,63	2,34	215,3	120,26	1,71	274,8	116,85	1,64	86,8	173,15	2,53	150,96	147,88	2,136
2	200x200	yes	1,285,201	24	24	1	1	266,2	108,37	1,58	148	125,78	1,85	207,2	120,7	1,76	147,4	131,61	1,96	115	125,51	1,84	176,76	122,39	1,798
2	200x200	yes	1,285,201	24	48	1	1	327,1	106,78	1,54	506,6	94,61	1,36	305,6	101,61	1,46	327,3	106,78	1,54	209	114,87	1,7	335,12	104,93	1,52
2	200x200	yes	1,285,201	24	96	1	1	508,2	103,04	1,5	182,7	130,76	1,95	166,7	136,69	2,05	282,7	114,93	1,67	303	115,5	1,74	288,66	120,18	1,782
2	200x200	yes	1,285,201	24	168	1	1	687	112,77	1,67	506,8	116,98	1,7	807,4	97,38	1,4	327,5	135,69	1,97	385	114,53	1,66	542,74	115,47	1,68

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	19,721	24	6	1	1	84,8	171,1	2,47	103,9	167,92	2,45	85	175,43	2,53	85,7	174,61	2,52	93,3	170,1	2,48	90,54	171,83	2,49
2	40x40	no	19,721	24	12	1	1	145	154,69	2,19	87,6	156,87	2,25	84,9	176,32	2,66	115,4	149,48	2,16	73,6	158,25	2,26	101,3	159,12	2,304
2	40x40	no	19,721	24	24	1	1	145,2	136,36	1,98	145,2	134,47	1,95	105,6	138,53	2,03	144,7	139,53	2,04	144,7	137,33	2,04	137,08	137,24	2,008
2	40x40	no	19,721	24	48	1	1	144,6	129,56	1,91	204,4	116	1,67	144,8	131,43	1,96	95,6	133,9	2,03	136,6	124,79	1,84	145,2	127,14	1,882
2	40x40	no	19,721	24	96	1	1	93	144,1	2,17	154,3	124,57	1,85	204,8	127,42	1,87	184	127,16	1,89	146,8	128,03	1,88	156,58	130,26	1,932
2	40x40	no	19,721	24	168	1	1	263,4	120,83	1,8	145,2	152,79	2,3	265,3	130,23	1,91	205	131,25	1,97	324,7	115,99	1,66	240,72	130,22	1,928

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	52,241	24	6	1	1	267,6	163,02	2,29	86,6	220,23	3,36	87,4	209,34	3,21	147,5	173,06	2,55	237,7	164,04	2,34	165,36	185,94	2,75
2	40x40	yes	52,241	24	12	1	1	147,5	159,49	2,32	267,3	136,91	1,94	207,3	145,76	2,07	148,7	151,23	2,14	293,5	142,1	2,04	212,86	147,1	2,102
2	40x40	yes	52,241	24	24	1	1	147,6	153,36	2,31	205,3	123,73	1,78	270,3	113,04	1,66	235,5	117,55	1,75	108,2	148,55	2,22	193,38	131,25	1,944
2	40x40	yes	52,241	24	48	1	1	206,8	134,85	1,99	147,1	149,81	2,2	270,9	110,35	1,58	327,1	109,94	1,6	266,4	116,96	1,69	243,66	124,38	1,812
2	40x40	yes	52,241	24	96	1	1	387,6	114,07	1,68	387,8	125,6	1,86	264,8	130,73	1,92	357	113,89	1,64	326,87	122,81	1,78	344,81	121,42	1,776
2	40x40	yes	52,241	24	168	1	1	202	158,47	2,37	345,2	134,96	2,01	327,4	2,01	2	387,6	138,69	2,01	386,7	124,68	1,81	329,78	111,76	2,04

Table 3 : LSTM (2h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	482,802	12	6	2	1	213,3	228,36	3,18	144,9	252,38	3,55	73,9	262,8	3,78	85,3	267,33	3,75	268,9	225,99	3,07	157,26	247,37	3,466
2	200x200	no	482,802	12	12	2	1	324,9	166,91	2,27	324,7	156,93	2,14	264,4	158,8	2,17	214,9	168,13	2,33	324,5	163,32	2,23	290,68	162,82	2,228
2	200x200	no	482,802	12	24	2	1	206,7	145,7	2,09	230,8	140,17	2,02	255,7	139,18	1,95	203,5	153,55	2,22	205	149,59	2,11	220,34	145,64	2,078
2	200x200	no	482,802	12	48	2	1	144,4	169,45	2,42	165,5	148,89	2,09	499,1	120,57	1,67	443,4	124,86	1,73	504,5	127,66	1,78	351,38	138,29	1,938
2	200x200	no	482,802	12	96	2	1	385,1	132,16	1,83	624,5	122,74	1,68	744	118,19	1,61	384,7	132,44	1,85	204,6	151,49	2,18	468,58	131,4	1,83
2	200x200	no	482,802	12	168	2	1	541,2	131,39	1,84	350,5	152,15	2,23	654,3	126,56	1,79	384,7	159,86	2,32	404,3	187,91	2,64	467	151,57	2,164

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN (Dropout No)			2nd RUN (Dropout No)			3rd RUN			4th RUN (Dropout No)			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	1,285,602	12	6	2	1	122,4	256,36	3,62	327,3	217,02	2,86	327,2	227,48	3,08	387,4	208,47	2,85	147,5	268,06	3,82	262,36	235,48	3,246
2	200x200	yes	1,285,603	12	12	2	1	447,5	171,5	2,37	327,3	159,07	2,15	687,4	150,15	2	446,1	158,99	2,24	507,4	158,25	2,14	483,14	159,59	2,18
2	200x200	yes	1,285,604	12	24	2	1	207,7	159,59	2,28	507	125,51	1,75	169,4	155,66	2,23	267,8	146,7	2,08	687,4	126,88	1,73	367,86	142,87	2,014
2	200x200	yes	1,285,605	12	48	2	1	447,4	131,14	1,87	387,2	134,7	1,89	479,3	134,81	1,86	506,1	134,96	1,92	266,9	160,46	2,33	417,38	139,21	1,974
2	200x200	yes	1,285,606	12	96	2	1	421,9	141,1	1,98	569,8	131,96	1,83	502,4	142,96	2,02	935,8	123,44	1,71	687,1	133,41	1,88	623,4	134,57	1,884
2	200x200	yes	1,285,607	12	168	2	1	1076,4	126,44	1,75	1466,9	123	1,67	1166,5	130,04	1,81	1168	127,12	1,76	508,4	161,46	2,38	1077,2	133,61	1,874

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	19,762	12	6	2	1	85,1	264,77	3,73	144,9	254,34	3,58	85,1	276,6	3,96	169,2	250,46	3,48	145,2	259,08	3,63	125,9	261,05	3,676
2	40x40	no	19,762	12	12	2	1	264,9	192,84	2,63	140,1	211,88	2,9	219,5	200,21	2,8	145,1	212,22	2,91	163,2	205,69	2,81	186,56	204,57	2,81
2	40x40	no	19,762	12	24	2	1	221,2	158,67	2,23	261,8	151,39	2,15	162,8	167,87	2,4	85,1	194,52	2,85	324,5	154,89	2,17	211,08	165,47	2,36
2	40x40	no	19,762	12	48	2	1	265	154,24	2,2	477,4	133,34	1,88	303,2	143,18	2,03	140,7	187,71	2,71	392,7	134,21	1,89	315,8	150,54	2,142
2	40x40	no	19,762	12	96	2	1	270,5	155,74	2,23	444,8	145,21	2,05	361,4	148,33	2,07	263,9	157,09	2,23	445	148,81	2,09	357,12	151,04	2,134
2	40x40	no	19,762	12	168	2	1	384,9	150,62	2,1	745,1	137,49	1,92	444,2	149,55	2,1	358,6	156,79	2,2	625,2	143,55	1,99	511,6	147,6	2,062

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN (Dropout No)			2nd RUN (Dropout No)			3rd RUN			4th RUN (Dropout No)			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	52,322	12	6	2	1	327,3	243,96	3,4	387,9	244,08	3,42	267	254,38	3,5	74,4	332,88	4,95	207,7	276,6	3,98	252,86	270,38	3,85
2	40x40	yes	52,322	12	12	2	1	387,3	198,72	2,72	274,4	220,97	3,09	447,8	195,54	2,67	262,7	213,7	2,93	567,3	184,68	2,51	387,9	202,72	2,784
2	40x40	yes	52,322	12	24	2	1	327,6	148,93	2,12	278,2	149,45	2,15	568,6	139,47	2,01	387,7	142,2	2	447,6	150,81	2,15	401,94	146,17	2,086
2	40x40	yes	52,322	12	48	2	1	447,5	148,19	2,1	447,8	151,07	2,13	206,6	184,08	2,65	386,6	143,92	2,05	206,9	193,92	2,83	339,08	164,24	2,352
2	40x40	yes	52,322	12	96	2	1	447,2	158,51	2,3	433,1	158,09	2,26	387,1	168,81	2,42	626,8	144,59	2,03	182,9	220,09	3,26	415,42	170,02	2,454
2	40x40	yes	52,322	12	168	2	1	867,2	141,1	1,96	1319,1	130,44	1,81	867,9	145,89	2,03	1046,6	132,93	1,86	1047,7	150,96	2,2	1029,7	140,26	1,972

Table 4 : GRU (1h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	363,201	24	6	1	1	123,8	149,6	2,11	80,3	160,31	2,3	87	180,73	2,73	145,6	156,26	2,24	44	180,27	2,62	96,14	165,43	2,4
2	200x200	no	363,201	24	12	1	1	78,4	138,72	1,97	69,2	138,83	1,97	126,3	123,11	1,74	151,5	112,27	1,56	145,4	117,31	1,63	114,16	126,05	1,774
2	200x200	no	363,201	24	24	1	1	77,7	125,78	1,82	44,9	144,12	2,15	43,3	160,34	2,36	80,5	125,91	1,85	91,2	120,49	1,75	67,52	135,33	1,986
2	200x200	no	363,201	24	48	1	1	324,3	86,1	1,23	56,3	147,23	2,18	324,1	87,24	1,27	168,2	100,36	1,5	220	93,66	1,35	218,58	102,92	1,506
2	200x200	no	363,201	24	96	1	1	124,9	117,47	1,73	264,7	96,59	1,4	327,4	91,76	1,31	324,2	90,37	1,3	144,6	127,15	1,86	237,16	104,67	1,52
2	200x200	no	363,201	24	168	1	1	504,9	86,45	1,25	445,3	94,81	1,38	188,3	117,06	1,72	172,8	126,62	1,86	264,6	112,74	1,65	315,18	107,54	1,572

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN (Dropout No)			2nd RUN			3rd RUN (Dropout No)			4th RUN			5rd RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	966,401	24	6	1	1	86,8	166,67	2,4	87,7	173,49	2,49	55,1	175,05	2,6	150,4	158,59	2,35	119	161,27	2,32	99,8	167,01	2,432
2	200x200	yes	966,401	24	12	1	1	184	117,76	1,62	147,2	132,1	1,85	208,2	115,13	1,59	143,9	125,02	1,8	207,5	121,67	1,68	178,16	122,34	1,708
2	200x200	yes	966,401	24	24	1	1	266	98,05	1,4	171,2	113,25	1,66	238,9	99,94	1,41	207,5	105,83	1,53	140,3	113,47	1,64	204,78	106,11	1,528
2	200x200	yes	966,401	24	48	1	1	207	113,92	1,65	151,5	124,14	1,78	368	92,78	1,34	190,2	114,31	1,68	195,1	114,43	1,67	222,36	111,92	1,624
2	200x200	yes	966,401	24	96	1	1	387	101,31	1,45	206,9	142,97	2,16	326,3	110,64	1,65	381,6	104,77	1,49	417,9	105,19	1,54	343,94	112,98	1,658
2	200x200	yes	966,401	24	168	1	1	665,7	99,52	1,48	567,1	111,87	1,58	627,8	99,55	1,42	506,8	106	1,53	984,2	88,66	1,27	670,32	101,12	1,456

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	15,041	24	6	1	1	84,8	172,09	2,48	85,2	176,78	2,57	69,5	173,78	2,52	85,8	169,2	2,46	55,4	173,17	2,52	76,14	173	2,51
2	40x40	no	15,041	24	12	1	1	69,3	157,6	2,25	36,2	200,46	2,93	71,5	156,7	2,25	76,3	156,24	2,22	145,5	151,43	2,13	79,76	164,49	2,356
2	40x40	no	15,041	24	24	1	1	54,8	150,46	2,24	84,2	145,95	2,17	144	133,48	1,96	144,3	137,97	2,04	144,5	129,89	1,92	114,36	139,55	2,066
2	40x40	no	15,041	24	48	1	1	144	125,28	1,82	145	134,14	1,95	139,2	124,43	1,83	179,6	118,49	1,73	74,9	144,4	2,09	136,54	129,35	1,884
2	40x40	no	15,041	24	96	1	1	145	132,25	1,96	154,6	127,29	1,86	180,8	116,81	1,69	145	138,74	2,02	173,5	124,23	1,79	159,78	127,86	1,864
2	40x40	no	15,041	24	168	1	1	265,6	124,16	1,82	325,4	122,06	1,78	172,4	131,18	1,95	264,7	121,3	1,77	227,1	125,91	1,84	251,04	124,92	1,832

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN (Dropout No)			3rd RUN			4th RUN (Dropout No)			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	39,681	24	6	1	1	144,2	170,55	2,47	91,2	179,59	2,61	146,3	170,54	2,49	100,1	174,51	2,52	99,5	171,07	2,53	116,26	173,25	2,524
2	40x40	yes	39,681	24	12	1	1	127,9	149,38	2,13	147,8	148,74	2,14	326,4	118,51	1,72	45,7	176,3	2,64	207	145,43	2,06	170,96	147,67	2,138
2	40x40	yes	39,681	24	24	1	1	62,1	149,53	2,2	207,8	123,52	1,78	159,2	121,29	1,76	147,2	134,79	1,99	268,1	114,64	1,65	168,88	128,75	1,876
2	40x40	yes	39,681	24	48	1	1	94,2	145,56	2,18	236,4	111,1	1,59	204,8	121,33	1,76	207,6	113,25	1,63	388,8	106,67	1,54	226,36	119,58	1,74
2	40x40	yes	39,681	24	96	1	1	327,4	120,7	1,74	447,3	105,96	1,55	387,6	112,74	1,64	143,3	138,99	2,07	327,9	122,4	1,79	326,7	120,16	1,758
2	40x40	yes	39,681	24	168	1	1	567,4	108	1,61	447,7	110,86	1,59	687,5	106,02	1,54	567,1	105,8	1,51	507,2	117,07	1,7	555,38	109,55	1,59

