NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Wind Energy Prediction Using Deep Learning Architectures

Georgios K. Floros

Supervisor: **Ioannis Emiris,** Professor

ATHENS

APRIL 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη Αιολικής Ενέργειας με τη Χρήση Αρχιτεκτονικών Βαθιάς Μάθησης

Γεώργιος Κ. Φλώρος

Επιβλέπων: **Ιωάννης Εμίρης,** Καθηγητής

ΑΘΗΝΑ

ΑΠΡΙΛΙΟΣ 2023

BSc THESIS

Wind Energy Prediction Using Deep Learning Architectures

Georgios K. Floros

S.N.: 1115201700178

SUPERVISOR: **Ioannis Emiris,** Professor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη Αιολικής Ενέργειας με τη Χρήση Αρχιτεκτονικών Βαθιάς Μάθησης

Γεώργιος Κ. Φλώρος
Α.Μ.: 1115201700178

ΕΠΙΒΛΕΠΩΝ: **Ιωάννης Εμίρης,** Καθηγητής

ABSTRACT

This thesis investigates the utilization of various deep learning architectures for energy prediction, utilizing Weather Research Forecasting (WRF) data and actual measurements from two wind farms. The preprocessing pipeline is thoroughly outlined, followed by experimentation with different feature extraction techniques such as CNNs, Mean Vector, and Central Vector, along with LSTM, Attention, and Transformer Blocks as temporal models. Attention-based models are emphasized as a notable contribution. The results show that efficient preprocessing is crucial for optimal performance and that attention-based models perform comparably or even better than LSTMs as temporal models in certain cases. Furthermore the study raises questions about the essentiality of CNNs feature extractor in some cases. It also suggests that transfer learning between nearby parks is a promising approach to address limited data, and can also be used for new parks where data are not available. Finally, the study also highlights certain limitations we faced and proposes avenues for future work.

SUBJECT AREA: Deep Learning

KEYWORDS: Machine Learning, Wind Energy Prediction, Neural Network, Attention, Transformer Encoder, WRF

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία διερευνά τη χρήση διαφόρων αρχιτεκτονικών βαθιάς μάθησης για πρόβλεψη ενέργειας, χρησιμοποιώντας δεδομένα πρόβλεψης καιρού (WRF) και πραγματικές μετρήσεις από δύο αιολικά πάρκα. Η μέθοδος προ-επεξεργασίας περιγράφεται λεπτομερώς, ενώ ακολουθούν πειράματα με διαφορετικές τεχνικές εξαγωγής χαρακτηριστικών όπως συνελικτικά νευρωνικά δίκτυα (CNN), μέσο διάνυσμα (mean vector) και κεντρικό διάνυσμα (central vector), μαζί με χρονικά μοντέλα όπως τα νευρωνικά δίκτυα μακράς βραχείας μνήμης (LSTM), το μηχανισμό προσοχής (Attention mechanism) και τα μπλοκ μετασχηματιστή (Transformer blocks). Τα αποτελέσματα δείχνουν ότι η αποτελεσματική προ-επεξεργασία είναι ζωτικής σημασίας για τη βέλτιστη απόδοση και ότι οι μέθοδοι που βασίζονται στον μηχανισμό προσοχής αποδίδουν συγκρίσιμα ή ακόμα καλύτερα από τα LSTM ως χρονικά μοντέλα σε ορισμένες περιπτώσεις. Επιπλέον, η μελέτη εγείρει ερωτήματα σχετικά με την ουσιαστικότητα του CNN ως εξαγωγέα χαρακτηριστικών σε μερικές περιπτώσεις. Υποδηλώνει επίσης ότι η μεταφορά μάθησης (transfer learning) μεταξύ κοντινών πάρκων είναι πολλά υποσχόμενη προσέγγιση για την αντιμετώπιση περιορισμένων δεδομένων και μπορεί επίσης να χρησιμοποιηθεί για νέα πάρκα όπου υπάρχουν ανεπαρκή δεδομένα. Τέλος, η μελέτη επισημαίνει ορισμένα προβλήματα που αντιμετωπίσαμε και προτείνει μελλοντικές κατευθύνσεις.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Βαθιά Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μηχανική Μάθηση, Πρόβλεψη Αιολικής Ενέργειας, Νευρωνικά Δίκτυα, Μηχανισμός Προσοχής, Κωδικοποιητής Μετασχηματιστή

Dedicated to my family and the people that supported me the most.

ACKNOWLEDGEMENTS

I would like to thank Professor Ioannis Emiris for supervising my thesis and his invaluable guidance and support throughout this research. I also want to thank the PhD candidate Konstantinos Tertikas for his valuable and constant assistance in this project. His contribution was essential for the completion of this study. Finally, I would also like to thank Dimitris Katsiros for providing me a first version of the code of the fundamental paper and for discussing with me certain aspects of the project.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This thesis was developed as a part of my undergraduate studies in the Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens, Greece, under the guidance and supervision of Professor Emiris.

1. INTRODUCTION

Wind energy is one of the fastest growing energy sources in the world. The demand is increasing year by year. Accurate wind energy prediction is crucial for the effective operation of wind farms. It enables optimal energy generation, reduces costs because it helps balance supply and demand, and improves the reliability of energy supply. Wind energy forecasting has been a critical concern since the early days of wind power generation, where it relied primarily on simple wind speed measurements. However, later on, more sophisticated models have been developed that take into account multiple environmental parameters, such as temperature, pressure, and wind speed, to produce accurate and reliable forecasts. In particular, persistence , numerical weather prediction and statistical models are some of the commonly used for that task. Recently, thanks to the advancements of technology, neural networks, including deep learning techniques, are used to perform such tasks due to their ability to handle large amounts of data and model complex relationships between input and output variables

In this study, we worked on wind energy production forecasting using deep learning architectures as a predictive model. Our task is hour-ahead wind energy forecasting provided time-series of 6 hourly steps. Our starting point is based on the method of a previous study [12] that helped us build on and test our methods. Our input data consisted of Weather Research Forecasting (WRF) meteorological predictions and our target data consisted of wind farm real energy values. We trained separate models for each park's data, with the aim of capturing the unique patterns and characteristics of wind energy production at each location. However, we also evaluated the performance of a single predictive model for both parks so we developed a generalized model.

Our study is divided into four sections. In section one, through a literature review, we provide an overview of wind energy forecasting and the neural network models utilized, as well as the source of our input data. In section two, we present extensively our approach. We describe the method we used for preprocessing our data, the problems we faced and the solutions we came up with. We address the models we used and the networks' architecture. This study uses a plethora of components, each designed to address specific aspects of the wind energy production forecasting problem. In particular, we categorize them into two distinct categories based on their functionality. Feature extractor (CNN, MLP) models, used to extract spatial information on input data features and temporal models (LSTM,Attention-based) used to capture temporal dependencies in the data. Although attention-based models are a relatively novel type of neural network architecture and have shown promise in other domains, their application to wind energy production forecasting is still not widely explored therefore our study contributes in this direction. In section three, we present the experimental results and provide a detailed analysis and interpretation of the findings. In section four, we draw conclusions based on our results.

2. BACKGROUND & RELATED WORK

This chapter provides a comprehensive overview of the previous studies and research conducted on the methods we will be using in this study and the topics related to it.

2.1 Wind Energy

The use of wind power, a renewable energy source generated by the wind, is anticipated to rise sharply in the future decades. Wind energy is a local resource that can help reduce dependence on fossil fuels [13].

Greece is a country with a very large number of islands so powerful winds occur in island and coastal regions. This is especially beneficial for the growth of wind energy in the nation. At the end of June 2022, Greece had 4,534 Megawatts of total wind power, according to statistics released by the Hellenic Wind Energy Scientific Association (HWEA) [1]. Figure 2.1 shows the statistical analysis broken down by year in more depth.

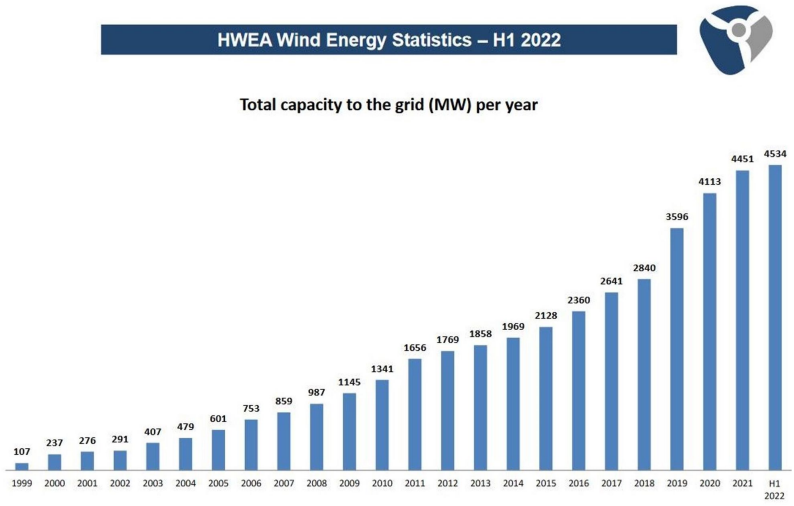

Figure 2.1: Capacity per year in Greece [1]

Also, Figure 2.2 shows the geographical distribution of wind power in Greece by Region. We can see from the map that Central Greece remains at the top of the wind installations since it hosts 1861 MW (41%) followed by the Peloponnese with 639 MW (14%) and Eastern Macedonia - Thrace with 534 MW (12%).

2.1.1 Wind Turbines

A wind turbine is a device that converts wind's kinetic energy into electricity. A group of wind turbines is called a wind farm. Turbines in a wind farm supply the electrical grid with energy. Onshore or offshore areas can host wind farms. The wind blows the turbine blades, which are attached to a rotor. The rotor then spins a generator to create electricity. There are two types of wind turbines: horizontal axis wind turbines (HAWTs) and vertical axis wind turbines (VAWTs) (Figure 2.3).

HAWTs are the most common type of wind turbines. They usually have two or three long, thin blades that look like airplane propellers. The blades are positioned so that they face

Figure 2.2: Distribution of wind power map [1]

directly into the wind. They have a high tower base that allows access to stronger wind resulting in increased power output and high efficiency, since the blades always move perpendicular to the wind, receiving power throughout the rotation. However, the cost of their construction, transportation and maintenance is quite high and requires a complex system of components with an additional mechanism to control the deflection of the wings in the direction of the wind. Also, their volume sometimes disturbs the aesthetics of the natural landscape [14].

