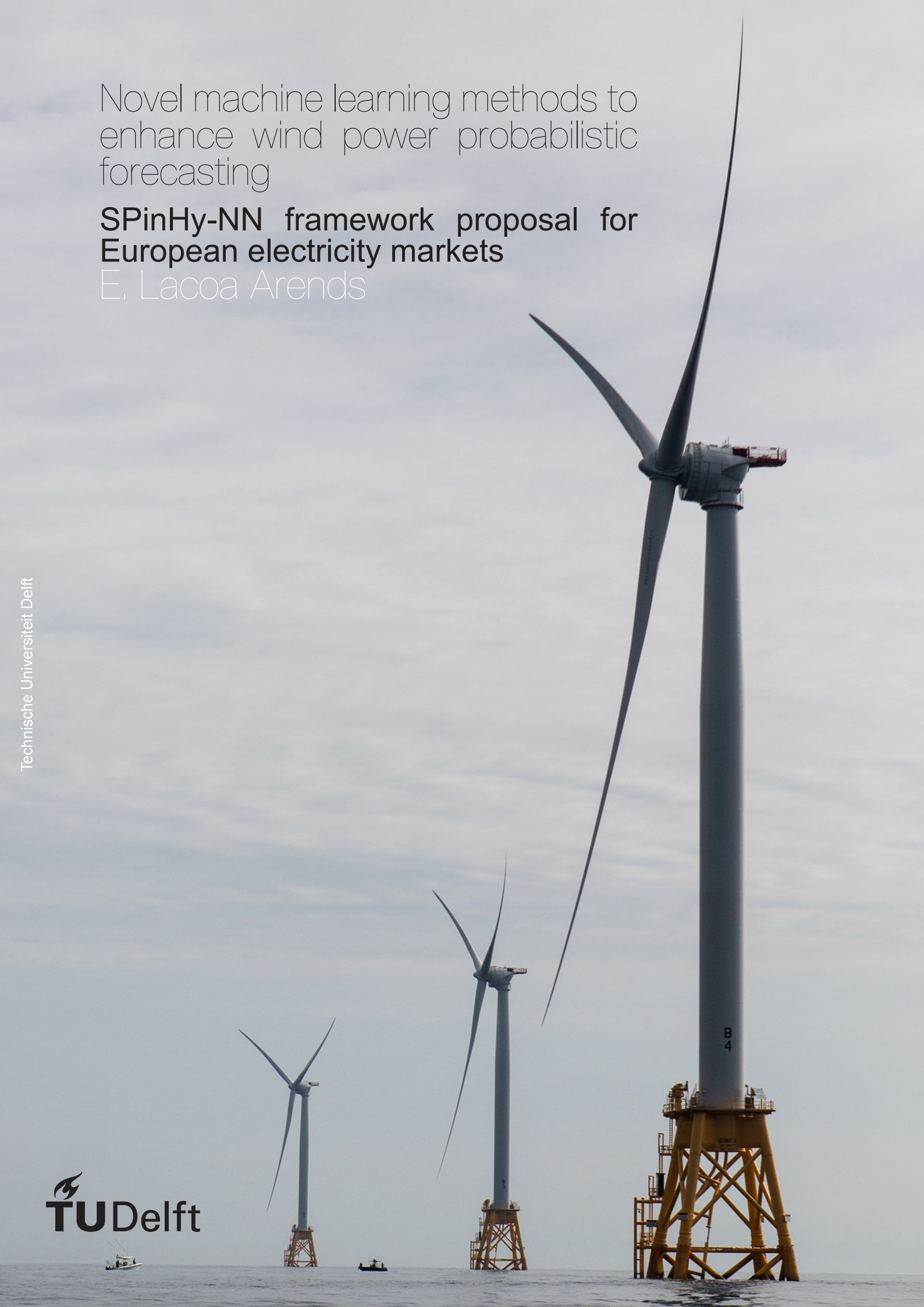
Novel machine learning methods to enhance wind power probabilistic forecasting

SPinHy-NN framework proposal for European electricity markets

E. Lacoa Arends



Technische Universiteit Delft

TU Delft

Novel machine learning methods to enhance wind power probabilistic forecasting

SPinHy-NN framework proposal for European electricity markets

by

E. Lacoa Arends

to obtain the degree of Master of Science in Sustainable Energy Technology
at the Delft University of Technology,
to be defended publicly on Friday, October 16, 2020 at 14:00.

Student number:	4772776	
Project duration:	November 18, 2019 – September 18, 2020	
Thesis committee:	Prof. dr. S.J. Watson,	TU Delft (AE) Supervisor
	Dr. S. Basu,	TU Delft (CiTG) Supervisor
	Dr. M.A. Schleiss,	TU Delft (CiTG) Thesis committee

Cover image courtesy: Kayana Szymczak for the New York Times.

TU Delft

Preface

My experience following the Sustainable Energy Technology master program has been quite a ride. A 22-year old boy from Caracas: ambitious and passionate about challenging topics, decided to leave a collapsing hometown to become a specialist in the energy system. The growth of the offshore wind energy sector and the emerging research of floating structures captured my attention to join the energy transition. An unconventional thought for someone born and raised in the land of the largest proven crude oil reserves in the world. There was more, though.

Following the Autonomous Systems track, I got introduced to different technologies and the importance of integrating them into a complex system. In particular, I was inspired by the future impact of high penetration of weather-dependent sources in electricity markets, as there must be a match between supply and demand at all times. On the other hand, I enjoy solving optimization problems, using my programming skills to model systems and find that local minimum. Therefore, I was closely following the emerging field of machine learning, seeing how data-driven approaches were solving intriguing problems. Just the simple handwritten digits recognition application amazed me enough to learn more. However, only a few projects were working with machine learning in my program. Consequently, I pitched my research topic at the Wind Energy Section of the Aerospace Faculty to the person who later became my main supervisor. The intention: enhance wind power forecasting methodologies using machine learning to aid power system operators' decision-making process. I must say I had to start from scratch. In this learning process, the MOOC in Machine Learning available in Coursera taught by Andrew Ng, gave me the fundamentals to dig deeper into the application I wanted to solve. Also, my second supervisor found the perfect setting to push myself by enrolling in the European Energy Markets Forecasting Competition 2020. These were three intense months, amidst the outbreak of COVID-19 in Europe. I failed, I learned, I failed once more until I succeeded. As a result, our team published two conference papers in this event, having the opportunity to present one of them as the main author, which is part of the outcome of this research project.

Eight months of work are summarized in this document. Working from home most of the time, adapting to the so-called "new normal", I made it to this point with the help of many people. Naming all of them in this short one-page is not enough. I will still take this space to acknowledge some names:

My supervisors: Simon J. Watson and Sukanta Basu. I want to thank Simon for always pushing me to provide a quality job. His professional character taught me how to be strategic beyond being smart. I am grateful to Sukanta for his energy, passion, and brilliant mind. He gave me the resources to find my way out during the bottlenecks of my project, encouraging me to follow new directions.

My family: Gardenia, Ulises, *agüe* Nancy. You have been part of the whole process, my evolution. I want to thank you for the emotional, rational, and financial support to follow my aspirations in life. Rodrigo, Nancy, and Juana: without you, this experience would not have been the same. I consider you all my godparents, a solid rock to facilitate my decisions. I owe you my academic formation.

Judith Claassen: for taking such good care of me during these months and reviewing my work. Your sense of caring and unconditional support has given me the energy to follow my dreams. It's you.

My Venezuelan friends: some still back home, most of them spread across the world. I miss you and keep you in my mind, the experiences we have lived together represent the most valuable treasure taken from home, important to overcome the difficulties of these times. My friends in the Netherlands: I have learned so much from you, diverse cultures, different ways of seeing life; you made me understand the world in new ways, to question my beliefs in a constructuve manner. I am fortunate that we all crossed paths and I felt the comfort of your presence.

Finally, on the eves of my 25th birthday, I reflect on this challenging yet beautiful road and cannot stop thinking about my father. As I am writing this, he would be turning 59 years old today. I want to acknowledge his teachings and discipline. After all, I think he would be very proud of me.

E. Lacoa Arends
Delft, October 2020

Summary

The promotion of sustainable energy as a means to tackle climate change is producing continuous growth of wind and solar. The increasing penetration of weather-dependent energy sources brings additional challenges to the operation of the power system. Wind power forecasting is a valuable resource for these operators: a tool that aids the decision-making process and facilitates risk management. However, the energy transition requires further developments over these predictive models to guarantee the security of supply and system adequacy. On the other hand, the progress of artificial intelligence algorithms and their success in different fields attracted research into wind power probabilistic forecasting. In particular, the Smooth Pinball Neural Network (SP-NN) is a non-parametric model for probabilistic applications, which introduces a customized objective function and addresses forecasting inconsistencies. Therefore, the objective of the research was to enhance wind power forecasting through data-driven or machine learning models, by proposing a new framework as an alternative to the existing ones, underlying its advantages and limitations, which is validated in different climate regions using as input grid-like topology data (weather images).

The project focused on the European Energy Markets 2020 Conference (EEM20) Forecasting Competition. The setting emulates the day-ahead electricity market and the location consisted of all four electricity price regions in Sweden, where three main sets of data were provided: (a) Numerical Weather Prediction (NWP) meteorological data (seven variables and ten ensemble members); (b) a wind turbine record; (c) aggregated wind power production by price region. The data timestamps are divided hourly for the years 2000 and 2001, serving each year as training and testing data sets, respectively. The wind installed capacity at each price region was 1.33, 3.01, 2.66, and 1.64 (GW). The forecast score uses the pinball loss (PL) function metric. Three main challenges are present in this competition, namely the large volume of data (the curse of dimensionality), the lack of information regarding wind turbine availability, and the growing evolution of wind installed capacity between data sets.

The methodology to create a framework consisted of three steps: (a) evaluating different methods in deterministic approach such as two benchmark models, three Convolutional Neural Networks (CNN), four Multi-layer Perceptrons (MLP), and k-means clustering; (b) feature engineering to make the best use of the information, enhanced by data analysis; (c) tuning the final model by concatenating model architectures based on weather images and additional features as input to compute quantile predictions, learning from the heteroscedasticity of data. The final framework is called the Smooth Pinball Hybrid Neural Network (SPinHy-NN), an extension of the SP-NN: introducing a CNN and a customized MLP architecture.

The results show that the LeNet-5 is the best CNN architecture, obtaining a Mean Absolute Error (MAE) at each price region of 104.65, 187.58, 150.94, and 109.44 (MW), for the period March-December 2001. The k-means clustering approach was optimized following the elbow method, obtaining a MAE of 152.95, 297.02, 283.84, and 170.59 (MW), respectively. This showcases the potential of CNN over k-means clustering. A high correlation is present between wind speed-wind gusts and power output. However, joining both variables did not improve the quality of the forecasts and increased computational costs. Hence, wind speed is the only input weather image. The SPinHy-NN was tuned for each price region, considering the sub-sampling layer, spatial dropout for regularization, batch size for stochastic gradient descent, and quantile margin (ϵ) for sharpness and consistency. The quantile cross-over problem has been evaluated through the crossing loss (CL) and the number of crossings (NC) metrics. The analysis shows that the quantile margin (ϵ) can correct this undesired behavior.

The conclusions validate the SPinHy-NN as a generalized framework for different climates through the final score achieved post-competition, managing to capture the spatial patterns in all cases except for rounds 1 and 4. Also, the global score of 57.54 is competitive with the top three teams of the competition. Moreover, the framework can be adapted to reach a desirable trade-off between accuracy, sharpness, and consistency by tuning the margin quantile parameter (ϵ) in different climates. Future recommendations aim to extend this research by employing satellite imaging, further feature engineering, novel neural networks, and exploring spatiotemporal models to capture atmospheric dynamics.

Contents

List of Figures

List of Tables

List of Abbreviations

AdaGrad	Adaptive Gradient optimization algorithm
Adam	Adaptive Moment Estimation optimization algorithm
AdaMax	Adam algorithm variation
AI	Artificial Intelligence
AR	Auto-Regressive model
ARIMA	Auto-Regressive Integrated Moving Average model
ARMA	Auto-Regressive Moving Average model
ANN	Artificial Neural Network
CFD	Computational Fluid Dynamics
CL	Crossing Loss metric
CNN	Convolutional Neural Network
ConvLSTM	CNN-LSTM
DAG	Directed Acyclic Graph
DNN	Deep Neural Networks
EEM20	European Energy Markets 2020 Conference
ECMWF	European Centre for Medium-Range Weather Forecasts
GBM	Gradient Boosting Machine
GPU	Graphical Processing Unit
HARMONIE	HIRLAM-ALADIN Research on Mesoscale Operational NWP in Euromed

HCM	Hybrid CNN-MLP
ICT	Information and Communication Technology
IDE	Integrated Development Environment
KDE	Kernel Density Estimation
k-NN	K-Nearest Neighbor
LES	Large-Eddy Simulation
LSTM	Long-Short Term Memory
MA	Moving Average model
MAE	Mean Absolute Error
MET Norway	Norwegian Meteorological Institute
MLP	Multi-layer Perceptron
MSLP	Mean Sea Level Pressure
MSE	Mean Squared Error
NetCDF	Network Common Data Form
NC	Number of Crossings metric
NWP	Numerical Weather Prediction
PCA	Principal Component Analysis
PDF	Probability Density Function
QGAM	Quantile Generalized Additive Model
QR	Quantile Regression
QRF	Quantile Regression Forest
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RH2M	Screen level (2-meter height) Relative Humidity
RMSprop	Root Mean Square Propagation optimization algorithm
RNN	Recurrent Neural Network
RES	Renewable Energy Sources
SE1	Swedish electricity price region 1
SE2	Swedish electricity price region 2

SE3	Swedish electricity price region 3
SE4	Swedish electricity price region 4
SGD	Stochastic Gradient Descent
SLAF	Scaled Lagged Average Forecasting
SP-NN	Smooth Pinball Neural Network
SPinHy-NN	Smooth Pinball Hybrid Neural Network
SVQR	Support Vector Quantile Regression
SVM	Support Vector Machine
SWEA	Swedish Wind Energy Association
T2M	Surface Temperature at 2-meter height
TCC	Total Cloud Cover
U10	Zonal 10-meter height wind component
V10	Meridional 10-meter height wind component
WS10	10-meter height Wind Speed

1

Introduction

Renewable Energy Sources (RES) find an opportunity to produce a transformational change in the energy sector. Despite global disruptive events that showcase the fragility of our current socio-economic system, the agenda to call for action to push resilient environmental-friendly solutions remains, as climate change poses itself as one of the biggest threats of the current generation.

In the context of COVID-19, renewable electricity has not been affected. In fact, during the first quarter of 2020, the global use of renewables grew by 1.5% [1]. Moreover, estimations suggest that renewable electricity generation will rise by 5% in 2020. In particular, wind energy positions itself as a dominant driver in the energy transition. By the end of 2019, Europe increased the total installed offshore wind capacity to 22 GW, connecting more than 5000 turbines, compared to 12.5 GW installed in 2016 [2]. Figure 1.1 shows the growth of the wind energy sector in Europe. In the Netherlands, more than 1.1 GW have been connected so far to commit to the National Offshore Wind Energy Roadmap to 2030, aiming at 11 GW of wind energy by the end of the decade [3].

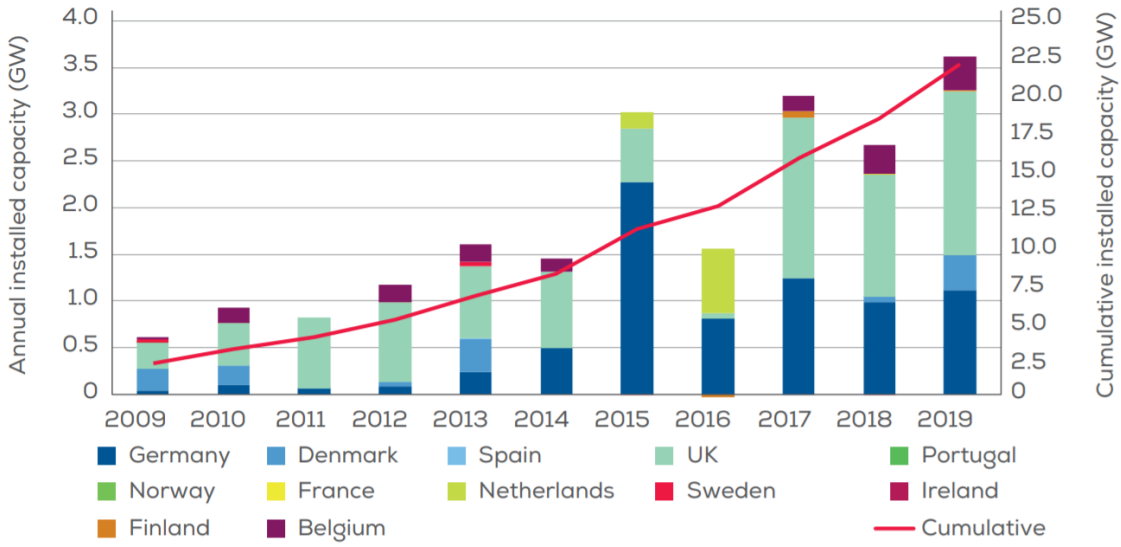


Figure 1.1: Offshore wind installed capacity growth by country [2].

Besides the promising growth of this infrastructure, one of the remaining challenges in the wind energy sector is to deal with the uncertainty of the output production due to its chaotic and intermittent nature. Power system operators still face the difficulty of integrating the variability of wind power into the grid, compromising scheduling, unit commitment, and management of reserve systems [4].

Wind power forecasting serves as a product to facilitate the decision-making of these operators: a tool for risk management in electricity markets [5]. Existing methodologies include deterministic, probabilistic, ensembles, and ramp event forecasts [6]. These are all mature techniques used by operators worldwide, for which advantages and limitations are understood.

Several gaps and bottlenecks are present in these forecasting tools, namely data streams, models, approaches, expertise, and visualization. These issues stimulate the research of new methodologies and frameworks to solve different problems, such as the value of data and the acquisition of relevant variables from Numerical Weather Prediction (NWP) products to improve the accuracy of forecasts [7]. Different strategies have been implemented to fulfill RES forecasting objectives, based on physical and data-driven models [8].

Machine (deep) learning and artificial intelligence (AI) has shown successful results in different fields and applications –including the energy sector–, to promote data-driven decision-making [4]. As computers improve their computation performance and algorithms become increasingly efficient, our society is shifting to an era of energy digitalization [9], where the use of massive Information and Communication Technology (ICT) plays an essential role to the energy transition, subject to the complexity of the system. In the sub-domain of deep learning, Artificial Neural Networks (ANN) are yielding research in an unprecedented way owing to their versatile methodology and the abundance of data, introducing novel optimization techniques, model architectures, and dealing with non-linearity. Furthermore, they are becoming more accessible through friendlier programming environments: simple yet powerful framework packages, attracting professionals from different fields to train their data.

The problem statement is summarized as follows. The energy transition promotes wind energy as a dominant technological driver to mitigate climate change. However, gaps and bottlenecks in power forecasting tools are still present in the value chain or pipelines of these frameworks. A high-penetration of RES challenge power system operators to balance the electricity demand effectively, introducing additional technical and financial risks that jeopardize the security of supply, as this gap for accurate forecasting methods remains open.

The objective of this report is to enhance wind power forecasting employing data-driven models, using different machine learning approaches that capture patterns in meteorological data, applied to different climate regions. The scope of this project compares novel approaches through different types of forecasts, giving special attention to probabilistic methodologies. Remote sensing, on-site measurements or simulation from NWP serve as possible data streams to train these models. Finally, creating a framework as an alternative to the existing ones, underlying its advantages and limitations, which is validated in different climate regions or data sets.

The outline of the report is structured by the following chapters:

- Chapter 2: offers a literature review and state-of-the-art of the two main pillars of this research project, namely deep learning and wind power forecasting.
- Chapter 3: introduces a novel neural network methodology, the Smooth Pinball Neural Network (SP-NN) for quantile regression problems.
- Chapter 4: explains the methodology and the approach followed to create the model for enhancing wind power forecasting, starting from the data stream to the forecast visualization, subject to the scope of this project, and time planning. Moreover, it introduces the programming frameworks (packages) and computational processing power.
- Chapter 5: analyses the data sets of different regions used to validate the created model framework to characterize them.
- Chapter 6: provides the results of the followed methodology, considering the case study. Moreover, it compares the architectures, discussing the feature engineering process and hyperparameter tuning of the model.
- Chapter 7: ends the report, concluding the research project with key takeaways, and provides recommendations for future works based on the learning outcomes of this work.

2

Literature Review

The following chapter aims to provide the literature review explored during the research project. Hence, it provides the necessary theory to understand the terminology and concepts utilized in this report. The outline of the chapter is divided in two main section: Section 2.1 provides a state-of-the-art summary of wind power forecasting. Section 2.2 overviews the field of machine learning.

2.1. Wind power forecasting

Wind power forecasting has been an active research area for the last decades. The interest to enhance forecasting models increases as a higher penetration of wind energy technologies and other renewable sources penetrate the electricity market [10].

Forecasting models in wind energy can be classified in two groups. The first group relies on historical time series of relevant wind data variables. The second group used predicted NWP models as input, making it a two-step forecasting process.

Another classification for wind forecasts is based on the approach [11]: models based on physical methods, conventional statistics, and machine learning, the latter given by the recent success of AI. Moreover, hybrid approaches that combine different strategies have served as a tool for enhancing these models. Usually, a simple method to predict wind is used to benchmark a set of approaches: persistence is one of them [8].

Persistence is a naïve measure defining the forecast value for the next time step (P_{t+1}) to be equal to the actual value (P_t), for which information is available. Despite its simplicity, persistence shows to be more accurate than NWP-based models for time horizons longer than 6 hours [10].

Given both classifications, the first group follows statistical approaches to forecast wind speed or electricity production directly. The second group focuses on building relationships from explanatory variables derived from another meteorological model that simulates wind dynamics to predict power production.

All these approaches have shown successful results. However, the performance varies according to the time horizon. Hence, some models are more suitable for certain cases than others. For instance, short-term horizons are more susceptible to atmospheric dynamics, making it convenient to employ meteorological models to enhance the accuracy of results [12].

Wind power forecasting then can also be classified in terms of the forecasting horizon [13]:

- **Very short-term** forecasting relates to models that predict for seconds to minutes ahead, finding most of their application in wind turbine control and power system frequency control.

- **Short-term** forecasting predicts for hours to days, having applications in Economic Dispatch, reserving units, and the day-ahead electricity (spot) market.
- **Medium-term** forecasting tries to predict for days up to weeks, being relevant for unit commitment and scheduling maintenance labors.
- **Long-term** forecasting relates to months or years time scale, finding applications in wind power planning and power system planning as a conceptual design or proposal to a new project.

2.1.1. Physical models

Physical models require detailed information of the lower atmosphere. It involves converting wind speed given by meteorological services or NWP to wind turbine clusters or wind farms. The conversion of wind speed has to go through an extrapolation process, correcting for hub height, which takes into consideration the terrain: logarithmic or power laws are suitable for this upscaling process. Once a proper wind speed is defined, the wind turbine power curve is used to relate this wind data with the power production [8].

More complex approaches involve Large-Eddy Simulation (LES) or Computational Fluid Dynamics (CFD), serving as an alternative method to adjust the local conditions. However, the complexity of the model increases the computational costs of the model, which depending on the application might be undesired. Some existing physical models for wind power forecasting are Prediktor (developed by Landberg, 2001) and SOWIE (developed by Eurowind GmbH in Germany, 2002) [11].

2.1.2. Statistical and hybrid models

In the statistical realm of wind power forecasting, big data sets are analyzed, ignoring meteorological variables. Therefore, these methods are also called 'black-box' approaches. Statistical models used include Auto-Regressive (AR), Moving Average (MA), Auto-Regressive Moving Average model, (ARMA), and Auto-Regressive Integrated Moving Average model (ARIMA) [8]. Table 2.1 shows an overview of some statistical models used in wind power forecasting.

Table 2.1: Overview of some statistical models for wind power forecasting [11].

Model name	Developer	Locations
WPPT	IMM & DTU	Denmark, Canada The Netherlands, Sweden
Sipreólico	University Carlos III & Red Eléctrica de España	Spain
WPMS	ISET, Germany	Germany
GH Forecaster	Garrad Hassan	Greece, Great Britain and USA
Alea Wind	Aleasoft at UPC, Catalunya	Spain

On the other hand, learning methods involve ANNs, fuzzy logic, Support Vector Machine (SVM), or a combination of methods. Therefore, these are so-called "grey-box" methods, as they are learning from the relationship between the forecasted values (wind production) and historical sequential data [14].

An example of learning methods is weather analogs based on a k-means clustering approach. An analog-based approach based on weather classification consists of predicting future values based on the most similar historical conditions. The use of a k-means method can enhance the traditional statistical tool, integrating images in a multivariable framework. Given this approach, the model can recognize the most similar patterns based on the target values among the training data [15].