Table 5 : GRU (2h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	363,402	12	6	2	1	145	250,93	3,51	204,3	228,32	3,14	205	226,92	3,07	139,2	232,34	3,19	145,2	235,51	3,27	167,74	234,8	3,236
2	200x200	no	363,402	12	12	2	1	204,8	178,99	2,49	180,8	182,57	2,51	208,8	165,68	2,29	249,9	163,46	2,22	180,6	171,67	2,36	204,98	172,47	2,374
2	200x200	no	363,402	12	24	2	1	264,7	127,43	1,78	324,5	126,88	1,74	196,2	132,9	1,84	312,8	125,16	1,72	145,6	151,87	2,16	248,76	132,85	1,848
2	200x200	no	363,402	12	48	2	1	237,7	126,11	1,78	246,6	132,14	1,84	204,8	151,47	2,17	264,3	135,08	1,85	289,6	127,66	1,77	248,6	134,49	1,882
2	200x200	no	363,402	12	96	2	1	477,9	119,15	1,62	324,7	126,94	1,77	444,9	131,34	1,81	281,2	130,97	1,83	324,9	130,21	1,81	370,72	127,72	1,768
2	200x200	no	363,402	12	168	2	1	351,3	151,64	2,21	265,2	153,59	2,22	384,5	139,6	1,98	745,2	118,67	1,63	587,4	123,78	1,71	466,72	137,46	1,95

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN (Dropout No)			2nd RUN (Dropout No)			3rd RUN			4th RUN (Dropout No)			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	966,802	12	6	2	1	290,6	214,65	2,96	268,4	214,71	2,89	387,1	206,81	2,78	387,4	211,16	2,86	241,8	225,29	3,1	315,06	214,52	2,918
2	200x200	yes	966,802	12	12	2	1	449,8	157,12	2,11	327,2	160,01	2,21	407,3	2,42	2,23	207,3	175,3	2,42	327,1	164,71	2,21	343,74	131,91	2,236
2	200x200	yes	966,802	12	24	2	1	370	125,08	1,74	265,2	143,53	2,04	183,1	149,69	2,14	448,3	124,36	1,72	147,2	174,38	2,51	282,76	143,41	2,03
2	200x200	yes	966,802	12	48	2	1	127,8	136,82	1,93	386,7	138,74	1,97	359,5	141,04	2,03	252,4	149,29	2,15	482	127,53	1,78	321,68	138,68	1,972
2	200x200	yes	966,802	12	96	2	1	421,9	121,99	1,68	268,8	163,77	2,32	367,7	142,5	1,98	536,7	128,13	1,81	507,6	136,62	1,93	420,54	138,6	1,944
2	200x200	yes	966,802	12	168	2	1	863,8	126,88	1,77	266,9	163,77	2,32	462,2	128,52	1,8	938,6	120,03	1,69	506,9	139,31	1,94	607,68	135,7	1,904

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	15,082	12	6	2	1	290,3	217,05	2,96	206,4	239,46	3,3	205,3	230,99	3,17	158	238,97	3,3	85,6	275,16	3,97	189,12	240,33	3,34
2	40x40	no	15,082	12	12	2	1	165	197,78	2,75	133	206,28	2,82	181,1	194,18	2,66	215,3	192,56	2,62	264,6	176,14	2,44	191,8	193,39	2,658
2	40x40	no	15,082	12	24	2	1	156,7	152,54	2,19	204,8	151,01	2,14	101,5	176,49	2,54	225,7	144,91	2,06	264,3	136,1	1,91	190,6	152,21	2,168
2	40x40	no	15,082	12	48	2	1	564,3	129,23	1,81	265,5	148,39	2,14	317,2	138,3	1,95	264,8	146,92	2,09	204,3	156,7	2,22	323,22	143,91	2,042
2	40x40	no	15,082	12	96	2	1	444,4	134,71	1,88	504,8	135,54	1,89	504,6	136,32	1,91	258,8	151,16	2,15	473,7	135,8	1,93	437,26	138,71	1,952
2	40x40	no	15,082	12	168	2	1	324,4	159,88	2,29	444,8	141,17	2	445,6	144,51	2,07	324,6	151,13	2,11	565,1	140,55	1,99	420,9	147,45	2,092

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN (Dropout No)			3rd RUN			4th RUN			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	39,762	12	6	2	1	327,3	224,47	3,12	88,1	275,74	3,9	138,8	265,1	3,81	389,6	217,61	2,93	386,6	218,36	2,97	266,08	240,26	3,346
2	40x40	yes	39,762	12	12	2	1	387,2	186,35	2,54	507,9	164,07	2,23	327,5	185,97	2,6	506,3	168,56	2,34	446,8	166,88	2,35	435,14	174,37	2,412
2	40x40	yes	39,762	12	24	2	1	267,3	152,94	2,2	555,9	130,59	1,82	387,1	144,98	2,1	200	154,86	2,26	447,1	132,82	1,87	371,48	143,24	2,05
2	40x40	yes	39,762	12	48	2	1	207,2	168,29	2,44	387,6	143,29	2,02	357,9	145,43	2,09	307,8	155,58	2,21	507,4	138,27	1,95	353,58	150,17	2,142
2	40x40	yes	39,762	12	96	2	1	296	161,42	2,32	567,8	146,91	2,07	385,6	154,42	2,21	626,5	142,49	2,04	506,5	146,5	2,08	476,48	150,35	2,144
2	40x40	yes	39,762	12	168	2	1	322,3	164,72	2,42	567,3	142,05	1,97	690,2	148,82	2,12	770,8	138,77	1,95	806,9	149,61	2,15	631,5	148,79	2,122

2.4.2 Output = 24h

Table 6 : RNN (24h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	125,424	24	24	24	24	28,68	408	5,94	20,94	426,49	6,15	43,08	403,51	5,94	43,16	402,84	5,8	15,93	428,9	6,2	30,358	413,95	6,006
2	200x200	no	125,424	24	48	24	24	20,82	451,66	6,88	18,71	445,71	6,7	38,97	398,78	5,85	36,1	426,54	6,15	43,25	421,4	6,12	31,57	428,82	6,34
2	200x200	no	125,424	24	96	24	24	29,35	522,62	8,29	72,41	406,99	5,92	90,8	440,54	5,89	92,8	399,5	5,72	84,47	421,57	6,31	73,966	438,24	6,426
2	200x200	no	125,424	24	168	24	24	204,93	446,98	6,71	144,98	424,17	6,13	144,85	421,16	6,17	144,9	427,23	6,38	144,9	414,8	5,94	156,91	426,87	6,266
2	200x200	no	125,424	24	384	24	24	95,99	678,92	10,17	146,61	631,31	9,93	326,15	420,9	6,27	206,21	502,9	7,93	146,19	817,13	13,24	184,23	610,23	9,508
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	125,424	48	24	24	24	14,26	410,89	5,91	11,05	442,11	6,6	9,13	427,35	6,24	12,8	420,36	6,16	12,34	423,23	6,21	11,916	424,79	6,224
2	200x200	no	125,424	48	48	24	24	23,11	425,63	6,39	20,7	414,12	6,05	22,74	426,34	6,29	21,98	426,08	6,4	22,76	408,89	6,02	22,258	420,21	6,23
2	200x200	no	125,424	48	96	24	24	16,33	542,89	8,75	32,46	407,57	6	15,98	522,12	7,78	84,44	422,57	6,23	29,37	480,23	7,52	35,716	475,08	7,256
2	200x200	no	125,424	48	168	24	24	101,69	413,49	6,03	71,17	405,74	5,84	145,25	396,53	5,74	91,96	410,33	5,99	55,98	417,45	6,26	93,21	408,71	5,972
2	200x200	no	125,424	48	384	24	24	146,6	463,59	7,38	81,51	478,07	7,21	62,94	599,88	9,45	146,25	457,26	6,78	146,2	606,35	9,28	116,7	521,03	8,02
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	125,424	96	24	24	24	12,34	429,75	6,31	12,36	425,14	6,23	6,09	464,12	6,93	7	437,52	6,46	12,45	431,15	6,31	10,048	437,54	6,448
2	200x200	no	125,424	96	48	24	24	10,06	441,65	6,57	13,33	414,95	6,11	10,71	437,44	6,56	13,16	432,47	6,51	22,73	436,87	6,38	13,998	432,68	6,426
2	200x200	no	125,424	96	96	24	24	43,5	424,84	6,2	33,33	415,87	6,16	18	445,58	6,72	23,06	463,3	6,81	33,02	415,74	6,15	30,182	433,07	6,408
2	200x200	no	125,424	96	168	24	24	84,81	391,19	5,69	46,09	445,14	6,67	35,2	476,01	7,28	16,83	583,45	9,12	84,94	404,52	6,02	53,574	460,06	6,956
2	200x200	no	125,424	96	384	24	24	44,97	769,75	12,58	86,31	503,4	7,51	30,19	829,88	13,14	37,42	666,79	10,89	86,53	79,71	12,91	57,084	569,91	11,406
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	330,824	24	24	24	24	44,59	390,39	5,56	23,8	438,02	6,25	44,96	444,32	6,83	20,03	438,27	6,41	350,54	408,12	5,82	96,784	423,82	6,174
2	200x200	yes	330,824	24	48	24	24	76,94	399,93	5,74	67,96	395,66	5,63	42,94	412,25	6,06	67,54	387,23	5,53	85,87	400,25	5,87	68,25	399,06	5,766
2	200x200	yes	330,824	24	96	24	24	120,9	523,82	7,71	146,41	404,18	5,93	206,78	438,92	5,79	102,52	399,45	5,84	146,69	439,58	6,36	144,66	441,19	6,326
2	200x200	yes	330,824	24	168	24	24	144,3	519,53	8,18	300,06	410,65	6,03	87,36	747,46	12,03	207,88	408,72	6,17	228,21	352,34	5,07	193,56	487,74	7,496
2	200x200	yes	330,824	24	384	24	24	450,75	627,76	9,32	510,18	539,16	8,57	296,34	840,17	13,49	510,27	417,55	6,14	329,98	476,47	7,13	419,5	580,22	8,93
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	330,824	48	24	24	24	19,05	415,85	5,99	24,1	432,84	6,28	24,06	407,22	5,96	33,76	379,07	5,38	20,55	395,65	5,67	24,304	406,13	5,856
2	200x200	yes	330,824	48	48	24	24	51,24	397,68	5,7	40,8	392,85	5,67	24,67	440,17	6,5	45,19	411,87	6,02	52,48	393,84	5,83	42,876	407,28	5,944
2	200x200	yes	330,824	48	96	24	24	42,37	433,13	6,48	57,68	405	6,01	50,41	422,34	6,34	90,42	405,61	5,86	90,55	399,95	5,84	66,286	413,21	6,106
2	200x200	yes	330,824	48	168	24	24	207,9	379,11	5,47	71,31	581,61	8,94	116,96	377,37	5,38	207,51	354,36	5,16	147,19	465,44	7,02	150,17	431,58	6,394
2	200x200	yes	330,824	48	384	24	24	150,19	810,86	12,62	210,74	570,74	8,61	449,99	459,29	6,91	390,62	398,45	5,96	310,01	406,25	5,85	302,31	529,12	7,99
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	330,824	96	24	24	24	24,79	411,05	5,97	14,59	424,64	6,25	11,61	426,18	6,31	16,53	405,16	5,89	14,29	429,73	6,2	16,362	419,35	6,124
2	200x200	yes	330,824	96	48	24	24	24,82	427,49	6,28	14,69	641,61	10,57	25,16	410,51	5,89	24,49	411,5	5,95	28,52	409,39	6,02	23,536	460,1	6,942
2	200x200	yes	330,824	96	96	24	24	49,71	406,87	5,98	47,06	407,14	5,98	45,82	443,9	6,63	38,86	420,74	6,18	86,42	408,37	5,94	53,574	417,4	6,142
2	200x200	yes	330,824	96	168	24	24	87,36	402,73	5,86	89,6	376,82	5,42	112,5	360,41	5	207,77	333,5	4,62	135,92	337,65	4,7	126,63	362,22	5,12
2	200x200	yes	330,824	96	384	24	24	66,89	772,06	12,18	137,6	752,54	11,94	91,89	752,17	11,67	270,32	405,14	5,79	209,82	426,41	6,37	155,3	621,66	9,59
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	5,904	24	24	24	24	31,64	426,3	6,28	43,07	406,94	5,89	84,13	406,2	5,9	54,39	406,17	5,92	43,17	421,24	6,19	51,28	413,37	6,036
2	40x40	no	5,904	24	48	24	24	75,14	411,63	6,06	84,25	418,91	6,21	41,84	448,07	6,69	31,68	459,22	6,83	43,74	449,87	6,78	55,33	437,54	6,514
2	40x40	no	5,904	24	96	24	24	84,95	490,04	7,29	144,49	419,46	6,06	99,07	435,14	6,6	145,01	458,37	6,81	57,21	451,86	6,73	106,15	450,97	6,698
2	40x40	no	5,904	24	168	24	24	205,23	426,91	6,36	175,21	424,71	6,35	360,21	393,73	5,64	248,06	403,07	5,88	264,96	418,42	6,18	250,73	413,37	6,082
2	40x40	no	5,904	24	384	24	24	547,5	414,91	6,02	326,24	474,25	7,19	566,17	421,06	6,17	686,6	423,83	6,33	626,31	413,74	6,02	550,56	429,56	6,346
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	5,904	48	24	24	24	25,94	421,69	6,21	29,96	409,05	5,99	43,16	420,28	6,19	22,76	424,41	6,34	22,65	445,36	6,69	28,894	424,16	6,284
2	40x40	no	5,904	48	48	24	24	43,29	456,49	6,87	43,23	452,07	6,85	43,29	434,04	6,44	44,81	432,89	6,47	14,68	484,67	7,37	37,86	452,03	6,8
2	40x40	no	5,904	48	96	24	24	84,54	452,3	6,86	43,51	476,23	7,29	59,12	445,54	6,63	84,6	436,31	6,48	84,89	447,14	6,55	71,332	451,5	6,762
2	40x40	no	5,904	48	168	24	24	113,85	438,77	6,6	123,85	423,1	6,33	145,29	459,78	6,97	101,59	448,71	6,82	85,32	465,34	7,03	113,98	447,14	6,75
2	40x40	no	5,904	48	384	24	24	206,7	476,58	7,16	446,64	400,68	5,9	266,28	454,53	6,73	386,63	432,24	6,41	206,18	485,49	7,34	302,49	449,9	6,708
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	5,904	96	24	24	24	23,05	440,33	6,57	12,37	463,12	6,97	12,41	455,73	6,89	10,1	478,41	7,23	12,48	450,18	6,72	14,082	457,55	6,876
2	40x40	no	5,904	96	48	24	24	15,41	459,2	6,97	22,75	498,3	7,63	43,26	426,67	6,32	16,23	474,75	7,15	14,7	474,04	7,09	22,47	466,59	7,032
2	40x40	no	5,904	96	96	24	24	43,56	471,95	7,07	84,47	460,01	6,83	56,81	429,96	6,4	84,6	458,37	6,91	44,44	434,71	6,51	62,776	451	6,744
2	40x40	no	5,904	96	168	24	24	85,03	503,85	7,75	60,92	445,13	6,64	54	462,91	6,96	122,23	425,3	6,32	144,94	425,5	6,44	93,424	452,54	6,822
2	40x40	no	5,904	96	384	24	24	326,3	425,99	6,28	266,25	416,79	6,13	122,94	467,68	6,93	138,58	457,32	6,87	146,61	495,53	7,47	200,14	452,66	6,736
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	14,984	24	24	24	24	42,88	409,79	5,92	30,87	441,63	6,58	44,72	437,25	6,42	44,84	415,09	6,02	34,27	413,69	6,17	39,516	423,49	6,222
2	40x40	yes	14,984	24	48	24	24	64,38	440,25	6,5	86,04	435,61	6,47	98,33	409,75	5,96	86,14	420,64	6,18	85,71	407,88	5,99	84,12	422,83	6,22
2	40x40	yes	14,984	24	96	24	24	223,11	400,94	5,75	146,75	436,8	6,54	229,89	405,08	5,85	326,22	353,91	5,01	206,68	395,87	5,77	226,53	398,52	5,784
2	40x40	yes	14,984	24	168	24	24	387,7	365,1	5,26	147,11	470,58	7,04	507,45	353,01	4,92	176,96	404,94	6,13	327,1	344,8	4,87	309,26	387,69	5,644
2	40x40	yes	14,984	24	384	24	24	1050,1	405,55	5,89	632,82	427,67	6,27	870,07	416,31	6,11	669,61	376,09	5,48	629,71	381,39	5,54	770,45	401,4	5,858
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	14,984	48	24	24	24	33,76	421,92	6,16	45,05	430,83	6,33	44,77	422,67	6,22	35	392,61	5,6	44,41	403,7	5,87	40,598	414,35	6,036
2	40x40	yes	14,984	48	48	24	24	55,43	411,46	5,99	85,63	414,58	6,04	86,04	439,21	6,44	71,93	387,7	5,58	31,42	435,92	6,42	66,09	417,77	6,094
2	40x40	yes	14,984	48	96	24	24	146,73	428,73	6,33	151,78	407,03	5,95	206,96	412,16	6,06	120,98	391,19	5,63	103,99	399,72	5,77	146,09	407,77	5,948
2	40x40	yes	14,984	48	168	24	24	140,58	393,09	5,74	215,15	374,94	5,5	327,35	391,43	5,7	267,56	339,99	4,76	147,08	375,07	5,49	219,54	374,9	5,438
2	40x40	yes	14,984	48	384	24	24	689,84	397,53	5,75	449,64	408,83	6,05	771,55	389,57	5,71	570,07	377,49	5,4	369,18	406,81	6,03	570,06	396,05	5,788
layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	14,984	96	24	24	24	24,72	419,63	6,12	44,5	416,12	6,1	44,71	417,85	6,12	31,61	405,99	5,84	24,64	409,65	6	34,036	413,85	6,036
2	40x40	yes	14,984	96	48	24	24	51,31	432,37	6,25	32,33	442,52	6,55	45,42	430,83	6,32	42,45	411,31	5,99	30,81	428,22	6,37	40,464	429,05	6,296
2	40x40	yes	14,984	96	96	24	24	86,05	428,7	6,36	60,24	445,42	6,69	86,87	425,33	6,3	75,28	419,6	6,13	97,12	414,74	6,12	81,112	426,76	6,32
2	40x40	yes	14,984	96	168	24	24	112,58	396,09	5,84	146,13	393,43	5,82	147,9	416,66	6,23	114,51	368,42	5,37	207,54	350,06	5,02	145,73	384,93	5,656
2	40x40	yes	14,984	96	384	24	24	448,93	393,33	5,74	329,98	417,81	6,17	629,87	409,04	6,06	329,67	404,01	5,98	329,71	405,58	5,9	413,63	405,95	5,97