VAWTs have smaller, wider curved blades that resemble the beaters used in an electric mixer [15]. Unlike HAWTs they can generate electricity in any wind direction. Also their construction, transportation and maintenance costs are much lower because the construction of the support tower is not demanding. This is because the generator, gearbox and other components are placed on the ground and no deflection mechanisms are needed (Figure 2.3) VAWTs are particularly suitable for areas with extreme weather conditions and as their wings move at relatively low speeds and so they do not pose a great accident risk to birds and humans. But compared to HAWTs they are less efficient due to the additional resistance created with the rotation of their blades [2].

Extensive research exists related to the improvement of the generated power of wind turbines, their mode of operation with emphasis on the wind power generator and the control of maximum power point tracking. A comprehensive review and comparison of the four most popular Maximum power point tracking (MPPT) control methods as well as improvements for each method was presented [16]. In addition, technologically advanced wind energy conversion generators, such as brushless dual-feed induction generator, modern permanent magnet stator generators, magnet generators, etc were presented. Also, three techniques for evaluating the performance of wind turbines (Experimental - Analytical - Computational) were analysed [17]. These methodologies aim for the wind turbine to be designed for optimal energy production with less cost [13].

A wind turbine's produced energy is strongly associated with wind speed.The produced energy increases as the wind speed increases. The design and operation of a turbine requires a certain wind speed range. The limits of this range are the cut-in speed and

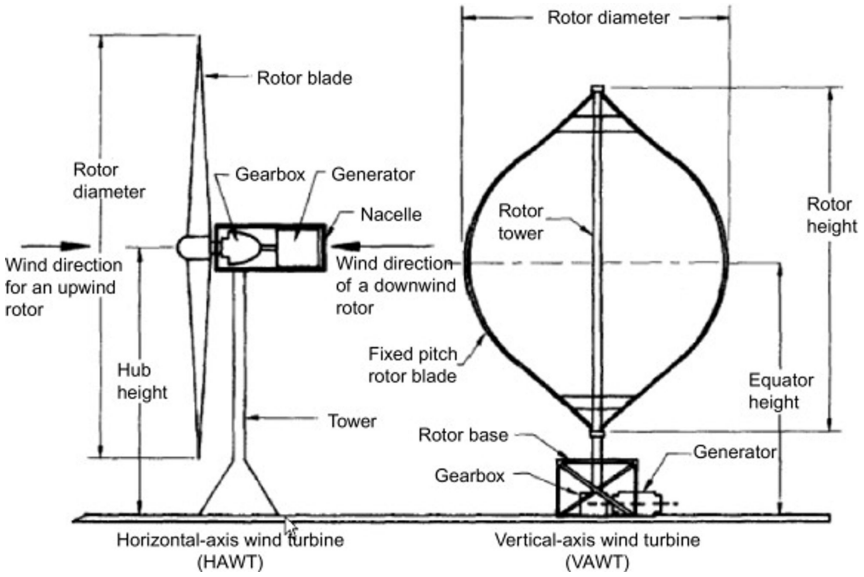

Figure 2.3: Two types of wind turbines unfolded [2]

the cut-out speed. In particular, as Figure 2.4 shows when wind speed is too low the wings of the turbine do not move and so power is not produced. Cut-in speed is when the

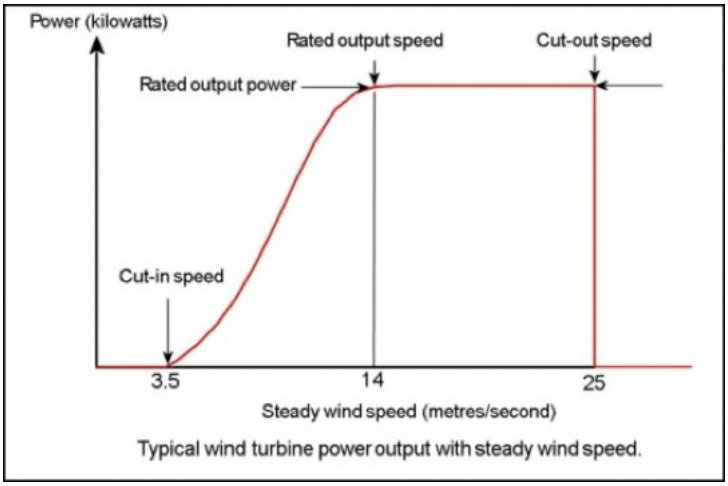

Figure 2.4: Turbine's produced energy [3]

turbine starts spinning for the first time and generates power. When the speed of the wind increases, the power gradually exceeds the cut-in speed and reaches the maximum limit the turbine design can support. To avoid the risk of damage on the rotor the cut-out speed is activated which stops the operation with the help of suitable auditors [3].

2.2 Supervisory Control & Data Acquisition

Supervisory control and data acquisition (SCADA) is an industrial process control system architecture. The main objectives of SCADA are the collection of real-time data from remote locations for the control of equipment and conditions, and the improvement of industrial automation. The architecture consists of :

- Sensors and actuators which detect device features and also work as switches.

- Field controllers which are directly connected to sensors or actuators. The controlers can be remote terminal units (RTUs) or logic controllers (PLCs),
- SCADA supervisory computers that collect data and issue commands on field controllers,
- HMI software suitable for presenting, processing and modification of data.
- Communication infrastructure for the communication of all system's elements

SCADA systems have been adopted by many businesses and organizations to make data-driven decisions about their industrial processes. Specifically they monitor and control infrastructure processes (e.g. water distribution, transport and distribution of solar energy - wind farms etc.) [18].

2.3 WRF

The Weather Research and Forecasting (WRF) model is a highly flexible and widely used numerical weather reflection system created through a collaborative effort between the National Center for Atmospheric Research (NCAR), the National Oceanic and Atmospheric Administration (NOAA), and several universities and government bodies. Its development began in the late 1990s as a successor to the MM5 (fifth-generation Mesoscale Model) [19]. The WRF model was officially launched in 2002 and has been continuously updated and improved since then. It has flexible, open source code adaptable to all computing systems [20]. It is used by researchers, meteorologists and other scientists worldwide to study weather and climate, as well as to create short- and long-term climate projections. Also, the WRF model can be used to predict the production of renewable energy systems, such as wind and solar energy, with the main purpose of optimizing these systems and integrating them into electricity distribution networks. Research reports that WRF provides confidence in the accuracy with which it can be used to predict the performance of operational wind farms [21].

2.4 Neural Network

Artificial intelligence has taken center stage in computer science thanks to advancements in technology and the creation of fast, powerful computers. Neural networks are a key tool in this field. A neural network leverages machine learning (ML) algorithms to simulate the functions of the human brain. Neural networks process large amounts of data and solve complex problems. Their structure is based on neuron connections of various levels and therefore mimics signaling processes of biological neurons [22].

2.4.1 Architecture

As shown in the Figure 2.5, a neural network consists of layers of nodes distributed in an input layer, one or multiple hidden layers, and an output layer. Neurons that connect to one another are called nodes. The output layer provides the information after the hidden layers have processed the data that was input by the neurons.

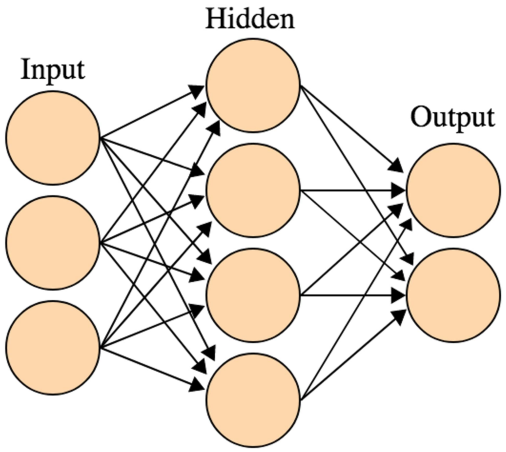

Figure 2.5: Neural network [4]

A typical neural network includes a fully connected layer of neurons, which means that every neuron's output serves as an input for every neuron in the next layer. The number of layers in a neural network varies and describes the depth of the network [23]. The architecture of neural networks is differentiated by various factors such as their depth, the number of their hidden layers and the input-output capabilities of each node [22]. The depth, number of hidden layers, and input-output capabilities of each node are just a few of the characteristics that distinguish neural network architecture [22].

2.4.2 Convolutional Neural Networks

Convolutional neural networks are mainly concerned with problems of advanced applications such as face recognition, natural language processing (NLP), optical character recognition (OCR) and image classification. A typical convolutional neural network (CNN) is made up of a variety of layers, each of which is in charge of a specific task within the network. Convolutional, pooling, and fully connected layers are the most frequent types of layers used to build CNNs [23].

A simple architecture of a CNN image recognition network is described in the Figure 2.6.

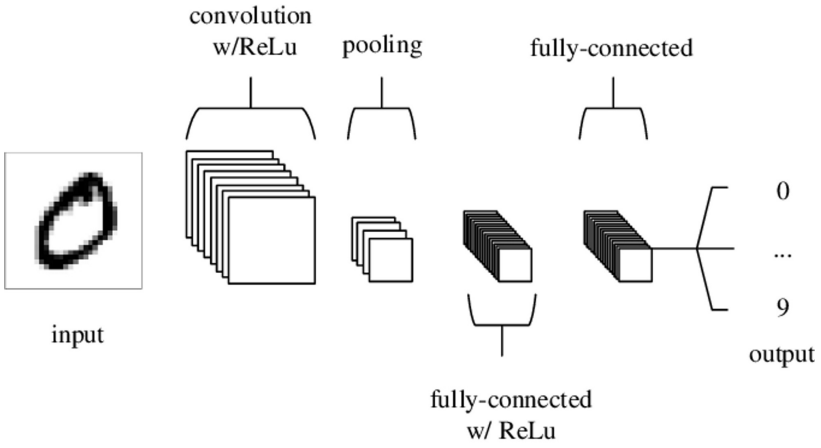

Figure 2.6: Convolutional neural network for classification [5]

2.4.2.1 Convolutional layer

The convolutional layer is the most important layer of a CNN, because it is responsible for feature detection. It accepts, as inputs, images or feature maps and applies different fixed size filters along these inputs. It does so by calculating the inner product of the weights of the filters with the corresponding area of the input. These filters can identify visual features in an image, such as oriented edges and corners, and when combined with an image, produce a feature map. By successively applying new filters to the existing feature maps, the network is able to detect higher order features and ”know” the input better [23].

The convolution operation applies successive transformations of the input with the help of a very small, local function called a ”kernel”. The “kernel” slides into each position and computes a new feature as a weighted sum of the input it is applied to (Figure 2.7).