Hybrid models combine physical and statistical models to enhance wind power forecasting. An example of a hybrid model is using a deep learning architecture to capture spatial patterns in NWP

data, learning from the real power prediction at a particular point in time that relates this grid-like topology data. Table 2.2 shows an overview of existing learning and hybrid models used in wind power forecasting.

Table 2.2: Overview of some learning and hybrid models used for wind power forecasting [4].

Inputs	Data set	Algorithms
Wind speed Wind power	Global Energy Forecasting Competition (GEFCom) 2014	Sparse Bayesian learning Kernel Density Estimation (KDE) Beta distribution fitting method [16]
Wind speed Wind power	Australian Energy Market Operator (AEMO) 2005	Random Forest Gradient Boosting SVM [17]
Wind speed Wind power	National Renewable Energy Laboratory (NREL) 2006	Ensemble method: wavelet transform partial least squares regression ANN [18]
Wind speed Wind power	GEFCom 2012	Gaussian Processes ANN [19]
Wind turbine data Wind speed Wind power	SCADA wind farm data 2012	K-means clustering, bagging ANN [20]
Wind power Weather forecasts	5 wind farms data Europe	Mutual information Deep auto-encoders Deep belief networks [21]
Day, hour Wind speed Wind direction Temperature Pressure Humidity Active turbines	MADE wind farm ITER Tenerife Spain 2014-2016	Multi-layer perceptron (MLP) with ReLU, Long-Short Term Memory (LSTM) [22]
Wind speed Wind direction Temperature Humidity Pressure	MADE wind farms ITER Tenerife Spain	Feed-forward ANN Convolutional Neural Network (CNN) Recurrent Neural Network (RNN) [23]

2.1.3. Probabilistic forecasting

The original intuition of a forecasting tool is perceived as a single deterministic point, expressing the expectation for the particular time ahead. This traditional methodology is known as **point forecast**. However, a different forecasting methodology exists, namely probabilistic or uncertainty forecasting, which instead of predicting one value at every time step, outputs the complete distribution of the explanatory variable. The distribution of a probabilistic forecast is represented by quantiles [6]. Figure 2.1 compares a point forecast (black) with a quantile forecast.

The basic terminology to assess the quality of probabilistic forecasts relies on forecast skill and sharpness (subject to calibration).

Forecast skill expresses the accuracy of the forecasting model respect to the real value. Since the observed value is unique and the probabilistic model provides a range from a density distribution, it is significant to reward how close these bands are to each other.

The closeness between boundary quantiles is expressed through the **sharpness**. However, forecast sharpness is subject to calibration, which refers to the level of consistency between the distribution (range) of the forecast with the real observed values [24].

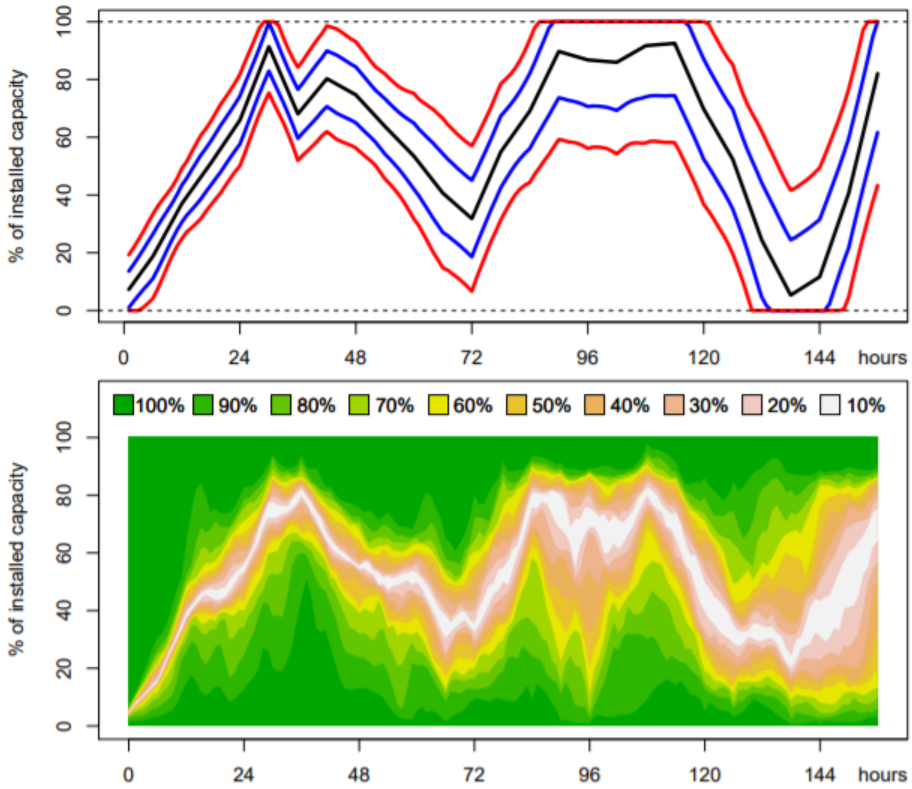


Figure 2.1: Forecasting methodologies: point forecast with bands (top); probabilistic forecast represented by quantiles (bottom) [6].

In comparison with point-value predictions, this methodology provides more information on future events, exhibiting it as a forecasting tool advantage. On the other hand, conventional deterministic wind forecasts are not able to provide enough accuracy, traduced into a complicated decision-making process for those who analyze them. As the precision of these models depends on the time horizon, high uncertainty levels for intermittent electricity production motivated the study of probabilistic tools. Therefore, uncertainty forecasting is getting more attention in developing wind power prediction models, as it turns to be of essential value for power system operators under high penetration of wind energy [13].

The traditional methodology for wind power probabilistic forecasting is based on an essential data stream, namely meteorological ensemble members. The vast amount of information available on these models does not imply, however, that all information of a particular scenario is present. Hence, producing relevant scenarios rely on the auto-correlation of data [6]. On the other hand, there are two main techniques to construct predictive distributions: parametric and non-parametric.

Parametric approach

Parametric approaches consist of assuming the 'shape' of the density distribution, Gaussian being the most common one. Therefore, the distribution is defined by a set of parameters that build the pre-existing function. As an example, Gaussian distributions are defined by two parameters: mean (μ) and deviation (σ) parameters. Several methods exist to determine these estimators, such as non-linear time series and the use of AI [13]. Active research in this area is focused on the shape assumption of the distribution, estimation, and evaluation of location and scale parameters [13].

Non-parametric approach

Non-parametric approaches, on the other hand, do not make any assumption of the shape of the distribution, or it is "shape-free". A Probability Density Function (PDF) is estimated discretely using a finite

number of points to interpolate between them and fit a function. Popular non-parametric approaches include quantile regression (QR), KDE, and AI. Aside from historical data, forecasts information is also necessary for this approach [13].

By comparing both approaches, parametric solutions offer a more simple methodology, depending on a few parameters and equations being more simply, translating in lower computational costs. However, the *a priori* assumption of the shape is not always a reasonable one, as the nature of wind does not resemble a Gaussian or any particular shape. Furthermore, complex atmospheric dynamics produce a change of these shapes in time. This phenomenon influences directly the accuracy of the parametric model. Therefore, without the need to make assumptions, a non-parametric version is preferred to facilitate the modeling process. On the other hand, the computational costs are higher, requiring many densities to be estimated. In cases, a particular model has to be trained for each quantile of the distribution. One famous non-parametric approach for wind power probabilistic forecasting is adaptive re-sampling [25], which consists of building an empirical distribution for similar conditions.

The development of forecasting based on AI methods brought research into wind power probabilistic forecasting. The introduction of neural networks and machine learning in this field allowed to model architectures in which the outputs are defined as the predictive quantiles. AI models train the data by minimizing the cost function. Moreover, the cost function in non-parameter approaches can be directly linked with the evaluation metric of the predictive interval. Therefore, these methods play a big role in the development of probabilistic forecasting methodologies, as wind power probabilistic forecasts are difficult to evaluate. Some scoring functions have been proposed to assess the quality of these models compared to deterministic forecasts. Furthermore, probabilistic scoring rules require a way to reward both the skill and sharpness of these predictive distributions [13].

2.1.4. Challenges of wind power forecasting

Forecasting models are reliable tools with significant value across its complete chain, from data acquisition to data visualization. Nonetheless, some pending challenges to enhance wind power forecasting have to be tackled when proposing new frameworks and methodologies [26], enumerated as follows:

1. Data cleansing: real-world data for energy forecasting is not free of error and mistakes. Using low-quality data have a direct impact on the prediction, as incorrect measurements or values from explanatory variables bias the decision-making.
2. Probabilistic forecasting methodologies: many fields within energy forecasting have reached different maturity levels in their methodologies forecasting. Finding suitable frameworks and identifying robust data processing pipelines, subject to the application and domain-knowledge, is relevant to mitigate the risks of variability in the renewable energy sector.
3. Forecast mix: using a combination of forecasts usually brings more accurate results and facilitates decision-making, especially in point forecasting. The trade-off between computational time and the complexity of the models, including ensemble models, has to be addressed to enhance their robustness and obtain the optimal methodology for a particular application.
4. Integration: probabilistic forecasting processes can be divided into several components in their pipeline or chain, namely data, assumptions, modeling, and visualization of the results. An optimal segment does not mean optimal behavior throughout the whole process. An optimal outcome from one component may not be the optimal one for the entire process. Therefore, following a holistic approach to combine these aspects and produce the best methodology represents one of the main goals of researchers in the field.

Figure 2.2 provide a scheme of the gaps and bottlenecks in the current forecasting value chain, provided by Smart4RES consortium.

In the illustration, five main bottlenecks can be visualized in red warning signs. These relate the connecting points in the value chain of forecasts, contrasting the challenges with the business gaps from a pipeline perspective.

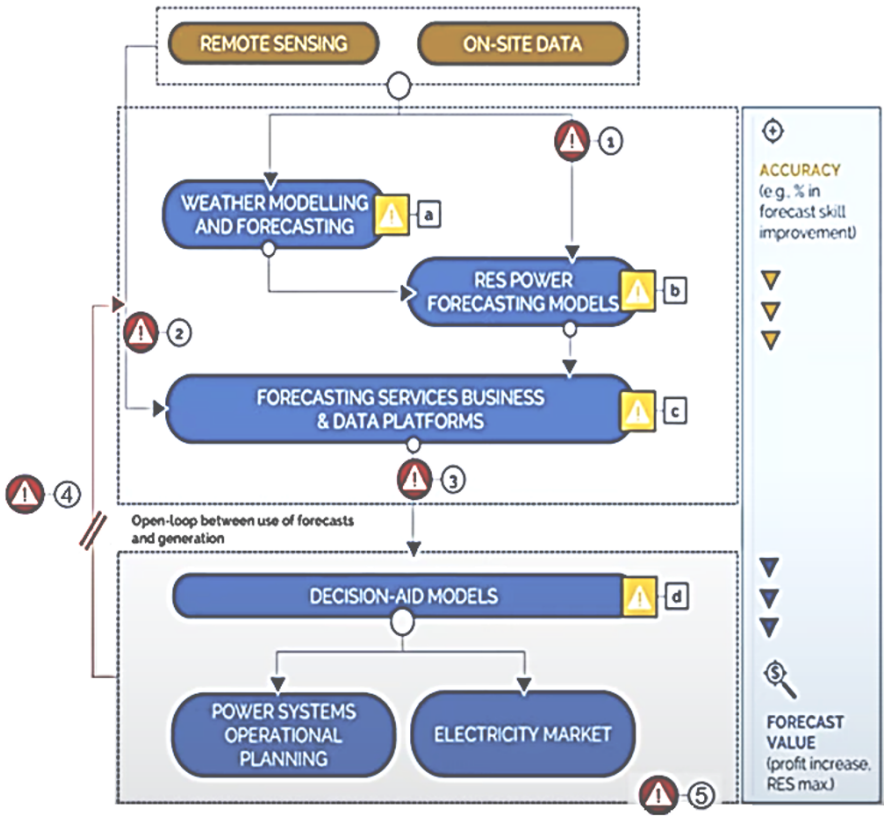


Figure 2.2: Gaps and bottlenecks of forecasting tools [7].

The first one relates to the lack of meaningful open data, generated by privacy issues and the value of data itself. This directly affects renewable forecasting models.

The second one is concerned with the lack of price incentives to share data. The associated costs break the forecasting business service model, despite the growth of remote sensing technologies.

The third aspect is the lack of standardization. Forecasting tools provide a broad spectrum of services given by the nature of models, thus they are difficult to generalize. As a result, the task of validating these instruments as decision-making models is laborious, forcing clients to diversify their portfolios to accumulate different sources of information.

The fourth aspect is the open-loop between the generation of forecast and their actual use. The end-user has little influence on the effect of the development of these models. Therefore, a communication channel is still missing to better cope with the expectations of forecasters and decision-makers.

The last aspect considers the need for business cases to showcase the value of data. This can be traduced to understanding the value of uncertainty.

Understanding the challenges of variable RES forecasting, machine learning algorithms represent an alternative to this overcome this situation. Recent research prefers to use these methods because they can adapt to changing patterns recognizable inside the data available, producing models based on specific information instead of generalizations. The following section provides an overview of machine learning, necessary to understand the approach followed during this research project.

2.2. Machine Learning

Machine learning, deep learning, and AI have made substantial improvements in various fields such as speech recognition, computer vision, and machine translation [27]. The ability of deep learning methods to capture intricate patterns in high-resolution data makes them successful at finding solutions to complex problems [28].

Researchers started exploring its applicability to model spatiotemporal dependencies [29], as this type of data can be found in different sciences: social studies, economics, biology, and naturally, meteorology. In consequence, a diverse community has devoted efforts to exploit the capabilities of these pattern recognition tools [30].

Machine learning can be classified into three groups based on the type of task: supervised, unsupervised, and reinforcement learning. Each task relates the type of application and goes beyond the data stream, implying that the same data set could serve different purposes.

Supervised learning corresponds to models where the output data is known, being the most common form of deep learning [28]. This task is further classified into two types: regression and classification. Regression consists of providing an output with infinite possible outcomes, while classification has a fixed set of outputs [31].

Unsupervised learning makes use of unlabeled data [31], organizing or categorizing it to detect features or group relationships from the data set. Clustering through k-means and Principal Component Analysis (PCA) for dimensionality reduction are common methods applied for this type of task [32]. It is common to confuse classification with unsupervised learning: the difference lays in the idea that classification is aided by predefined labels or tags, while clustering organizes the data without making an *a priori* assumption.

Reinforcement learning rewards particular signals, namely the machine learns how to map situations based on a set of actions to maximize such reward. This idea resembles the positive reinforcement concept from psychology, where a subject associates actions based on the stimulus of the recompense. However, these rewarding actions are not predefined in advance, but instead, the algorithm has to discover what yields the highest compensation by trial and error [33].

2.2.1. Artificial Neural Networks (ANN)

ANN are inspired by the way our neurons work. Dendrites bring in the information processed in the nucleus of the cell, and transmits the information through the axons, which is received by other neurons across a synapse [32]. Analogous to the neural system, the structure of an ANN is composed of several connected layers that contain units. These units are activated based on the information they receive from the previous layers [31]. A basic ANN architecture is shown in Figure 2.3.

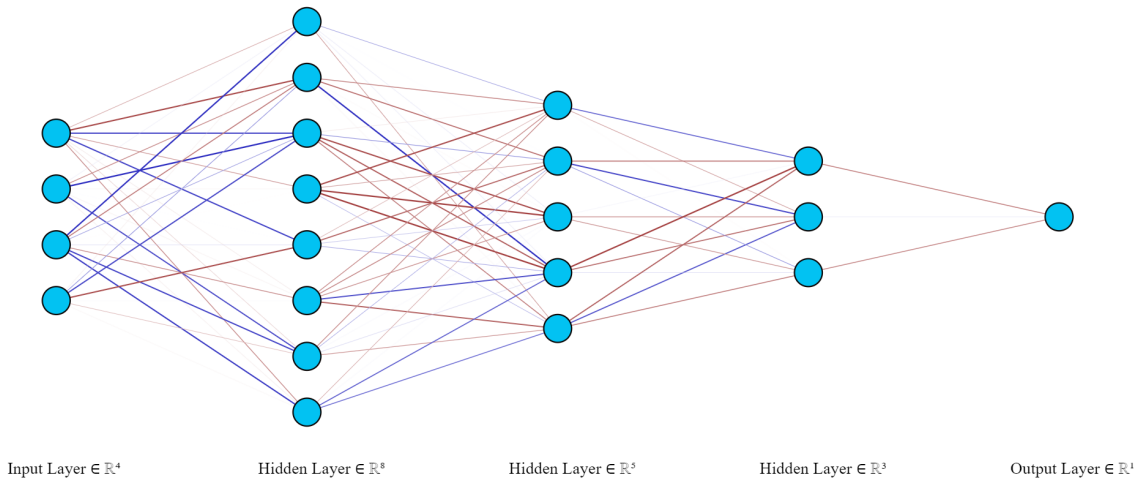


Figure 2.3: Architecture of an Artificial Neural Network (ANN).

The structure of the architecture is described employing the following terms:

The first layer is called the input layer, carrying the data stream features. The feature is a measurable variable used to characterize the problem [32].

The last layer is called the output layer, providing the results subject to the type of task [32]. For supervised learning, the output label computes the classification label or value from a regression problem. In the case of unsupervised learning, the output provides an encoded version of the data, so a new representation of the input data.

Between the input and output layers, hidden layers are located. These layers recognize patterns and communicate the information to subsequent layer units [32]. The amount of hidden layers depends on different aspects, such as the application and computational power. Hence, it is defined by the machine learning engineer as a design choice [31]. Architectures with many hidden layers receive the name of Deep Neural Networks (DNN), the realm of deep learning.

A vectorized notation can describe the computation process. It consists of an addition and multiplication process, following linear algebra operations, transformed into a non-linear system through the activation function. Equation 2.1 [31], describe two-step process for every unit in the architecture:

$$\begin{aligned} z^{[l+1]} &= W^T \cdot y^{[l]} + b \\ y^{[l+1]} &= g(z^{[l+1]}) \end{aligned} \tag{2.1}$$

The first step expresses a linear operation, where a matrix of weights (W) is updated through an optimization algorithm to fit the data. This matrix is multiplied by the output units of the previous layer. A bias vector term (b) is added to this matrix multiplication, resulting in an output vector of the same shape (z). In some cases, the bias factor is neglected [28].

The second step consists of applying an activation function ($g(z)$) to flatten the output between a range of values, resulting in the values of the following layer ($y^{[l+1]}$). This process is repeated from the input layer until the last output layer, which provides the final output of the task.

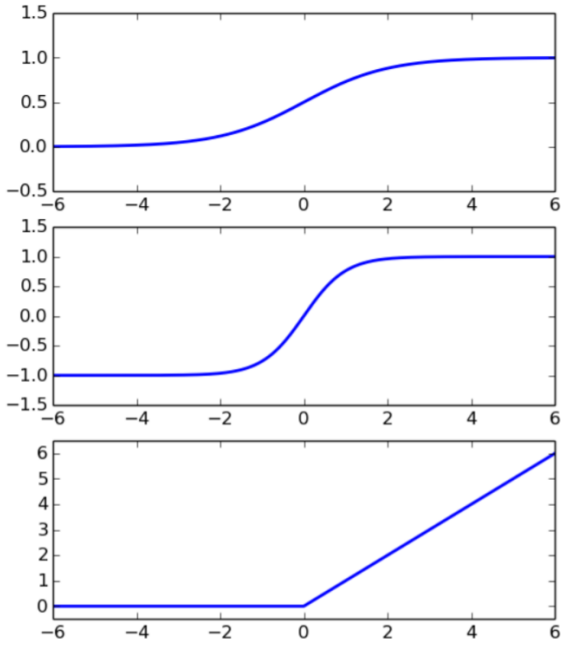
Activation functions

The activation function ($g()$) introduces non-linear properties to the ANN, which plays an essential role in training deep learning models [34]. The purpose of introducing non-linear functions is to break the linearity of the operations, otherwise, the model would consist of purely linear relationships, meaning that it would not be able to capture complex patterns. Also, the lack of non-linear components removes the necessity for hidden layers. Therefore, non-linear functions discover relevant information in these layers. Frequently used activation functions are shown in Figure 2.4.

Common activation functions are the hyperbolic tangent (tanh) and the sigmoid function. The sigmoid function transforms the input domain $(-\infty, \infty)$ into the range domain (0, 1). On the other hand, the tanh function maps the same input domain onto a slightly different domain (−1, 1). In other words, these functions are flattening the input, replicating an ON/OFF switch. For example, the neurons under an OFF state, are not transferring information to deeper layers.

Recent methods are moving towards novel non-linear activation function, being the Rectified Linear Unit (ReLU) function a popular choice, given two main advantages:

1. Deals with the so-called “exploding/vanishing gradient” problem [36]. Traditional activation functions are subject to optimization functions that rely on the calculation of gradient or derivatives, as will be explained in the following subsection. However, these functions have segments where the derivative is zero (see Figure 2.4), meaning there is no learning process where the matrix of weights is unable to update its values. ReLU manages this by ensuring learning updates, regardless of the range in which the activation function falls. A variant of this function is the leaky ReLU version.
2. Accelerates the convergence speed if combined with an appropriate optimization algorithm. This is a derivation of the first point: as the slope of the ReLU ensures learning, the weights of the matrix (W) are always updated in every iteration.



Sigmoid

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Hyperbolic Tangent

$$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Rectified Linear

$$\phi(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases}$$

Figure 2.4: Overview of common activation functions [35].

Optimization of Neural Networks

Training consists of updating the matrix of weights (W), subject to an objective function, also known by its variations: lost or cost function.

The algorithm to minimize this cost function is called backpropagation. In essence, once the output is computed through the feed-forward propagation algorithm using a particular weight initialization matrix, the inverse process occurs. The weights are updated by comparing the target value with the prediction, then calculating their respective derivatives at every layer, starting from the output until reaching the input layer, going through all the hidden layers. Next, the ANN computes a new output based on the actualization of these weights, repeating this process to minimize the cost function as desired. Consequently, the backpropagation algorithm is a way to memorize the training data set, as large iterations tend to make the training error close to zero [32].

Equation 2.2 shows the general form to compute the gradient in neural networks following the backpropagation algorithm [37]. The derivation and intuition of the backpropagation algorithm can be found in Appendix A.

$$\nabla_{ij}^{(l)} = \sum_{k} \theta_{ki}^{(l+1)} \delta_{k}^{(l+1)} * \left(a_{i}^{(l)} \left(1 - a_{i}^{(l)} \right) \right) * a_{j}^{(l-1)} \tag{2.2}$$

Figure 2.5 illustrates the feed-forward pass and backward propagation flow in a particular node or unit of the neural network. The optimization algorithm can be summarized as follows [32]:

- Step 1 - Forward propagation: computing predictions from the input layers to the output layer using a particular random initialization of the matrix of weights (W).
- Step 2a - Backward propagation: computing errors (δ) from the resulting output in Step 1.
- Step 2b - Backward propagation: computing the gradient of the cost function using the results from Step 1 and 2a (predictions and errors), following Equation 2.2.

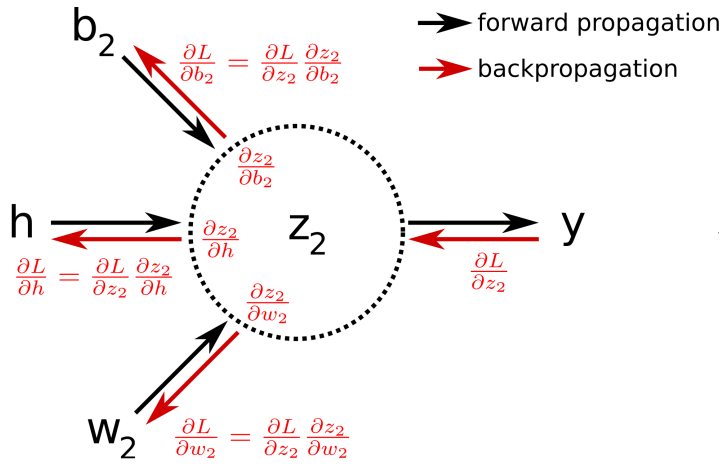


Figure 2.5: Feed-forward pass and backward propagation in a unit of the neural network following the chain rule [38].

- Step 3 - Optimization technique: obtain new weights to build a new matrix (W) that computes new predictions, going back to Step 1.

Many loss functions can be evaluated to compare the output value of the model with the target value. The following functions represent the most commonly used ones for regression problems:

- Mean Absolute Error (MAE, or L1-loss).
- Mean Squared Error (MSE, or L2-loss).
- Huber loss, a smooth version of L1-loss.
- Quantile loss, used for probabilistic forecasting.

Figure 2.6 compares these objective (cost) functions.