Table 7 : LSTM (24h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	487,224	24	24	24	24	8,6	477,83	7,15	20,59	457,46	6,71	21,2	469,88	6,92	8,86	467,58	6,93	28,39	421,79	6,13	17,528	458,91	6,768
2	200x200	no	487,224	24	48	24	24	11,06	434,3	6,4	11,26	456,48	7,12	11,8	430,47	6,28	10,61	454,14	6,9	9,63	447,27	6,54	10,872	444,53	6,648
2	200x200	no	487,224	24	96	24	24	9,9	487,08	7,25	15,88	440,27	6,65	14,56	438,45	6,6	25,23	427,3	6,3	25,51	438,74	6,49	18,216	446,37	6,658
2	200x200	no	487,224	24	168	24	24	25,97	442,97	6,7	18,61	439,34	6,4	15,09	469,45	7,31	18,59	437,7	6,49	10,24	494,61	7,52	17,7	456,81	6,884
2	200x200	no	487,224	24	384	24	24	46,67	442,94	6,63	26,19	505,7	7,8	26,05	460,36	6,75	23,98	463,58	7,01	32	445,19	6,64	30,978	463,55	6,966

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	487,224	48	24	24	24	15,11	441,54	6,7	10,99	429,73	6,37	7,48	501,92	7,91	8,5	470,13	6,94	9	476,97	7,14	10,216	464,06	7,012
2	200x200	no	487,224	48	48	24	24	6,79	507,16	7,51	8,68	502,24	7,88	15,24	425,14	6,18	7,21	537,36	8,56	10,33	491,06	7,42	9,65	492,59	7,51
2	200x200	no	487,224	48	96	24	24	12,95	432,21	6,38	9,82	492,63	7,5	8,97	491,07	7,39	15,44	463,41	6,84	9,87	488,62	7,43	11,41	473,59	7,108
2	200x200	no	487,224	48	168	24	24	14,97	474,81	7,16	15,71	489,15	7,4	25,35	445,54	6,68	13,86	460,85	6,84	15,16	491,35	7,46	17,01	472,34	7,108
2	200x200	no	487,224	48	384	24	24	25,62	493,88	7,6	26,14	491,23	7,58	21	451,64	6,71	17	493,02	7,38	15,82	488,36	7,32	21,116	483,63	7,318

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	487,224	96	24	24	24	8,9	458,29	6,73	8,58	464,86	6,79	7,25	491,84	7,37	10,42	491,97	7,43	9,76	487,3	7,36	8,982	478,85	7,136
2	200x200	no	487,224	96	48	24	24	9,31	486,78	7,34	10,05	495,69	7,53	9,75	484,45	7,28	8,4	481,41	7,27	8,86	485,12	7,23	9,274	486,69	7,33
2	200x200	no	487,224	96	96	24	24	9,72	491,4	7,48	10,36	507,36	7,71	15,55	444,69	6,54	15,8	443,49	6,72	10,54	491,25	7,58	12,394	475,64	7,206
2	200x200	no	487,224	96	168	24	24	15,91	501,66	7,65	8,53	640,4	9,54	8,53	690,06	10,11	15,31	486,42	7,33	9,66	512,84	8	11,588	566,28	8,526
2	200x200	no	487,224	96	384	24	24	32,1	456,76	6,91	27,88	468,85	7,03	18,04	488,46	7,39	16,62	495,69	7,47	19,42	488,17	7,47	22,812	479,59	7,254

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	1,294,424	24	24	24	24	19,11	456,51	6,73	18,97	465,72	7,07	30,2	422,91	6,14	35,52	425,47	6,11	32,15	483,02	7,02	27,19	450,73	6,614
2	200x200	yes	1,294,424	24	48	24	24	24,56	424,87	6,11	16,4	436,04	6,54	19,02	448,87	6,7	29,2	425,29	6,35	17,73	432,7	6,4	21,382	433,55	6,42
2	200x200	yes	1,294,424	24	96	24	24	30,16	435,57	6,49	21,22	44,35	6,57	27,09	418,67	6,09	27,77	416,83	6,08	31,98	418,69	6,07	27,644	346,82	6,26
2	200x200	yes	1,294,424	24	168	24	24	50,75	365,14	5,12	25,93	420,27	6,07	49,6	373,6	5,3	50,52	374,51	5,34	50,9	397,06	5,66	45,54	386,12	5,498
2	200x200	yes	1,294,424	24	384	24	24	70,68	415,68	6,03	95,27	418,86	6,13	62,52	433,42	6,53	81,36	424,02	5,84	48,62	434,18	6,38	71,69	425,23	6,182

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	1,294,424	48	24	24	24	14,5	449,23	6,51	19,22	419,54	6,1	19,18	419,4	6,12	14,17	456,49	6,92	18,16	427,27	6,37	17,046	434,39	6,404
2	200x200	yes	1,294,424	48	48	24	24	14,9	476,61	7,22	19,21	425,59	6,37	17,76	435,6	6,48	13,73	486,74	7,52	13,29	483,77	7,15	15,778	461,66	6,948
2	200x200	yes	1,294,424	48	96	24	24	30,45	454,08	6,92	25,59	430,47	6,35	19,62	437,47	7,06	23,58	436,62	6,48	15,48	482,99	7,32	22,944	448,33	6,826
2	200x200	yes	1,294,424	48	168	24	24	30,96	407,41	5,87	30,56	437,03	6,61	29,69	414,75	5,99	30	463,11	6,95	50,97	391,93	5,49	34,436	422,85	6,182
2	200x200	yes	1,294,424	48	384	24	24	51,44	462,81	6,99	74,92	416,79	6,16	92,11	418,97	6,01	65,19	428,78	6,32	31,81	426,76	6,35	63,094	430,82	6,366

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	1,294,424	96	24	24	24	19,06	423,62	6,12	13,93	469,77	7,13	12,73	485,76	7,35	19,54	438,17	6,46	12,67	479,85	7,25	15,586	459,43	6,862
2	200x200	yes	1,294,424	96	48	24	24	20,21	427,16	6,37	15,35	474,08	7,21	29,73	421,48	6,15	19,34	473,19	7,17	16,1	468,05	7,06	20,146	452,79	6,792
2	200x200	yes	1,294,424	96	96	24	24	23,42	445,92	6,7	29,69	468,49	7,25	33,36	421,3	6,17	17,84	474,34	7,25	25,65	440,29	6,63	25,992	450,07	6,8
2	200x200	yes	1,294,424	96	168	24	24	20,04	488,03	7,6	36,32	403,48	5,75	33,02	432,25	6,53	50,66	404,78	5,84	37,59	408,12	5,91	35,526	427,33	6,326
2	200x200	yes	1,294,424	96	384	24	24	46,28	482,51	7,27	48,22	457,83	6,86	92,07	428,8	6,45	41,69	475,07	7,32	50,96	476,29	7,29	55,844	464,1	7,038

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	20,664	24	24	24	24	30,68	464,09	3,89	14,86	448,07	6,69	19,37	519,47	8,19	20,88	504,52	7,5	18,12	495,58	7,43	20,782	486,35	6,74
2	40x40	no	20,664	24	48	24	24	9,91	497,42	7,39	11,56	491,89	7,49	25,16	433,73	6,42	16,44	434,6	6,39	25,3	433,88	6,3	17,674	458,3	6,798
2	40x40	no	20,664	24	96	24	24	14,38	481,5	7,23	11,11	500,59	7,75	12,62	488,02	7,34	12,62	497,97	7,68	11,51	491,54	7,35	12,448	491,92	7,47
2	40x40	no	20,664	24	168	24	24	15,1	481,09	7,24	11,39	499,68	7,62	10,3	506,48	7,66	10,94	509,64	7,7	10,26	529,57	7,96	11,598	505,29	7,636
2	40x40	no	20,664	24	384	24	24	46,54	475,39	7,16	19,04	488,13	7,38	22,45	483,43	7,2	46,66	469,47	7,21	14,87	511,16	7,62	29,912	485,52	7,314

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	20,664	48	24	24	24	8,84	482,98	7,22	8,23	494,93	7,39	10,06	501,3	7,64	7,89	500,37	7,61	12,18	468,77	7,14	9,44	489,67	7,4
2	40x40	no	20,664	48	48	24	24	11	505,01	7,68	7,62	504,23	7,66	8,5	495,52	7,44	10,79	488,38	7,35	10,39	495,78	7,51	9,66	497,78	7,528
2	40x40	no	20,664	48	96	24	24	25,4	458,75	6,78	13,55	482,84	7,28	15,08	480,51	7,26	15,8	479,88	7,26	11,36	490,82	7,37	16,238	478,56	7,19
2	40x40	no	20,664	48	168	24	24	15,09	505,82	7,56	9,75	510,52	7,79	9,88	500,88	7,63	15,35	490,36	7,62	15,08	499,61	7,55	13,03	501,44	7,63
2	40x40	no	20,664	48	384	24	24	21,64	490,38	7,46	15,76	495,84	7,45	13,98	505,45	7,76	25,82	490,96	7,5	46,41	437,7	6,49	24,722	484,07	7,332