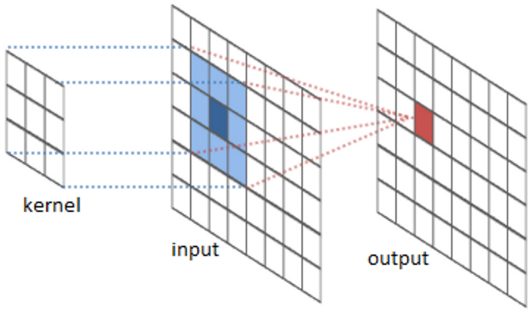

Figure 2.7: Convolutional layer [6]

2.4.2.2 Pooling Layer

The pooling layer, which usually follows a set of successive convolutional layers, is in charge of downsampling the feature maps. Pooling layers produce as an output the result of a function on a neighboring region of the feature map. The max function and the average function are frequently used functions on these nearby adjacent map regions. Each function defines a different type of pooling layer, a max-pooling layer (Figure 2.8) and an average pooling layer respectively [23].

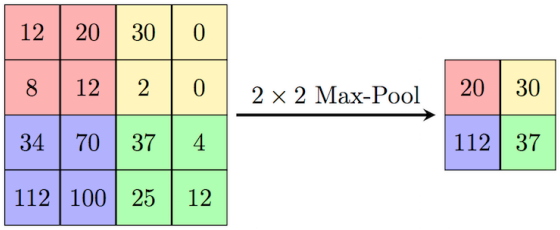

Figure 2.8: Max pooling Layer

2.4.3 Long Short Term Memory Networks

A special type of recurrent neural network (RNN), long short-term memory networks (LSTM) are capable of learning ”long-term dependencies,” a problem that simple RNNs (Figure 2.9) could not handle. They fall under the category of recurrent neural networks, which

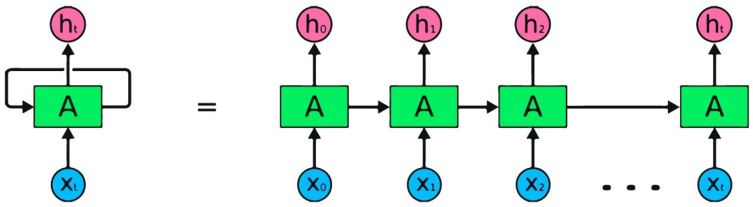

Figure 2.9: Simple RNN architecture [7]

includes networks that work with sequences of data and whose length of inputs and outputs can be variable. They were initially introduced in 1997 [24], improved, and are now often utilized in a range of difficult issues. Long-term memory systems are intended to retain knowledge over time, and due to the way they are built, they can be trained to do so considerably easier [8].
In the Figure 2.10 we see that the recurrent unit A of an LSTM network has a different structure than a typical RNN.

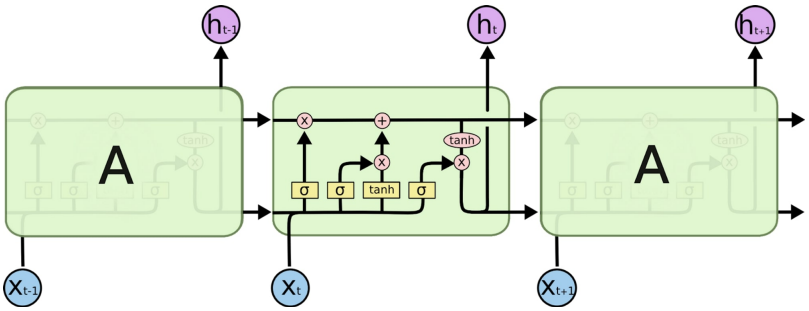

Figure 2.10: LSTM [8]

More specifically, in the Figure 2.11 the horizontal line that runs through the top of the

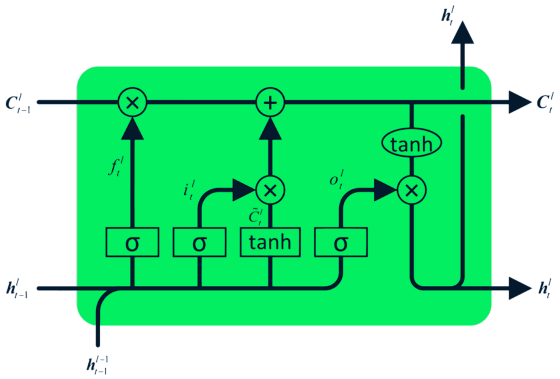

Figure 2.11: LSTM unfolded [7]

diagram and the entire chain is called the cell state, and easily conveys information with small linear interactions. The σ and $tanh$ layers with the help of a series of gates have the ability to remove or add information to the state of the cell. For example, at the decision

level whether to 'keep' or 'throw' information, the cell state will be taken by the forget gate(f_t). Accordingly, the decision regarding which new information should be stored in the cell state will be taken by the input gate(i_t) in cooperation with the $tanh$ layer. Next is the update of the state of the cell(c_t). The decision about which data will be sent to the output is made by the output gate(o_t) with the help of the $tanh$ layer [8].

The LSTM architecture described above has a number of variations. An important extension is the bi-directional LSTM (Figure 2.12), which trains two LSTMs simultaneously,

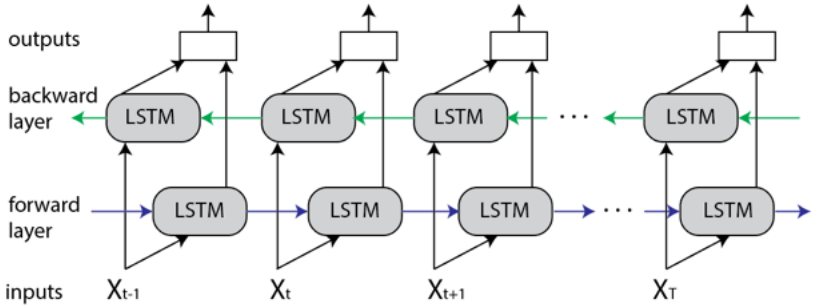

Figure 2.12: Bi-directional LSTM architecture [9]

one in positive and one in negative time direction, merging the results for each time step. Another important architecture based on LSTMs is the encoder - decoder architecture (Figure 2.13) [25]. This architecture uses a multi-layered short-term memory LSTM network

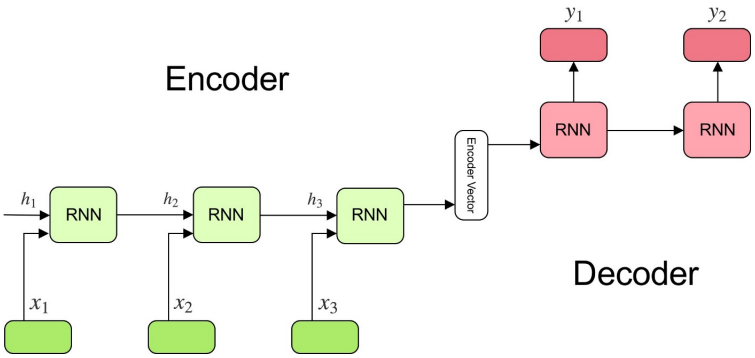

Figure 2.13: Encoder - Decoder architecture [10]

to map the input sequence to a fixed-dimensional vector and then another deep LSTM to decode the target sequence from the vector. This architecture therefore creates a type of Seq2Seq model, a model that takes a sequence of items (words, letters, time series, etc) and outputs another sequence of items. The encoder and decoder are trained together to maximize the conditional probability of a target sequence given a source sequence. This architecture is often used in problems that have to do with Machine translation, speech recognition, video captioning, text summarization and a lot more.

2.4.4 Transformers - Self-Attention

2.4.4.1 Attention

Even though the attention mechanism is now used in various problems like image captioning and others it was initially designed for Neural Machine Translation tasks using

Seq2Seq Models to permit the decoder to utilize the most relevant parts of the input sequence in a flexible manner in recurrent neural network.

The attention mechanism is introduced [26] and defined as the calculation of a context vector. In order to do so, a weighted average is computed from the encoder's hidden states at each step of the decoder. Each state's contribution to this average is determined by an alignment score. Each state of the encoder is specified as a key and value vector, with the previous (last) hidden state of the decoder typically being represented as a query vector.

The result is a weighted average of the value vectors, where the weights are set by the function that compares the query and key vector compatibility [27]. Keys and values can be represented by different sets of vectors. The equations described are shown below: For an input sequence $X = (x_1, x_2, ..., x_n)$ of length n The context vector c is computed as:

$$c = \sum_{j=1}^{n} a_j v_j \tag{2.1}$$

The weight a_j is computed by:

$$a_j = \frac{\exp(e_{q_j,k_j})}{\sum_{i=1}^{n} \exp(e_{q_i,k_i})} \tag{2.2}$$

$e(q, k)$ is the alignment score function , researchers have proposed dot product attention [28] while a more recent work [11] proposed it scaled by $1/\sqrt{d_k}$, where d_k is the dimension of key vector in order to counteract really small gradients.

$$e(q, k) = \frac{(q)^\intercal k}{\sqrt{d_k}} \tag{2.3}$$

2.4.4.2 Self-Attention

The method by which we apply the attention mechanism to each position of the same sequence is known as the self-attention mechanism [11]. For each step of the sequence, we generate three vectors (query, key, and value) and use the x_i query, key and value vectors to implement the attention mechanism.

This results in the transformation of an input sequence of n time steps X into a sequence of n time steps Y, where each step y_i includes the information of x_i along with how x_i is related to the other positions of X. By accumulating the values of the query, key, and value vectors in the tables Q, K, it is possible to calculate the aforementioned relationships for the full sequence in parallel.