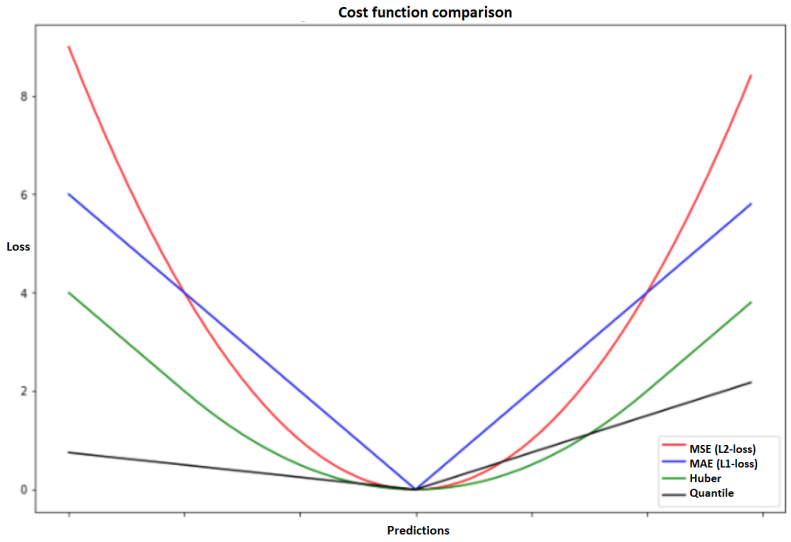


Figure 2.6: Comparison of common loss functions.

The Huber loss includes a smoothening parameter (δ) to deal with the original non-convex L1-loss function. Converting the L1-loss function into a convex optimization facilitates the convergence process of the algorithm, as the L1-loss by itself has a non-differentiable point at zero.

The quantile loss, also similar to L1-loss, includes a quantile parameter (τ) accounting for an under(over)estimation criterion. This function accommodates output in such a way that lower quantiles parameters, i.e. quantiles below the median (Q50) are rewarded by underestimating the target value, while upper quantiles are rewarded by overestimation.

There is an on-going extensive research regarding optimization techniques to integrate it on the backpropagation algorithm.

Having the classical gradient descent technique as a starting point, the first variant is Stochastic Gradient Descent (SGD), which randomly selects a subset of the training data set to calculate the partial derivatives [32]. This strategy reduces computational time significantly, as fewer calculations are required. However, the disadvantage of SGD is the trouble of finding local optimal points in cases where the steepness of surfaces between different axes is not equally distributed. [39].

Motivated by this, Qian (1999) discovered that using momentum can help to speed up this process, adding a γ fraction component to the updated gradient [40], resulting in the Adagrad method [41].

In 2014, Diederik Kingma (University of Amsterdam) and Jimmy Lei Ba (University of Toronto) introduced a revolutionary algorithm in the field, intended to solve ever more complex applications: the so-called Adam optimizer [42]. Adam is a combination of two popular methods: AdaGrad and RMSProp. RMSProp is an adaptive learning rate method developed by Hinton (2012). The creators of the Adam algorithm also proposed the AdaMax algorithm [42]. The AdaMax uses an L-infinity norm instead of the L2-norm, to enhance a stable optimization algorithm.

Regularization of Neural Networks

.

Regularization consists of a set of strategies developed to rectify the data in such a way that enhances performance in the validation and test sets. Therefore, there is a trade-off between improving the error of the test set at the expense of the training error [37]. The end goal is to produce a generalized model, beyond the information trained.

Two concepts are used to assess the need for regularization: underfitting and overfitting.

Underfitting refers to a model that is not able to characterize the problem because the training error is too high, hence the training data still have many points outside of the fitted model. Underfitting is also defined as a bias problem.

Overfitting refers to a model that is not able to characterize the problem because the cross-validation error or test error is high when the training error is low. In other words, the fitted model is more intricate than it is in reality. Overfitting is also defined as a variance problem, where the training data set has been memorized but fails to generalize for new examples.

The goal of a machine learning engineer is to produce an appropriate fit, which means not underfitting, such that the training error is too high, nor overfitting, such that it over complicates the problem and produces still a high test error. Depending on the case, regularization techniques are introduced in the model to correct both situations. Figure 2.7 illustrates the underfitting and overfitting problem, as displays an ideal scenario. Figure 2.8 shows a plot illustrating the optimal complexity of the model, highlighting the bias and variance errors.

A basic regularization technique is to include a penalty factor in the objective function. This penalty is linked to the weights, preserving the important ones as the least relevant ones have a high cost, undesired for obtaining local minima. Equation 2.3 illustrates the regularization term introduced to the objective function of a linear regression problem [37].

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right] \tag{2.3}$$

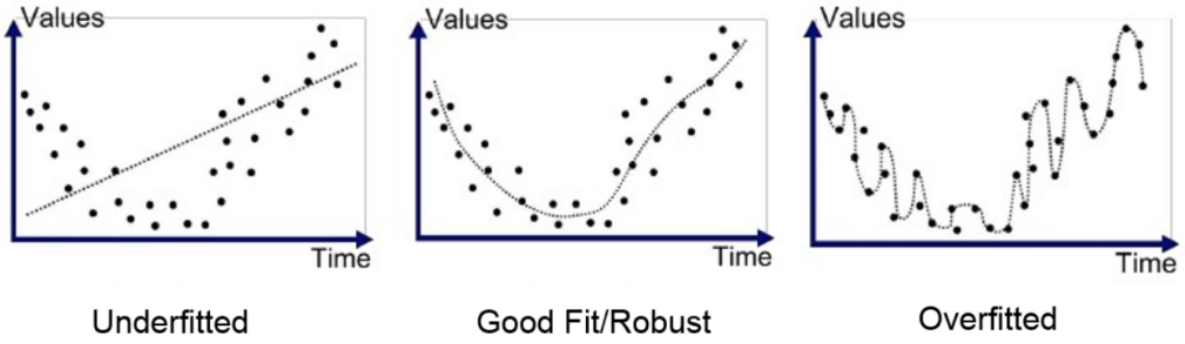


Figure 2.7: Different fitting model scenarios: underfitting (left), robust model (center) and overfitting (right).

The parameter λ is the penalty factor, which is multiplied by the square of each weight. This term is added to the basic objective function, affecting the minimization criteria. Defining λ depends on the judgment of the modeler, in an effort to produce a robust (proper) fit.

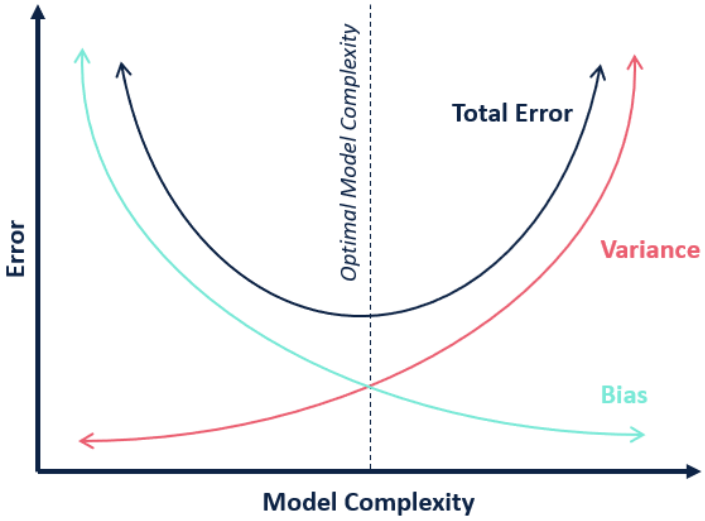


Figure 2.8: Bias and variance trade-off.

Dropout is another strategy, mostly used in DNNs. It consists of randomly turning ON/OFF units across the architecture. This strategy aims to generalize the model, even under circumstances in which some patterns are not fully captured by it as the units are unavailable. Additionally, it pushes the optimization algorithm to find basic the most basic patterns from the data without tackling intricate ones, as this might result in overfitting. The dropout ratio is a hyperparameter that randomly determines the number of units turned off or ”dropped out” from the algorithm. Figure 2.9 illustrates the process of dropout in a neural network architecture.

2.2.2. Convolutional Neural Networks (CNN)

A particular type of ANN architecture is the CNN. These are mainly used in the field of computer vision, given its remarkable ability to capture and extract spatial patterns in grid-like topology data [31]. Similar to ANNs, they contain multiple layers, designed to progressively learn higher-level features. Famous CNN architectures are LeNet-5, AlexNet, and VGG-16 [32]. Figure 2.10 shows an AlexNet CNN architecture.

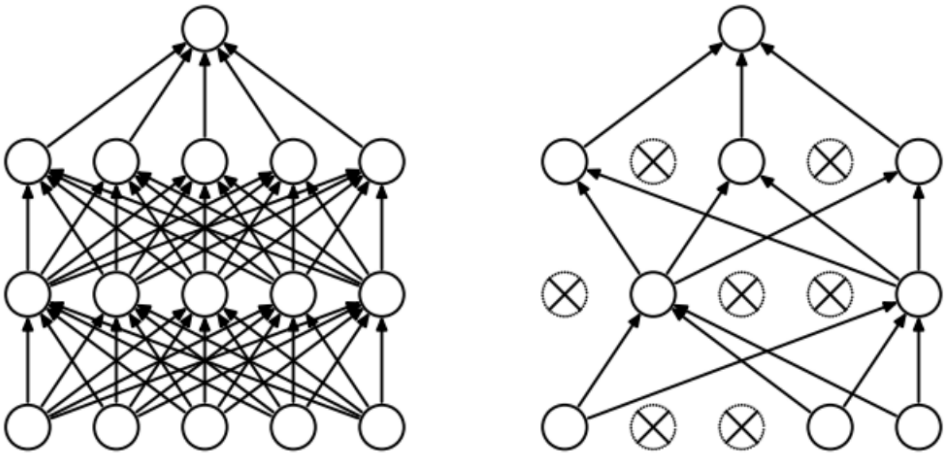


Figure 2.9: Dropout strategy in a neural network architecture: (a) standard neural network (left); (b) applying dropout (right).

Traditionally, images are used as input for these neural networks. The first layer can contain multiple channels, also called feature maps. In the case of images, it typically denotes the use of red-green-blue (RGB) values as three feature maps. Furthermore, it makes use of three operations, namely the convolution layer, sub-sampling layer, and fully-connected layer.

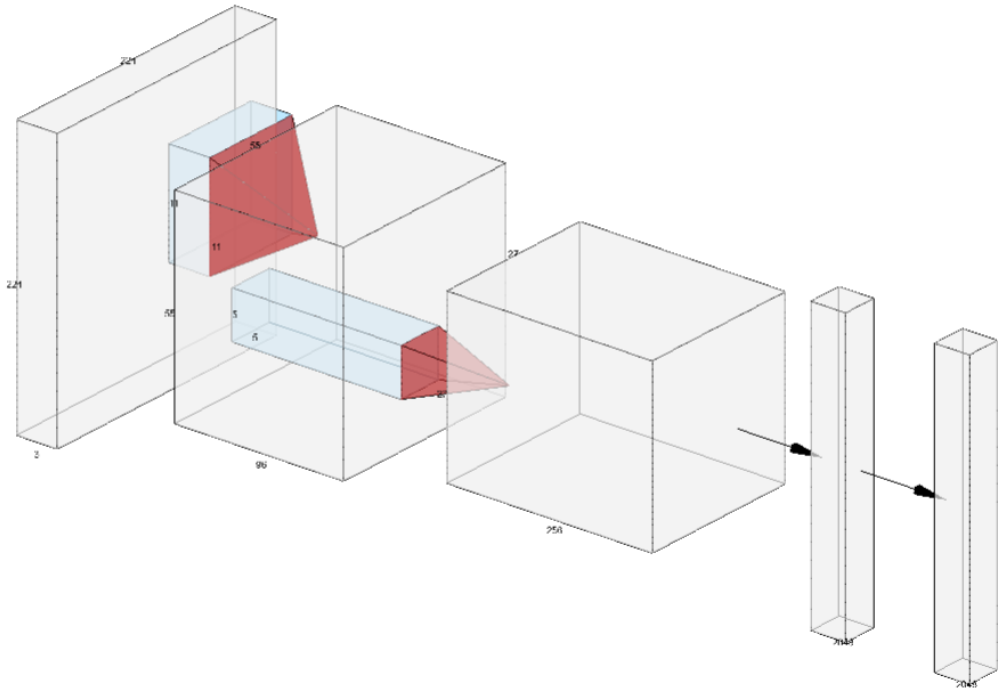


Figure 2.10: AlexNet: Convolutional Neural Network (CNN) architecture.

Convolution layer

The convolution layer extracts features from the input image using multiple filters or kernels. These symmetric-size kernels are small compared to the input data, moving as a sliding window through the input feature map covering the whole grid. Applying element-wise multiplication, the value in the output layer is computed. A visual example of the convolution operation is shown in 2.11.

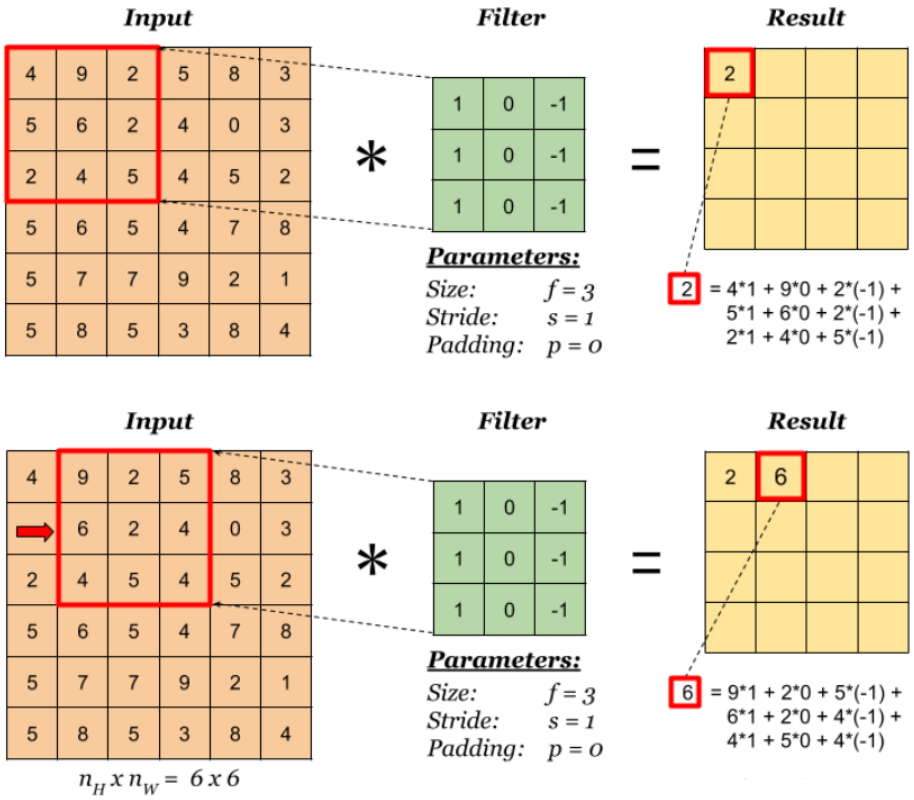


Figure 2.11: Convolution layer operation example [43].

The new feature map is determined by element-wise multiplication, where the kernel window slides to the neighboring cells given the stride parameter. Stride determines the shift or number of positions that the window moves to calculate the following cell. Moreover, to counteract the shrinkage of the feature map, padding can be used. Padding refers to an addition of zero-value cells around the feature map.

The amount of filters used to extract patterns is a design choice, making this value another hyper-parameter, next to the size of the kernel. Every value or cell inside the kernel is optimized using the optimization techniques described in subsection 2.2.1.

Sub-sampling layer

The function of the sub-sampling layer is to take average or maximum values from a window cell, multiplied by a trainable scalar. To avoid overlap, a common practice is to match the size of the window cell with the stride (shift). The operation reduces the size of the output cell, keeping the number of channels or feature maps. Therefore, sub-sampling results in lower resolution filters. However, alternating the convolution layers and the sub-sampling layers is an approach to preserve the number of features in order to enhance robustness from variability [44]. Additionally, sub-sampling layers reduce the computational cost of the model. The use of averaging in machine learning is called Average Pooling [44], while taking the maximum value corresponds to Max Pooling [45].

Fully-connected layer

A fully connected layer restructures the architecture to a basic ANN version, as shown in Figure 2.3. The last layer of spatially-connected features is ”flattened” in a way that the grid-like topology characteristic is lost. The Flatten operation promotes the dimensional reduction of the data when the feature maps have shrunken enough, so the number of filters is comparable to the number of features or pixels in the feature map. The subsequent layers reduce the number of nodes to provide a final output.

2.2.3. K-means clustering

K-means clustering is a simple machine learning algorithm. The objective of the method consists of saving training examples to group them in *k* clusters. The flexibility of the approach allows its implementation for both unsupervised -clustering- and supervised learning. In the latter, it can be used also for regression problems, if a continuous target value is available.

This strategy should not be confused with k-Nearest Neighbor (k-NN), which is a classification algorithm that employs a voting system based on the output labels. In the case of k-means, the grouping is based on a similarity measure or distance function. The algorithm has been used in pattern recognition for non-parametric weather prediction, directly related to analog-based approaches.

The algorithm process of this method can be described as follows [46]:

- Step 1: specify a k-value for the number of groups or clusters.
- Step 2: assign by randomness each sample to a particular cluster.
- Step 3: compute the cluster centroid coordinates.
- Step 4a: determine the distances of each data point to the centroids.
- Step 4b: re-assign each point to the closest cluster centroid based upon minimum distance.
- Step 5: calculate cluster centroids again.
- Step 6: Iterate steps 4 and 5 until reaching global optima, where no improvements are possible, or there is no switching of data points from one cluster to another.
- Step 7: optimize the k-value based on the elbow method.

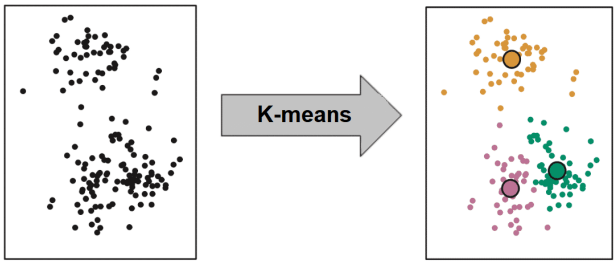


Figure 2.12: K-means clustering algorithm [46].

Figure 2.12 illustrates the grouping process of this algorithm. The original data is clustered based on the algorithm described above, resulting in k-clusters with their respective centroids, highlighted with black borders. In this case, K = 3 is defined to cluster the data.

A distance function has to be defined to maximize the similarity between samples. The most common approach is to use Euclidean distance, which is expressed by the following formula:

$$d = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.4}$$

Equation 2.4 takes the distance between two points (p, q) connected by a straight line. Moreover, this expression is generalized for i-components, allowing for multiple dimensions or variables. This expression can be expanded to include weights or corrections based on the nature of the data.

In reality, the last step of the k-means process is outside of the algorithm. However, it is important to determine optimal k-values. A common approach is to use the elbow method. Figure 2.13 illustrates the selection of the optimal k-value. From the illustration, the error diminishes with increasing the number of k-clusters. The optimal case is located in the red dot. This red dot can be defined as the point in which the k-value stops decreasing the error drastically by increasing its value.

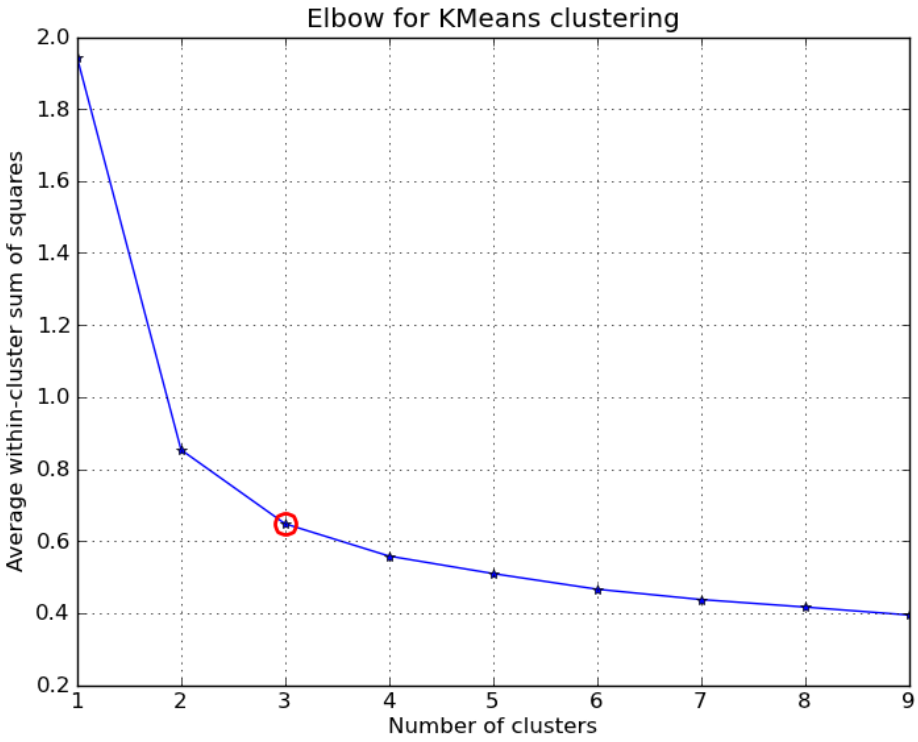


Figure 2.13: Elbow method to select optimal k-value of clusters [47].

The advantages of this model are seen in its simplicity, being able to learn from data with low computational effort. Moreover, no assumptions have to be made; a single k-group value has to be optimized. Last, it is versatile for different learning tasks. On the other hand, the main disadvantage is seen when the number of examples or the amount of features increases. Hence, for big data problems, it can result in a computational prohibitive tool.

3

Smooth Pinball Neural Network Framework

The following chapter describes a deep learning-based probabilistic forecasting framework: the Smooth Pinball Neural Network (SP-NN). This novel architecture was first proposed by Kostas Hatalis, Alberto Lamadrid, Katya Scheinberg, and Shalinee Kishore in 2017 [48].

The outline of the chapter consists of four parts. First, it provides an overview of this particular machine learning architecture. Second, it explains the custom loss function employed in the optimization algorithm. Third, it introduces the quantile cross-over problem, showing the way the model corrects this inconsistency. Finally, the architecture parameters and the original structure are shown.

3.1. Framework overview

The SP-NN was inspired in the traditional Support Vector Quantile Regression (SVQR) model [48, 49]. This model was developed for probabilistic problems, introducing a combination of the Huber loss (smooth L1-loss) and quantile (pinball) loss, following a non-parametric approach. According to the researchers [48], some features of the model include:

- Introduction of a custom objective function for neural networks;
- Strategy to deal with the quantile cross-over problem;
- Integration of multiple neural networks for probabilistic wind forecasting applications;
- Improvement of the skill, reliability, and sharpness over various benchmarks.

Figure 3.1 illustrates the algorithm process of the SP-NN.

3.2. Loss function

The core of this neural network lies in the customized objective function. The so-called smooth pinball loss function results from a combination between the Huber loss (smooth L1-loss) and the quantile loss. An approximation of this loss is described in Equation 3.1.

$$S_\tau = \tau \cdot u + \alpha \cdot \log(1 + \exp(-\frac{u}{\alpha})) \tag{3.1}$$

In Equation 3.1, τ represents the quantile target, while α expresses the smooth parameter. Additionally, u reflects the difference between the predicted value and the real value. The reasoning behind

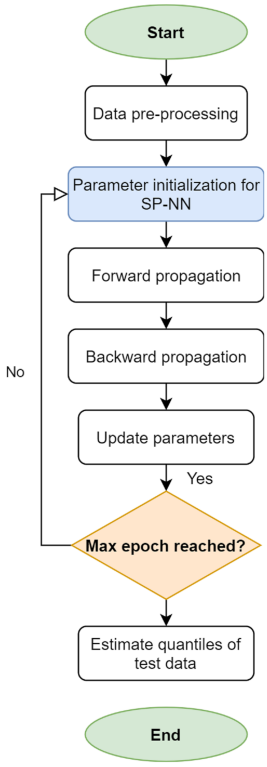


Figure 3.1: SP-NN optimization algorithm [49].

the smooth parameter obeys the same idea of the Huber loss: dealing with the non-convex segment of L1-loss and formally treat the problem as a convex optimization case. The final expression of the minimization problem is:

$$\min_{W,b} \frac{1}{N} \sum_{t=1}^{N} S_{\tau,\alpha} \left(y_t - \hat{q}_t^{(\tau)} \right) \tag{3.2}$$

Equation 3.2 refers to the optimization of the weights (W, b), based on Equation 3.1. The difference is that for the case of QR, the latter evaluates the difference between the predicted quantile value and the actual value, considering every target parameter.

However, it is important to clarify that in current Neural Network packages (e.g. TensorFlow, Keras), the implementation of advanced and state-of-the-art optimization algorithm such as Adam, easily deal with non-convex functions [42]. Despite the powerful optimization algorithms, a formal approach to express the cost function as a traditional non-convex optimization problem is presented.