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	20,664	96	24	24	24	10,91	452,95	6,73	9,99	485,46	7,3	7,26	508,52	7,49	8,02	500,09	7,57	14,52	494,5	7,34	10,14	488,3	7,286
2	40x40	no	20,664	96	48	24	24	10,15	499,7	7,52	8,06	503,84	7,7	8,51	499,49	7,54	8,11	498,49	7,52	9,29	495,9	7,45	8,824	499,48	7,546
2	40x40	no	20,664	96	96	24	24	9,76	495,46	7,44	8,98	495,97	7,51	10,43	513,53	7,69	10,87	496,35	7,57	7,77	510,67	7,69	9,562	502,4	7,58
2	40x40	no	20,664	96	168	24	24	15,64	496,8	7,5	8,72	514,15	7,79	13,1	486,42	7,34	16,11	507,39	7,72	9,73	499,05	7,54	12,66	500,76	7,578
2	40x40	no	20,664	96	384	24	24	15,72	498,84	7,55	15,27	499,29	7,58	12,22	502,23	7,59	15,03	505,27	7,72	20,94	479,53	7,21	15,836	497,03	7,53

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	54,104	24	24	24	24	35,57	469,74	7,19	18,9	476,11	7,24	31,47	457,19	6,9	22,31	420,48	6,08	21,88	499,66	7,41	26,026	464,64	6,964
2	40x40	yes	54,104	24	48	24	24	29,03	429,93	6,43	29,22	436,64	6,47	29,43	423,31	6,17	16,18	480,7	7,21	18,89	498	7,62	24,55	453,72	6,78
2	40x40	yes	54,104	24	96	24	24	27,57	447,24	6,61	33,41	417,42	6,04	30,07	463,36	7,03	22,29	445,03	6,7	29,42	443,56	6,67	28,552	443,32	6,61
2	40x40	yes	54,104	24	168	24	24	36,82	411,81	5,92	49,45	379,3	5,36	35,27	413,92	6,07	19,15	471,64	7,12	30,35	431,82	6,34	34,208	421,7	6,162
2	40x40	yes	54,104	24	384	24	24	50,87	489,39	7,33	30	501,9	7,93	51,29	40,45	6,77	91,36	430,6	6,32	48,2	457,29	6,87	54,344	383,93	7,044

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	54,104	48	24	24	24	17,65	445,1	6,58	17,82	436,1	6,34	14,47	487,43	7,34	19,24	447,32	6,54	14,2	485,08	7,42	16,676	460,21	6,844
2	40x40	yes	54,104	48	48	24	24	16,13	485,23	7,28	16,65	476,04	7,22	17,11	477,67	7,2	18,97	429,87	6,34	14,71	483,82	7,35	16,714	470,53	7,078
2	40x40	yes	54,104	48	96	24	24	19,78	492,89	7,53	19,11	494,19	7,39	19,96	487,97	7,39	18,73	478,23	7,37	19,9	467,82	7,01	19,496	484,22	7,338
2	40x40	yes	54,104	48	168	24	24	29,97	453,23	6,88	21,95	438,7	6,44	22,32	452,21	6,74	16,77	474,43	7,13	49,82	411,07	6,03	28,166	445,93	6,644
2	40x40	yes	54,104	48	384	24	24	51,1	447,06	6,68	50,52	477,3	7,46	60,76	416,13	6,1	30,37	475,49	7,47	30,36	476,28	7,21	44,622	458,45	6,984

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	54,104	96	24	24	24	14,57	487,48	7,45	13,78	483,61	7,21	14,61	485,15	7,27	13,01	484,14	7,25	12,42	488,37	7,3	13,678	485,75	7,296
2	40x40	yes	54,104	96	48	24	24	16,56	471,77	7,16	14,08	521,64	7,71	14,66	491,21	7,42	11,92	536,94	8,2	13,86	507,1	7,68	14,216	505,73	7,634
2	40x40	yes	54,104	96	96	24	24	18,98	486,51	7,74	14,5	493,13	7,42	17,17	481,84	7,3	15,7	477,98	7,23	16,46	480,62	7,21	16,562	484,02	7,38
2	40x40	yes	54,104	96	168	24	24	19,07	465,87	6,91	19,45	489,61	7,43	19,41	469,26	7,13	19,33	465,94	6,91	19,6	467,23	6,96	19,372	471,58	7,068
2	40x40	yes	54,104	96	384	24	24	29,59	482,21	7,28	20,62	511,13	7,82	30,81	497,97	7,62	29,38	481,47	7,32	30,52	483,92	7,33	28,184	491,34	7,474

Table 8 : GRU (24h Out)

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	367,824	24	24	24	24	7,28	497,96	7,73	10,84	409,57	5,91	10,68	410,12	5,9	14,51	392,67	5,56	14,2	384,09	5,39	11,502	418,88	6,098
2	200x200	no	367,824	24	48	24	24	12,6	402,76	5,71	25,75	402,29	5,69	15,26	402,12	5,65	25,25	389,13	5,5	25,46	384,36	5,47	20,864	396,13	5,604
2	200x200	no	367,824	24	96	24	24	16,73	393,62	5,53	15,15	411,6	5,97	16,81	403,78	5,75	13,61	416,38	6,08	25,03	388,87	5,47	17,466	402,85	5,76
2	200x200	no	367,824	24	168	24	24	25,08	406,72	5,78	22,32	401,68	5,78	15,58	442,46	6,67	25,38	398,93	5,66	8,78	502,86	7,68	19,428	430,53	6,314
2	200x200	no	367,824	24	384	24	24	45,97	410,87	5,92	46,4	394,16	5,56	33,42	402,85	5,76	46,36	409,33	5,86	31,97	412,36	5,95	40,824	405,91	5,81

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	367,824	48	24	24	24	6,93	495,91	7,44	7,86	494,54	7,59	9,85	434,58	6,44	14,68	399,06	5,72	14,98	409,54	5,84	10,86	446,73	6,606
2	200x200	no	367,824	48	48	24	24	14,65	405,86	5,77	10,92	412,47	5,93	15,18	423,83	6,35	6,94	496,59	7,43	15,12	405,43	5,76	12,562	428,84	6,248
2	200x200	no	367,824	48	96	24	24	12,73	417,72	6,11	13,99	409,53	5,92	15,34	416,42	5,99	15,3	408,56	5,86	12,4	417,75	6,07	13,952	414	5,99
2	200x200	no	367,824	48	168	24	24	19,53	403,18	5,73	7,94	498,93	7,66	17,79	412,32	5,91	25,38	414,39	5,99	18,03	414,12	6,1	17,734	428,59	6,278
2	200x200	no	367,824	48	384	24	24	26,05	435,58	6,46	15,66	493,48	7,71	28,71	408,39	5,83	26,82	417,8	6,05	46,49	399,16	5,67	28,746	430,88	6,344

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	no	367,824	96	24	24	24	7,02	495,27	7,6	9,33	421,09	6,14	6,62	490,35	7,47	7,63	484,49	7,33	6,89	498,77	7,67	7,498	477,99	7,242
2	200x200	no	367,824	96	48	24	24	7,09	498,95	7,54	6,8	503,63	7,71	10,16	419,96	6,08	9,53	481,28	7,3	10,5	418,84	6,09	8,816	464,53	6,944
2	200x200	no	367,824	96	96	24	24	15,22	418,23	6,09	9,78	479,36	7,38	8,15	487,92	7,54	15,07	412,41	5,95	9,65	490,65	7,57	11,574	457,71	6,906
2	200x200	no	367,824	96	168	24	24	10,45	494,93	7,46	10,67	505,48	7,82	11,66	478,42	7,16	26,24	404,98	5,78	25,44	427,32	6,35	16,892	462,23	6,914
2	200x200	no	367,824	96	384	24	24	12,37	495,81	7,62	16,57	486,72	7,37	15,09	487,53	7,43	17,98	486,45	7,47	17,79	487,57	7,39	15,96	488,82	7,456

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	975,624	24	24	24	24	16,33	416,47	6,01	20,04	391,29	5,57	18,7	422,45	6,08	28,94	394,24	5,6	21,03	382,97	5,37	21,008	401,48	5,726
2	200x200	yes	975,624	24	48	24	24	19,23	430,63	6,35	19,12	414,32	6,04	29,41	402,28	5,73	20,9	398,34	5,62	28,92	401,93	5,79	23,516	409,5	5,906
2	200x200	yes	975,624	24	96	24	24	28,91	399,84	5,68	24,48	396,8	5,63	49,68	399,66	5,72	49,72	391,29	5,52	26,44	408,07	5,91	35,846	399,13	5,692
2	200x200	yes	975,624	24	168	24	24	35,4	345,42	4,6	49,95	330,39	4,47	49,86	347,81	4,79	36,24	340,5	4,56	30,89	355,71	4,91	40,468	343,97	4,669
2	200x200	yes	975,624	24	384	24	24	91,87	397,66	5,62	91,66	395,33	5,52	48,17	409,51	5,9	50,67	406,64	5,73	50,89	396,58	5,63	66,652	401,14	5,68

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	975,624	48	24	24	24	13,98	429,99	6,27	15,96	414,83	5,96	14,09	420,05	6,13	18,85	394,64	5,59	13,45	420,43	6,1	15,266	415,99	6,01
2	200x200	yes	975,624	48	48	24	24	18,95	407,33	5,85	18,91	419,31	6,2	18,95	407,34	5,89	19,02	417,83	6,15	17,89	397,94	5,7	18,744	409,95	5,958
2	200x200	yes	975,624	48	96	24	24	29,63	412,32	5,92	15,48	455,09	6,75	29,42	408,45	5,91	29,42	412,85	6	25,57	401,65	5,76	25,904	418,07	6,068
2	200x200	yes	975,624	48	168	24	24	30,77	371,62	5,19	29,5	410,19	5,92	50,08	342,86	4,75	16,53	465,51	6,96	49,86	338,93	4,64	35,348	385,82	5,492
2	200x200	yes	975,624	48	384	24	24	92,02	401,93	5,68	50,66	407,1	5,84	91,64	405,12	5,73	50,9	406,32	5,87	46,45	411,17	6,04	66,334	406,33	5,832

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	200x200	yes	975,624	96	24	24	24	14,81	420,99	6,12	13,73	430,33	6,31	14,46	419,2	6,12	18,87	433,04	6,37	15,3	415,4	6,08	15,434	423,79	6,2
2	200x200	yes	975,624	96	48	24	24	29,21	405,94	5,86	18,37	409,12	5,94	29,14	407,79	5,86	18,98	422,88	6,25	19,72	401,74	5,81	23,084	409,49	5,944
2	200x200	yes	975,624	96	96	24	24	29,59	412,59	6,03	29,65	407,65	5,88	26,29	413,74	5,98	29,57	410,99	6	16,45	455,02	6,82	26,31	420	6,142
2	200x200	yes	975,624	96	168	24	24	26,83	421,19	6,29	50	360,34	4,9	29,72	413,49	6,11	36,86	364,89	5,07	44,99	341,62	4,61	37,68	380,31	5,396
2	200x200	yes	975,624	96	384	24	24	50,82	456,27	6,87	91,71	400,74	5,77	50,7	465,44	7,08	91,73	408,88	5,87	50,67	456,87	6,8	67,126	437,64	6,478

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	15,984	24	24	24	24	24,91	419,83	6,08	15,42	411,23	5,94	14,73	427,29	6,25	10,1	489,63	7,44	7,28	508,85	7,78	14,488	451,37	6,698
2	40x40	no	15,984	24	48	24	24	14,46	421,18	6,09	9,18	478,69	7,28	24,92	409,07	5,91	8,54	493,22	7,59	8,86	490,31	7,42	13,192	458,49	6,858
2	40x40	no	15,984	24	96	24	24	10,9	482,43	7,42	20,1	406,96	5,88	9,7	492,06	7,55	16,94	426,03	6,27	8,38	496,76	7,58	13,204	460,85	6,94
2	40x40	no	15,984	24	168	24	24	45,64	401,78	5,76	12,54	484,17	7,44	11,59	484,7	7,27	33,06	401,61	5,74	21,87	423,62	6,26	24,94	439,18	6,494
2	40x40	no	15,984	24	384	24	24	46,04	401,85	5,77	25,31	482,67	7,38	44,76	406,34	5,87	18,16	482,15	7,22	46,31	411,1	5,87	36,116	436,82	6,422

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	15,984	48	24	24	24	7,23	496,71	7,62	7,84	488,21	7,4	8,81	478,44	7,14	10,04	484,21	7,35	9,57	493,62	7,53	8,698	488,24	7,408
2	40x40	no	15,984	48	48	24	24	10,12	480,56	7,2	7,33	489,38	7,43	10,08	476,39	7,23	10,34	492,26	7,54	9,53	488,89	7,46	9,48	485,5	7,372
2	40x40	no	15,984	48	96	24	24	9,85	484,38	7,21	9,75	488,9	7,42	9,66	483,28	7,3	10,23	487,23	7,32	9,98	489,03	7,38	9,894	486,56	7,326
2	40x40	no	15,984	48	168	24	24	10,19	490,7	7,49	25,53	409,13	6	9,71	475,21	7,37	10,48	489,23	7,37	10,59	484,23	7,38	13,3	469,7	7,122
2	40x40	no	15,984	48	384	24	24	15,33	487,7	7,35	15,18	479,73	7,29	15,1	483,26	7,34	30,38	416,46	6,05	33,35	413,81	6,05	21,868	456,19	6,816

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN			5th RUN			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	no	15,984	96	24	24	24	9,64	484,71	7,34	14,91	413,33	6	7,61	488,12	7,41	7,52	488,3	7,32	6,58	495,97	7,63	9,252	474,09	7,14
2	40x40	no	15,984	96	48	24	24	9,64	481,44	7,19	7,24	498,12	7,76	7,47	493,12	7,48	9,05	480,97	7,33	8,99	479,35	7,17	8,478	486,6	7,386
2	40x40	no	15,984	96	96	24	24	8,79	487,82	7,39	10,68	477,71	7,36	10,08	486,55	7,41	10,06	489,21	7,39	9,78	491,97	7,49	9,878	486,65	7,408
2	40x40	no	15,984	96	168	24	24	8,54	495,55	7,58	10,2	488,64	7,45	16,95	436,29	6,53	9,95	489,01	7,42	8,6	487,35	7,38	10,848	479,37	7,272
2	40x40	no	15,984	96	384	24	24	13,62	487,84	7,35	15,53	496,61	7,57	15,6	493,84	7,57	15,65	489,79	7,55	15,09	492,33	7,53	15,098	492,08	7,514