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{2.4}$$

2.4.4.3 Multi-head Attention

The idea behind Multi-head Attention is allowing the model to receive information from different representation spaces at different positions [11]. Instead of applying the attention mechanism with d_{model} sized keys, values and queries it was proposed that we project the

keys, values, and queries h times with various linear projections of size d_k, d_k, and d_v [11]. In each of these projections, we apply the attention mechanism, which considers d_v - output values. The result is obtained after concatenating the h linear projections and projecting them one final time (Figure 2.14). The idea behind Multi-head Attention is

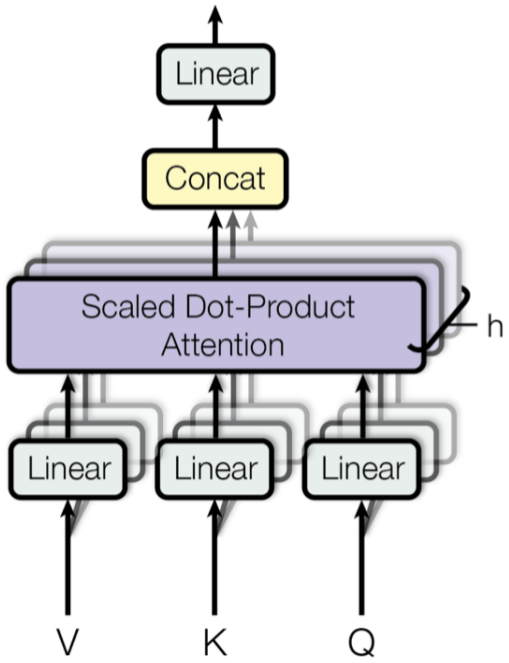

Figure 2.14: Multi-head Attention [11]

allowing the model to receive information from different representation spaces at different positions.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \tag{2.5}$$

where

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.6}$$

For each of the projection matrices, $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. It is further mentioned in the paper that due to the reduced dimension of each head the computational cost is similar to that of single-head attention.

Another attention-based architecture introduced [11] is the transformer encoder, or transformer block. Essentially, a transformer block consists of self-attention mechanism followed by a multilayer perceptron (MLP).

2.4.4.4 Positional Encodings

Attention mechanism lacks positional information and therefore treats each element of the data sequence as independent of the other. Therefore, position information needs to be added to the model solely to preserve information about the order of time steps. Positional encoding is the means by which knowledge of the order of the data in a sequence is maintained. Positional encodings can be both fixed and learned, they are usually aggregated and added to the input data values.In the case of fixed positional encodings, the following Trigonometric Sine and Cosine functions were proposed [11].

$$PE(pos, 2i) = \sin(pos/n^{2i/d_{model}}) \tag{2.7}$$

$$PE(pos, 2i+1) = \cos(pos/n^{2i/d_{model}}) \tag{2.8}$$

where pos defines the position in the input sequence, i is the dimension and n refers to a user defined variable(recommended value is 10.000).

Learned positional encodings originally proposed in convolutional seq2seq [29] learn a mapping function through the training process.

2.5 Wind Energy and Speed Prediction

Wind energy forecasting is closely related to wind speed forecasting. Therefore, various methods included in this section refer to wind forecasting. Wind energy forecasting can be distinguished into short (up to 6 hrs), medium (up to a day), long (up to a week), and very-long term forecasts. Several techniques have been developed for wind energy prediction, ranging from traditional statistical methods to advanced artificial intelligence techniques such as neural networks.

2.5.1 Traditional Methods

One of the traditional methods used for wind speed prediction is the persistence method, where the future wind speed is assumed to be equal to the current wind speed. However, this method is not accurate as wind speed can vary significantly over time. Another traditional method is the autoregressive integrated moving average (ARIMA) model, which is a statistical method used for time-series forecasting. A study was conducted in US to examine the use of those statistical time-series models in order to predict short-term wind power output up to six hours in advance [30]. In Egypt, a similar study was conducted comparing the two methods for short-term predictions [31]. For Day-ahead prediction, f-ARIMA model, an variation of ARIMA was able to improve significally the accuracy of wind speed forecasting significally compared to the persistence method [32]. Machine learning algorithms such as SVM were also used for short term prediction wind prediction with good results [33].

2.5.2 Neural Network Methods

One of the most commonly used neural network models is the multilayer perceptron (MLP). Experiments using several MLP based neural networks for 1-h-ahead forecast of wind speed were conducted using data observations in North Dakota (US) [34], and concluded that combining forecasts from different artificial neural network (ANN) models is needed to overcome the inconsistency issue in model selection. In Greece, 24-hours ahead hourly wind speed predictions using an MLP was made in Tilos using wind speed and pressure historical data [35]. A CNN-LSTM architecture for day ahead wind speed forecasting relying on WRF predictions produced some interesting results [12].

3. METHOD

In this chapter we present our method. We adapted the method suggested by [12] by adding new models and using a different approach on data preprocessing. We have structured this section in 3 subsections 3.1, 3.2, 3.3. In the first subsection we present the method we used for preprocessing our data, in the second we discuss the models we adopted to achieve our goal and in the third we describe the training and evaluation of our models.

3.1 Data Preprocessing

We present our input and target data and the methods we use to prepare them for the experiments. We work with data for two different parks. At first we discuss the target data preprocessing in 3.1.1. Secondly we describe the input data preprocessing in 3.1.2 and lastly we present the data normalization techniques we apply in 3.1.3.

3.1.1 Target Data

The target data used in this study was produced from a SCADA system and consists of real energy measurements from two different wind energy farms located in central Greece. Park A consists of 16 wind turbines with a total of 12,000 kW maximum capacity while park B consists of 17 wind turbines with a total of 10,200 kW maximum capacity. This target data have weaknesses regarding the turbines availability and wind speed when it was generated. More specifically over a period of time, turbines may not perform at one hundred percent due to maintenance or economy reasons. As a result, it is possible some of our energy values are inaccurate (considered as noise), so we need to identify and eliminate them. Moreover, we need to exclude energy measurements for wind speeds when turbines do not work.

First, we are provided with real wind speed measurements for 10 minute time intervals and energy measurements for every 15 minute intervals and we adjust them to 30-minute time intervals. To do this adaptation we calculate the sum of two near 15-minute points energy measurements and the average wind speed of the corresponding points. Through this adjustment we have the same time scale and a good number of data points for filtering. Using these adjusted values we plot the energy - wind speed diagram (Figure 3.1). The points on the diagram form a dense curve.

Second, by using the diagram we filter the noise. We consider the points outside of the curve as outliers and possible low availability points. In order to aggregate them we draw the line segments of Figure 3.2. These segments correspond to the following functions:

$$f_1(x) = \begin{cases} 700 \cdot (x - 7.5) & \text{if } 7.5 \leq x \leq 12.5 \\ 57.14 \cdot x + 2,785.75 & \text{if } 12.5 \leq x \leq 30 \end{cases} \tag{3.1}$$

$$f_2(x) = 635.14 \cdot x - 970.28 \text{ if } 2 \leq x \leq 9.4 \tag{3.2}$$

We remove from our data set the data points outside these line segments. We also remove data points that did not fulfill the following criteria:

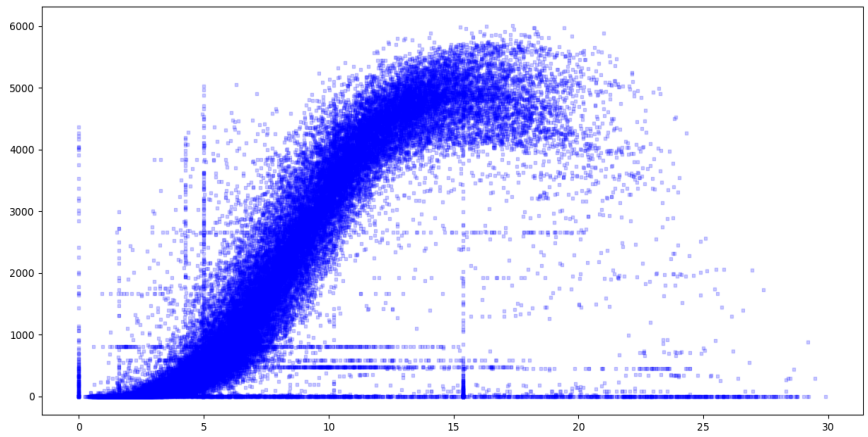

Figure 3.1: Power - Wind plot - 30 minutes window

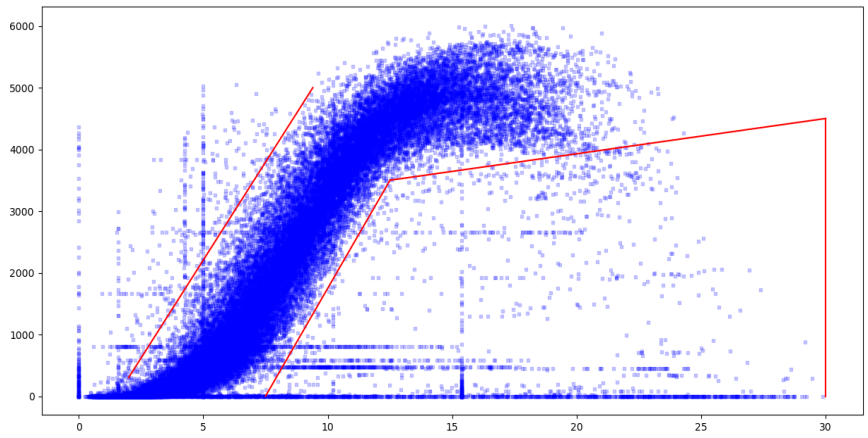

Figure 3.2: Power - Wind plot - 30 minutes window with lines

- Minimum energy value of 200kW. The reason we do this is that when the park is closed it is possible for the wind turbine blade to be slightly displaced by the wind. In these cases we have very low energy values that we want to rule out.
- Wind speed values less than 5 m/s and more than 25 m/s, The reason for this is that these values are turbines' cut-in and cut off speeds.

The outliers removed are displayed in Figure 3.3.

Thirdly , we adjust the energy and wind speed values from 30-minute time intervals to 1-hour using the same formula as we did earlier. Further elimination of the data is also done in this phase. In particular if any sub-value of energy of this hour is missing (it has been removed in the previous step) then the entire time period of one hour is removed from the data set as it is considered a point of low availability.