The pinball loss function serves as a useful metric for probabilistic forecasting, being able to penalize the overestimation and underestimation of particular quantiles. To better understand the nature of this formula, an example for estimating a single quantile is given: when a prediction lies above a reported quantile, such a lower bound Q10 (or decile 1), the loss computed results in the difference from the estimate multiplied by the correspondent target 0.1 value of the quantile. Otherwise, by falling below the reported value, the loss is calculated as the same difference multiplied by one minus its probability target (0.9 in the case of Q10). Therefore, the pinball loss function penalizes low-probability quantiles more for overestimation than for underestimation and vice versa in the case of upper quantile targets. This represents a simple yet elegant way to reward for underestimation and overestimation, according to the evaluated quantile.

3.3. Quantile cross-over

One challenge of non-parametric forecasting methodologies in deep learning is the quantile cross-over [49]. It happens when the prediction output values for higher quantiles are smaller than lower quantiles (e.g. Q20 < Q10). This behavior becomes common when explanatory variables are heteroscedastic [30]. Naturally, these estimations are undesired, as they do not follow the nature of a probability distribution function, hence it reduces the reliability of the forecast. To tackle this situation, a penalty factor has been applied in the objective function to stimulate particular local minima, similar to the idea of reinforcement learning, leaving the algorithm to learn by itself which scenarios to repeat. The penalty factor summed to the objective function is described in Equation 3.3:

$$penalty = \kappa \max[0, \epsilon - (q^{<\tau-1>} - q^{<\tau>})]^2 \tag{3.3}$$

From Equation 3.3, κ represents the penalty factor, while ϵ represents the least amount that two quantiles should differ between them. Therefore, if higher quantiles are greater than smaller quantiles, the penalty becomes zero. Otherwise, a high penalty is applied, thus omitting that particular local optimum of the original objective function.

The procedure is repeated by comparing the differences of all consecutive quantiles for the target quantiles considered. Including this in the original loss function converts this methodology in a multitask learning approach, in which the algorithm is learning to capture patterns in the data and simultaneously avoids inappropriate results in the quantile estimation.

A way to evaluate the cross-over problem for probabilistic models is shown in [30]. In this paper, they use deep learning to produce a joint mean and QR for spatiotemporal problems. They addressed the cross-over problem and assessed the quality of their model by introducing two relevant metrics: the crossing loss (CL) and the number of crossings (NC).

The CL is defined as in Equation 3.4:

$$\mathrm{CL} = \sum_{i=1}^{N_{\text{pred}}} \sum_{j=1}^{J-1} \max\left(0, \hat{q}_i^{(\tau_j)} - \hat{q}_i^{(\tau_{j+1})}\right) \tag{3.4}$$

Equation 3.4 represents the absolute error between two consecutive quantiles, assuming that the desired behaviour is $\tau_{j+1} > \tau_j, \forall j \in 1, \ldots, J-1$. This metric serves as a calibration instrument to improve the sharpness of probabilistic frameworks, especially in deep learning approaches. Furthermore, it validates the multitask learning approach of the loss function, enforcing coherence and consistency.

The NC metric counts the cross-over incidences, regardless of the magnitude of the crossing. Therefore, it serves as a metric to produce a sensitivity analysis of the ϵ parameter from Equation 3.3. As a consequence, optimal values for this equation could be determined by analyzing this value. The objective is to develop a methodology that has a negligible CL, subject to accuracy.

3.4. Architecture parameters

For this novel neural network architectures, additional parameters have to be trained to tune the model subject to the application. Therefore, Table 3.1 shows an overview of the defined parameters, alongside a short description, based on the equations shown in this chapter.

These parameters have to be tuned next to the traditional hyperparameters of a conventional neural network. The tuning depends on the application, accuracy, and sharpness of the results. For example,

Table 3.1: Additional parameters of the SP-NN.

Parameter	Name	Description
κ	Penalty factor	Value to penalize for quantile cross-over
ϵ	Quantile margin	Minimun margin between quantiles
α	Smoothening parameter	Convexity level of the loss function

increasing the quantile margin parameter (ϵ) reduces the sharpness of the forecast. The same idea can be derived for the penalty factor (κ), as it directly influences the objective function. The higher κ, the less sharp the forecast becomes.

In the case of α, the difference could facilitate faster convergence of the algorithm. However, an appropriate optimization algorithm can deal with the non-convexity of the optimization problem. Hence, under a robust optimization setting, this parameter does not have greater influence.

Figure 3.2 illustrates a basic architecture of the SP-NN. It consists of a traditional ANN, differentiated by the number of output nodes, which corresponds to the number of quantile targets. Every node is optimized by a particular objective function, using the respective target parameter (τ) from the general expression in Equation 3.1.

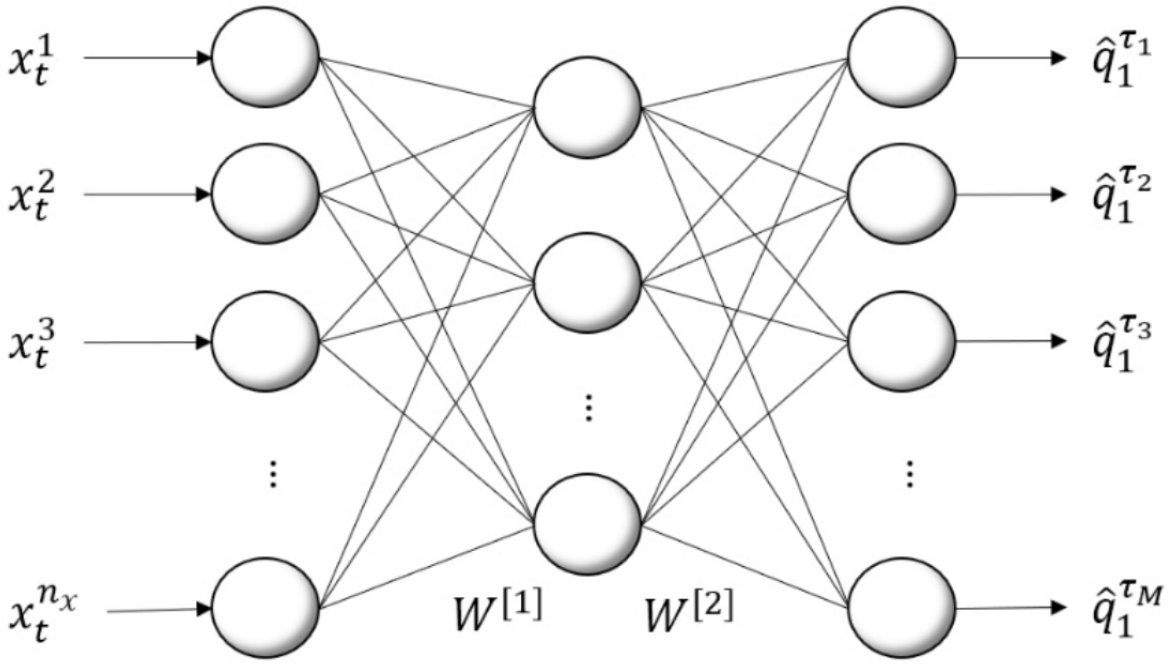


Figure 3.2: Basic architecture of a SP-NN with quantile output nodes [49].

4

Methodology

The following chapter discusses the methodological approach followed during this research project. Based on the problem statement in Chapter 1, the investigation is focused on used state-of-the art machine (deep) learning methods to enhance wind power forecasting tools. The approach is motivated by the recent success of this field in different applications, including renewable energy forecasting, as explained in Chapter 2. Therefore, based on the novel SP-NN framework, discussed in Chapter 3, an extension of this approach is explored to answer the research question.

The nature of the research project consisted of using quantitative methods to assess the quality of the predictive models. The quality of a forecast is directly related to its accuracy, or the relationship existing between the real and predicted values. Moreover, to compare the quality of different models, they are benchmark with traditional approaches, namely physical and statistical models.

4.1. European Energy Markets 2020 Forecasting Competition

The aim of the project tackled a practical problem, proposed by the European Energy Markets 2020 Conference (EEM20) [50]. Motivated by the increasing penetration of RES in the European electricity grid –especially wind energy–, the organizers promoted a competition to produce novel probabilistic forecasting methodologies that improve existing wind power production forecasts.

The setting of the competition simulates a real-life scenario. The location is based in Sweden mainland, consisting of four electricity price regions (SE1, SE2, SE3, SE4). To keep fair participation amongst competitors, the organizers provide three main sets of data, namely:

- NWP variables.
- Wind turbine record.
- Aggregated wind power production by region.

Given this available data, the intention was to replicate the day-ahead electricity market. To achieve this, the organizers provided these three main data sets in advance, corresponding to the year 2000, to train initial models. On the other hand, the objective of the competition was to accurately predict wind power production for the year 2001, divided into six submissions.

Every submission rounds accounted for two months of predictions (thus, six submission rounds make up for twelve months). Moreover, forecast submissions occurred every week, having deadlines on Tuesdays, results of the forecasts, and newly available NWP data for the following submission round on Wednesdays, and preliminary ranking on Fridays. The timeline of the competition is illustrated in Figure 4.1.

Figure 4.1: Timeline of the EEM20 Forecasting competition [50].

The final ranking consists of the best five submission rounds of every team. As can be seen from Figure 4.1, the data corresponding to the year 2000 was made available two months before the first submission deadline. The end of the competition finished with the last submission deadline in June 2019.

4.1.1. Data overview

The first part of the data set consists of NWP variables. The grid-like topology is centered in Sweden, having a spatial size of 169 x 71. Furthermore, ten ensemble members are available for seven explanatory variables, shown in Table 4.1

Table 4.1: NWP variables of the EEM20 Competition.

Variable name	Long name	Units
Temperature	Surface temperature (T2M)	K
Wind U	Zonal 10-meter wind (U10)	m/s
Wind V	Meridional 10-meter wind (V10)	m/s
WindGustSpeed	Wind Gust	m/s
Pressure	Mean Sea Level Pressure (MSLP)	Pa
CloudCover	Total Cloud Cover (TCC)	[-]
RelativeHumidity	Screen level relative humidity (RH2M)	[-]

The NWP data was divided into daily NetCDF files, containing hourly explanatory variables. Hence, one meteorological variable is structured in 24 time steps, 10 ensemble members, 169 y-coordinate values (latitude), and 71 x-coordinate values (longitude). These files were archived by *Greenlytics*, facilitating the storage of big data.

The ensemble members were produced by *MET Norway* (Norwegian Meteorological Institute), to cope with the chaotic nature and its intrinsic uncertainty. As part of their traditional weather forecasts, they employed the HIRLAM-ALADIN Research on Mesoscale Operational NWP in Euromed (HARMONIE) based-model [51], characterized by a 10-kilometer spatial resolution.

The first ensemble member constitutes a control forecast based on the best initial and boundary conditions taken from the European Centre for Medium-Range Weather Forecasts (ECMWF) model [52]. The remaining nine ensemble members were developed by modifying these conditions using a Scaled Lagged Average Forecasting (SLAF) approach, focused on scaling lateral boundary perturbations differences between forecasts [53].

The second part of the data consisted of the wind turbine record, a spreadsheet containing the following information:

- Wind turbine ID.
- Terrain height.
- Nacelle height.
- Rotor diameter.
- Maximum (rated) power.
- Price region.
- Installation date.
- Longitude.
- Latitude.

The record accounted for 4004 wind turbines, representing 8640 MW on installed capacity. However, according to the Swedish Wind Energy Association (SWEA), the actual value was 4099 wind turbines, comprised of 8984 MW of installed capacity. Therefore, a mismatch existed between the record provided and the real situation. However, it served for multiple purposes in classifying the production beyond the price region, namely the dynamic installed capacity, and estimating a physical-based model.

The third part of the data consisted of aggregated power production. This spreadsheet was organized based on the price regions of the Swedish electricity market. Since the installed capacity varied in every submission round, integration of the record and the power production was necessary to treat the increasing installed capacity of wind power.

Based on the data available for the competition, the first step consisted of visualizing and understanding this information. Hence, data analysis of these three components was developed in Chapter 5. These results provided comparisons between different price regions, quality of data, and the relationship between input and output variables. Moreover, it gave a more profound insight into the challenges of the competition setting. Alongside this, it allowed the team to define a strategy and assist the selection of input variables.

4.1.2. Evaluation method

The organizers defined the competition in a probabilistic forecasting framework, expecting the submission of nine quantiles for every time step –from Q10 to Q90–, representing an 80% range-bound. As a result, the simplest way to assess the accuracy of forecasts was through the pinball loss function, or quantile loss function. The formula expressing this function is shown as follows in 4.1:

$$\rho^{i}_{k,a}\left(q^{i}_{k,a}, y_{k,a}\right) = \begin{cases} (i/100)\left(y_{k,a} - q^{i}_{k,a}\right), & y_{k,a} \geq q^{i}_{k,a} \\ (1 - i/100)\left(q^{i}_{k,a} - y_{k,a}\right), & q^{i}_{k,a} < y_{k,a} \end{cases} \tag{4.1}$$

The pinball loss function is suitable for probabilistic forecasts, as it contains a target parameter (i), relating a particular quantile. It expresses the difference between the predicted value (q) and the real value (y), multiplied by the corresponding target quantile. Furthermore, it always gives a positive value, penalizing differently when the predictions are overestimated and underestimated, depending on the target.

The final score was determined by the average loss over the best five submission rounds, meaning that the worst submission score of every participant team was not accounted for the final metric. Moreover, the pinball loss function in every submission round was averaged for every quantile, price region, and time step, hence a global average. The process of calculating the final score is shown in 4.2-4.4:

$$\rho_{k,a} = \frac{1}{9}\sum_{i=10}^{90} \rho_{k,a}^{i}\left(q_{k,a}^{i}, y_{k,a}\right), \quad \forall k, \forall a \tag{4.2}$$

$$Score_{w,a} = \frac{1}{T}\sum_{k=1}^{T} \rho_{k,a}, \forall a \tag{4.3}$$

$$Score_{w} = \frac{1}{4}\sum_{a=1}^{4} Score_{w,a} \tag{4.4}$$

In this process, *a* refers to the price region, *k* refers to the time step, and (*i*) refers to the quantile target. The decision from the organizers to use this particular metric was based on its simplicity.

4.2. Deterministic forecasting approach

The challenges of the EEM20 forecasting competition were addressed following a machine deep learning approach, using the data provided by the organizers. Moreover, every price region (SE1, SE2, SE3, and SE4) was trained separately. Hence, the following machine learning strategies were implemented to model wind power forecasting:

- Fully-connected architectures: MLP.
- Shift-invariant architectures: CNN.
- K-means clustering.

This represented the first step of the research, namely identifying potential architectures that could capture relevant patterns from the available EEM20 data set. The goal was to select the best deterministic models to integrate them in a probabilistic framework.

The metric to assess the accuracy of deterministic forecasts is the MAE. Equation 4.5 presents how the errors are computed:

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n} \left|y_j - \hat{y}_j\right| \tag{4.5}$$

From the equation above, *n* represents the number of samples considered, while the terms between the absolute represent the difference between the real –observed– value (y_j) and the predicted value ($\hat{y}_j$).

The MSE metric has been left out of the analysis. Two reasons support this decision. First, this metric does not align with the pinball loss function used in the forecasting competition. Second, since the deterministic approach only represents an initial approach to assess the accuracy of forecasts, the use of MAE is sufficient to quantify the capabilities of different models.

The strategy to standardize the quality of different neural network models consisted of characterizing them as follows:

1. Data cleansing: the raw data provided had missing or NaN (not-a-number) values, as commonly named in programming environments. Therefore, these values were substituted by interpolation. In some cases, interpolation could not resolve the problem, hence as a further step, these values were replaced by standard meteorological conditions.

2. Feature scaling: to facilitate the convergence of the optimization algorithm to find local optima, the input variables were normalized by removing the mean and scaling to the variance of the data set. As a result, the data set was more symmetric, reducing possible skewness between variables.

3. Optimization: the adaptive learning rate Adam algorithm was implemented to converge in suitable local optima, motivated by its success in non-convex optimization problems among different application fields. Moreover, as already discussed, the loss function used for this process is the MAE. In machine learning terminology, this is also called the L1-loss.

4. Activation function: to ensure the learning process of the algorithm, a ReLU function was employed in all the layers of the architectures tested. The reason was to avoid gradient vanishing or explosion during the backpropagation algorithms, a typical challenge in DNN structures.

5. Regularization: to avoid overfitting of data, the introduction of regularization terms are necessary to find a balance between bias and variance. However, as the input data is the same in all cases, regularization terms are initially neglected in this stage, unless otherwise mentioned. The introduction of regularization terms was only considered in the last stage of the forecasting framework.

6. Training data: the values used for machine learning processes consisted of the information provided by the forecasting competition concerning the year 2000. Next to this, a validation split was implemented to evaluate the bias and variance of the model. Hence, 25% of the training data set was separated to convert it into a cross-validation data set.

7. Testing data: the values used to test the models trained consisted of the information provided by the forecasting competition through every submission round, corresponding to the year 2000. Moreover, instead of testing every submission round, the testing data was split into two segments. The first segment corresponded to submission round 1, while the second segment corresponded to submission round 2-6.

To compare the reliability of these models, they are benchmarked against different traditional statistical models. The most basic approach is a similarity model [54], which can be expressed by the following formula:

$$\hat{y}_{t+1} = y_t \tag{4.6}$$

Although the competition replicated a day-ahead situation, the power output information was given for every two months. Hence, it was not appropriate to define a persistence-based model on the day or hour before, but the year before. As a result, the best way to formulate a naive approach was by defining the normalized power production to the same value corresponding to the previous year. For instance, the normalized aggregated power production for price region 3 (SE3) on October 16, 2001, at 14:00 is assumed to be equal to the value reported that same day at the same time.

On the other hand, these models are also benchmarked against a physical-based approach, following the nature of wind power production and using the wind turbine record to relate the meteorological conditions of a particular location. This process can be shown in Figure 4.2.

The process of building the physical model was inspired by the work of Sukanta Basu during the same competition [55], being illustrated in Figure 4.2. It starts with the wind turbine record, identifying the locations of all wind turbines for a particular price region. These location coordinates were linked with the meteorological variables available from the NWP grid-like data. Therefore, every wind turbine corresponds to specific 10-meter wind speed, temperature, and pressure values. Note that the wind speed value is the magnitude of both zonal (U10) and meridional (V10) vector components. The second step was to correct the wind speed at 10 meters to the hub height of every wind turbine. This vertical extrapolation was achieved using the power law. Consequently, a power-law coefficient (α) had to be defined for this purpose. Although the surface roughness can vary across the country and in time, a

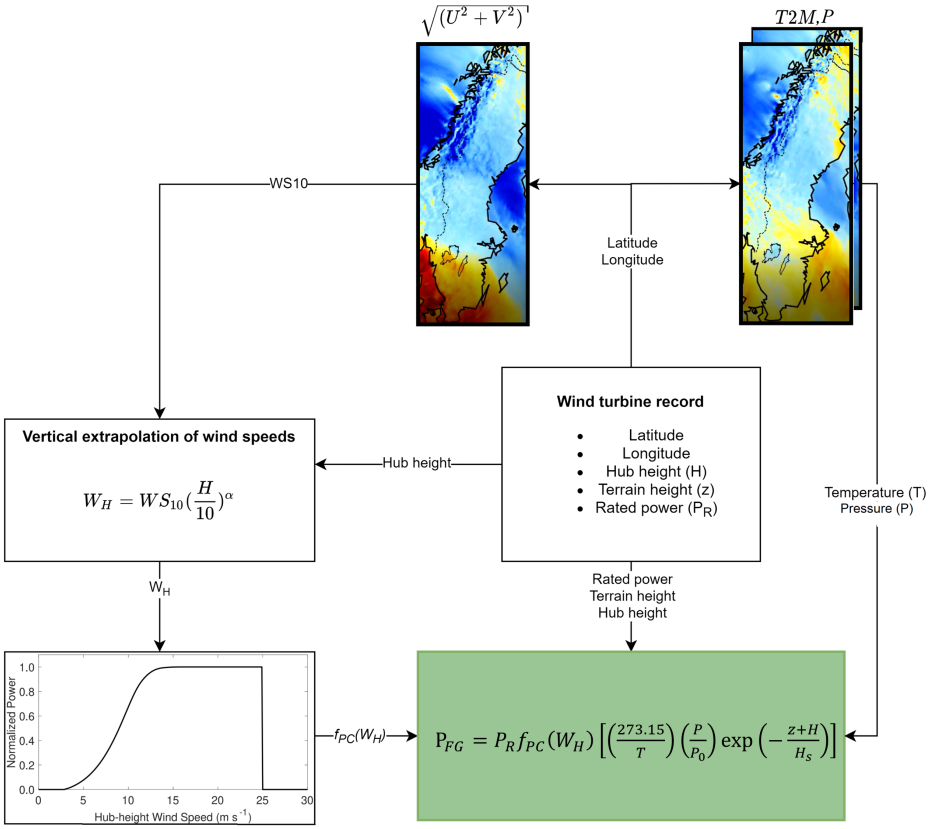


Figure 4.2: Physical-based model flowchart.

value of $\alpha = 0.3$ was assumed constant to facilitate these calculations. This simplification would affect the effective wind speed at hub height. However, it was not possible to directly determine the surface roughness based on the available data. The resulting wind speed at the corresponding hub height was inputted in a standardized class I power curve, extracted from IEC 61400. This power curve is normalized, following traditional control regimes, namely, torque and pitch control. The final step was to compute the power produced by the wind turbine in a particular time step by multiplying the normalized power production with the rated power, derived again from the wind turbine record. Moreover, since this power curve was based on standard meteorological conditions, the following corrections were introduced:

- A temperature correction: air density is inversely proportional.
- A pressure correction: air density is directly proportional. approximating to Equation 4.7. This correction is an addition of the original approach followed by [55].
- Elevation correction: air density decreases in an exponential approximation. Hence, based on terrain and hub height $(z + H)$, it follows the following relationship:

$$\rho(P) = \rho_0(\frac{P}{P_0}) \tag{4.7}$$

$$\rho(z+H) = \rho_0 \exp\left(-\frac{z+H}{H_s}\right) \tag{4.8}$$

Equation 4.7, relates a density correction based on mean sea level conditions. Here, ρ_0 = 1.225 kg/m^3, while P_0 = 101.325 kPa. On the other hand, Equation 4.8 corrects the density respect to sea level (ρ_0) to the desired elevation, assuming the scale height of density, H_s = 8550 m. Given the nearly exponential relationship, this term is relevant to properly estimate the production in mountainous regions.

4.2.1. Shift-invariant architectures: CNN

The CNN architectures were used to capture spatial patterns in the data provided. Given the grid-like topology nature of the NWP data, the methodology focused on using proven architectures that are suitable for identifying spatial patterns. Therefore, a LeNet-5, VGG-16, and AlexNet-based architectures were tested. Figure 4.3 shows a schematic of the LeNet-5 based architecture tested. Moreover, Figure 4.4 and Figure 4.5 illustrate the schematics of the VGG-16 and AlexNet architectures, respectively.

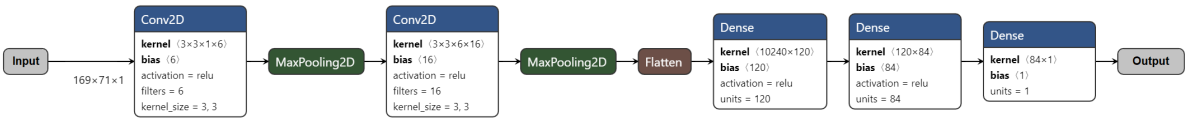


Figure 4.3: Schematic of the LeNet-5 based neural network architecture.

The LeNet-5-based architecture was the most basic CNN tested. It consists of two convolutional layers with 6 and 16 kernels, respectively. Moreover, the size of the kernel was 3x3. After each convolutional layer, a sub-sampling layer is introduced, using the Max Pooling strategy. Next, the structure is flattened to couple it into two fully-connected layers of 120 and 84 nodes, respectively. The last layer consists of one output node, corresponding to the aggregated power production of the relevant price region.

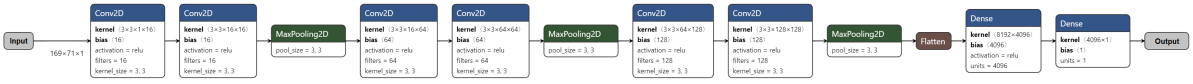


Figure 4.4: Schematic of the VGG-16 based neural network architecture.

The VGG-16-based architecture was the most complex CNN tested. Given its deepness, a simplified version was built to align with the computer processor resources. It consists of three convolutional layers with 16, 64, and 128 kernels, respectively. Similar to the LeNet-5 architecture, the size of the kernel was 3x3, while a sub-sampling layer is introduced after the convolutional layer, using the Max Pooling strategy. In comparison to the previous architecture, this model introduced a stride of 2 for the sub-sampling section. For the last segment, the structure is flattened to couple it into one fully-connected layer of 4096 nodes. Again, the last layer consisted of the output node corresponding to the power production in the relevant price region.