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	41,544	24	24	24	24	18,9	418,6	6,12	28,77	415,16	6,04	19,25	421,8	6,14	19,02	415,01	5,96	16,46	432,02	6,39	20,48	420,52	6,13
2	40x40	yes	41,544	24	48	24	24	29,3	423,97	6,25	49,38	400,69	5,71	19,49	427,81	6,27	25,16	408,42	5,9	20,97	423,2	6,19	28,86	416,82	6,064
2	40x40	yes	41,544	24	96	24	24	32	408,95	5,92	33,08	401,12	5,75	28,81	411,49	5,97	29	412,74	5,94	37,48	399,92	5,66	32,074	406,84	5,848
2	40x40	yes	41,544	24	168	24	24	41,53	361,34	4,88	38,4	364,85	5,07	50,17	388,19	5,51	31,04	391,38	5,53	28,05	394,72	5,63	37,838	380,1	5,324
2	40x40	yes	41,544	24	384	24	24	91,28	405,59	5,77	91	400,74	5,69	56,72	403,99	5,78	91,15	394,38	5,6	71,16	395,73	5,67	80,262	400,09	5,702

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	41,544	48	24	24	24	17,33	420,46	6,09	18,78	429,77	6,25	19,05	416,9	6,02	18,47	420,44	6,16	18,81	425,49	6,26	18,488	422,61	6,156
2	40x40	yes	41,544	48	48	24	24	19,23	422,29	6,23	18,63	423,62	6,3	17,41	426,16	6,29	17,39	419,17	6,15	14,52	453,37	6,86	17,436	428,92	6,366
2	40x40	yes	41,544	48	96	24	24	19,12	477,01	7,23	19,27	436,89	6,48	29,53	405,89	5,83	28,89	425,74	6,21	29,5	411,05	5,93	25,262	431,32	6,336
2	40x40	yes	41,544	48	168	24	24	36,94	358,8	4,88	18,97	461,44	6,91	38,15	358,85	4,94	28,72	414,46	6,07	24,38	405,16	5,85	29,432	399,74	5,73
2	40x40	yes	41,544	48	384	24	24	46,43	412,7	5,97	91,02	403,01	5,81	30,21	485,15	7,34	91,16	393,56	5,58	39,02	427	6,26	59,568	424,28	6,192

layers	Neurons	bidirectional	Params	batch	Inputs	Outputs	Win_Size	1st RUN			2nd RUN			3rd RUN			4th RUN (Dropout No)			5th RUN (Dropout No)			AVG		
								sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE
2	40x40	yes	41,544	96	24	24	24	19,51	412,29	5,99	13,06	476,47	7,11	19,15	422,25	6,2	18,5	424,58	6,21	13,7	450,27	6,65	16,784	437,17	6,432
2	40x40	yes	41,544	96	48	24	24	18,79	418,48	6,16	16,11	430,32	6,34	19,16	463,14	6,95	18,54	440,67	6,47	15,52	439,32	6,52	17,624	438,39	6,488
2	40x40	yes	41,544	96	96	24	24	21,33	424,2	6,26	29,03	434,91	6,43	19,35	456,32	6,8	28,93	418,3	6,14	22,3	417,94	6,05	24,188	430,33	6,336
2	40x40	yes	41,544	96	168	24	24	29,55	411,9	5,99	19,04	462,51	6,95	29,77	402,03	5,87	18,95	462,5	7,01	15,47	464,04	6,97	22,556	440,6	6,558
2	40x40	yes	41,544	96	384	24	24	27,49	457,21	6,87	29,51	451,42	6,76	30,3	476,81	7,24	40	418,09	5,99	18,94	495,34	7,64	29,248	459,77	6,9

Regarding to the results of the above tables we can comment that:

- In addition to the RMSE & MAPE scores we have also kept the time it took to run each model with each set of parameters. The RNN architecture makes them excruciatingly slow. In contrast, the new and more modern GRU models are at least 5 times faster than RNNs. From theory we knew about RNNs being slow to train. We confirmed this experimentally.
- The initial impression that we had previously gotten with the FFN Network construction was that by increasing the input vector, we would also get more accuracy. We saw previously that with 24h input we had a MAPE score of 1.91% while with 12h we had 2.28%. But as we can see in the tables above, this is not the norm. In many cases we see that the prediction is better with an input vector of 48h compared to an input vector of 168h and the same set of parameters.
- Adding more layers to a model can increase its capacity, allowing it to learn more complex representations of the data, but it also increases the risk of overfitting. This happens when the model becomes too complex and starts to fit the noise in the training data rather than the underlying patterns. Additionally, with more layers, the model may require more data to generalize well, and it may also become computationally expensive to train. In general, increasing the number of layers in a model can be a good way to improve its performance, but it is important to be aware of the trade-offs involved. It's worth noting that adding more layers does not always improve the performance, and sometimes it may even decrease the performance.
- Bidirectional networks can improve prediction performance in certain tasks. This is because they take into account both past and future context when making a prediction, whereas a traditional unidirectional network only considers past context. However, the improvement in prediction performance will depend on the specific task and dataset being used. It is worth testing a bidirectional network and comparing its performance to a unidirectional network to see if it provides any benefit for a specific use case.

- GRUs are simpler and faster to train than LSTMs because they have one less gate, which makes them more efficient in terms of computation time. However, in terms of accuracy, there is no clear advantage between GRUs and LSTMs as it depends on the specific task and dataset. Both architectures have been used to achieve state-of-the-art results on various tasks, and the choice between them often comes down to personal preference and experimentation.
- In comparison with other 2 architectures like LSTMs or GRUs, RNNs have less number of gates/memory cells which make them less powerful in terms of performance, especially when it comes to long-term dependencies in the data. That is the reason that RNNs seem to have the worst prediction results among these 3 models. But it is worth noting that RNNs are still widely used in various applications, such as language modeling and speech recognition, and they can be very effective when used in the right context.

- Increasing the number of neurons in a layer can lead to a more powerful model, as it allows for more complex representations to be learned. However, it is not always the case that increasing the number of neurons leads to better performance. This is because increasing the number of neurons also increases the risk of overfitting, which occurs when a model is too complex and starts to fit the noise in the training data rather than the underlying patterns. Another reason is the concept of over-parametrization, where a model with too many parameters will not be able to generalize well to new data, because it would have learned the noise rather than the underlying pattern of the training data. As we can see in the tables above, a model with fewer neurons can often be simpler and more generalizable, which can lead to better performance on unseen data.

- Because of their ability to maintain a "memory" of the previous inputs, LSTMs and GRUs can predict two subsequent load values with as much accuracy as feed-forward neural networks (FFNNs) need to predict only one subsequent value. This is because LSTMs and GRUs can use the

information from previous inputs to better understand the context of the current input and make more accurate predictions.

- The batch size is a hyperparameter that controls the number of samples used in one forward/backward pass. A batch size of 2^n (n is an integer) is often used in practice because it is more efficient to process data in batches that are a power of 2, as it can take advantage of the memory hierarchy in modern CPUs and GPUs.

- It is generally expected that increasing the length of the prediction vector (i.e. the number of steps ahead that the model is trying to predict) in RNNs, LSTMs, and GRUs will increase the error in the predictions. As the length of the prediction vector increases, the model is required to make more predictions and maintain a longer-term memory of the input sequence, which can be more challenging. Longer prediction vectors also require the model to make predictions that are based on more complex dependencies between the input and output. As a result, the model may be less able to accurately predict the future values, and the error may increase. However, it's worth noting that this relationship is not always linear and it depends on the specific task, dataset and model.

  It is important to note that the error rate may not be the only metric that matters, in some cases the model may be able to predict the future values with high accuracy but the actual values may be different from the predicted ones, in such cases the model may still be useful depending on the use case. In general, it's important to experiment with different prediction vector lengths, and use techniques such as regularization and early stopping to mitigate the impact of increasing the prediction vector length on the error.

- From all the models with an output vector length of 24h, the majority of MAPE scores are above 5%. As we can see in the tables only GRU models achieved MAPE scores of less than 5% in this category. Specifically, they achieved up to 4.66% (344MWh RMSE)

After many runs with different combinations of parameters we came up with 2 best load prediction models for the next 1h. From the GRUs the best model includes:

- 2 layers
- 500 neurons per layer
- input lenght = 32h
- batch size= 16

Table 9 : Best 5 GRU models

BEST 5 MODELS for Out=1 & Win_Size=1																	
layers	Neurons	bidirectional	Params	batch	Inputs	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	AVG		
						1st RUN			2nd RUN			3rd RUN			sec	RMSE	MAPE
2	150x150	no	271,951	12	48	445,2	92,79	1,36	444,9	88,83	1,26	684,9	84,59	1,2	525	88,74	1,273
2	400x400	no	1,925,201	12	48	452,3	*87,11*	*1,25*	615,4	*84,84*	*1,23*	505,3	*86,31*	*1,22*	524,3	86,09	1,233
2	500x500	no	3,006,501	16	32	444,7	*82,83*	*1,17*	324,4	*84,54*	*1,19*	274,3	*82,74*	*1,17*	347,8	83,37	1,177
2	200X200	yes	1,285,201	16	32	446,9	83,22	1,19	505,9	84,18	1,19	506,9	82,23	1,16	486,6	83,21	1,18
2	200x200	no	363,201	16	32	210,1	86,86	1,25	324,3	81,63	1,14	282,5	82,4	1,17	272,3	83,63	1,187

The best GRU model has RMSE 83,2 MWh. From the LSTMs the best model includes:

- 2 layers
- 500 neurons per layer
- input lenght = 32h
- batch size= 16

Table 10 : Best 5 LSTM models

BEST 5 MODELS for Out=1 & Win_Size=1																	
layers	Neurons	bidirectional	Params	batch	Inputs	sec	RMSE	MAPE	sec	RMSE	MAPE	sec	RMSE	MAPE	AVG		
						1st RUN			2nd RUN			3rd RUN			sec	RMSE	MAPE
2	150x150	no	271,95	12	48	645	89,37	1,27	564,7	90,12	1,29	774,7	85,06	1,21	661,5	88,18	1,257
2	400x400	no	1,925,201	12	48	685,3	*90,84*	*1,28*	745	*88,3*	*1,28*	852,2	*84,53*	*1,22*	760,8	87,89	1,26
2	500x500	no	3,006,501	16	32	505,7	*83,41*	*1,17*	503,4	*85,17*	*1,19*	744,1	*82,84*	*1,18*	584,4	83,81	1,18
2	200X200	yes	1,285,201	16	32	565,1	90,05	1,29	503,7	88,11	1,25	566,5	85,02	1,19	545,1	87,73	1,243
2	200x200	no	482,601	16	32	624,9	85,55	1,2	444,7	89,35	1,28	444,5	85,12	1,2	504,7	86,67	1,227

The best LSTM model has RMSE 83,8 MWh.

Section 3: Dynamic Power Bi Report

3.1 Power Bi report

We enriched our original testing data with the following metadata:

- Year
- Month
- Day
- Hour
- Season
- Week Day
- Date

for further statistical analysis. A snapshot of all these additional features is shown in the picture below.

Table 11 : Preview of metadata

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
INDEX	Year	Month	Day	Season	Week Day	Date	Y-REAL	Hour	LSTM 6IN	LSTM 12IN	LSTM 24IN	LSTM 48IN	LSTM 96IN	LSTM 168IN
1	2016	1	9	4	5	9/1/2016	5172	0:00:00	5569,52	5619,8	5495,791	5640,929	5510,854	5675,4517
2	2016	1	9	4	5	9/1/2016	4756	1:00:00	4400,22	4935,753	4972,785	4921,194	4852,802	4967,4316
3	2016	1	9	4	5	9/1/2016	4680	2:00:00	4319,67	4886,706	4747,69	4626,477	4767,303	4744,7358
4	2016	1	9	4	5	9/1/2016	4482	3:00:00	4502,08	4697,491	4617,211	4487,786	4511,794	4710,1489
5	2016	1	9	4	5	9/1/2016	4334	4:00:00	4417,81	4584,224	4435,813	4372,135	4536,977	4605,7368
6	2016	1	9	4	5	9/1/2016	4366	5:00:00	4343,65	4617,004	4427,77	4610,301	4571,725	4604,5586
7	2016	1	9	4	5	9/1/2016	4523	6:00:00	4473	4931,813	4763,439	5090,193	4863,326	5033,2969
8	2016	1	9	4	5	9/1/2016	4770	7:00:00	4717,08	5277,738	5260,87	5167,688	5116,593	5106,5059
9	2016	1	9	4	5	9/1/2016	4978	8:00:00	5026,49	5142,637	5304,979	5131,729	5111,891	5031,7617
10	2016	1	9	4	5	9/1/2016	5051	9:00:00	5155,99	5156,543	5270,398	5174,333	5215,31	5135,9634
11	2016	1	9	4	5	9/1/2016	4940	10:00:00	5091,87	5152,19	5242,664	5166,231	5161,843	5139,1719
12	2016	1	9	4	5	9/1/2016	4845	11:00:00	4859,36	4821,107	4963,708	4980,583	4905,244	4908,6006
13	2016	1	9	4	5	9/1/2016	4850	12:00:00	4805,65	4800,744	4735,767	4838,184	4756,551	4708,21
14	2016	1	9	4	5	9/1/2016	4978	13:00:00	4904,05	4853,861	4775,292	4829,06	4716,823	4648,564
15	2016	1	9	4	5	9/1/2016	4967	14:00:00	5148,1	5087,192	5083,763	5030,694	4989,132	4938,6743
16	2016	1	9	4	5	9/1/2016	5493	15:00:00	5040,98	5082,287	5245,659	5182,654	5174,57	5183,9526
17	2016	1	9	4	5	9/1/2016	5836	16:00:00	5944,18	5946,074	6061,233	5955,995	5933,362	5890
18	2016	1	9	4	5	9/1/2016	6242	17:00:00	6051,66	6137,758	6603,147	6463,932	6367,33	6410,3472
19	2016	1	9	4	5	9/1/2016	6806	18:00:00	6442,17	6481,808	7045,934	6878,542	6761,7	6886,7002
20	2016	1	9	4	5	9/1/2016	6810	19:00:00	6940,31	6662,705	7317,048	7144,706	6954,381	7038,0464
21	2016	1	9	4	5	9/1/2016	6603	20:00:00	6618,05	6431,716	7029,604	6919,198	6759,666	6818,0039
22	2016	1	9	4	5	9/1/2016	6239	21:00:00	6367,28	6265,381	6567,871	6502,842	6382,98	6428,231
23	2016	1	9	4	5	9/1/2016	5751	22:00:00	5889,68	5822,299	6013,896	5985,473	5900,681	5926,5972
24	2016	1	9	4	5	9/1/2016	5402	23:00:00	5388,9	5354,244	5379,413	5444,818	5326,084	5400,7168
25	2016	1	10	4	6	10/1/2016	4920	0:00:00	5158,62	5112,202	4980,743	5079,632	4886,956	5045,4341
26	2016	1	10	4	6	10/1/2016	4505	1:00:00	4628,14	4755,328	4647,387	4701,165	4503,094	4692,1143
27	2016	1	10	4	6	10/1/2016	4425	2:00:00	4267,43	4434,378	4373,728	4371,588	4246,416	4407,5942
28	2016	1	10	4	6	10/1/2016	4216	3:00:00	4390,01	4311,783	4282,057	4293,206	4218,701	4347,791
29	2016	1	10	4	6	10/1/2016	4093	4:00:00	4154,9	4135,909	4141,25	4162,306	4123,371	4195,6313
30	2016	1	10	4	6	10/1/2016	4037	5:00:00	4109,98	4212,166	4156,593	4261,262	4095,334	4199,3853
31	2016	1	10	4	6	10/1/2016	4121	6:00:00	4086,63	4261,632	4293,828	4367,647	4206,813	4348,894