Finally, due to the specifications of the park, we limit the maximum price of energy to 12,000 kWh for park A and 10,200 kWh for park B. We set all exceeded values to this value. The final energy-wind speed plot is presented in Figure 3.4

The most recent energy data were provided with the percentage of turbines availability. If

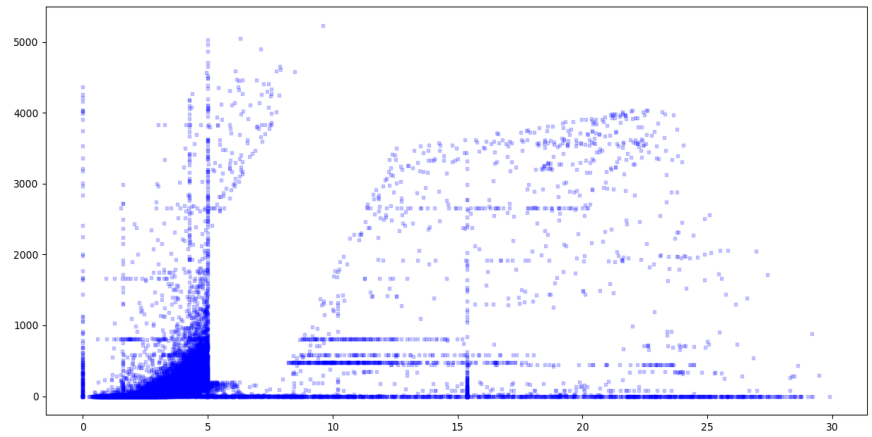

Figure 3.3: Power - Wind speed plot outliers

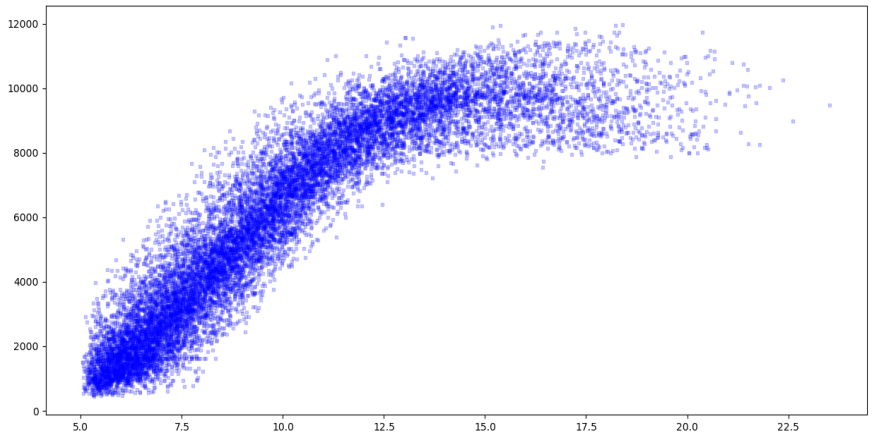

Figure 3.4: Power - Wind speed plot 1-hour interval

availability was not 100 percent then we transform them using the formula before starting the filtering process:

$$f(x) = \frac{x}{\text{availability}} \cdot 100 \tag{3.3}$$

The variation in energy values before and after the transformation is depicted in Figure 3.5.

3.1.2 WRF Input Data

The input data used in this study consisted of 18 WRF predicted features : X and Y components of wind, temperature, and pressure, all at 10,80,100,120 meters height, surface pressure, snow water equivalent and daily total snow and ice.

It was extracted from parameterized WRF models executed on a rectangular region covering a large fraction of mainland Greece, provided by the work of [36]. The model produces a WRF grid that records weather patterns, such as temperature or wind patterns that may be influenced by the environment. In particular, we get forecasts at every location of the

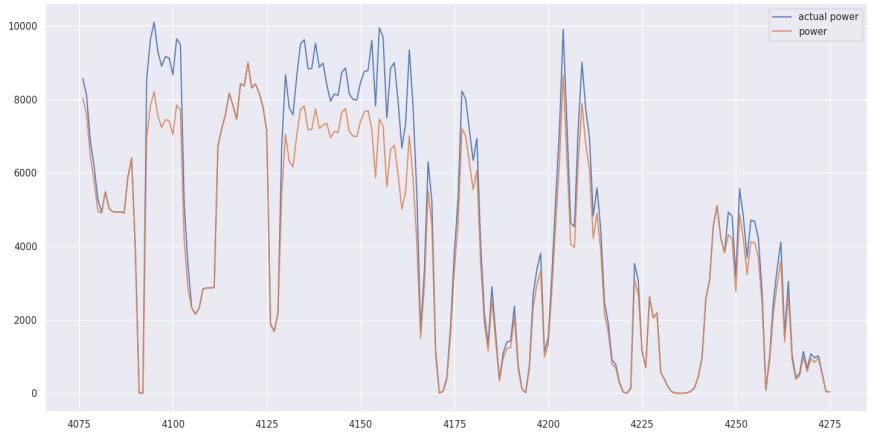

Figure 3.5: Availability transformation plot for a portion of data

grid with a 1 km resolution. Using the specified coordinates (longitude and latitude) of the wind parks we obtain the grid for the area of interest from the initial WRF (Figure 3.6). For

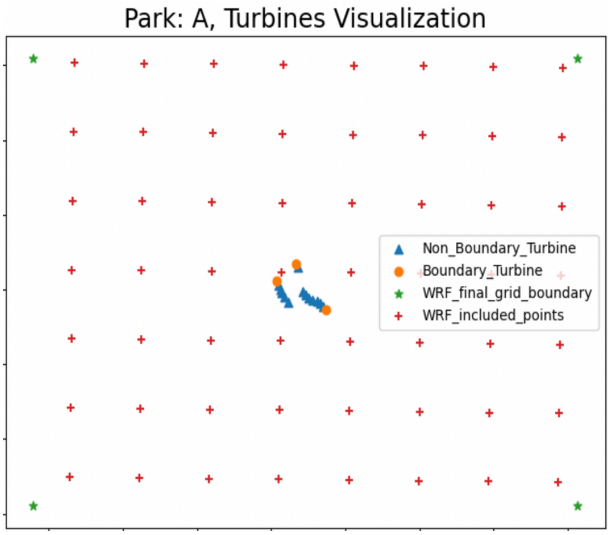

Figure 3.6: Park A WRF grid

each point of this grid the 18 WRF predicted features were generated for an hour interval.

In the preprocessing phase the input data was matched with the target data using time criteria. In particular, for each target label corresponding to a timestamp of t hours we saved the WRF predictions for the $t, t-1, ..., t-5$ hours.

As a result, each input element forms the following vector shape:

$$\text{timesteps} \times \text{park\_grid\_height} \times \text{parks\_grid\_width} \times \text{wrf\_features} \quad (3.4)$$

For our parks of interest, park A has a 7 $\times$ 8 grid, while park B has a 6 $\times$ 8 grid. As mentioned above, the input data was generated for 6 time steps and a total of 18 features for each grid point.

3.1.3 Normalization - Standardization

After aggregating our data, we normalize them in $[0, 1]$ with the following techniques. We use feature-wise normalization by applying ”max - min”(3.5)

$$X = \frac{X - \min X_{train}}{\max X_{train} - \min X_{train}} \tag{3.5}$$

or ”z-score”(3.6) normalization

$$X = \frac{X - \mu(X_{train})}{\sigma(X_{train})} \tag{3.6}$$

where μ describes the mean value and σ the standard deviation, depending on the experiment. We always normalize our input data, however depending on the experiment, our target data may not be normalized.

3.2 Models and components

For the implementation of the neural networks we used python’s programming language keras library. Our architecture overview consists of a feature extractor component with a purpose of identifying geographical trends and a temporal model whose goal is to capture temporal patterns. An architecture overview is displayed in Figure 3.7. The models output is described as:

$$y = h \circ g \circ f(x) \tag{3.7}$$

Where f represents the feature extractor, g represents the temporal model and h is the output MLP. This section is structured in 3 subsections. In the first we describe the feature extractor methods (3.2.1), in the second we discuss the temporal models (3.2.2) and in third we describe some extra components we used (3.2.3).

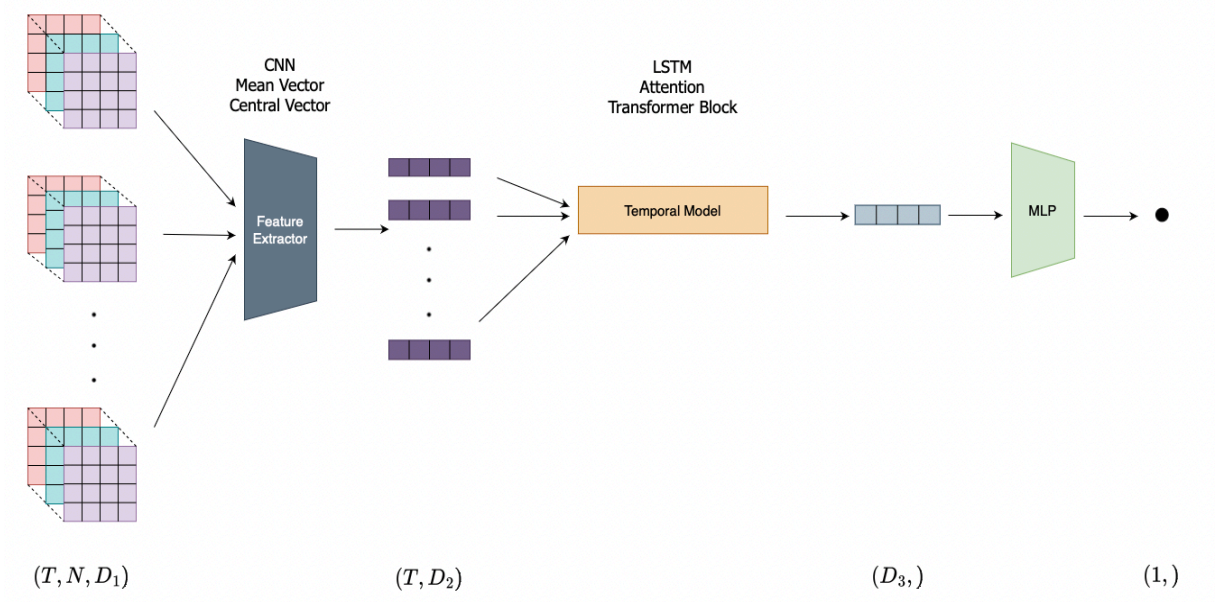

Figure 3.7: Architecture Overview

3.2.1 Feature Extractors

To extract meaningful spatial information we use several methods. We use a Convolutional neural network (3.2.1.1), the mean vector (3.2.1.2), and the central vector (3.2.1.3). We analyse them below.

3.2.1.1 Convolutional Neural Network

Convolutional Neural Network (CNN) is defined as

$$f : \mathbb{R}^{H \times W \times D} \longmapsto \mathbb{R}^{D'} \tag{3.8}$$

where H and W represent the WRF grid's height and width and D represents the input features dimension. H and W are set to 7 and 8 for park A and 6 and 8 for park B.D represents the input features number that is equal to 18 and D' represents output features dimension that is equal to 64.

We use an architecture that includes convolutional, pooling layers and dropout layers. Specifically, each CNN building block includes a convolutional 2D layer with 16 or 32 filters of 3 $\times$ 3 kernel size followed by a Leaky ReLU activation function and a max pooling layer of size 2 $\times$ 2 to reduce the grid's dimensions. Finally, dropout layers are used between blocks to prevent overfitting. Convolutional layers were applied seperately on each time step. The choice behind using small convolutional filters and a small pooling window is an attempt to reduce information loss. By using Leaky ReLU instead of ReLU we avoid the dying ReLU problem(cite).