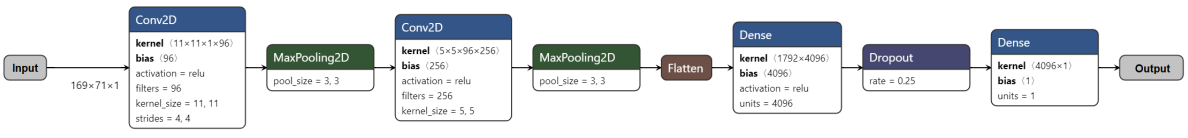


Figure 4.5: Schematic of the AlexNet based neural network architecture.

The AlexNet-based represented a model with a complexity level between the previous architectures. It consisted of two convolutional layers with 96 and 256 kernels, respectively. In comparison to the

previous architecture, this model employs variable kernel sizes of 11x11, and 5x5, respectively. A sub-sampling layer was introduced after the convolutional layer, using the Max Pooling strategy. Similar to the VGG-16 model, this structure imposes a stride of 2. For the last segment, the structure is flattened, fully-connected to a layer of 4096 nodes, alongside a Dropout factor of 25% for this last hidden layer. The output layer consisted of a single node corresponding to the power production in the relevant price region.

4.2.2. Fully-connected architectures: MLP

Different MLP architectures were tested with the expectation of finding relationships following a non-grid-like topology data approach, neglecting the spatial relationship between the values in the array.

The conventional ANN model served as a complementary way to relate to the output data. Additionally, it gave the possibility of capturing temporal patterns in the data without building a computationally expensive model. A series of models with different amount of nodes and layers were evaluated. The selection of MLP architectures was based on the best models obtained by Torres and Aguilar, where they tested a variety of traditional architectures to predict wind farm generation [22]. Moreover, one personal choice architecture was designed. Figures 4.6, 4.7, 4.8, and 4.9 illustrate schematics of the MLP architectures.

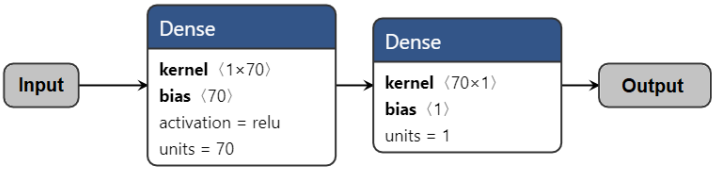


Figure 4.6: Schematic of MLP-1 architecture.

The first MLP architecture tested –identified as MLP-1– consists of a single hidden layer with 70 nodes. This structure resulted in the best model employed in [22] for ReLU-based structures.

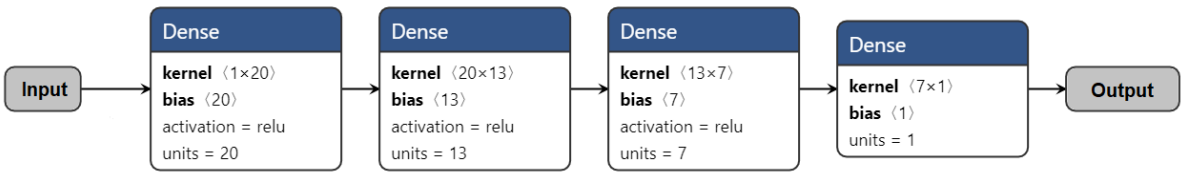


Figure 4.7: Schematic of MLP-2 architecture.

The second MLP architecture tested –identified as MLP-2– consists of three hidden layers with 20, 13, and 7 nodes, respectively. This structure resulted in the best model employed in [22] for a different type of activation function. However, given the methodology of this project, it was replicated using a ReLU activation function.

The third MLP architecture tested –identified as MLP-3– represented the most complex structure of this type of architecture tested for this application. It consists of 6 hidden layers: the node distribution

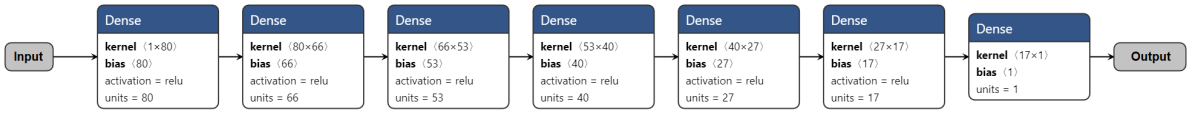


Figure 4.8: Schematic of MLP-3 architecture.

for every layer is [80, 66, 53, 40, 27, 14]. This represented the best model using the ReLU activation function but with a feature engineered data set in [22].

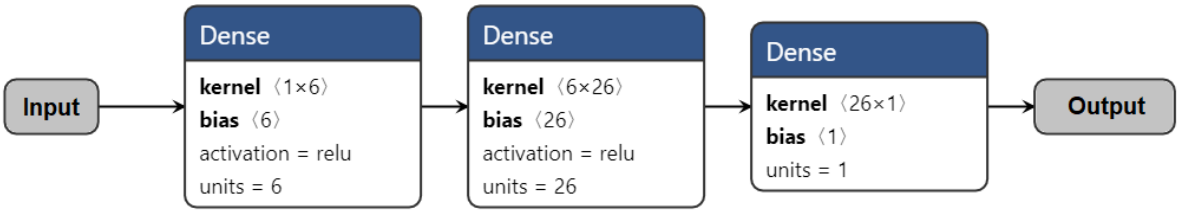


Figure 4.9: Schematic of MLP-4 architecture.

The last MLP architecture (MLP-4) was a personal design choice, hence does not have any supporting documentation. However, the reason to use this architecture was to elaborate a structure that could capture temporal patterns in the data set. The architecture consists of two hidden layers with 6 and 26 nodes, respectively.

Since all price regions were trained separately, the output layer consisted of a single node. However, the input node could vary based on feature engineering. For these visualizations, the default input layer also consists of a single node.

4.2.3. Clustering: k-means

The k-means clustering unsupervised method is introduced to organize and reduce the dimensionality of data. The objective of this technique was two-fold:

On the one hand, to tackle the traditional clustering problem by selecting relevant data points from the NWP data grid. Therefore, a weighted k-means method was utilized to consider not only the number of wind turbines but also its rated power, thus, to make wind turbine clusters based on wind power concentration. Therefore, this decision served for feature engineering (further discussion is provided in the following Section 4.3).

On the other hand, to build a k-means regression-based predictive model. Hence, a simple implementation by calculating the cluster average of the numerical target. The motivation to use this method was to test algorithms outside the supervised learning realm. Moreover, it was meant as a method of weather classification. Figure 4.10 illustrates the analog-based approach for weather classification, in which every class provides an average wind power production. The analog-based approach is based on the strategy followed by the winning team of the EEM20 forecasting competition: MINES ParisTech [56, 57].

As shown from the figure, a new atmospheric situation is inserted, related to a cluster of weather class based on its similarity or analogous conditions. The similar patterns derived from the training set

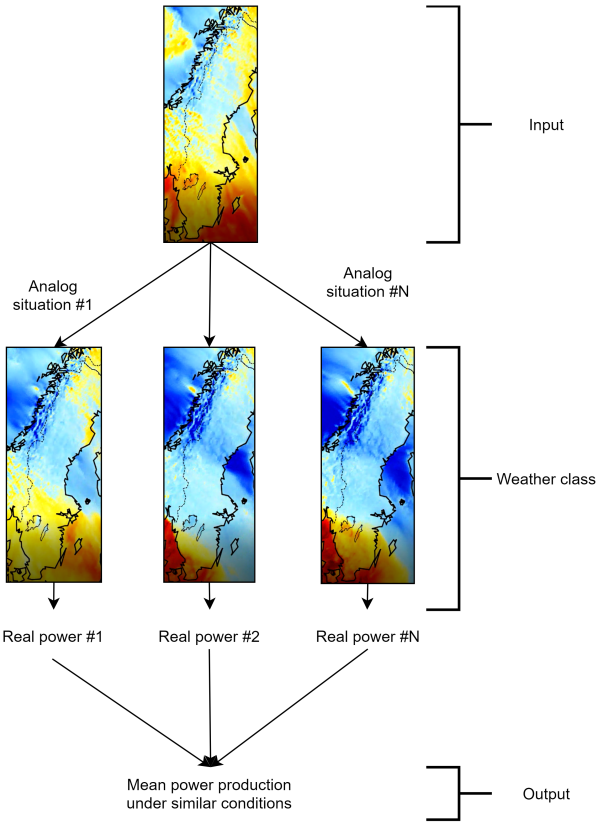


Figure 4.10: Weather analog-based approach using a simple k-means clustering method.

were found using the Euclidean distance formula. Finally, the average power production of the weather class (containing a set of similar events) represents the output of the new situation.

The process of weather classification consisted of organizing the available NWP data to enhance wind power prediction. In other words, to select averaged past values of power production during similar weather conditions. As a consequence, the meteorological variables were selected based on the performance of the weather classes to predict wind power production.

The selection of the number of weather classes was based on forecasting accuracy. Therefore, different class values (k) were evaluated. As a result, the regression method aimed at optimizing the number of classes, introducing various input data available from the NWP meteorological variables.

4.3. Feature and output engineering

The second step of the research focused on feature/target engineering. Forecasting competitions have shown that winning teams tend to stand out for working out the available data, relating the input and output data. Beyond data cleansing and normalization, employing optimal features is relevant to enhance the accuracy of these models. Therefore, domain knowledge of wind power facilitates the feature design process.

An extensive exploration of the wind power drivers based on various physical phenomena was performed. Table 4.2 summarizes different factors inspected during this phase of the research.

As seen from the table, feature variables had to be pre-processed before inserting them in the different models tested. For instance, the wind speed variable was obtained from calculating the module

Table 4.2: Drivers of wind power for feature engineering.

Physical phenomena	Feature engineering
Wind speed (WS10)	Resulting magnitude of U10 & V10
Spatial dependencies (WD10)	Wind direction from U10 & V10
Extreme wind values (Gust)	Gust wind speeds
Atmospheric conditions (T, P)	MSLP & T2M
Wind power (WP)	Vertical extrapolation of WS10 for hub height (W_H) Cube of wind speed (W_H^3) Air density for T2M and terrain height (z) Standard IEC power curve class 1 Turbine location (latitude, longitude) Rated power
Operational constraints Temporal dependencies	Time of the day Day of the year
Weather classes	k-means analog-based approach averaged historical production

of the zonal and meridional components of wind speed. Equation 4.9 shows the process to create a grid of wind speeds:

$$WS10_{ij} = \sqrt{U10_{ij}^2 + V10_{ij}^2} \quad \forall i, \forall j \tag{4.9}$$

From the equation above, U10 represents the zonal wind component, while V10 represents the meridional wind component. Both components are related to a 10-meter height. Hence, for every coordinate point (i,j), the 10-meter wind speed magnitude was computed.

The same approach was followed to determine the wind direction. Equation 4.10 shows the formula used to compute this feature:

$$WD10_{ij} = \arctan\left(\frac{V10_{ij}}{U10_{ij}}\right) \quad \forall i, \forall j \tag{4.10}$$

The same variables from 4.9 are used in 4.10. In the latter, the inverse tangent function served to calculate the angle between both vectors. Additionally, to extend the range from 0 to 360°, for negative meridional components, a factor 2π is summed, while for negative zonal components a factor π is summed. When both factors are negative, a factor 1.5π is summed. The remaining meteorological variables were not altered concerning the raw NWP data.

The wind power features were used following the same methodology described in Figure 4.2. The motivation is to use this data as a new input for a machine learning approach, hence a neural network architecture.

Operational constraints were introduced to capture certain operational logistics of wind power, such as demand and maintenance. In the case of the physical model, a 100% availability was assumed. However, **the availability of each wind turbine was one of the principal challenges** given the competition settings. Hence, the simplest way to address the behavior of each turbine consisted of computing a time proxy pair, based on a sine-cosine function. Equations 4.11 and 4.12 show the computations for time pairs based on the time of the day and day of the year, respectively:

$$h \rightarrow \left(\sin \frac{h}{24}, \cos \frac{h}{24} \right) \tag{4.11}$$

$$d \rightarrow \left(\sin \frac{d}{365}, \cos \frac{d}{365} \right) \tag{4.12}$$

The equations above relate the hour of the day (h) in a pair of sine-cosine functions. The same applies to the day of the year (h). In consequence, 4 time-proxy features were considered as temporal dependencies.

The second main challenge of the EEM20 forecasting competition was the size of the data. In machine learning terminology, this setting would be considered as a Big Data problem. The available NWP training data set consisted of 840.000 values per hour, considering the meteorological variables, the number of ensemble members, and the size of the grid. Hence, to reduce the dimensionality of the data, the following decisions were made:

- Understanding the compilation of the ensemble members, only the median values of these ensembles were taken into account. This simple strategy already processes the data in a way that reduces it to 10% of the original size.
- Although every price region was trained separately, the NWP grid size (169 x 71) was kept unaltered when introduced in convolutional layers. Hence, determining the optimal grid size for every price region was left out of the scope of this research. However, the k-means approach was employed to reduce the dimensionality of data, but this input data was only introduced in fully-connected layers, such as in MLP architectures.
- Meteorological variables were selected based on initial steps, given time constraints and computational resources. In other words, feature engineering was not developed in all machine learning models, but guided by domain knowledge and literature.

The third challenge of the forecasting competition was to address the dynamic installed capacity during the submission rounds. The best approach to manage the growth of installed wind power was by means of normalizing the production. Hence, the forecasting models would predict the capacity factor of the respective price region. After the prediction outcome, to comply with the competition requirements and facilitate visualization, the capacity factor is multiplied by the installed capacity in the respective time step. In this step, a 100% availability was also assumed.

4.4. SpinHy-NN: Integrating the SP-NN probabilistic framework

After exploring different architectures, the last step consisted in developing a methodology for probabilistic forecasting. As explained in the literature review, two approaches are possible: a parametric and non-parametric approach.

As the objective of this research is to introduce a new methodology that can be generalized for different data sets, a non-parametric approach was followed, making the model learn the shape of the PDF. Particularly for wind power forecasting, this decision seems appropriate as the complex nature of wind does not allow to fit a traditional ”shape curve” varying in every climate. Moreover, the shape for the same location could change in time [25], having to re-adapt existing parametric models, making non-parametric models suitable for wind applications. However, the non-parametric approach requires the tuning of additional parameters and hence, increased computational costs.

On the other hand, under the context of the Swedish case study, another possibility was to develop an ensemble forecast model. However, ensemble members require substantial amounts of data from meteorological variables that are not always available. Although it was possible to follow this methodology, the approach was discarded to promote a generalized method for future case studies.

The metric to assess the accuracy of forecasts during the competition is similar to the cost function employed in the SP-NN. The only difference lies in its smoothening. In this sense, the final framework trained directly the evaluation method, or in other words, the setup of the neural network guaranteed the optimization of the model following the competition scoring guidelines. As a consequence, the evaluation of this method expresses the score for a particular round.

To introduce both spatial and temporal data, a variation of the original SP-NN was created to forecast wind power production under a probabilistic framework: the **Smooth Pinball Hybrid Neural Network (SPinHy-NN)**. The differences between both architectures are:

1. Implementation of a CNN to capture spatial patterns of NWP data.
2. Concatenation of a traditional MLP to use it as a proxy for additional variables and capture operational availability and maintenance;
3. Adjustment of tailor-made hyperparameters to account for sharpness and reliability.

The CNN-branch of the structure was selected based on the best initial model of the convolutional neural networks tested. Similarly, the tailor-made MLP branch was selected.

Regarding the hyperparameters, the smoothening parameter was assumed to α = 0.001, as employed in [49]. The reason is that the Adam optimization algorithm managed to adapt the learning rate to converge particular local optima, proving its capabilities in non-convex optimization problems. On the other hand, a penalty factor of κ = 1000 was assumed [49]. Hence, the only SPinHy-NN parameter to tune was the quantile margin. This strategy corrected the undesired crossing of quantiles, which is an inconsistent behavior in non-parametric approaches. The SPinHy-NN included a regularization term for the MLP branch, which is assumed to be (λ) = 0.0001, following the approach in [49]. This term is used in the kernel initializer as well, which is in charge of reducing the influence of certain terms during the random initialization of weights.

Finally, the following hyperparameters were tuned: spatial dropout, batch size, and sub-sampling layer, based on the method followed in [37]. In this method, batch sizes of 32, 64 are typically used, depending on the data set size. Moreover, both Max Pooling and Average Pooling sub-sampling layers were tested to reduce the size of the convolutional operations. The spatial dropout was kept small, from no dropout factor up to 25%.

The best results obtained post-competition are compared against the three best models during the event, which all turned out to be based on machine learning approaches, namely:

- MINES ParisTech: Quantile Regression Forest (QRF) [56].
- University of Strathclyde: Quantile Generalized Additive Model (QGAM) & Gradient Boosting Machine (GBM) [58].
- TU Delft: Hybrid CNN-MLP (HCM) neural network [55].

The last team was led by Sukanta Basu, assisted by Simon J. Watson, Bedassa Cheneka, and the author of this report. Regarding the team representing this thesis project (*DeepWinds*), submission mistakes occurred during the competition, hence these are not reported in this document. Therefore, the best outcome of this research project is benchmarked against the best forecasts during the competition timeline, providing a time advantage and expertise to improve the accuracy using the same data set. However, the probabilistic methodology during the re-forecasting process did not change, meaning that the outcome was already constrained by the same approach.

4.5. Programming framework

A comparison between different models was performed during this research. These models are characterized by their complexity, hence framing the computational tools available was relevant to deal with the time constraints of the project. The trade-off between accuracy and computational time represented a constraint on the decision-making and research direction for quality results.

The algorithms were executed on an HP ZBook Studio G5. This device is integrated with an Intel i7-8750H processing unit, holds 16 GB of RAM, and two Graphical Processing Units (GPU): an Intel UHD Graphics 630 and NVIDIA Quadro P1000 GPU. The operative system of the computer is Windows 10.

GPUs are a valuable resource in the development of spatial based deep learning models, such as the case of CNN. They can execute many computations in parallel that have a grid-like topology. Therefore, the computational effort was reduced by including them in the framework.

The code was written in Python language, using a GPU-enabled Jupyter Notebook Integrated Development Environment (IDE) under the Anaconda management software, useful for the deployment of this programming language. Jupyter Notebooks facilitate the sharing of files, as the environment allows the inclusion of notes, headers, and images to characterize every section of the script. Furthermore, the following library package versions were installed:

- Python: 3.7.4
- Keras: 2.3.1
- netCDF4: 1.4.2
- scikit-learn: 0.23.1

The main Python code is available in Appendix B and C.

4.5.1. Keras library

Keras is an application programming interface (API) designed for AI. It follows innovative practices to reduce cognitive load through simple APIs, diminishing the number of actions required by the user with clear and actionable error messages. It also has extensive documentation and developer guides [59]. Therefore, Keras has emerged as one of the most powerful libraries to develop these algorithms in a simple manner. The Keras functional API is a way to create flexible models compared to sequential ones. The functional API can handle models with non-linear topology, shared layers, and models with multiple inputs or outputs. Since deep learning models have become directed acyclic graph (DAG) of layers, the tool allowed to build graphs of layers. Figure 4.11 shows a summary model example of the SPinHy-NN using this package.

Layer (type)	Output Shape	Param #	Connected to
space (InputLayer)	(None, 169, 71, 1)	0	
time (InputLayer)	(None, 4)	0	
sequential_5 (Sequential)	(None, 84)	1240024	space[0][0]
sequential_6 (Sequential)	(None, 26)	212	time[0][0]
concatenate_3 (Concatenate)	(None, 110)	0	sequential_5[1][0] sequential_6[1][0]
dense_15 (Dense)	(None, 9)	999	concatenate_3[0][0]

Total params: 1,241,235
Trainable params: 1,241,235
Non-trainable params: 0

Figure 4.11: Model summary of the SPinHy-NN using the Keras package.

5 Data Analysis

Given the big data problem of this case study, understanding the available information and using optimal variables is relevant to enhance the accuracy of a predictive model. Therefore, the following chapter is focused on analyzing the main data sets through clear visualizations.

The outline of this chapter is divided into two sections. The first section provides an overview of the NWP meteorological variables, assessing the resolution and quality of data. Additionally, a correlation between variables is performed to understand the relationship between the input and output of the model. The second section illustrates the wind turbine record: showcasing the installed capacity operating in Sweden for the period 2000-2001. Moreover, it visualizes the terrain height, evaluating its influence on the predictive model. Finally, the last part addresses the installation dates of wind turbines across the submission round periods.

5.1. Meteorological variables

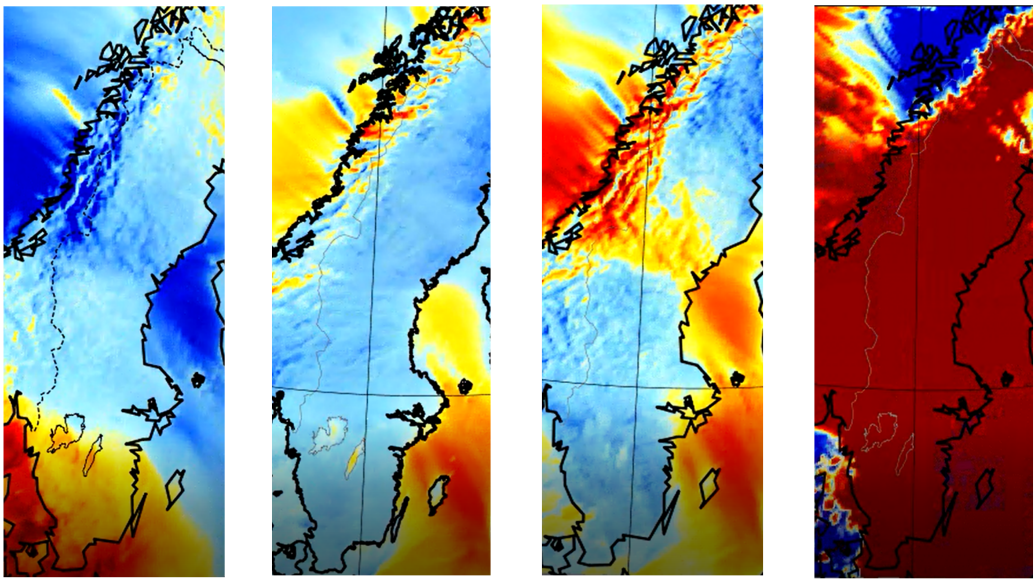
The most crucial information provided by the competition was the NWP data. The following section provides an analysis of this information, to assess the quality of the data archived by Greenlytics. Moreover, a correlation between these variables and output power production.

The Swedish case study data set can be defined as a big data problem. Given seven meteorological variables, each of them containing ten ensemble members of hourly values for a whole year (except two missing dates) in a 169 x 71 spatial grid. This represents 839,930 variables every hour, hence 7.3 billion values. An overview of the seven meteorological variables available is shown in Figure 5.1. These visualizations correspond to January 1st, 2000 at 00:00. For this purpose, only the median of the ensemble members is shown.

An initial assessment of the quality of these images for the training data set counts the number of NaN-values for every variable present in the NetCDF files. By performing this calculation, the results showed that all the variables contained simultaneous missing hours. In addition to the two missing days, 128 hours did not contain any information, meaning that one week of the year 2000 data was missing in the training set.

A correlation matrix is shown in Table 5.1 to assess relationship between input-output variables. The input meteorological variables relate the full grid size (169 x 71), while the output represents the aggregated power production at every price region. On the other hand, the wind component variables have been derived into wind speed magnitude and direction, as the vectors have positive-negative values. Additionally, the matrix has been normalized between a range of [0, 1], following mutual information theory.

As expected, wind speed (WS10) resulted to have the greatest correlation with wind power. Alongside wind speed, gust wind shows a very similar correlation as well (> 0.5), which can be understood



(a) U10 m/s (b) V10 (m/s) (c) Gust (m/s) (d) Cloud Cover (-)

Figure 5.1: Overview of NWP meteorological variables for a particular time step.

Table 5.1: Correlation matrix between input and output variables.

Feature variable	SE1	SE2	SE3	SE4
Wind speed (WS)	**0.523**	**0.659**	**0.696**	**0.560**
Wind direction (WD)	0.060	0.060	0.016	0.164
Zonal 10-meter wind (U10)	0.336	0.453	0.545	0.527
Meridional 10-meter wind (V10)	0.416	0.512	0.511	0.338
Wind gust	**0.521**	**0.657**	**0.687**	**0.548**
Mean sea level pressure (MLSP)	0.063	0.240	0.315	0.234
Screen level rel. humidity (RH2M)	0.050	0.017	0.012	0.051
Surface temperature (T2M)	0.034	0.002	0.103	0.199
Total cloud cover (TCC)	0.065	0.149	0.244	0.197

by the methodology used in weather forecasts to produce NWP data sets. On the other hand, other variables like wind direction show almost no correlation, which is an interesting indication for the problem. As a result, based on the NWP information provided by the organizers, wind speed and wind gust are the only variables relating to the aggregated power production.