FFNN | RNN40 | RNN200 | LSTM40 | LSTM200 | GRU40 | GRU200 | BEST

We uploaded into Power Bi all the metadata with load predictions for the next ONE hour. The models involved are:

➔ 3 FFNN (100 x 50 neurons) models which are:

- FFNN with 24h Input lenght
- FFNN with 12h Input lenght
- FFNN with 48h horizon

➔ 5 RNN (40 x 40 neurons) models which are:

- RNN with 6h Input lenght
- RNN with 12h Input lenght
- RNN with 48h Input lenght
- RNN with 96h Input lenght
- RNN with 168h Input lenght

➔ 5 RNN (200 x 200 neurons) models which are:

- RNN with 12h Input lenght
- RNN with 24h Input lenght
- RNN with 48h Input lenght
- RNN with 96h Input lenght
- RNN with 168h Input lenght

➔ 6 LSTM (40 x 40 neurons) models which are:

- LSTM with 6h Input lenght
- LSTM with 12h Input lenght
- LSTM with 12h Input lenght
- LSTM with 48h Input lenght
- LSTM with 96h Input lenght
- LSTM with 168h Input lenght

➔ 6 LSTM (200 x 200 neurons) models which are:

- LSTM with 6h Input lenght
- LSTM with 12h Input lenght
- LSTM with 12h Input lenght
- LSTM with 48h Input lenght
- LSTM with 96h Input lenght
- LSTM with 168h Input lenght

➔ 1 LSTM (500 x 500 neurons) model which is:

- LSTM with 32h Input lenght

➔ 6 GRU (40 x 40 neurons) models which are:

- GRU with 6h Input lenght
- GRU with 12h Input lenght

- GRU with 12h Input lenght
- GRU with 48h Input lenght
- GRU with 96h Input lenght
- GRU with 168h Input lenght

➔ 6 GRU (200 x 200 neurons) models which are:

- GRU with 6h Input lenght
- GRU with 12h Input lenght
- GRU with 12h Input lenght
- GRU with 48h Input lenght
- GRU with 96h Input lenght
- GRU with 168h Input lenght

➔ 1 GRU (500 x 500 neurons) model which is:

- GRU with 32h Input lenght

Therefore, we uploaded all 39 model variations to the tool. The Home page we designed has the following format.

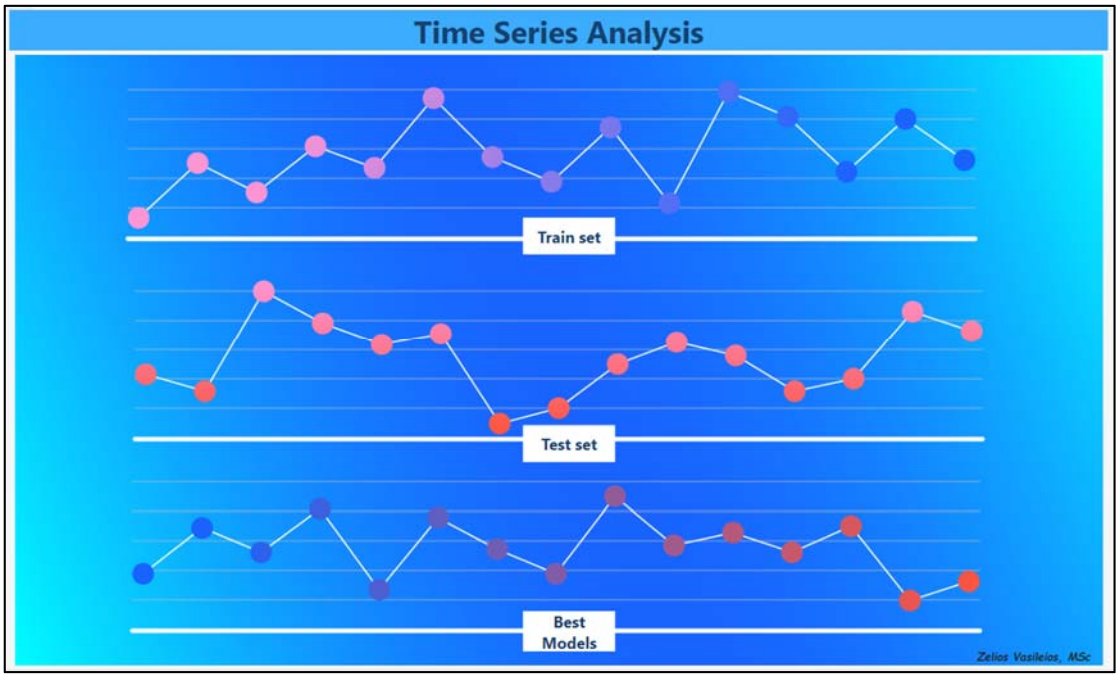

Figure 34 : Report's HomePage

We can see 3 buttons. The first one is about the dataset used to train the models, the second one is about the predictions of the models in the test set and the third one focuses on the 2 best models mentioned in chapter 2. It is reminded that we have uploaded to Power Bi predictions of 1h due to the high

accuracy of the models. Selecting the Train set button leads to the following screenshot.

Figure 35 : Train dataset overview

At the top of the screen, we can find the filters of:

- the Seasons (a dropdown list containing winter, spring, summer and autumn),
- the Load (a between Min-Max slicer of Actual Load)
- the Date and (a calendar of training data)
- the Day (a dropdown list containing the names of the 7 days)

In the filter area there is also a “2013-2014-2015 COMPARISON” button which, by pressing it, leads us to the next snapshot.

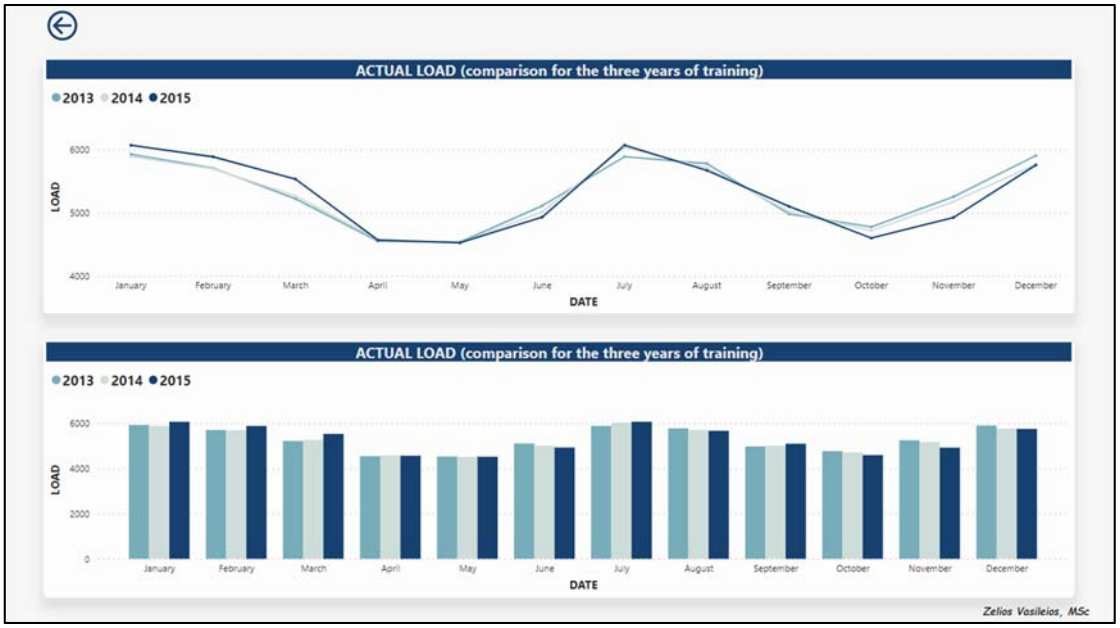

Figure 36 : 3-year comparison

Here the user can see the load during the 3 years of training in a line chart and bar chart. We note that in all 3 years the pattern of the time series is similar. In spring and autumn, the demand for power is low while in summer and winter the demand is high. This is normal because in our country we use electricity in the summer to cool our homes with air conditioners and in the winter to heat them with electric stoves and other electrical appliances. The return button on the top-left of the screen takes us back to the previous page.

Below the filter area we will find a box that contains some statistical measures relating to the time series for the years 2013 to 2015. These measures are:

- Average
- Median
- Max
- Min
- Sum

These values change dynamically depending on the choice of filters. This is the power provided by the Bi tool. For example, by selecting from the seasons the “winter” and from the days “Thursday” we see that the values of the statistical measures have been adjusted accordingly.

Figure 37 : Apply filters

We also noticed that the average load on Sundays (and only on Sundays) is less than 5 thousand MWh. High load demand occurs on weekdays and mainly on Thursdays. This is explained because in our country Sunday is a day of rest and most companies are closed, so they do not consume energy.

Figure 38 : Sunday consumption

Clicking on the master's logo on the top-left, will take you back to the home page. By clicking this time on the test set button lead us to the screenshot below.

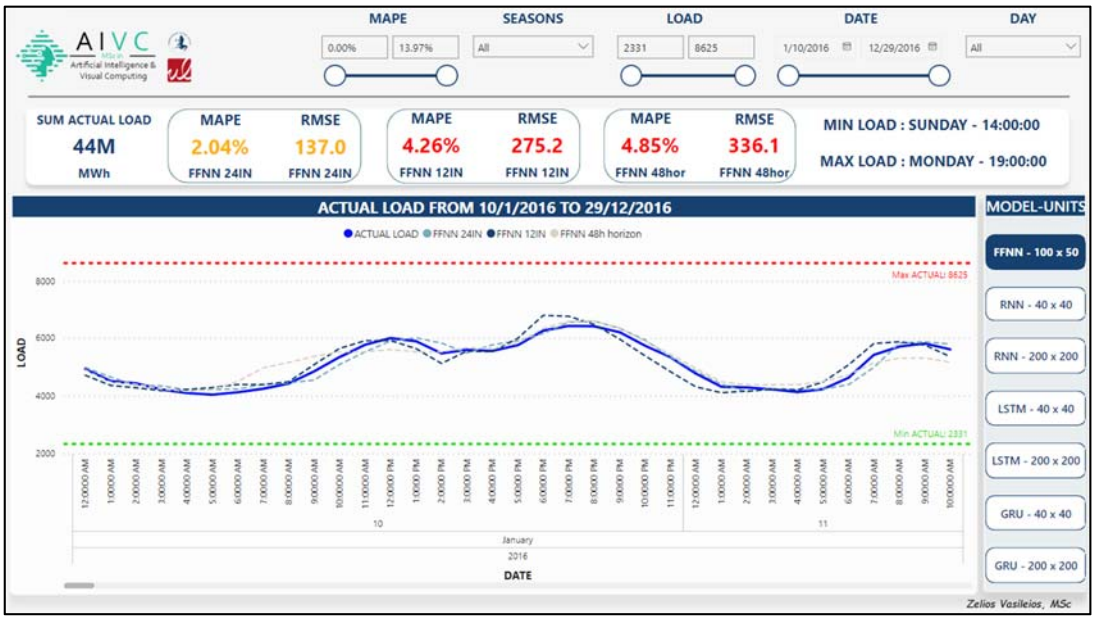

Figure 39 : Test dataset overview

The environment is similar to that of the train. On the right there is a list of the 7 available models that were trained with different input vector lengths each time. As we have already mentioned in chapter 2, we also see here that increasing the input length of the models does not necessarily mean that we will achieve better prediction. As we can see below, choosing a GRU model with 200 x 200 neurons we confirm that the GRU model with an input vector of 168h is not efficiently the best.

Figure 40 : GRU (200 x 200)

By making a horizontal scroll on the graph's bar we can generally observe that all the GRU (200 x 200) models adapt smoothly to the changes of the ACTUAL LOAD curve. We also notice that for short input length we get high error. Increasing input length from 6h to 12h the error decreases. Increasing the length again to 24h the error grows and reaches about the same percentage as when the input has a length of 6h. With vector 48h as input we get the best results in prediction.

Below is a screenshot of the LSTM model (40 * 40 units) without the application of any filter. We see that 4 of the 6 variants of the model have up to ~2% MAPE score (<135MWh MAPE).