3.2.1.2 Mean Vector

Another approach we deploy is to utilize the mean 18 WRF features of all the points of the WRF grid instead of using all the feature vectors in the grid.

$$\overrightarrow{v} = \frac{1}{H \cdot W} \sum_{i}^{H} \sum_{j}^{W} v_{i,j} \tag{3.9}$$

H and W represent height and width. This results in an output shape of 18 features for each of our 6 time steps.

3.2.1.3 Central Vector

Similarly with the mean vector method, we keep the central 18 WRF features of all the points of the WRF grid for each time step. To determine the central WRF point we compute the turbines' mean coordinate and choose the WRF point with the smallest manhattan distance(L1 norm) to it. That is our central point.

$$\begin{aligned}
\overrightarrow{y_i} &= (lat_i, long_i) \\
\overrightarrow{\mu} &= (\frac{1}{N}\sum_{k=0}^{N} lat_k, \frac{1}{N}\sum_{k=0}^{N} long_k) \\
\mathsf{L}_1(\overrightarrow{y_i}, \overrightarrow{\mu}) &= \sum_{j=0}^{2} |{y_i}^j - \mu^j| \\
\mathsf{c}(\overrightarrow{y_i}, \overrightarrow{\mu}) &= \arg\min_i \mathsf{L}_1(y_i, \overrightarrow{\mu})
\end{aligned} \tag{3.10}$$

Where N represents the number of wind turbines, μ represents their mean coordinates, y_i represents the coordinates of each WRF points and c represents the index of the central vector. We keep the features of point (3, 3) for park A, and point (2, 4) for park B. This results in an output shape of 18 features for each of our 6 time steps.

All the above methods have been used in the different experiments of the study. In the case of mean and central vector method a Multi-layer perceptron (MLP) with one hidden layer and output size 64 with a ReLU activation function follows. The MLP is described as followed

$$f : \mathbb{R}^D \longmapsto \mathbb{R}^{D'} \tag{3.11}$$

where D represents the input's number of features and D' the output features number. The purpose of using MLP is to feed our temporal model with extra features. The MLP in our method receives 18 input features and outputs 64.

3.2.2 Temporal Model

After extracting the output from our feature extractor component, we feed it to our temporal model component. We employ the LSTM (3.2.2.1), Self-Attention (3.2.2.2), and Transformer block (3.2.2.3) architectures that are analysed below.

3.2.2.1 LSTM

Our LSTM is defined as:

$$g : \mathbb{R}^{T \times D} \longmapsto \mathbb{R}^{D'} \tag{3.12}$$

where T represents the number of time steps, D the number of input features and D' the number of output features. In our method we input 64 features, we use an LSTM with a hidden size of 256 nodes followed by a dropout layer of 0.2 dropout rate. That results in an output shape of 256 features. We also experiment with bi-directional LSTMs of 256 hidden size nodes. In that case our output features are 512 since the two direction LSTM outputs are concatenated.

3.2.2.2 Self-Attention

To compute attention, we use keras' MultiHeadAttention based on the paper [11]. In our method we use as key and value dimension the value 64 the same as the features input dimension. Our model is defined as:

$$g : \mathbb{R}^{T \times D} \longmapsto \mathbb{R}^{T \times D'} \tag{3.13}$$

where T represents the number of time steps, D the number of input features and D' the number of output features. The model's output is computed and added back on the input. We flatten the output and end up with 384 final features.

3.2.2.3 Transformer Block

A transformer block method is an extension of self-attention, as we add on it an MLP. For the needs of our study, we choose an MLP with one hidden layer and output size 64 with a ReLU activation function. Transformer blocks have the same input and output shape. As in the case of attention, here also, we flatten the output and end up with 384 output features as a final shape.

3.2.3 Extra Components

Some additional components featured in our experiments are described below.

3.2.3.1 Positional Encodings

To receive positional information for the attention and transformer block models, we also experiment with learnable and fixed positional encodings used in other works [37, 11] . We receive input information from feature extractor models and we add the computed positional information of the same shape to it. We then forward the output on attention or transformer block. Positional encodings are incorporated into the input data using the following formula:

$$\begin{aligned} & w = PE(t) + x, \\ & t \in \mathbb{Z} \\ & PE : \mathbb{R} \rightarrow \mathbb{R}^D \\ & w : \mathbb{R}^{N \times D} \rightarrow \mathbb{R}^{N \times D} \end{aligned} \tag{3.14}$$

where x denotes the input data and w represents the final input data after the addition of positional encodings.

3.2.3.2 Output Layer

The output layer, is a fully connected layer (an MLP) followed by an activation function that outputs one value. It is implemented after our Temporal Model and generates the model's projected energy value. We conduct tests with a ReLU activation function, that outputs values on $[0, +\infty)$, and Sigmoid activation function, that outputs values on $[0, 1]$. We also conduct some experiments without using any activation function, that outputs values on $[-\infty, +\infty]$.

3.3 Training & Evaluation

In this subsection we describe the configuration we use to train and evaluate the architectures we presented in 3.2.

Regarding the loss function, we use mean absolute error (MAE) which is the L1 norm and we add the loss over training batches . MAE is given by

$$\mathrm{MAE} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} |(y_i - y_p)| \tag{3.15}$$

Where y_i is the actual energy value and y_p is the predicted value.
In some experiments we use sigmoid activation function that outputs values in [0,1]. Therefore, we slighty modify mae to normalise y_i, so that it is in the same range as y_p

$$\mathrm{MAE}_{\mathrm{custom}} = \sum_{i=1}^{n_{samples}} |(\frac{y_i - \mathrm{min}_y}{\mathrm{max}_y} - y_p)| \tag{3.16}$$

We also use Adam as our optimiser, with learning rate 0.001 and batch size is set to 48. In regard to the validation and test sets we use during training, they are a part of the data set that consist of data equally distributed through seasonality. That is because we want our network to predict accurate values for all seasons without biases. On both parks we select data from the same exact months for validation and test set. We have no overlaps. All models are trained for 100 epochs and the model with the lowest validation loss is saved and evaluated. The metric used to evaluate the test set is Mean normalized absolute error(MNAE)

$$\mathrm{MNAE} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \frac{|y_i - y_p|}{c} \tag{3.17}$$

Where y_i is the actual energy value and y_p is the predicted value and c is a constant value. It is set to 12.000 in our experiments.

4. RESULTS & DISCUSSION

In this chapter we present the results we obtained from the experiments and discuss them.

4.1 New / Old Data Comparison

First, we present the results we achieved for applying our method of preprocessing. We compare them with the previous method of preprocessing. We are using minmax normalization and the CNN + LSTM model as described in section 3.2. The models are tested over the same test set and we are also maintaining the same validation set.

Table 4.1: Comparison of data preprocessing park A

Data	MNAE
Old preprocessing	14.1656 ± 0.3885
Ours	12.3939 ± 0.2880

Table 4.2: Comparison of data preprocessing park B

Data	MNAE
Old preprocessing	13.3116 ± 0.4404
Ours	13.0028 ± 0.6116

We noticed significant changes of the metrics we acquired with the old data compared to the new in both parks. In particular, the metric improved on both parks, especially in park A (12.3939 compared to 14.1656) , but also in park B (13.0028 compared to 13.3116).

These results highlight the importance of data preprocessing as a critical step in the development of accurate and reliable neural network models. They also show the potential benefits that can be achieved by implementing appropriate preprocessing techniques. Our suggested preprocessing method removed irrelevant or erroneous information from the dataset and ensured that the neural network is not fed with any biases.

4.2 Default Model Finetuning

In this section we will present the results we obtained from fine-tuning our default model. Those results helped us choose the configuration we used on the next experiments. We tried out bi-directional LSTM, different output functions and normalization techniques.

Table 4.3: Fine-tuning activation function, park A

Normalization	Output Activation function	MNAE
Min-max	ReLU	12.3939 ± 0.2880
z-score	Sigmoid	11.7516 ± 0.3657
Min-max	-	12.2239 ± 0.2290

Based on those experiments we selected the normalization technique and output activation function we used for the next experiments. Z-score normalization and and sigmoid output activation function outperformed the rest. While the differences are not significant, it seems like that configuration fits the distribution of our data better.

Table 4.4: Fine-tuning activation function, park B

Normalization	Output Activation function	MNAE
Min-max	ReLU	13.0028 $\pm$ 0.6116
z-score	Sigmoid	12.6021 $\pm$ 0.4108

4.3 Feature Extractors

In this section we will present the results we achieved from experimenting with different feature extractor methods that we described in subsection 3.2.1. We are using LSTM as our temporal model.

Table 4.5: Comparison of feature extractors park A

Feature extractor	MNAE
CNN	11.7516 $\pm$ 0.3657
Mean vector	12.1674 $\pm$ 0.2057
Central vector	12.0207 $\pm$ 0.2573

Table 4.6: Comparison of feature extractors park B

Feature extractor	MNAE
CNN	12.6021 $\pm$ 0.4108
Mean vector	12.4049 $\pm$ 0.3539
Central vector	12.1578 $\pm$ 0.0937

The results we obtained are interesting. For park A, the central vector method performed a bit better than mean vector method (12.0207 compared to 12.1674). Compared to the other methods, CNN gives us the best metric (11.7516). Despite that, the results of CNN do not have big differences with the other methods. An explanation is the fact that the input WRF values are close from one point of the grid to another and the spatial patterns the CNN is designed to capture are not that significant. For park B, central vector outperformed the other methods (12.1578). CNN in that case did not help our model as expected. A possible explanation is, the geographical distance between some points of the grid and the location of the turbines created instability. This instability could be due to the fact that the CNN may have difficulty capturing the spatial relationships between these distant points, leading to poor performance.

The results of the experiments raise questions about the essentiality of CNNs for our problem. As our experiments showed that a simpler model achieved similar or even better performance. To further improve the performance models that use CNN, it is worth considering the possibility of modifying the dimensions of the WRF grid.

4.4 Temporal Model

In this section we will present the results we obtained from experimenting with the different temporal models described in subsection 3.2.2. We are using CNN as our feature extractor and Sigmoid output activation with z-score normalization and custom mae as a loss function.