5.2. Wind turbine record

Next to the NWP meteorological variables, relevant information can be collected from the wind turbine record. The first step consists of visualizing the location of the wind turbine, alongside the price region they belong to. This is shown in Figure 5.2.

As it has been stated in Chapter 4, the record provided contains information of 4004 wind turbines. However, the record held by the SWEA accounts for 4099 wind turbines. Regardless of the discrepancies, this information provides valuables insights into the case study presented in the competition.

The subsequent sub-sections will provide further insights into the installed capacity, power concentration, terrain height variations, and installation dates. The visualizations are classified by price region, motivated by the approach of training the models separately.

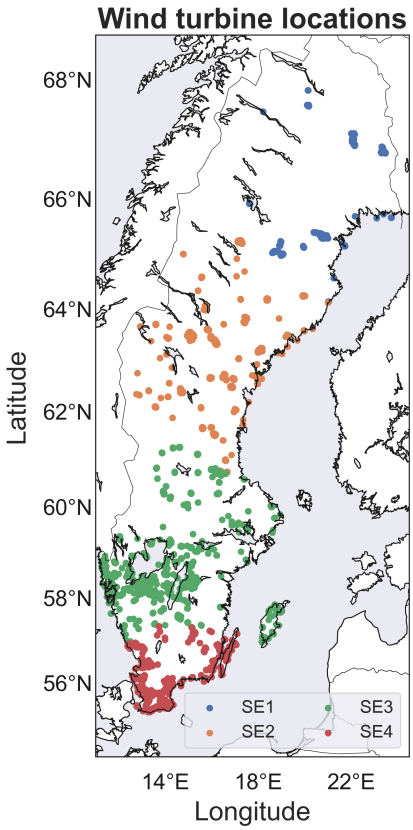


Figure 5.2: Wind turbine locations classified for every price region.

5.2.1. Installed capacity

The share of installed wind capacity in Sweden varies per price region. Figure 5.3 illustrates the number of wind turbines concentrated in clusters in Sweden. It can be seen that price regions SE2 and SE3 have the highest share of wind turbines. In fact, the share is 12%, 28%, 36%, and 24%, respectively. In detail, the concentration of wind turbines per region is shown in 5.4.

Based on Figure 5.4, it can be seen that wind turbine clusters are present throughout the country, especially in price regions SE1, SE2, and SE3. For SE4, the amount of wind turbines is spread roughly in equal proportions. The total number of wind turbines per price region is 472, 1112, 1450, and 970, respectively.

On the other hand, the rated power of these wind turbines varies from 0.01 MW to 4.2 MW. Hence, it is more appropriate to visualize power concentration in Sweden. Figure 5.5 shows the power concentration in Sweden. It can be seen that the same clusters are present, especially in the North. Figure 5.6 shows in detail the power concentration for every price region. The share of installed wind power capacity is 15%, 35%, 31%, and 19%, respectively for every price region.

Similar to the number of wind turbines in Figure 5.5, wind turbine clusters are present throughout the country, especially in price regions SE1, SE2, and SE3. For price region SE4, wind power concentration is spread roughly in equal proportions. The installed capacity per price region is 1.33 GW, 3.01 GW, 2.66 GW, and 1.64 GW, respectively. In consequence, the ratio between installed capacity and the number of turbines [MW/turbine] is 2.82, 2.70, 1.83, and 1.69. As a result, the biggest turbines are located in the North of Sweden, corresponding to the price regions SE1 and SE2, in that order. In total, the case study aims to predict the performance of a system with 8.64 GW of wind power installed capacity.

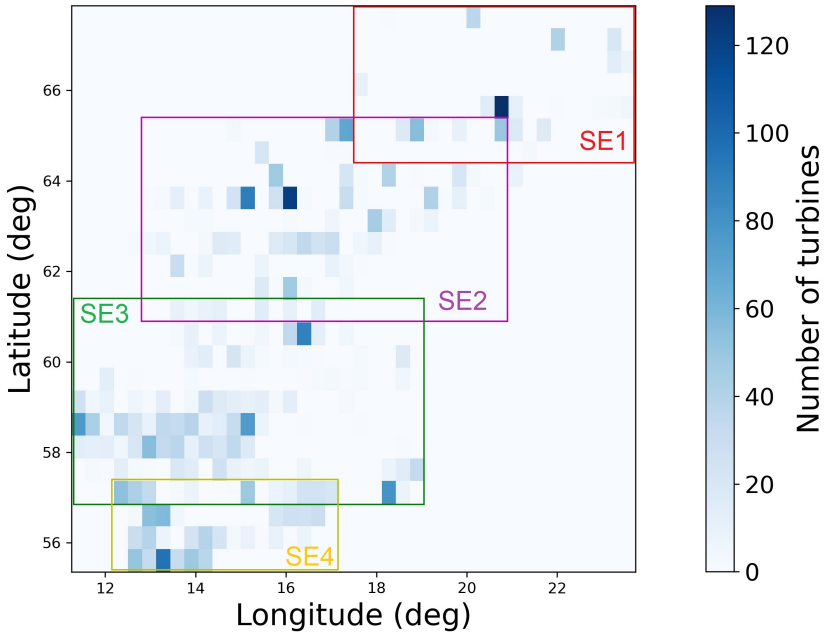
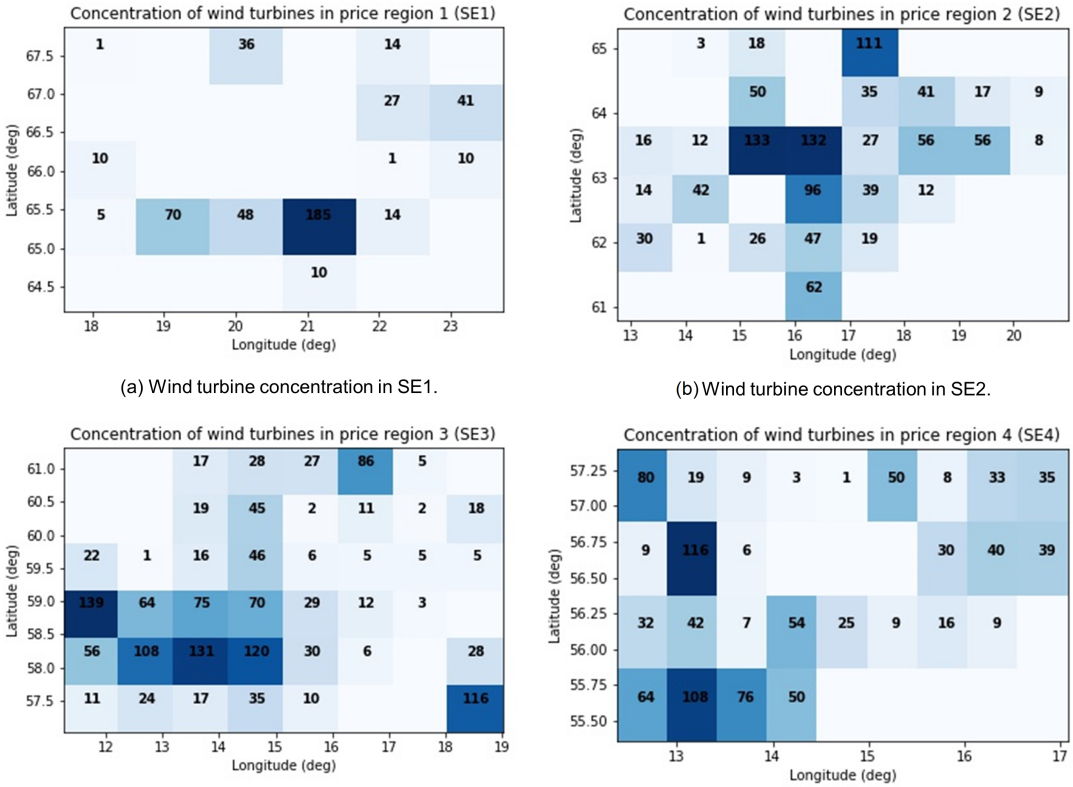


Figure 5.3: Concentration of wind turbines in Sweden.



(a) Wind turbine concentration in SE1.

(b) Wind turbine concentration in SE2.

(c) Wind turbine concentration in SE3.

(d) Wind turbine concentration in SE4.

Figure 5.4: Concentration of wind turbines in every price region of Sweden.

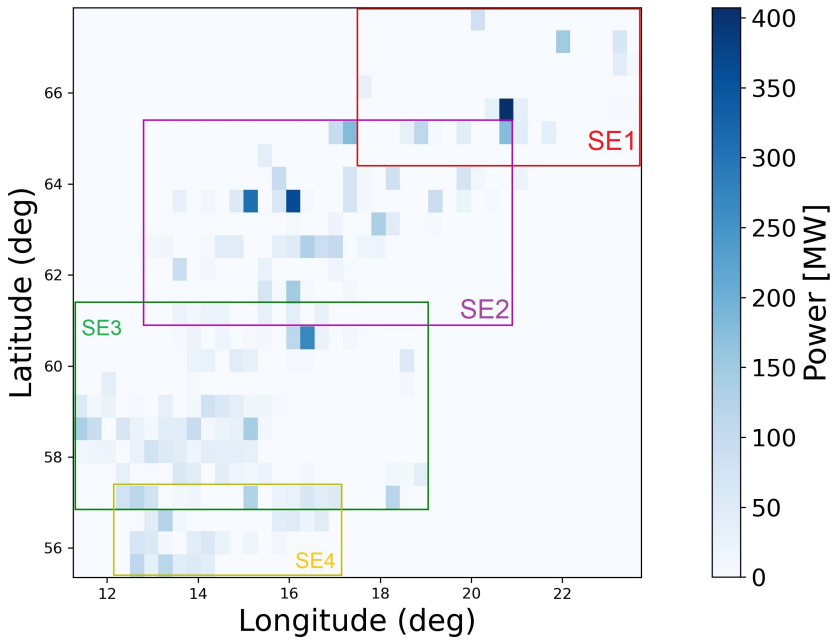
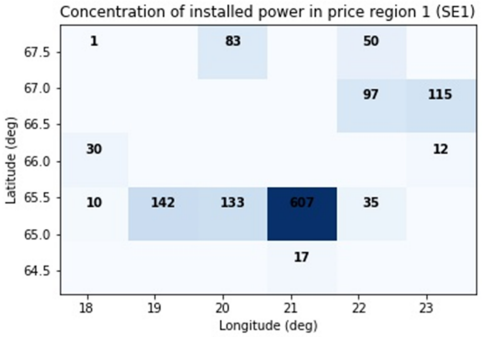
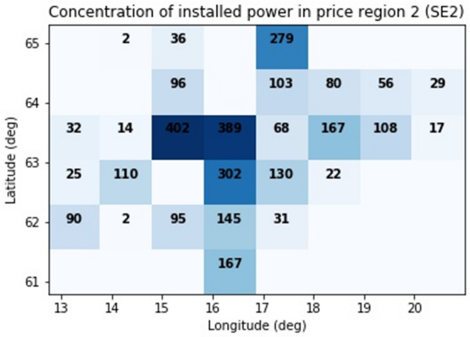


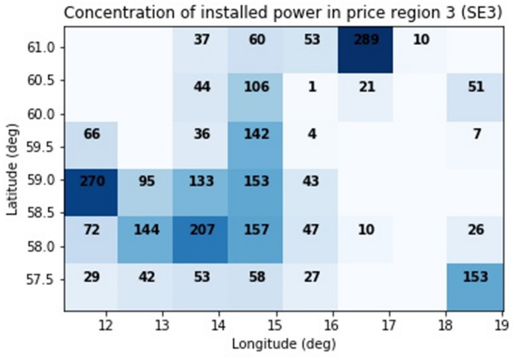
Figure 5.5: Installed capacity concentration in Sweden.



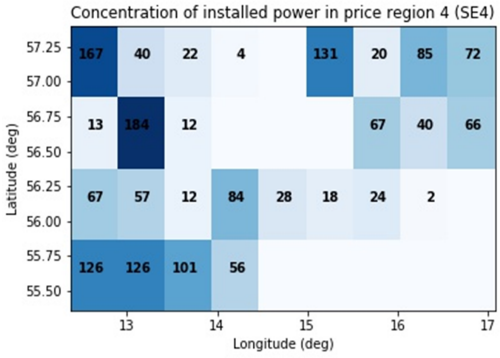
(a) Installed power concentration on SE1 (MW).



(b) Installed power concentration on SE2 (MW).



(c) Installed power concentration on SE3 (MW).



(d) Installed power concentration on SE1 (MW).

Figure 5.6: Installed capacity concentration in every price region of Sweden.

5.2.2. Terrain height

Sweden is the third-largest country in the European Union, accounting for 450,295 km^2 of land. Moreover, it is the largest country in Northern Europe [60]. Hence, the elevation greatly varies along its surface area. Figure 5.7 shows the terrain height distribution of wind turbines in Sweden (2000-2001). The average terrain height for every price region is 348 m, 476 m, 170 m, and 73 m, respectively.

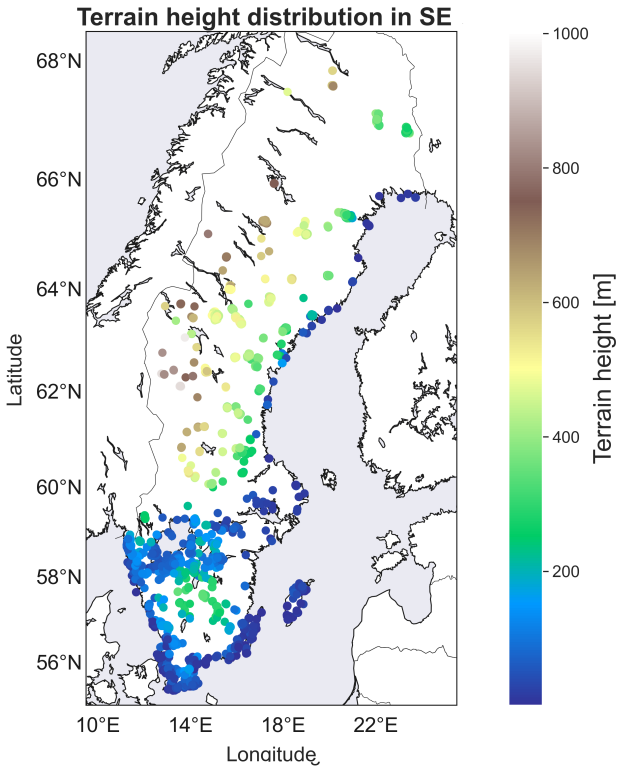


Figure 5.7: Terrain height distribution of wind turbines in Sweden.

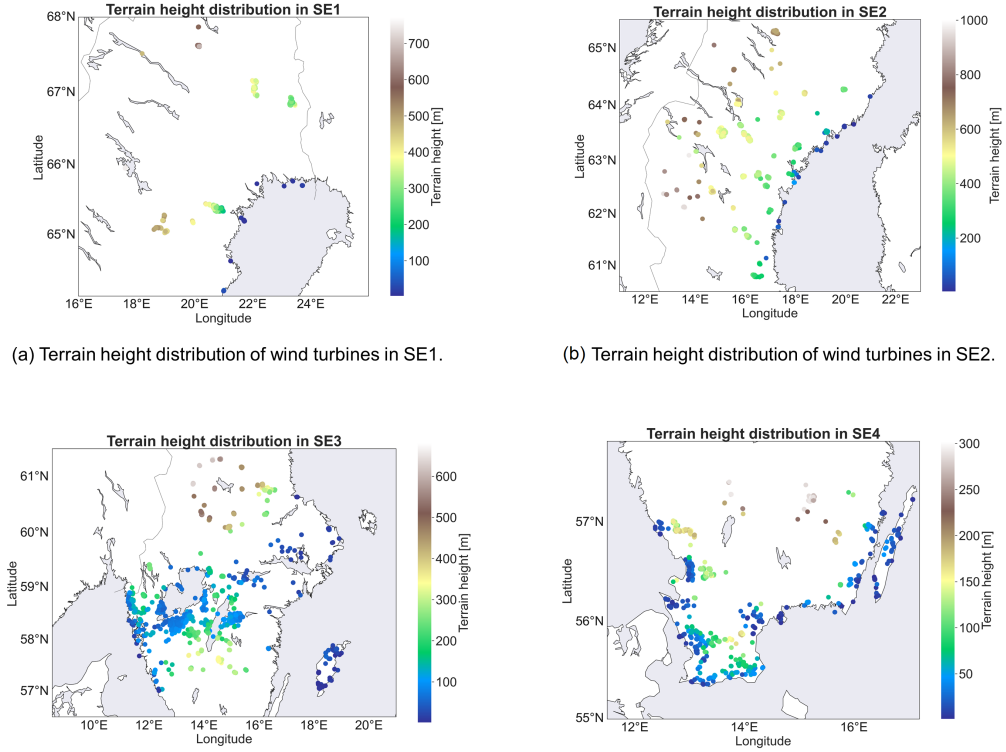
Wind turbines in SE1 and SE2 are mostly located far inland, while wind turbines for price regions 3 and 4 are situated close to sea level. Furthermore, in the south of the country, most of the wind turbines are located near the coast. Figure 5.8 zooms the terrain height distribution for every price region in Sweden.

Figure 5.8 details that wind turbine locations for SE1 and SE2 are in mountainous areas. In particular, the location with the highest terrain height is in SE2 at 1003 m MSL. On the other hand, the locations for SE3 and SE4 are close to sea level. As a result, these low-land regions have a better correlation with the wind components provided by the NWP meteorological data, given for 10-meter elevations.

5.2.3. Installation dates

The last assessment of the wind turbine record consists of determining the operational time of the wind turbine, which has a direct impact on its availability for the long term. Older turbines have a higher downtime, hence the effective capacity factor is reduced.

Figure 5.9 shows a histogram based on the installation date of wind turbines for the period 1991-2001, as the operational age has been determined with respect to the last day of the year 2001. The histogram shows that 68% of the wind turbines have an operational age below ten years, equivalent to 2725 turbines. Moreover, almost 15% of the wind turbines were installed in the period 2000-2001



(a) Terrain height distribution of wind turbines in SE1.

(b) Terrain height distribution of wind turbines in SE2.

(c) Terrain height distribution of wind turbines in SE3.

(d) Terrain height distribution of wind turbines in SE4.

Figure 5.8: Close-up of terrain height distribution of wind turbines in Sweden.

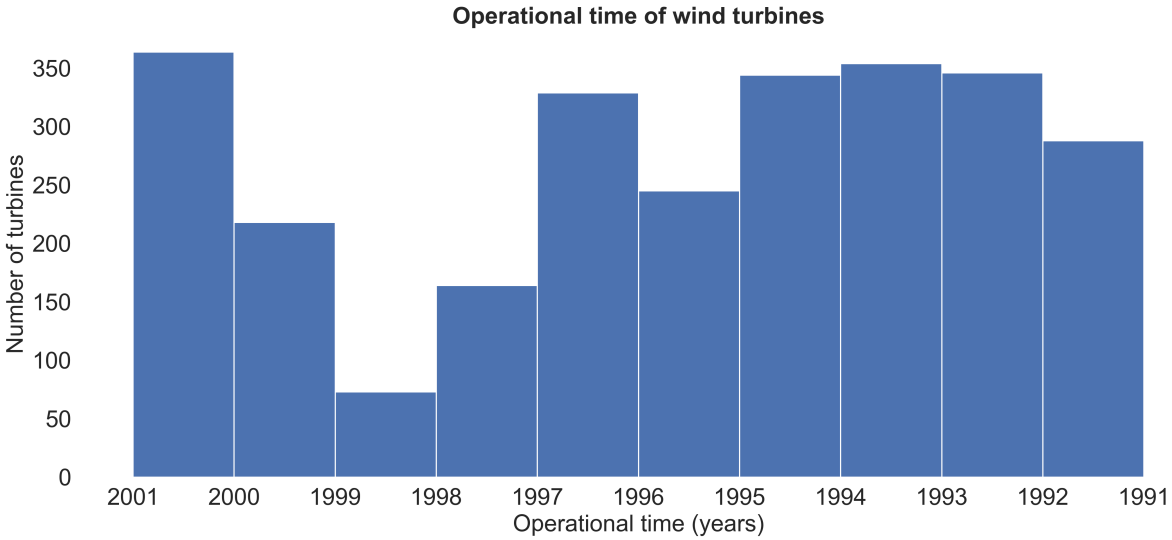


Figure 5.9: Histogram of operational age of wind turbines in Sweden for period 1991-2001.

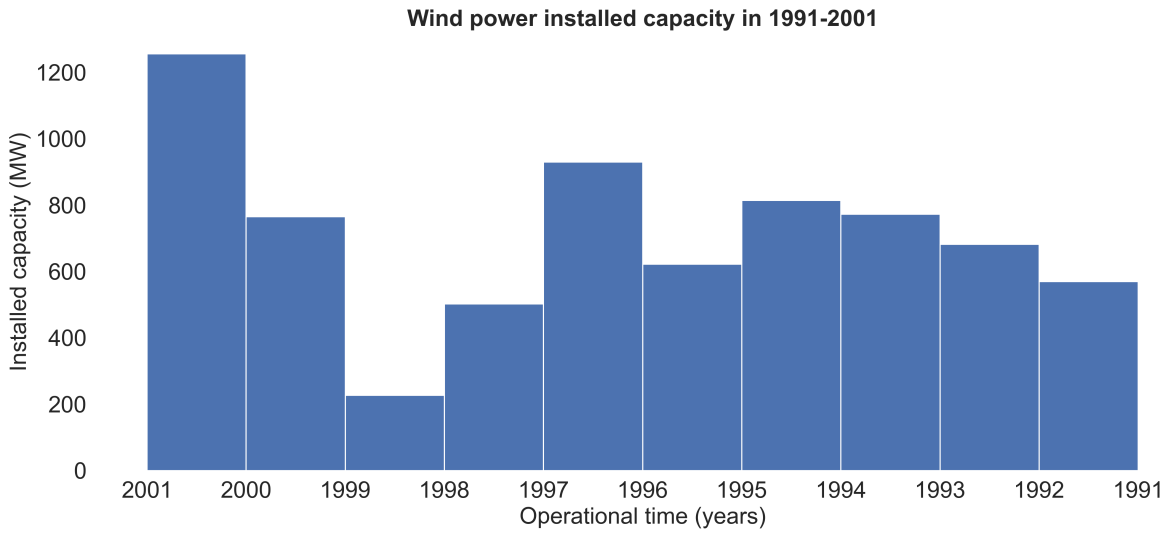


Figure 5.10: Histogram of installed capacity in Sweden for period 1991-2001.

–equivalent to 582 turbines–, which concerns the competition setting. On the other hand, the installed power capacity for the period 1991-2001 is shown in Figure 5.10.

In the installed power capacity histogram for the period 1991-2001, 81% of the wind power capacity was installed in the last ten years, equivalent to 7.04 GW. Additionally, 2.02 GW of installed capacity was installed in the period of the competition (2000-2001) Hence, there is a big fluctuation between the training data and testing data installed capacity, as this value is equivalent to 23% of the power capacity.

Regarding the testing data period, 364 wind turbines were installed during the submission rounds period, accounting for 1.26 GW, equivalent to almost 15% of the total installed capacity available from this record. On the other hand, the majority of the wind turbines are reasonably new, considering that the lifetime for a megawatt-scale wind turbine is 20 years [61], although this can be extended with additional maintenance and increased downtimes of the machine (less availability).

Table 5.2 summarizes the information provided by the histograms, considering every price region. Most of the wind turbines and power capacity were installed in SE1, SE2, SE3. On the contrary, SE4 did not introduce additional capacity throughout the period 2000-2001. In particular, SE1 installed 47% of its wind turbines during the competition setting period, accounting for 59% of its total wind power installed capacity. This represents the biggest wind energy addition in relative terms amongst all price regions in Sweden for the given period.

Table 5.2: Number of wind turbines and power capacity installed by price region for 1, 2 and 10 years.

Region	SE1	SE2	SE3	SE4	Total
Amount of turbines < 1 year	115	170	70	9	364
Wind power capacity installed < 1 year (MW)	387.5	590.2	259.9	19.0	1256.6
Amount of wind turbines < 2 years	224	211	134	13	582
Wind power capacity installed < 2 years (MW)	779.9	737.9	475.3	28.9	2022
Amount of wind turbines < 10 years	414	954	863	494	2725
Wind power capacity installed < 10 years (MW)	1240.1	2637.1	2066.4	1093.9	7037.5

6

Results & Discussion

The following chapter presents the results derived from this project. Once the data available for this case study has been analyzed, the next step consists of comparing the different machine learning-based predictive models for wind power production, alongside an optimal feature engineering and tuning of hyperparameters. Therefore, the outline of this chapter comprises three main sections. Firstly, introducing an overview of the results obtained from the deterministic models. Secondly, the performance of feature engineering to define the most suitable variables for this application, subject to computational effort. Lastly, the integration of these models into a probabilistic framework called SPinHy-NN, based on the novel SP-NN.