Figure 41 :LSTM (40 x 40)

Now, for the same model (LSTM 40 x 40) we choose from the DAY filter the "Sunday"

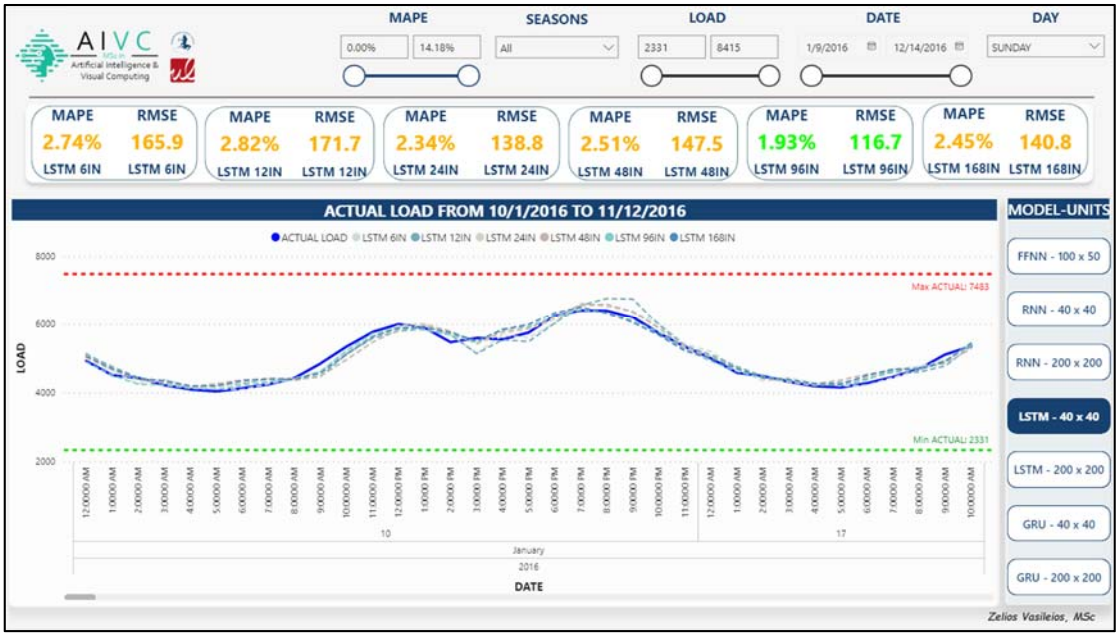

Figure 42 : LSTM (40 x 40) - Sunday

We can see here that on Sundays the model doesn't work so well. Only the run with 96h as input vector has RMSE <135MWh. We also test the RNN (40 x 40) without filters to compare the results with those of Sunday.

Figure 43 : RNN (40 x 40)

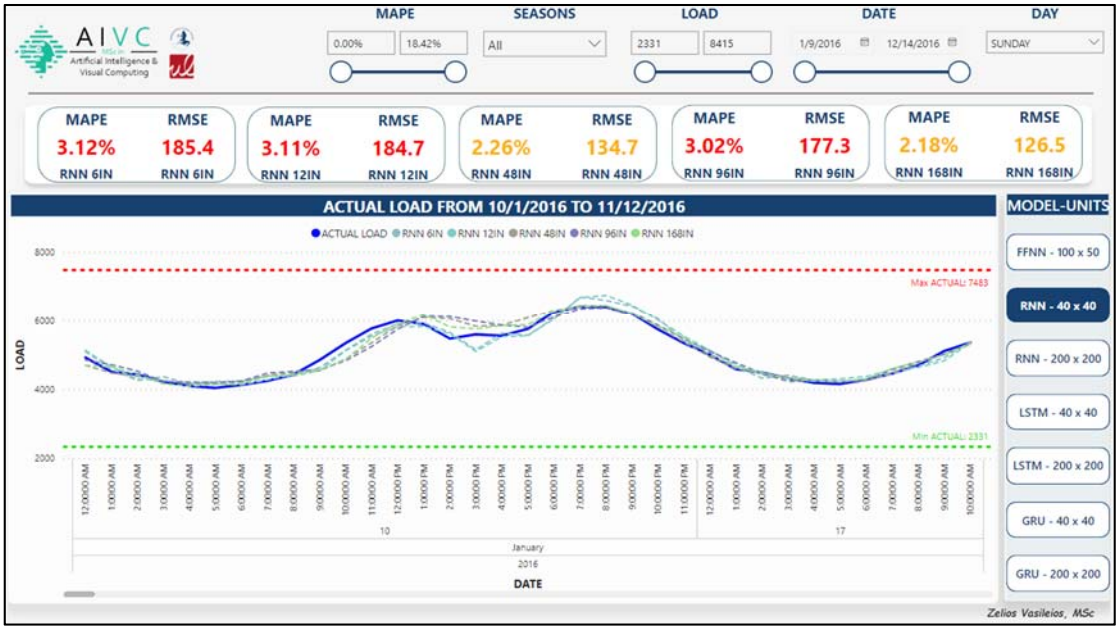

Figure 44 : RNN (40 x 40) – Sunday

Figure 45 : RNN (40 x 40) – Sunday

This model doesn't work well enough. It has difficulties capturing patterns in the data that occur infrequently, such as on weekends or holidays [26]. So, to generalize this we try the 15th of August for the same model.

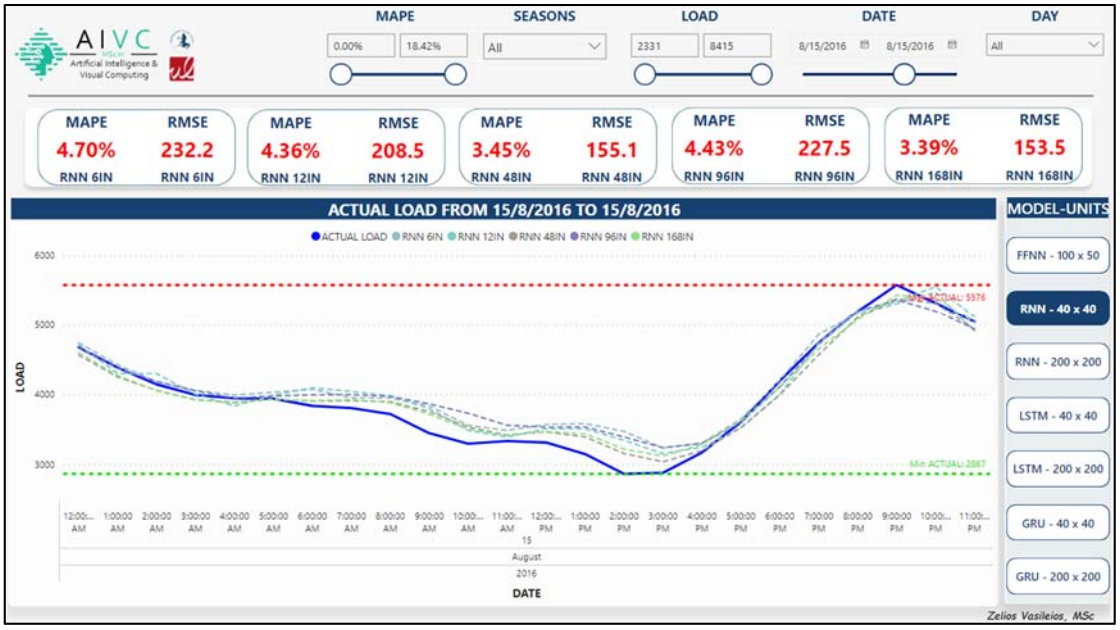

Figure 46 : RNN (40 x 40) - 15th August

The problem here is greater than on Sundays. This is because the model might not have seen enough examples of this type of pattern during the training process, making it difficult for the model to learn to recognize them. We try the same day (August 15th) for both LSTM (40 x 40) and GRU (40 x 40) models and we see that the “problem” of August 15th remains for these models as well.

Figure 47 : LSTM (40 x 40) - 15th August

Figure 48 : GRU (40 x 40) - 15th August

After this we tested LSTM and GRU with 200 x 200 units to see what they did on August 15th.

Figure 49 : GRU (200 x 200) - 15th August

Figure 50 : LSTM (200 x 200) - 15th August

The GRU model had more difficulty than the LSTM in predicting load on 15 August but in general, the "problem" of August 15th remains for these models as well. For May 1st the problem is worse. See the following series of images

Figure 51 : FFNN (100 x 50) – 1st May

Figure 52 : LSTM (200 x 200) - 1st May

Figure 53 : LSTM (40 x 40) - 1st May

Figure 54 : GRU (200 x 200) - 1st May

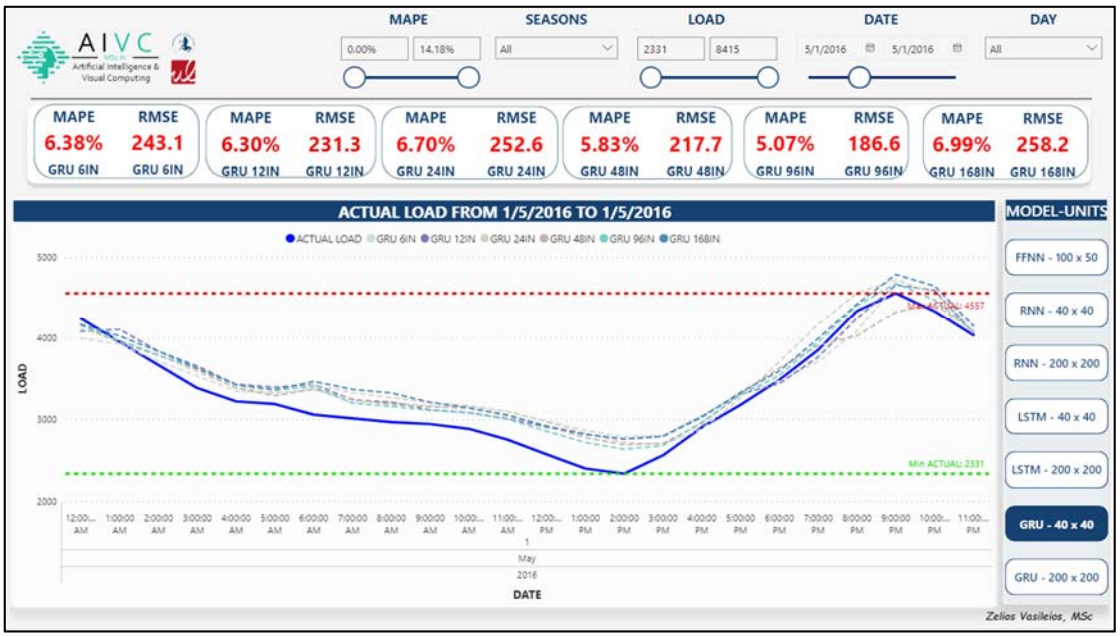

Figure 55 : GRU (40 x 40) - 1st May

Figure 56 : RNN (200 x 200) - 1st May

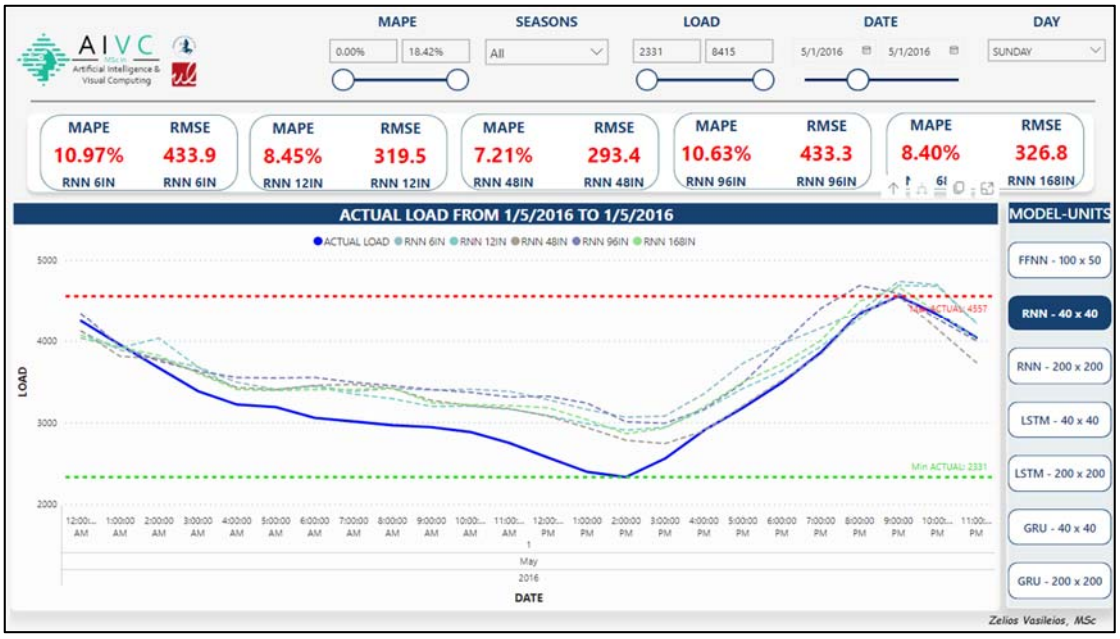

Figure 57 : RNN (40 x 40) - 1st May

Figure 58 : Best LSTM & GRU (500 x 500) - 1st May

None of the 7 models and consequently of the 39 model variants correctly predicted the load for May 1st. The reader will reasonably wonder what can happen on Sundays or even what happens on May 1 and August 15. As we have seen above, Sundays are generally days of rest and low energy consumption. But the models are not aware of this. In fact, the model does not know what 'Sunday' or 'Thursday' really means. The models learn from the time series pattern. As far as May 1 or August 15th are national holidays for the whole country, therefore the energy consumption is very low. This is not known to the model. The model predicts load prices according to past values without considering calendar holidays, national holidays, etc. There are several strategies that can be used to address this issue of RNNs:

1. Incorporate information about national holidays into the preprocessing of the data. This could involve adding a binary indicator variable for each holiday or encoding the holiday information in a more sophisticated way.
2. Use a more complex model architecture that can better handle the added complexity of modeling national holidays [14]. For example, a hybrid model that combines RNNs with other types of neural networks, such as convolutional neural networks (CNNs) or attention mechanisms, might be more effective.