Table 4.7: Comparison of temporal models park A

Temporal model	Pos. encodings	MNAE
LSTM	-	11.7516 $\pm$ 0.3657
Bi-LSTM	-	12.4621 $\pm$ 0.5107
Attention	-	12.2100 $\pm$ 0.5640
Attention	Learnable	11.9796 $\pm$ 0.3330
Attention	Fixed	12.0599 $\pm$ 0.3525
Transformer Block	-	12.2671 $\pm$ 0.4067
Transformer Block	Learnable	11.8685 $\pm$ 0.2706
Transformer Block	Fixed	11.6368 $\pm$ 0.2032
2 Transformer Blocks	-	11.9848 $\pm$ 0.2606
2 Transformer Blocks	Learnable	11.9009 $\pm$ 0.1874
2 Transformer Blocks	Fixed	12.0535 $\pm$ 0.3532

Table 4.8: Comparison of temporal models park B

Temporal model	Pos. encodings	MNAE
LSTM	-	12.6021 $\pm$ 0.4108
Bi-LSTM	-	12.4230 $\pm$ 0.3290
Attention	-	13.0851 $\pm$ 0.3866
Attention	Learnable	13.0798 $\pm$ 0.4014
Attention	Fixed	12.8692 $\pm$ 0.2602
Transformer Block	-	12.9023 $\pm$ 0.6318
Transformer Block	Learnable	12.5219 $\pm$ 0.4044
Transformer Block	Fixed	12.9150 $\pm$ 0.5425
2 Transformer Blocks	-	13.1800 $\pm$ 0.6846
2 Transformer Blocks	Learnable	12.6517 $\pm$ 0.3731
2 Transformer Blocks	Fixed	12.7696 $\pm$ 0.4674

For park A, we observe that metrics are close in all cases. The best performance was achieved using a Transformer block with fixed positional encodings as it outperforms the LSTM (11.6368 compared to 11.7516). We observe that two stacked transformer blocks compared to one slightly improve the metrics (11.9848 compared to 12.2671). Positional encodings tend to improve results in both attention and transformer block components. In particular, learnable positional encodings help attention (11.9796 compared to 12.2100) and fixed positional encodings help single transformer block significantly (11.638 compared to 12.2671). However, major improvement is not observed when combining them with two stacked transformer blocks. Learnable slightly help the models (11.9009 compared to 11.9848) while fixed produce slightly worse results (12.0535 compared to 11.9009).

We obtain similar results for park B. Fixed positional encodings improve attention model (12.8692 compared to 13.0851) and learnable positional encodings improve Transformer Block significantly (12.5219 compared to 12.9023). Two Stacked transformer blocks in this case do not improve the model as they produce a worse results compared to a single transformer block (13.1800 compared to 12.9023). In that case, using fixed positional encodings helps the model (12.7696 compared to 13.1800) and learnable positional encodings also seem to help (12.6517 compared to 13.1800). The best metric though is obtained by a bidirectional LSTM (12.4230). Combining information from both time directions proved useful for park B, long term dependencies captured by bi-LSTM helped the

model improve the performance, differences are not significant though.

Overall, LSTM architectures and attention-based architectures showed comparable performance when used to process and analyze our time-series data. That suggests that both architectures can be considered as viable options as temporal models in our task. As for positional encodings, our models improved considerably with their addition in most of the cases. That is somewhat expected since they offer valuable positional information. Comparing learnable and fixed positional encodings, we do not observe major differences between them. Learnable position encodings thought improved the model in all cases they were used.

4.5 Multiple Parks - All in One Model

In this section we share the findings from combining the data from two models and creating an all-in-one model. Parks A and B are 6km distance apart, and meteorological WRF predictions should be similar we combine park data and create an all-in-one model. We monitor the metrics over both test sets and the best model when both validation losses decrease. We use the CNN + LSTM model, sigmoid output activation function with z-score normalization and custom mae.

Table 4.9: All-in-one model

	MNAE park A	**MNAE park B**
Single, A	11.7516 $\pm$ 0.3657	-
Single, B	-	12.6021 $\pm$ 0.4108
All-in-one	11.9604 $\pm$ 0.3097	11.9030 $\pm$ 0.3462

The findings of this experiment are really promising. The metric of park A gets a bit worse (11.9604 compared to 11.7516). This has to do with the fact we have different max values on park A and park B. In particular, park A values are capped on 12 kWh and park B values are capped on 10.2 kWh. As a result the all-in-one model misses some high values for park A.

However, park B benefits a lot (11.9030 compared to 12.6021) from this merge. That leads us to think that the concept of merging the park train data might work if we encode the information of which park data come from for our model.

This experiment provides us with some interesting insights. It shows that transfer learning between two nearby parks is a viable option. By leveraging data from two different parks, we were able to achieve partial improvement in our forecasting accuracy. Moreover, we understand that this transfer learning approach can also be used to make predictions about a new park that has limited data available by using existing data from a nearby park with similar weather patterns.

4.6 Visualization

In this section we are presenting two visual representations of test set predictions.

In Figure 4.1 we have included a visual representation of park A best model's (CNN + transformer block + fixed pos.encodings) predicted values compared to the actual ground truths. We have also included in Figure 4.2 a visual representation of the predictions of the

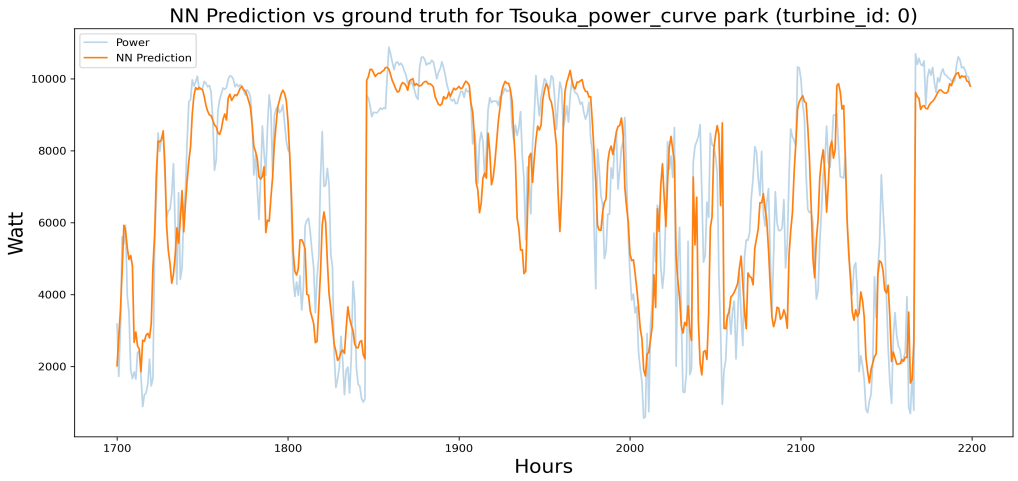

Figure 4.1: Ground truth - Prediction lines park A

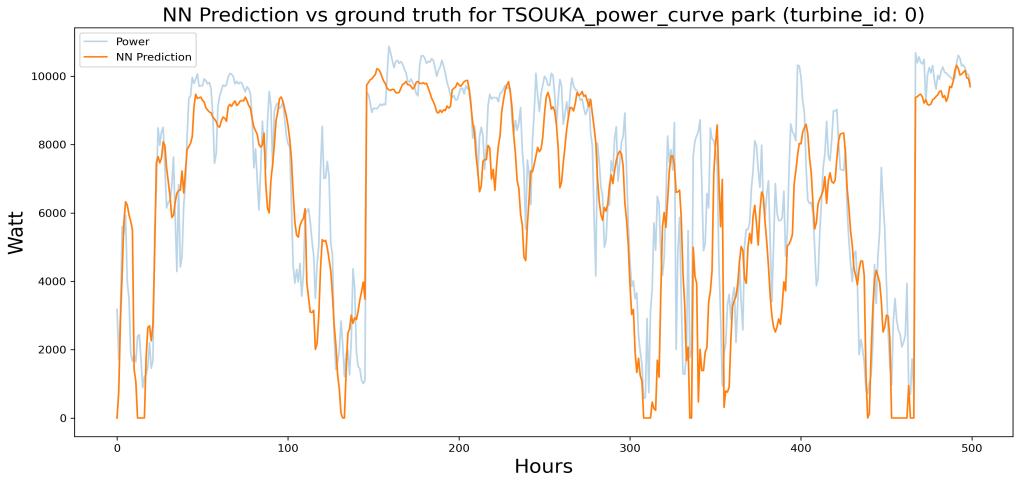

Figure 4.2: Ground truth - Prediction lines old model park A

model trained with old preprocessing data (Table 4.1). As previously mentioned, the visual predictions of the old data model a certain degree of noise, which is also supported by the evaluation metric. In addition, it can be observed that the low and high energy prices are not modeled as accurately as they are with our best model. As we have already mentioned and the metric suggest, the predictions we acquire in the second case are are noisier. Furthermore, we can observe that especially low energy prices but also high energy prices are modeled better with our best model.

5. CONCLUSIONS & FUTURE WORK

The primary objective of this thesis was to investigate the effectiveness of various deep learning architectures for energy prediction using WRF data and actual measurements from two wind farms.

The first part of our project involved developing a data preprocessing method based on our foundation paper [12]. The testing was performed in comparison with this paper. The results showed that our proposed method exhibited better performance on the same exact test set. These findings indicate the crucial role of efficient preprocessing in such tasks as it made a great impact on the performance of our models. Overall we consider it as one of the most important aspects in such tasks.

In the second part we implemented the model architecture. Our model architecture consisted of a feature extractor and a temporal model. In particular, as feature extractors we implemented the CNN, mean vector and central vector approaches. Our results were similar in all cases as in the first park CNN performed to some extent better while in the second park the central vector approach outperformed the rest. That raises questions about the essentiality of CNNs for our problem as our experiments showed that a simpler approach achieved similar or even better performance. For temporal models, we compared LSTM, attention, and transformer blocks with the addition of positional encoding. Our results showed that attention-based perform comparably and even better than LSTM architectures as temporal models. In addition while no clear preference emerged between learnable and fixed positional encodings, their addition to attention-based models was essential.

Finally, we built an all-in-one model using data from both wind farms, which showed that transfer learning between nearby parks is a promising approach. By leveraging data from multiple sources, we achieved partial improvement in forecasting accuracy. That also demonstrated the viability of using transfer learning to make predictions for a new park with limited data by utilizing existing data from a nearby park with similar weather patterns.