6.1. Deterministic forecasts

The following section provides results for the first part of the methodology described in Chapter 4. To compare the models proposed in this project, basic approaches are followed to serve as a benchmark. Hence, Table 6.1 provides an overview of the results for the similarity model and the physical-based model.

Table 6.1: Overview of results for benchmark predictive models (values in MW).

Models	Testing data	MAE SE1	MAE SE2	MAE SE3	MAE SE4
Similarity	01/02-2001	230.37	707.35	642.61	405.85
	03/12-2001	220.86	611.48	537.91	358.78
Physical	01/02-2001	226.84	587.73	613.94	360.01
	03/12-2001	213.93	489.84	517.18	348.34

From Table 6.1, both benchmark models shows similar results. In particular, the physical-based model returns slightly better predictions, but still fails to capture value in the training data. However, as previously stated, these models serve as a reference for future models. Figure 6.1 compares a forecast example of the physical-based model benchmark with the observed values.

The next group of models consists of the structures that require grid-like topology data, namely three CNN-based architectures: LeNet-5, VGG-16, and AlexNet. In this case, only WS10 is considered as input data. Table 6.2 shows an overview of the results for the CNN-models.

The CNN-based models show reasonable improvement concerning the benchmark. The most simple CNN architecture, LeNet-5, shows the best MAE for price regions SE1, SE2, and SE3. The AlexNet architecture only performs better for the first segment of the testing data set corresponding to January-February 2001. In price region SE4, the VGG-16 based approach shows the most promising results, closely followed by the AlexNet and LeNet-5 based architectures. The more complex architectures

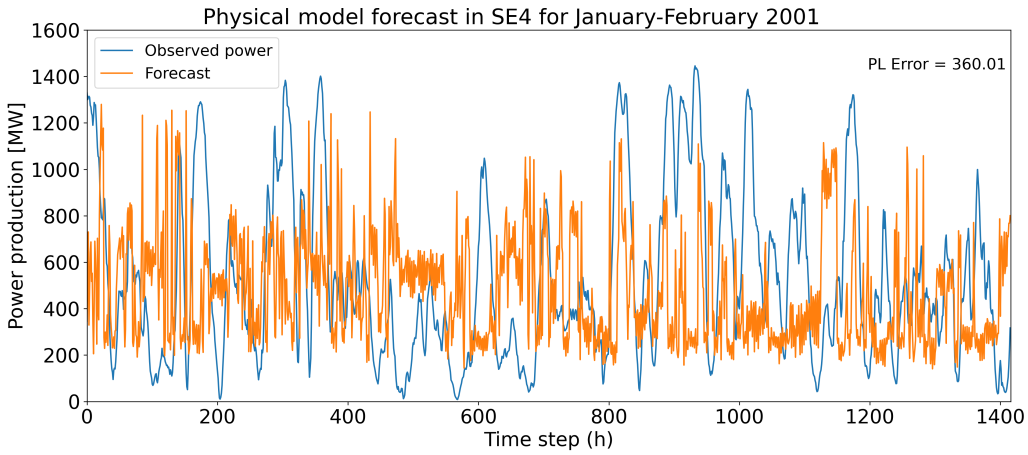


Figure 6.1: Physical-model forecast of price region SE4 in the period January-Febraury 2001, compared to the real observed value.

Table 6.2: Overview of results for CNN-based predictive models (values in MW).

Models	MAE SE1	MAE SE2	MAE SE3	MAE SE4
LeNet-5	99.68	326.73	258.78	153.35
	104.65	187.58	150.94	109.44
VGG-16	114.62	379.30	217.17	145.09
	123.91	206.36	160.23	100.43
AlexNet	102.66	238.51	238.51	152.24
	116.10	207.89	169.61	107.14

(i.e. AlexNet and VGG-16) generalized the data, failing to enhance the predictions despite their intricate structures. Moreover, the computational time for the LeNet-5 architecture was almost 2 hours. In other cases, it required 2.5 hours for the AlexNet architecture and over 6.5 hours for the VGG-16 (the most complex architecture), compiling for 200 epochs in every instance. In consequence, the LeNet-5 is used as the CNN branch to build a probabilistic framework for all price regions. This architecture reduces the error from the best benchmark model to 47%, 38%, 29%, and 31% of the initial predictions, respectively. Figure 6.2 compares a forecast example of the LeNet-5-based CNN model with the observed values.

The k-means clustering following a weather analog-based approach is evaluated. The first step consists of defining optimal k-clusters. The elbow method is used for the training data to define these values. In this case, two different input groups are considered. The first input group considers only WS10, while the second input group considers WS10, T2M, MSLP. On the other hand, only the first testing data group was used, namely January-February 2001. Figure 6.3 illustrates the results for different k-values in all price regions for the multiple-input case.

In the multiple-input case, no elbow was clearly defined, given the irregular shape of the results. However, the error follows a decreasing trend up to the last value tested (364), dropping until 241.37%, 69.98%, 55.09%, and 72.75%, respectively at each price region. In particular, price region SE1 did not manage to capture the spatial patterns of the clustering algorithm, while in other cases shows promising results.

Figure 6.4 illustrates the elbow method results for a single input (WS10), under the same conditions defined above. Again, the irregular shape did not facilitate the decision for an optimal k-cluster. However, in this case, the results improved respect to the multiple-input approach, reducing the percentage error to 140.63%, 63.42%, 48.94%, and 56.55%, respectively at each price region. In consequence, the single input k-means clustering shows the most promising approach to build the probabilistic forecast framework. In this case, K = 364 is selected.

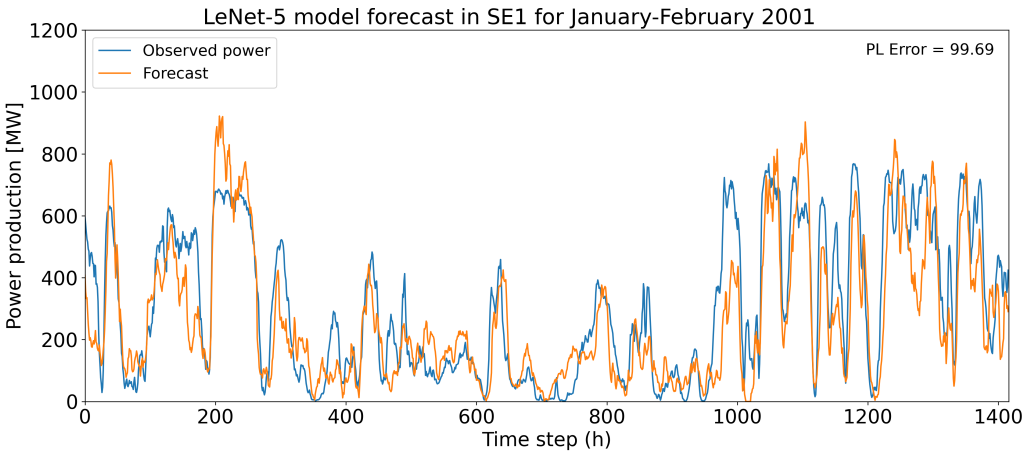


Figure 6.2: Physical-model forecast of price region SE1 in the period January-Febraury 2001, compared to the real observed value.

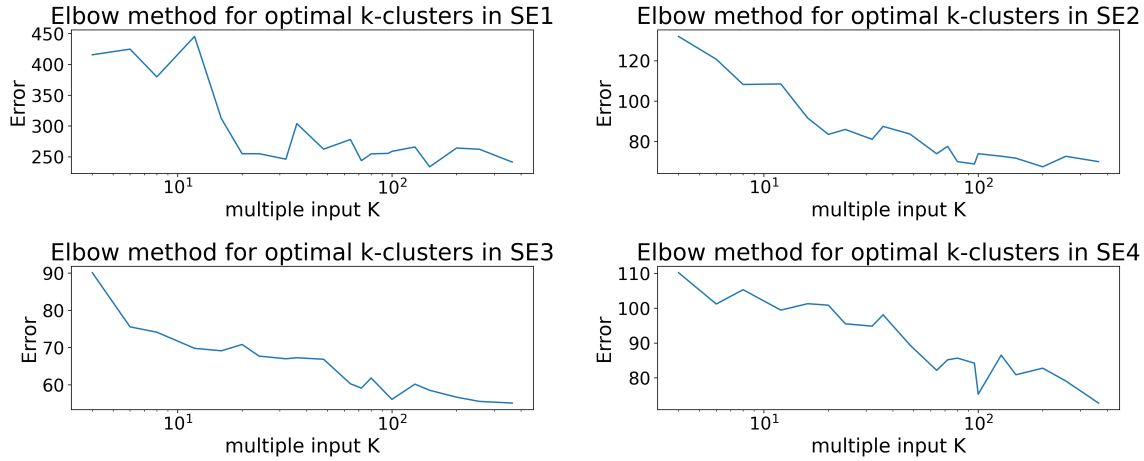


Figure 6.3: Elbow method for multiple inputs (WS10, T2M, MSLP) in all price regions.

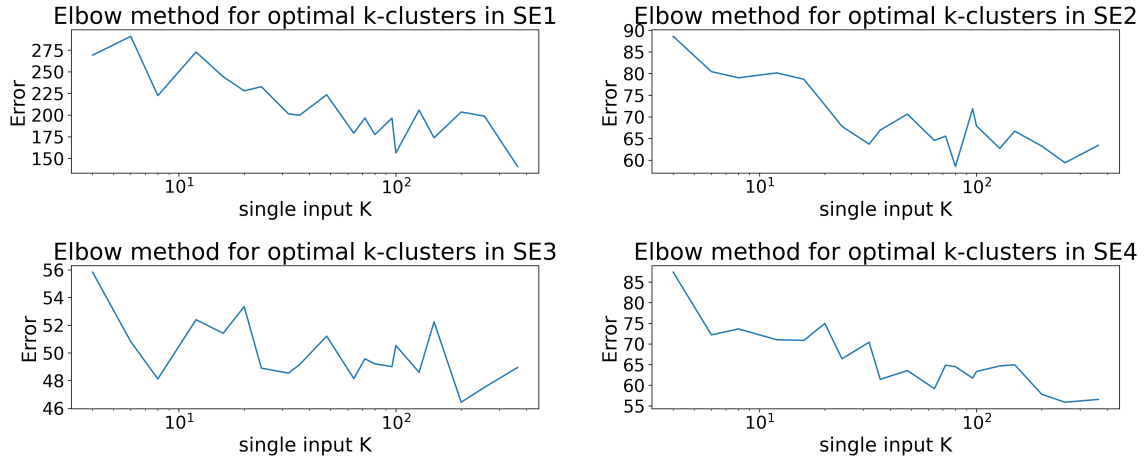


Figure 6.4: Elbow method for single input (WS10) in all price regions.

Using a single input approach with K = 364 relates to averaging the daily power production of the training data to predict future values based on similarity. Hence, the most similar wind conditions of a particular day are chosen for forecasting the production in the future. Table 6.3 shows the results for the analog-based approach.

Table 6.3: Results for analog-based approach using optimal k-clusters (values in MW).

Testing data	MAE SE1	MAE SE2	MAE SE3	MAE SE4
01/02-2001	163.77	400.07	355.37	190.31
03/12-2001	152.95	297.02	283.84	170.59

Comparing the results with benchmark models, the analog-based approach provides a significant improvement. However, it does not enhance the predictive model concerning the grid-like topology data approach (NWP images). In perspective, the analog-based inputs enhance the forecasts respect to the physical-based benchmark by 24%, 39%, 45%, and 51%, respectively at each price region, considering the second testing set (March-December 2001). On the other hand, the CNN-based model is 32%, 37%, 47%, and 36% better, respectively at each price region, considering the same data set. Figure 6.5 compares a forecast example of the analog-based approach model with the observed values.

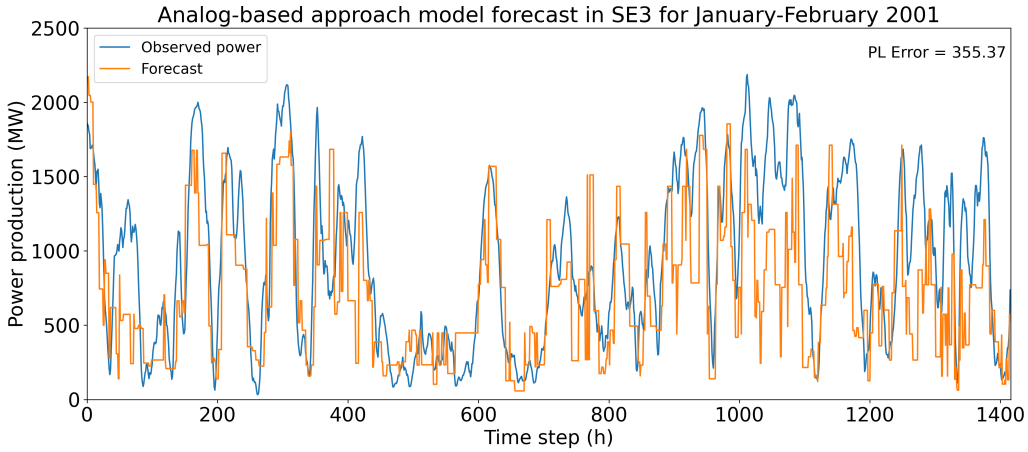


Figure 6.5: Analog-based approach model forecast of price region SE3 in the period January-February 2001, compared to the real observed value.

6.2. Feature engineering

The second main task was to determine the optimal features to introduce in the MLP branch. In this case, four MLP architectures are tested to assess the performance of such a model. In this scenario the goal was not to understand the optimization algorithm performance, thus an Early Stopping callback is implemented in case the iterative process does not improve the validation split after 20 iterations.

The first approach consisted of using the output of the physical model as input for the different MLP-based architectures. However, no learning was achieved. Hence, these results are excluded as they do not provide additional information. Also, feature engineering by combining different features with the results of the physical-based model output was not tested further, as the latter variable does not enhance the accuracy of the forecasts.

The second approach included the output of the analog-based approach. Using the output of the values provided by the k-means clustering algorithm, these are introduced as an input for the MLP-based architecture. Moreover, the single input optimal k-value is employed for these features, as they

provided the best performance given the elbow method. Table 6.4 shows the results for different MLP-based models using analog-based inputs.

Table 6.4: Overview of results for MLP-based predictive models using analog-based features (values in MW).

Models	Testing data	MAE SE1	MAE SE2	MAE SE3	MAE SE4
MLP-1	01/02-2001	129.63	382.88	360.23	194.32
	03/12-2001	145.34	299.23	277.78	172.21
MLP-2	01/02-2001	130.24	383.45	359.65	192.34
	03/12-2001	145.78	298.98	277.34	172.92
MLP-3	01/02-2001	129.73	382.72	359.81	192.45
	03/12-2001	146.12	298.22	279.02	172.38
MLP-4	01/02-2001	129.36	382.64	359.57	193.29
	03/12-2001	145.08	297.76	277.25	172.07

The analog-based inputs did not show relevant differences in respect to the original approach, meaning that these features did not learn significantly to relate the input features with the power production. Moreover, all architectures show almost identical results, implying that the architecture is not further enhancing the predictions. However, it exhibits a small improvement in comparison to the initial analog-based approach, except in SE4. Consequently, this is an encouraging feature for the MLP branch of the final architecture.

A more intricate combination of input features consists of the analog-based inputs and time dependencies. The objective is to capture the wind turbine availability and maintenance based on seasonal patterns learned from the training sample. Table 6.5 provides the results obtained combining the analog-based input with the temporal features.

Table 6.5: Overview of results for MLP-based predictive models using analog-based features and time dependencies (values in MW).

Models	Testing data	MAE SE1	MAE SE2	MAE SE3	MAE SE4
MLP-1	01/02-2001	133.54	400.02	367.06	202.16
	03/12-2001	142.20	285.38	270.27	170.08
MLP-2	01/02-2001	130.52	407.82	367.29	206.12
	03/12-2001	140.99	292.90	269.49	171.14
MLP-3	01/02-2001	131.09	404.64	369.05	190.37
	03/12-2001	136.90	283.14	272.67	168.11
MLP-4	01/02-2001	132.68	399.19	370.98	199.53
	03/12-2001	142.50	288.10	265.35	170.90

The results for this case improve with respect to the previous test, namely using only analog-based input features. However, the improvement in percentage terms is not remarkable, changing between 1% and 4%. Nevertheless, considering that the temporal dependencies do not promote any additional computational expense and provide a small improvement to a reasonably good predictive model, these features are kept to build the final framework. Figure 6.6 compares a forecast example providing analog-time inputs to the MLP-4 model with the observed values.

Regarding the architectures, using the time proxy produces more fluctuations in the results compared to the analog-base single input, which provides stagnant outcomes regardless of the architecture. Given the output similarity in all MLP-based models, MLP-4 is selected in this stage, as it is a tailor-made architecture design for this application, motivated by personal choice during this project. Hence, the remaining feature engineering is analyzed using this fully-connected architecture.

The last feature engineering approach consists of concatenating a CNN branch using NWP input with the MLP branch, using analog-based and time proxy inputs. In this setting, only the LeNet-5 based

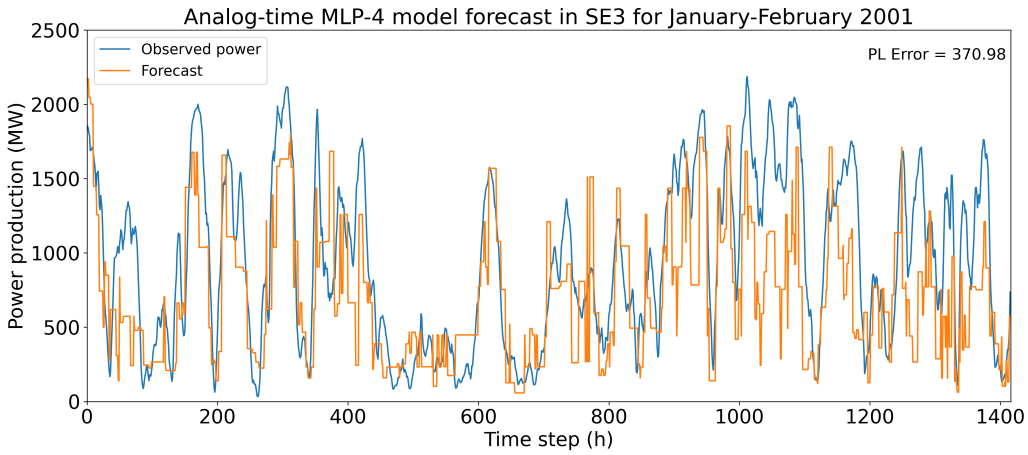


Figure 6.6: Analog-time MLP-4 model forecast of price region SE3 in the period January-February 2001, compared to the real observed value.

Table 6.6: Results of the hybrid model combining CNN-MLP architectures introducing NWP images, analog-based and time proxy data (values in MW).

Testing data	MAE SE1	MAE SE2	MAE SE3	MAE SE4
01/02-2001	109.11	350.72	257.81	159.81
03/12-2001	91.89	190.25	150.41	101.83

architecture and MLP-4 are considered as one hybrid architecture. Moreover, this hybrid structure represents the general model for the SPinHy-NN. Results for this approach appear in Table 6.6.

The results from the hybrid approach are similar to the LeNet-5 based approach. In SE3 and SE4, it manages to improve the accuracy in the period March-December 2001 (the second segment of the data set). On the other hand, for the first testing data set, all cases showed lower performance in comparison to the best CNN model. Figure 6.7 compares a forecast example of the hybrid model with the observed values.

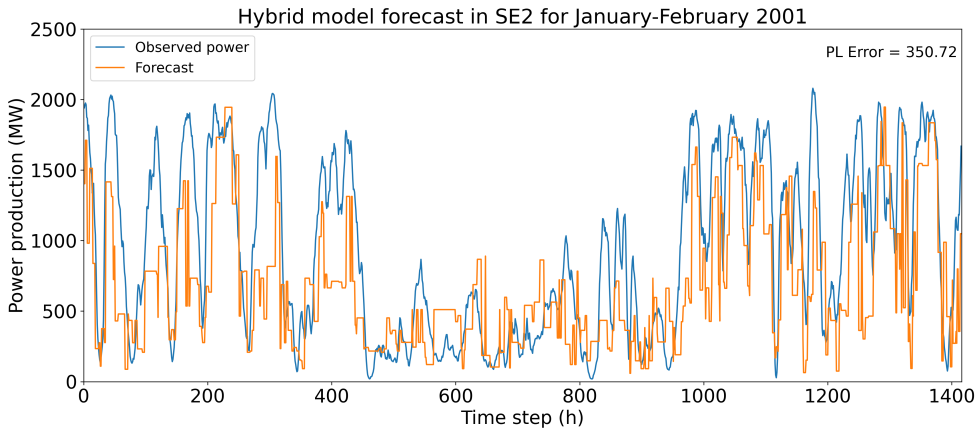


Figure 6.7: Hybrid model forecast of price region SE2 in the period January-February 2001, compared to the real observed value.

6.3. SPinHy-NN performance

The final step of the strategy consists of building a framework based on a hybrid architecture: the SPinHy-NN. Figure 6.8 illustrates the probabilistic framework of the final architecture.

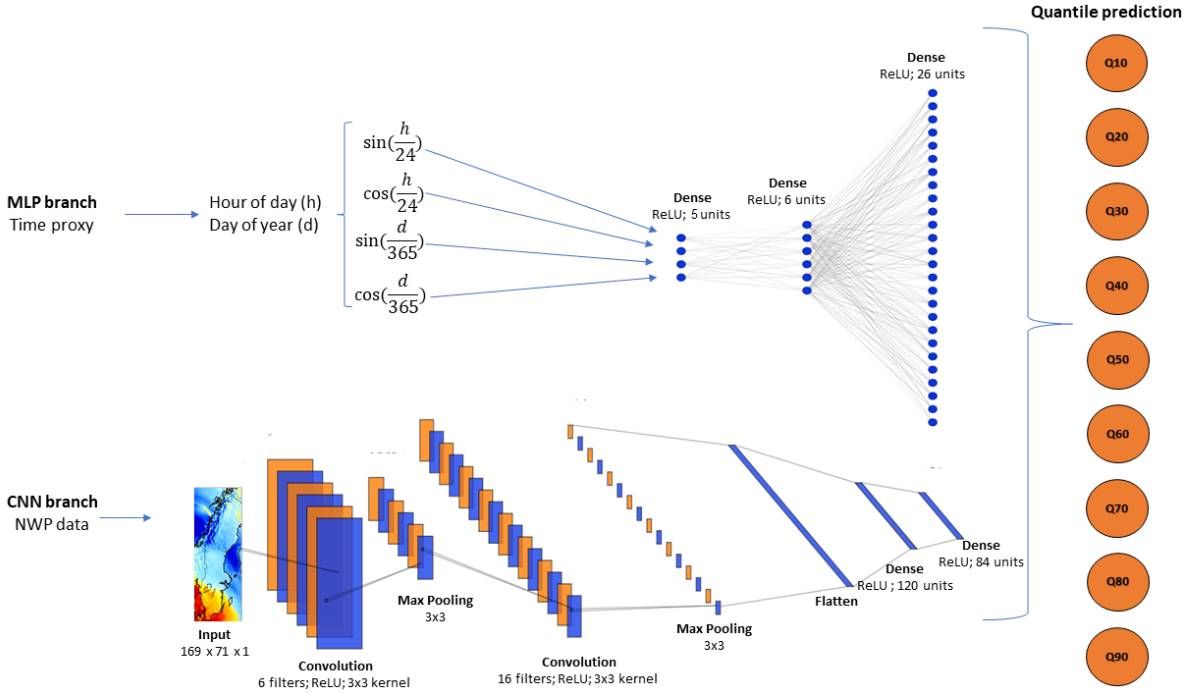


Figure 6.8: Structure of the SPinHy-NN predictive-based model probabilistic framework.

In this step, the final structure is tuned by the following hyperparameters: batch size, sub-sampling layer, spatial dropout, and quantile margin (ϵ). Table 6.7 shows the best results tested for January-February 2001, based on this sensitivity analysis. The error metric used for the probabilistic setting is the pinball loss (PL) function. These results indicate that every price region in Sweden is tuned using different hyperparameters. In particular, Model C fits SE2 and SE3, while Model E fits SE1 and SE4. Table 6.8 shows the results for the same models using the second part of the testing data set, corresponding to March-December 2001.

Based on the sensitivity analysis for both testing sets, various models are considered. In all cases, a Max Pooling sub-sampling layer approach provides the best results as a strategy to reduce the size of the feature maps. However, differences between the hyperparameters can be noted:

- In SE1, the first testing set shows better results with Model E; no spatial dropout, a smaller batch size of 32, and a quantile margin of 0.001. For the second testing data set, all parameters change in Model A; increasing the spatial dropout to 0.25, batch size of 64, and quantile margin of 0.002 produces the best results.
- In SE2, the first testing set performs better with Model C; no spatial dropout, batch size of 64, and small quantile margin of 0.001. The second testing data improves with Model B; increasing the spatial dropout to 0.25 while keeping the other parameters the same.
- In SE3, Model C shows the best results for both testing data sets.
- In SE4, Model E also shows the best results for both segments of the testing data set.