3. Use a larger dataset that includes more examples of load consumption patterns on national holidays.
4. Use transfer learning techniques where pre-trained models on similar data can be fine-tuned on the current task.
5. Use an ensemble of models with different architectures to capture different patterns and generalize better.
6. Add external data sources such as weather information, socio-economic indicators and population density to the model to get a more comprehensive understanding of the data.

It's important to note that, depending on the specific data and the complexity of the problem, a combination of these strategies may be necessary to achieve accurate load consumption predictions on national holidays.

Section 4: Conclusion

4.1 Observations and future study

In this thesis, a comprehensive and in-depth study on the application of Recurrent Neural Networks was undertaken for the purpose of forecasting short-term electric load in Greece. The study was carried out using an extensive dataset and various techniques, with a focus on evaluating the performance of recurrent neural networks. Through the utilization of grid search algorithms, the model that resulted in the minimal statistical error was determined as the optimal model. That model is a 2-layer GRU with 500 neurons each taking as input 32h values. The model yields RMSE of 83MWh (1.17%).

Upon conducting experimental evaluations, it was determined that the runtime of Recurrent Neural Networks (RNNs) is inferior to that of Long Short-Term Memory networks (LSTMs), and LSTMs exhibit longer execution times when compared to Gated Recurrent Units (GRUs).

Similarly, the performance of Recurrent Neural Networks (RNNs) is inferior to that of Long Short-Term Memory Networks (LSTMs), while LSTMs show minimal deviation compared to Gated Recurrent Units (GRUs). RNNs models outperformed LSTMs only in the 24-hour forecast horizon. This does not impress us because RNNs are effective in predicting short-term load. The 24-hour horizon belongs to the category of short-term forecasting.

Table 12 : Performance comparison of RNN-LSTM-GRU

units	200 x 200																
outputs	1h						2h						24h				
inputs	6IN	12IN	24IN	48IN	96IN	168IN	6IN	12IN	24IN	48IN	96IN	168IN	24IN	48IN	96IN	168IN	384IN
RNN	2,98	2,38	2,19	1,72	1,82	2,28	-	-	-	-	-	-	5,86	5,77	6,11	5,12	7,99
LSTM	2,51	2,13	1,79	1,52	1,73	1,61	3,2	2,18	2,01	1,94	1,83	1,87	6,4	6,42	6,26	5,49	6,18
GRU	2,4	1,71	1,53	1,51	1,52	1,46	2,92	2,23	1,84	1,89	1,77	1,9	5,72	5,6	5,69	4,66	5,68

units	40 x 40																
outputs	1h						2h						24h				
inputs	6IN	12IN	24IN	48IN	96IN	168IN	6IN	12IN	24IN	48IN	96IN	168IN	24IN	48IN	96IN	168IN	384IN
RNN	2,69	2,31	1,99	1,73	1,74	1,92	-	-	-	-	-	-	6,04	6,09	5,78	5,44	5,79
LSTM	2,49	2,1	1,94	1,81	1,78	1.93	3,67	2,78	2,09	2,14	2,13	1,97	6,74	6,78	6,61	6,16	6.98
GRU	2,51	2,13	1,88	1,74	1,76	1,59	3,34	2,41	2,05	2,04	1.95	2,09	6,13	6,06	5,84	5,32	5,7

From the Power BI results we saw that it is possible that all the models might predict some seasons more accurately than others. One possible reason for this is that the changes in weather patterns during the summer and winter

seasons are more drastic and consistent, making them easier for the model to learn and predict. On the other hand, the autumn and spring seasons are known to be more transitional, with more variability in weather patterns and temperature fluctuations, which can be more difficult for the model to learn and predict.

Table 13 : Seasonal performance

season	Winter						Spring						Summer						Autumn					
units	200 x 200																							
outputs	1h						1h						1h						1h					
inputs	6IN	12IN	24IN	48IN	96IN	168IN	6IN	12IN	24IN	48IN	96IN	168IN	6IN	12IN	24IN	48IN	96IN	168IN	6IN	12IN	24IN	48IN	96IN	168IN
RNN	-	2,47	2,16	1,69	2,02	2,19	-	2,75	2,53	1,89	2,16	2,28	-	2,08	1,73	1,26	1,4	1,53	-	2,39	2,13	1,73	1,94	2,05
LSTM	3,09	1,93	1,4	1,98	1,41	1,43	3,18	2,05	1,67	1,81	1,69	1,53	2,15	1,58	1,22	1,65	1,12	1,32	2,81	2,12	1,53	1,77	1,5	1,49
GRU	2,16	1,58	2,33	1,36	2,69	2,11	2,35	1,88	2,42	1,54	2,61	2	1,89	1,33	1,73	1,11	1,8	1,56	2,26	1,81	2,16	1,44	2,27	1,86

We also noticed something called "Sunday effect". It refers to the phenomenon where energy consumption patterns on Sundays tend to differ from those on other days of the week. This can be caused by a variety of factors, such as changes in human activity levels, building occupancy, and industrial production. As a result, it can be more difficult for Recurrent Neural Network models to predict energy consumption patterns on Sundays due to the unique characteristics of this day. This phenomenon is observed especially on the days characterized as national holidays (May 1st, August 15th, etc.). In order to possibly be able to make better predictions on these days, we will have to train the models with many more such days as "Sunday effect".

In order to optimize prediction algorithms there are several additional types of data that we could use, for further study, to improve the accuracy of short-term electric load prediction with RNNs. Some examples include:

1. Weather data: Information such as temperature, humidity, wind speed, and precipitation can have a significant impact on electric load.
2. Demographic data: Population density, age distribution, and socio-economic factors can also affect electric load.
3. Economic data: Data such as GDP, unemployment rate, and energy prices can provide important context for understanding electric load patterns.
4. Holiday and events data: Holidays and events can affect electric load.

5. Building's occupancy and temperature data: knowing the occupancy and temperature of the building can give us a better understanding of the energy consumption.
6. Smart meter data: Smart meter data can provide fine-grained consumption data, allowing us to make more accurate predictions.

In conclusion, combining different types of neural networks, such as feedforward neural networks and recurrent neural networks, a model can take advantage of the benefits of each type of network and potentially improve the accuracy of load forecasting. This can also be combined with other techniques, such as fuzzy logic or support vector machine, to make it more efficient. This combination is known as hybrid models [18].

Bibliography

1. Bakirtzis A.G, Kiartzis S.J, Petridis V., Short term load forecasting using neural networks, Aristotle University of Thessaloniki, 1994. doi:10.1016/0378-7796(95)00920-D.
2. Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. “Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches,” Energies, vol. 11, no. 7, 2018. doi :10.3390/en11071636.
3. Brownlee, J. Deep Learning for Time Series Forecasting. 2018.
4. Brownlee, J. Machine Learning Mastery With Python. 2021.
5. Chollet F. Deep Learning with Python. 2017.
6. C.N. Lu, H.T. Wu and S. Vemuri, "Neural network based short term load forecasting," in IEEE Transactions on Power Systems, vol. 8, no. 1, pp. 336-342, Feb. 1993, doi: 10.1109/59.221223.
7. D. C. Park, M. A. El-Sharkawi, R. J. Marks, L. E. Atlas and M. J. Damborg, "Electric load forecasting using an artificial neural network," in IEEE Transactions on Power Systems, vol. 6, no. 2, pp. 442-449, May 1991. .doi: 10.1109/59.76685.
8. D. Srinivasan, A. C. Liew, and C. S. Chang, “A neural network short-term load forecaster,” Electr. Power Syst. Res., vol. 28, no. 3, pp. 227–234, 1994. doi:10.1016/0378-7796(94)90037-X.
9. D. D. Highley and T. J. Hilmes, "Load forecasting by ANN," in IEEE Computer Applications in Power, vol. 6, no. 3, pp. 10-15, July 1993, doi: 10.1109/67.222735.
10. Georgouli, A. (2015). Τεχνητή νοημοσύνη. Kallipos, Open Academic Editions. https://hdl.handle.net/11419/3381.
11. Géron, A. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2019.
12. G. Gross and F. D. Galiana, "Short-term load forecasting," in Proceedings of the IEEE, vol. 75, no. 12, pp. 1558-1573, Dec. 1987. doi: 10.1109/PROC.1987.13927.
13. Grus, J. Data Science from Scratch: First Principles with Python. 2019.

14. W. He, “Load Forecasting via Deep Neural Networks,” in Procedia Computer Science, 2017, vol. 122, pp. 308–314. doi: 10.1016/j.procs.2017.11.374.
15. Hagan, M.T. and Behr, S.M. The Time Series Approach to Short Term Load Forecasting. IEEE Transactions on Power Systems, 2, 785-791. 1987 doi: 10.1109/TPWRS.1987.4335210.
16. Hong, T. and Shahidehpour, M. Load Forecasting Case Study. EISPC, U.S. Department of Energy. 2015.
17. Hippert, H.S., Pedreira, C.E. and Souza, R.C. Neural Networks for Short-Term Load Forecasting: A Review and Evaluation. IEEE Transactions on Power Systems, 2001 16, 44-55. doi: 10.1109/59.91078
18. Kandilogiannakis G., Mastorocostas P., Voulodimos A., ReNFuzz-LF: A Recurrent Neurofuzzy System for Short-Term Load Forecasting. Energies 2022, 15(10), 3637. doi: 10.3390/en15103637.
19. Ke Li, Wei Huang, Gaoyuan Hu, Jiao Li. Ultra-short term power load forecasting based on CEEMDAN-SE and LSTM neural network, Energy and Buildings 2023, doi : 10.1016/j.enbuild.2022.112666.
20. K. Y. Lee, Y. T. Cha and J. H. Park, "Short-term load forecasting using an artificial neural network," in IEEE Transactions on Power Systems, vol. 7, no. 1, pp. 124-132, Feb. 1992, doi: 10.1109/59.141695.
21. Livieris Ioannis, Evaluation of artificial neural network training methods and applications, University of Patras, 2009.
22. Park, D.C.; El-Sharkawi, M.; Marks, R.; Atlas, L.; Damborg, M. Electric load forecasting using an artificial neural network. IEEE Trans. Power Syst. 1991, 6, 442–449 doi: 10.1109/59.76685.
23. Powell B. Mastering Microsoft Power BI: Expert techniques for effective data analytics and business intelligence. 2018.
24. P. Xiuyan, Z. Biao and C. Yanqing, "The short-term load forecasting of electric power system based on combination forecast model," The 27th Chinese Control and Decision Conference (2015 CCDC), Qingdao, China, 2015, pp. 6509-6512, doi: 10.1109/CCDC.2015.7161993.
25. S. Rahman and R. Bhatnagar, "An expert system based algorithm for short term load forecast," in IEEE Transactions on Power Systems, vol. 3, no. 2, pp. 392-399, May 1988, doi: 10.1109/59.192889.

26. S. Singh, S. Hussain and M. A. Bazaz, "Short term load forecasting using artificial neural network," 2017 Fourth International Conference on Image Information Processing (ICIIP), Shimla, India, 2017, pp. 1-5, doi: 10.1109/ICIIP.2017.8313703.
27. Roger, F. Silva. Power BI - Business Intelligence Clinic: Create and Learn. 2018.
28. D. J. Sobajic and Y. . -H. Pao, "Artificial neural-net based dynamic security assessment for electric power systems," in IEEE Transactions on Power Systems, vol. 4, no. 1, pp. 220-228, Feb. 1989, doi: 10.1109/59.32481.
29. Shukla, N. Machine Learning with TensorFlow. 2018.
30. Vasilev. I, Slater. D, Spacagna. G, Roelants. P, Zocca. V. Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition. 2019.
31. Verykios, V., Kagklis, V., & Stavropoulos, E. (2015). Η επιστήμη των δεδομένων μέσα από τη γλώσσα R. Kallipos, Open Academic Editions. https://hdl.handle.net/11419/2965.

.

Webliography

1. https://www.deeplearning.ai/
2. https://towardsdatascience.com/time-series-analysis-visualization-forecasting-with-lstm-77a905180eba
3. https://r2rt.com/recurrent-neural-networks-in-tensorflow-ii.html
4. https://colah.github.io/posts/2015-08-Understanding-LSTMs/
5. https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/
6. https://powerbi.microsoft.com/en-us/
7. https://energy.stonybrook.edu/facts/demand.php
8. https://en.wikipedia.org/wiki/Recurrent_neural_network
9. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
10. https://ethw.org/Milestones:Pearl_Street_Station,_1882
11. https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
12. https://machinelearningmastery.com/how-to-grid-search-deep-learning-models-for-time-series-forecasting/
13. https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/
14. https://keras.io/examples/timeseries/
15. https://en.wikipedia.org/wiki/Long_short-term_memory
16. https://intellipaat.com/blog/what-is-lstm/
17. https://wiki.pathmind.com/lstm
18. https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide
19. https://towardsdatascience.com/exploring-the-lstm-neural-network-model-for-time-series-8b7685aa8cf
20. https://neptune.ai/blog/arima-vs-prophet-vs-lstm
21. https://gallery.azure.ai/Tutorial/Forecasting-Short-Time-Series-with-LSTM-Neural-Networks-2
22. https://medium.com/@pratik.asija1234/time-series-forecasting-using-keras-9ffae6c53bfc

23. https://codeit.us/blog/machine-learning-time-series-forecasting
24. https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/
25. https://www.analyticsvidhya.com/blog/2022/01/tutorial-on-rnn-lstm-gru-with-implementation/
26. https://www.alpha-quantum.com/blog/long-short-term-memory-lstm-with-python/long-short-term-memory-lstm-with-python/
27. https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM
28. https://www.datacamp.com/tutorial/lstm-python-stock-market
29. https://towardsdatascience.com/a-quick-deep-learning-recipe-time-series-forecasting-with-keras-in-python-f759923ba64
30. https://colab.research.google.com/
31. https://en.wikipedia.org/wiki/Feedforward_neural_network
32. https://en.wikipedia.org/wiki/Smart_grid
33. https://neptune.ai/blog/select-model-for-time-series-prediction-task
34. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac
35. https://www.geeksforgeeks.org/difference-between-model-parameters-vs-hyperparameters/
36. https://www.turing.com/kb/mathematical-formulation-of-feed-forward-neural-network
37. https://www.baeldung.com/cs/hidden-layers-neural-network
38. https://machinelearningknowledge.ai/brief-history-of-deep-learning/
39. https://electricalacademia.com/electric-power/electrical-power-system-components/