As a whole, it was a really interesting project since we had the freedom to experiment with a variety of aspects of the problem at hand. We gained valuable insights into the role of efficient preprocessing and the behavior of several deep learning architectures for feature extraction and temporal modeling. Certain limitations this study faced is the lack of data, the data provided were valuable but a larger dataset would have provided more robust and reliable results. In addition, is worth noting that the data provided were noisy and required denoising. While our method addressed this issue, it remains a challenge that requires further exploration. In addition, although the WRF provided valuable meteorological features for our input data, it is a forecasting model that contains inherent errors. To address this, one potential improvement could be to incorporate actual meteorological data in order to model the WRF's historical prediction errors. By doing so, we could potentially improve the accuracy of our model by accounting for the discrepancies between the predicted and actual meteorological values. Additional improvements to our model could involve leveraging aerial images of the wind farm to further encode the park's location and geography. This extra signal can provide valuable information about nearby terrain features, such as hills or valleys, that can affect meteorological patterns and improve the model's ability to make accurate predictions. Another interesting approach could be the use of unsupervised pre-training tasks (self-supervised learning) [38]. By training the model on WRF data from several areas in Greece for a pre-training task, the model can learn common patterns and features in the data that are relevant to wind energy prediction. This can

provide a strong foundation for the model to make accurate predictions for individual wind parks. After pre-training, the model can be fine-tuned using the specific data from each wind park. Fine-tuning in this way allows the model to adjust to the unique characteristics of each park, and produce better results than if the model were trained from scratch.

Regarding model architectures, our study used a sigmoid output activation function. However, with deep networks that involve multiple transformer blocks, issues such as the vanishing gradients problem [39] may arise. While our experiments did not observe this issue, it is possible that it could occur if we deepen the network with multiple transformer blocks. In such cases, it may be worth considering the use of no activation function. To sum up, we believe that our findings related to the importance of preprocessing, the effectiveness of attention-based models, and the potential for transfer learning between neighboring parks provide valuable insights for those interested in conducting research in this field.

ABBREVIATIONS - ACRONYMS

WRF	Weather Research Forecasting
CNN	Convolutional Neural Network
LSTM	Long-Short Term Memory
MLP	Multi-Layer Perceptron
HAWT	Horizontal Axis Wind Turbine
VAWT	Vertical Axis Wind Turbine
SCADA	Supervisory Control and Data Acquisition
ML	Machine Learning
RNN	Recurrent Neural Network
Seq2Seq	Sequence to sequence
ARIMA	Auto-Regressive Integrated Moving Average
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
MAE	Mean Absolute Error
MNAE	Mean Normalized Absolute Error
ΕΚΠΑ	Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

APPENDIX A. FIRST APPENDIX

A.1 Additional Experiments

The section of the appendix presents additional experiments and results that were conducted but not included in the main body of the study. Those experiments were conducted using CNN as our feature extractor and Min-Max normalization

Table A.1: Extra experiments, park A

Temporal	Activation	Augmentation	MNAE
Bi-LSTM	ReLU	-	12.24 ± 0.38
Bi-LSTM	-	-	11.90 ± 0.27
LSTM	ReLU	horizontal flip	12.42 ± 0.56

In one of the experiments we employed a data augmentation technique called horizontal flip to provide the model with more diverse and informative representations of the same data. This technique involves flipping the grid horizontally, which enables the model to recognize patterns regardless of their orientation. Our goal was to enable the model to learn new representations and improve its robustness, allowing it to better generalize to unseen data and ultimately enhancing its overall performance.

We also present some extra all-in-one model experiments.

Table A.2: Extra experiments all-in-one model

Feat.Extractor	Temporal	Activation	MNAE,A	MNAE,B	Normal.
CNN	LSTM	ReLU	12.53 ± 0.62	12.43 ± 0.62	min-max
Mean vector	LSTM	ReLU	12.02 ± 0.28	12.58 ± 0.29	min-max
CNN	Tr.Block	Sigmoid	11.92 ± 0.34	12.05 ± 0.30	z-score

The results obtained from the experiments presented in this appendix exhibit a certain degree of resemblance, to the ones we have highlighted in the main body of our study as they share similar patterns.

A.2 Metric Calculation

For our metric calculation we used MNAE:

$$\text{MNAE} = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \frac{|y_i - y_p|}{c} \tag{A.1}$$

The constant value c is set at 12.000, which is the maximum value recorded for park A. Typically, energy providers set c as the maximum possible value measured for a given park. However, in our study, we used the same c value for both parks A and B to demonstrate the relative differences between the models and to train an all-in-one model. It is worth noting that the metrics can be easily transformed to per-park basis since the maximum value recorded for park B is 10.200.

A.3 Libraries Used

We implemented the models using Python programming language, with the `Keras` library for MLP, CNN, LSTM, and Attention and learnable positional encodings. For fixed positional encodings, we were inspired by [40], and implemented them using `TensorFlow` and `NumPy` libraries. Transformer blocks were inspired by the work of [41].

BIBLIOGRAPHY

[1] H. W. E. Company, “Wind energy statistics in greece for the 1st half of 2022,” Aug. 10 2022. [Online]. Available: https://eletaen.gr/d-t-statistiki-eletaen-first-semester-2022/

[2] I. Dincer and H. Ishaq, “Chapter 4 - wind energy-based hydrogen production,” in *Renewable Hydrogen Production*. Elsevier, 2022, pp. 123–157. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323851763000159

[3] S. Cole, “Wind turbine power curve.” [Online]. Available: https://theroundup.org/wind-turbine-power-curve/

[4] C. Olah, “Neural networks, manifolds, and topology,” Apr. 6 2014. [Online]. Available: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

[5] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” 2015.

[6] C. Olah, “Understanding convolutions,” Jul. 13 2014. [Online]. Available: https://colah.github.io/posts/2014-07-Understanding-Convolutions/

[7] DataCamp, “Recurrent neural network tutorial (rnn).” [Online]. Available: https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network

[8] C. Olah, “Understanding lstm networks,” Aug. 27 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[9] “Differences between bidirectional and unidirectional lstm.” [Online]. Available: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm

[10] S. Kostadinov, “Understanding encoder-decoder sequence to sequence model,” Feb. 5 2019. [Online]. Available: https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346

[11] A. Vaswani *et al.*, “Attention is all you need,” 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1706.03762

[12] E. Christoforou, I. Z. Emiris, A. Florakis, D. Rizou, and S.Zaharia, “Spatio-temporal deep learning for day-ahead wind speed forecasting relying on wrf predictions,” *Energy Systems*, pp. 1–21, Sep. 15 2021. [Online]. Available: https://doi.org/10.1007/s12667-021-00480-6

[13] G. J. Herbert *et al.*, “A review of wind energy technologies,” *Renewable and sustainable energy Reviews*, vol. 11, no. 6, pp. 1117–1145, 2007.

[14] K. Wind, “202 types of wind turbines, their advantages & disadvantages.” [Online]. Available: https://kohilowind.com/kohilo-university/202-types-of-wind-turbines-their-advantages-disadvantages/

[15] N. G. Society, “Wind energy,” Oct. 9 2012. [Online]. Available: https://education.nationalgeographic.org/resource/wind-energy/

[16] M. Cheng and Y. Zhu, “The state of the art of wind energy conversion systems and technologies: A review,” 2014.

[17] V. Sharma, S. Sharma, and G. Sharma, “Recent development in the field of wind turbine,” *Materials Today: Proceedings*, vol. 64, pp. 1512–1520, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785322037798

[18] P. Loshin, “Scada(supervisory control and data acquisition),” 2016. [Online]. Available: https://www.techtarget.com/whatis/definition/SCADA-supervisory-control-and-data-acquisition

[19] J. Powers *et al.*, “The weather research and forecasting model: Overview, system efforts, and future directions,” *Bull. Am. Meteorol. Soc.*, vol. 98, no. 8, pp. 1717–1737, Aug 2017, doi: 10.1175/BAMS-D-15-00308.1.

[20] W. Skamarock *et al.*, “A description of the advanced research wrf model version 4.3,” 2021, doi:10.5065/1dfh-6p97.

[21] G. Cuevas-Figueroa, P. K. Stansby, and T. Stallard, “Accuracy of wrf for prediction of operational wind farm data and assessment of influence of upwind farms on power production,” *Renewable Energy*, vol. 94, pp. 111–124, May 2016.

[22] H. Ashtari, “What is a neural network?definition,working,types and applications,” Aug. 3 2022. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-a-neural-network/

[23] K. Tertikas, “Implementation of dense sift as a convolutional neural network,” 2017.

[24] S. Hochreiter and J. Schmidhuber, “Long short-term memory (lstm),” 1997.

[25] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[26] D. Bahdanau *et al.*, “Neural machine translation by jointly learning to align and translate,” 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1409.0473

[27] A. Ambartsoumian and F. Popowich, “Self-attention: A better building block for sentiment analysis neural network classifiers,” in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Oct. 2018, pp. 130–139. [Online]. Available: https://aclanthology.org/W18-6219

[28] M. Luong *et al.*, “Effective approaches to attention-based neural machine translation,” 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1508.04025

[29] J. Gehring *et al.*, “Convolutional sequence to sequence learning,” 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1705.03122

[30] M. Milligan and S. Watson, “Statistical wind power forecasting models: Results for us wind farms,” *Journal of solar energy engineering*, vol. 125(1), pp. 172–179, 2003.

[31] A. Abdelaziz, “Short term wind power forecasting using autoregressive integrated moving average modeling,” *Electric Power Systems Research*, vol. 92, pp. 69–77, 2012.

[32] R. G. Kavasseri and K. Seetharaman, “Day-ahead wind speed forecasting using f-arima models,” *Renewable Energy*, vol. 32(15), pp. 2555–2567, 2007.

[33] M. A. Mohandes, T. O. Halawani, and S. Rehman, “Support vector machines for wind speed prediction,” *Renewable Energy*, vol. 34(2), pp. 392–397, 2009.

[34] G. Li and J. Shi, “On comparing three artificial neural networks for wind speed forecasting,” *Applied Energy*, vol. 87, pp. 2313–2320, Jul. 2010.

[35] K.Moustris *et al.*, “24-h ahead wind speed prediction for the optimum operation of hybrid power stations with the use of artificial neural networks,” in *Perspectives on Atmospheric Sciences*, 2017, pp. 409–414.

[36] Research team of A. Anastasiou and I. Z. Emiris, “Wrf model parameterization and adaptation for the mainland of greece, with final feature extraction.”

[37] J. Devlin *et al.*, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1 (Long and short papers), pp. 4171–4186, 2019.

[38] D. Erhan *et al.*, “Why does unsupervised pre-training help deep learning?” *Journal of Machine Learning Research*, vol. 11, no. 19, pp. 625–660, 2010. [Online]. Available: http://jmlr.org/papers/v11/erhan10a.html

[39] C.-F. Wang, “The vanishing gradient problem,” 2019. [Online]. Available: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[40] Tensorflow. [Online]. Available: https://www.tensorflow.org/text/tutorials/transformer

[41] A. Nandan. [Online]. Available: https://keras.io/examples/nlp/text_classification_with_transformer/