In general, the regions with the highest installed capacity performed better with a bigger batch size of 64, except in the case of SE1, which also shows a better performance for the second segment of the

Table 6.7: Overview of results based on tuning of hyperarameter values for each price region in January-February 2001 testing data set (values in MW).

SPinHy-NN Model description	PL SE1	PL SE2	PL SE3	PL SE4
Model A a. Max Pooling b. Spatial Dropout = 0.25 c. Batch size = 64. d. ϵ = 0.002	39.9	179.8	109.8	74.1
Model B a. Max Pooling b. Spatial Dropout = 0.25 c. Batch size = 64. d. ϵ = 0.001	38.9	184.2	117.4	77.3
Model C a. Max Pooling b. Spatial Dropout = 0 c. Batch size = 64. d. ϵ = 0.001	34.6	142.2	92.8	60.5
Model D a. Average Pooling b. Spatial Dropout = 0 c. Batch size = 64. d. ϵ = 0.001	34.9	143.3	106.0	59.6
Model E a. Max Pooling b. Spatial Dropout = 0 c. Batch size = 32. d. ϵ = 0.001	32.2	148.3	106.0	58.2

testing data set. Moreover, a quantile margin of 0.001 seems to exhibit better accuracy in all forecasts, except in the second testing data set of SE1. Lastly, the models perform better without spatial dropout terms, except in the second segment of the testing data set of SE1.

The same models are used introducing the analog-based approach input features as an alternative for comparison of two SPinHy-NN based architectures. The results are summarized in Table 6.9. However, since the Average Pooling approach did not show significant improvements, Model D was discarded from this analysis.

Despite the results obtained by including the analog-based production input features, these do not enhance the accuracy of forecasts respect to the time proxy dependencies. The CNN branch seems to dominate the neural network architecture, blocking the learning process of the MLP input features. Hence, in all price regions, the time proxy approach suits best these predictive models.

The final model forecasts are shown in Table 6.10. These values extract the best models from both

Table 6.8: Overview results of SPinHy-NN models for each price region in March-December 2001 testing data set (values in MW).

SPinHy-NN Model	PL SE1	PL SE2	PL SE3	PL SE4
Model A	34.4	137.6	64.6	47.7
Model B	35.1	137.4	65.2	50.7
Model C	37.8	149.8	57.4	40.9
Model D	112.4	274.1	296.3	108.1
Model E	41.0	138.5	59.3	39.7

Table 6.9: Overview of results of SPinHy-NN models for each price region in March-December 2001 testing data set considering analog-based data (values in MW).

SPinHy-NN Model	Testing data	SE1	SE2	SE3	SE4
Model A	01/02-2001	46.2	205.8	158.9	81.0
	03/12-2001	38.7	80.4	91.0	60.7
Model B	01/02-2001	52.2	207.5	152.3	80.1
	03/12-2001	37.8	79.2	97.2	59.4
Model C	01/02-2001	41.3	139.4	85.7	60.3
	03/12-2001	36.3	78.4	58.5	40.4
Model E	01/02-2001	39.5	149.8	86.7	59.4
	03/12-2001	39.6	71.9	59.1	39.9

cases evaluated in the probabilistic framework setting considering six rounds. The global score is given by the average of the best five rounds. Figures 6.9-6.14 visualize the resulting probabilistic forecasts for rounds 1-6 of the forecasting competition.

Table 6.10: Final pinball loss results of the SPinHy-NN framework in six subsets of 2001 (values in MW).

Round	R1	R2	R3	R4	R5	R6
SE1	39.97	42.70	33.92	41.76	33.19	46.94
SE2	149.78	83.55	69.62	102.60	72.06	83.44
SE3	92.79	62.53	55.87	79.92	62.08	83.00
SE4	58.20	55.65	35.34	50.95	46.39	61.41
Round score	85.19	61.11	48.69	68.81	53.43	68.70

6.3.1. Clipping factors

Historical output production data shows that most of the time the wind turbines are not operating at the maximum capacity factor. Hence, to avoid the overestimation of certain quantiles, a clipping factor was applied to avoid outliers in the forecasts. For instance, the power production in a particular time step cannot be higher than the installed capacity available at that moment. Therefore, upper quantiles are constrained such that $\hat{y} < IC_{max}$. Given the historical data, Table 6.11 shows the clipping factors applied in rounds 1-6.

Table 6.11: Clipping factors applied in rounds 1-6 based on historical data.

Round	R1	R2	R3	R4	R5	R6
SE1	0.877	0.896	0.900	0.872	1.00	1.00
SE2	0.827	0.855	0.874	0.792	0.832	0.852
SE3	0.946	0.826	0.891	0.866	0.929	0.902
SE4	0.904	0.885	0.763	0.795	0.888	0.886

Clipping factors between 76% and 95% have been applied to increase the accuracy of forecasts. Table 6.12 shows the percentage improvements for every round and each price region. In consequence, the application of the clipping factor enhanced these models, despite the unnatural strategy. It also implies that these models tend to overestimate predictions concerning the observed values. On the other hand, the same holds valid to underestimation. In this case, the power output must always have a positive output. Therefore, lower quantiles are constrained such that $\hat{y} > 0$. The improvement in clipping factors range from zero to 8.47%. Additionally, the regions that did not require clipping factors imply a recognition of the NWP data. As a result, given the framework proposed in this research project, the predictive model managed to properly capture intricate spatial patterns in rounds 3 and 5, while failing to recognize them in rounds 1 and 4. In the case of round 6, the performance did not show good results. However, all teams during the competition faced the same situation, relating the inaccuracy to the original NWP data of that period.

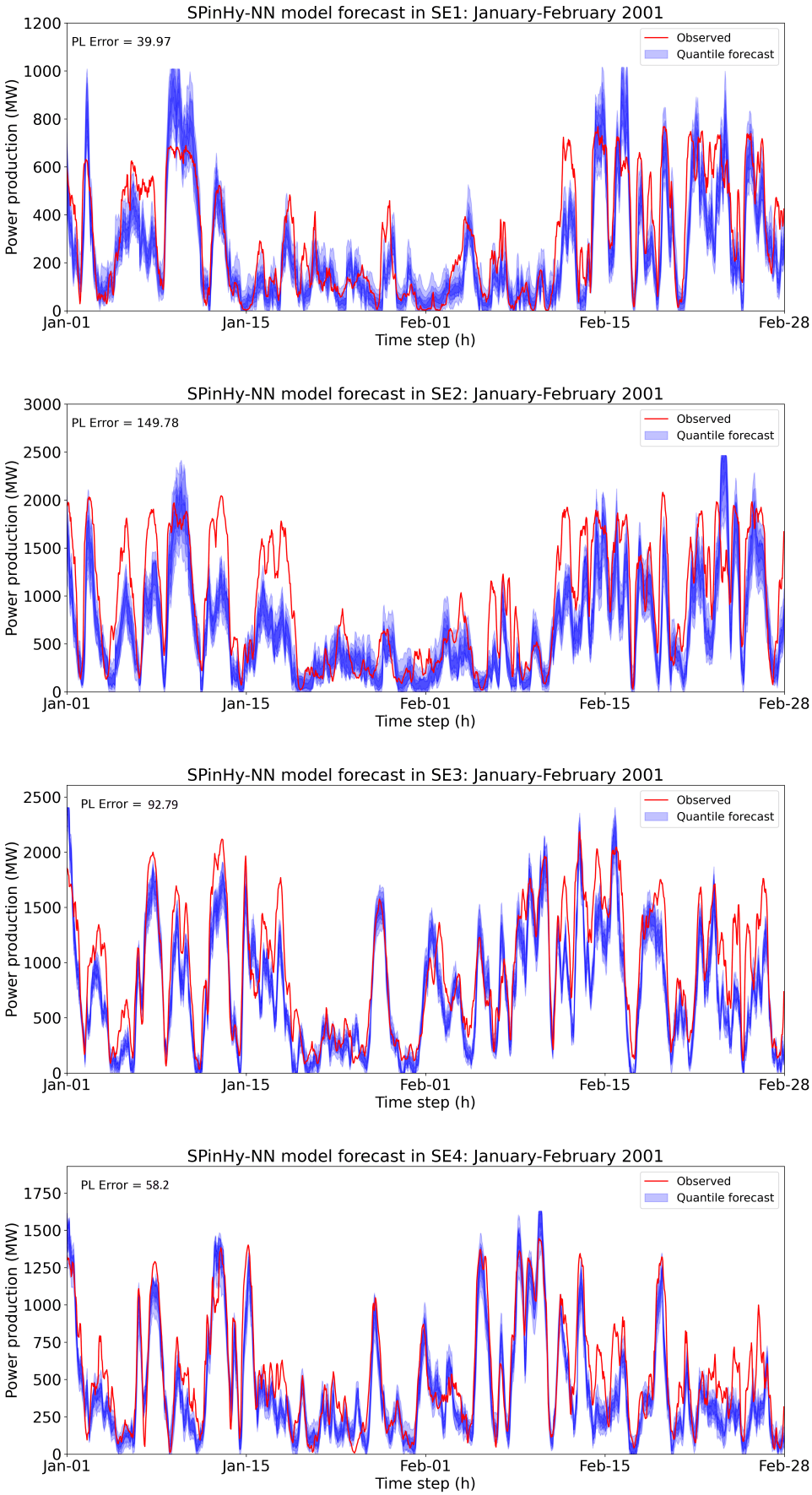


Figure 6.9: Resulting probabilistic wind power forecasts in Sweden for period January-February 2001.

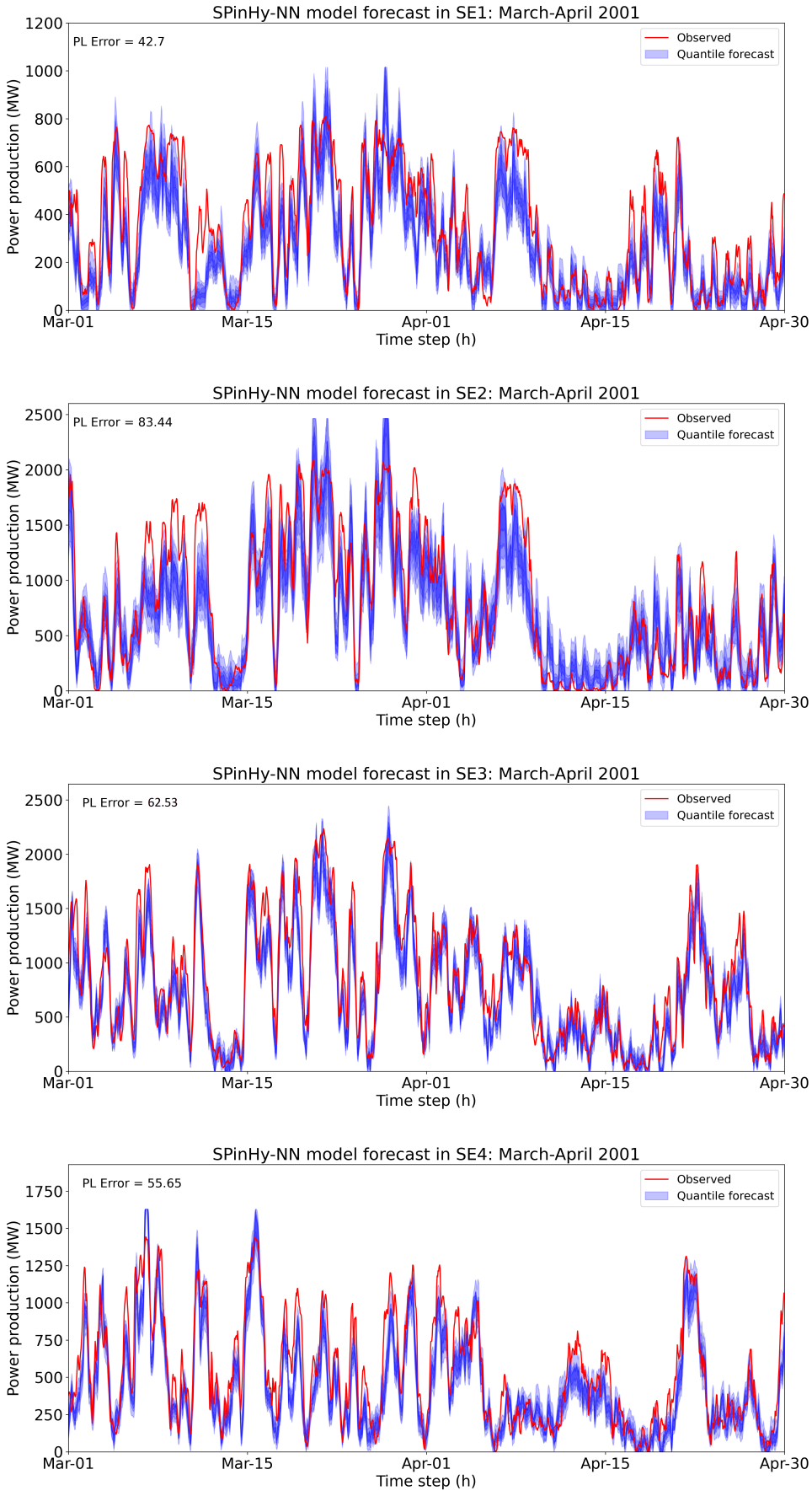


Figure 6.10: Resulting probabilistic wind power forecasts in Sweden for period March-April 2001.

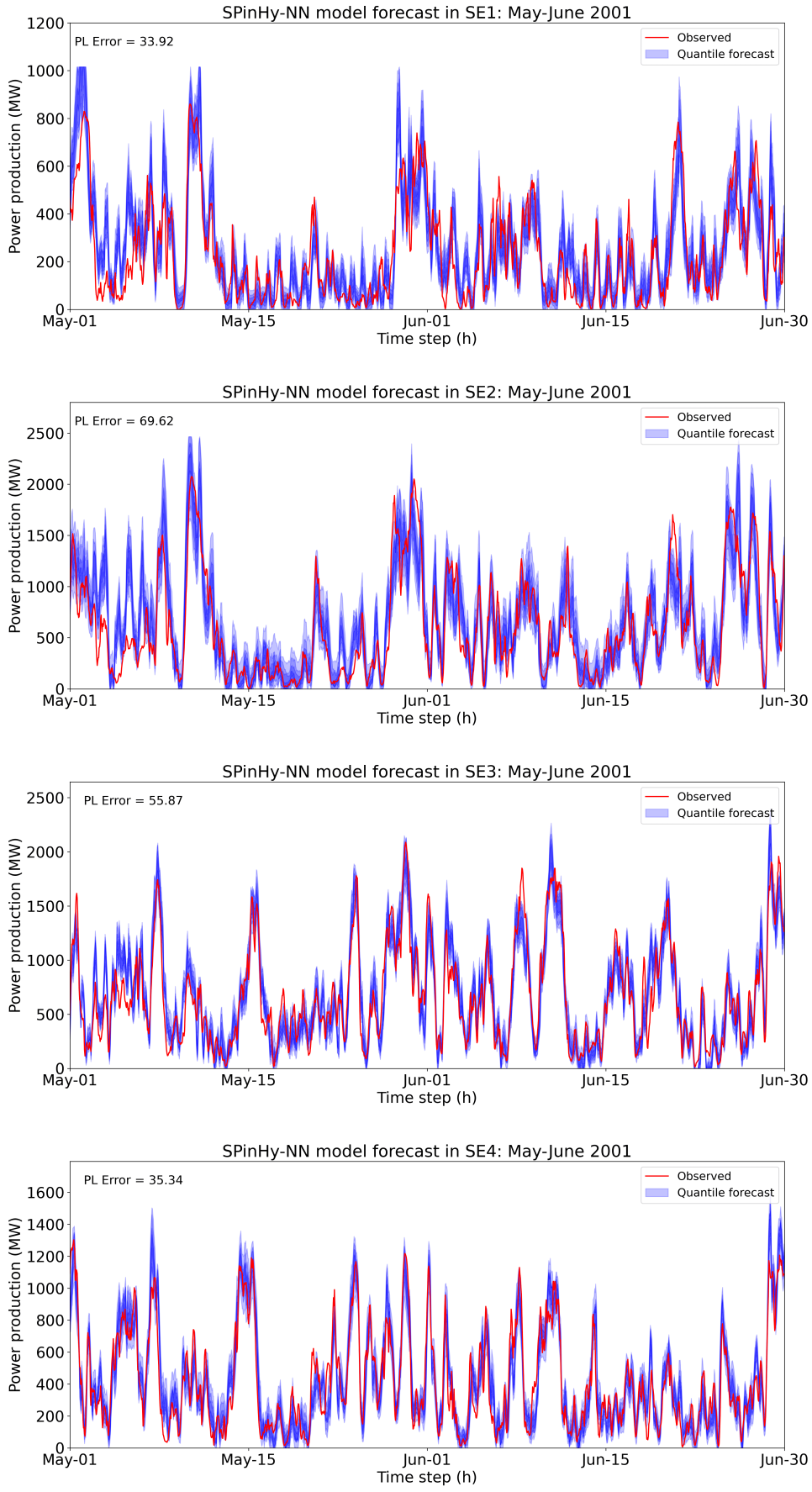


Figure 6.11: Resulting probabilistic wind power forecasts in Sweden for period May-June 2001.

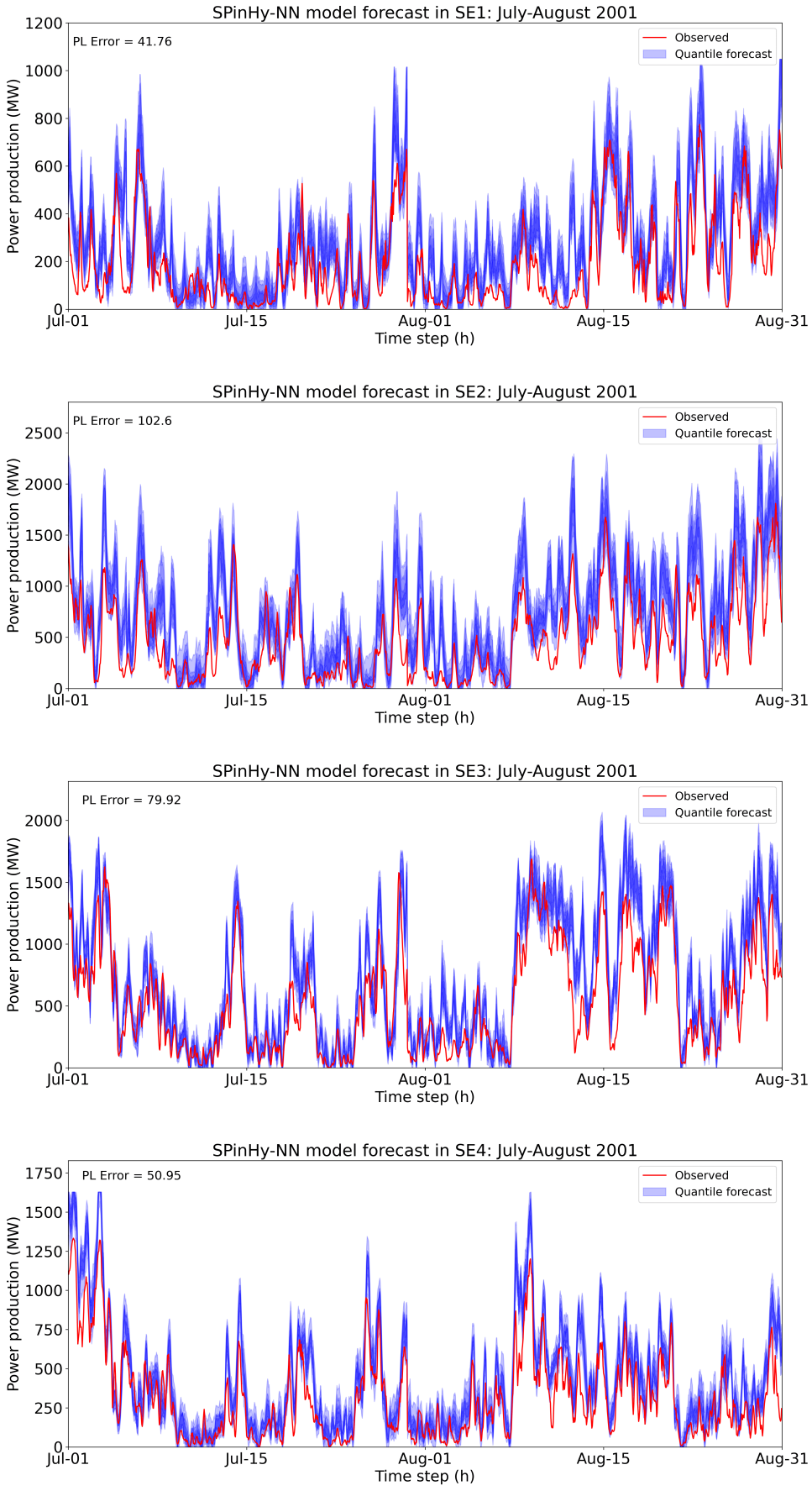


Figure 6.12: Resulting probabilistic wind power forecasts in Sweden for period July-August 2001.

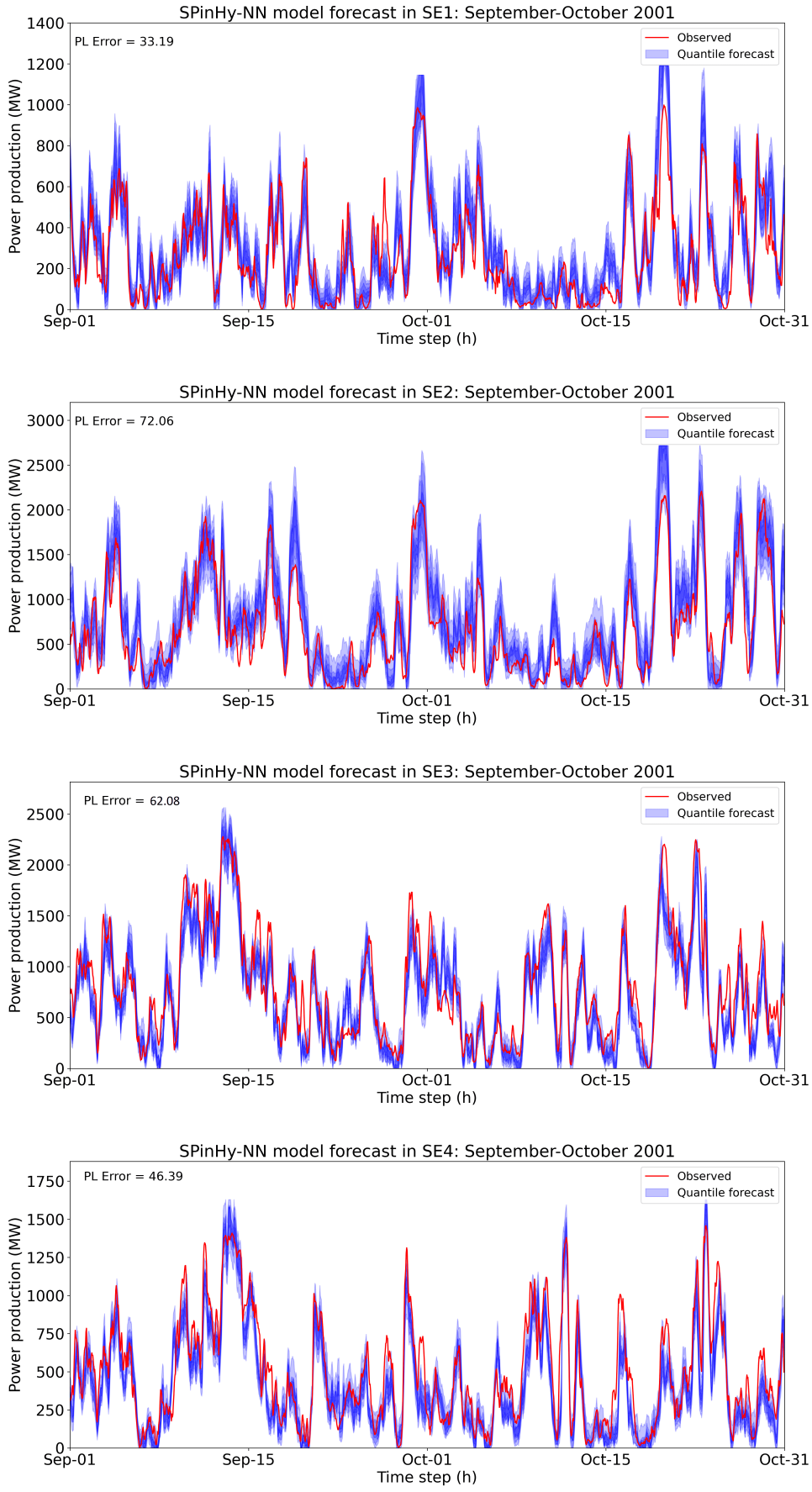


Figure 6.13: Resulting probabilistic wind power forecasts in Sweden for period September-October 2001.

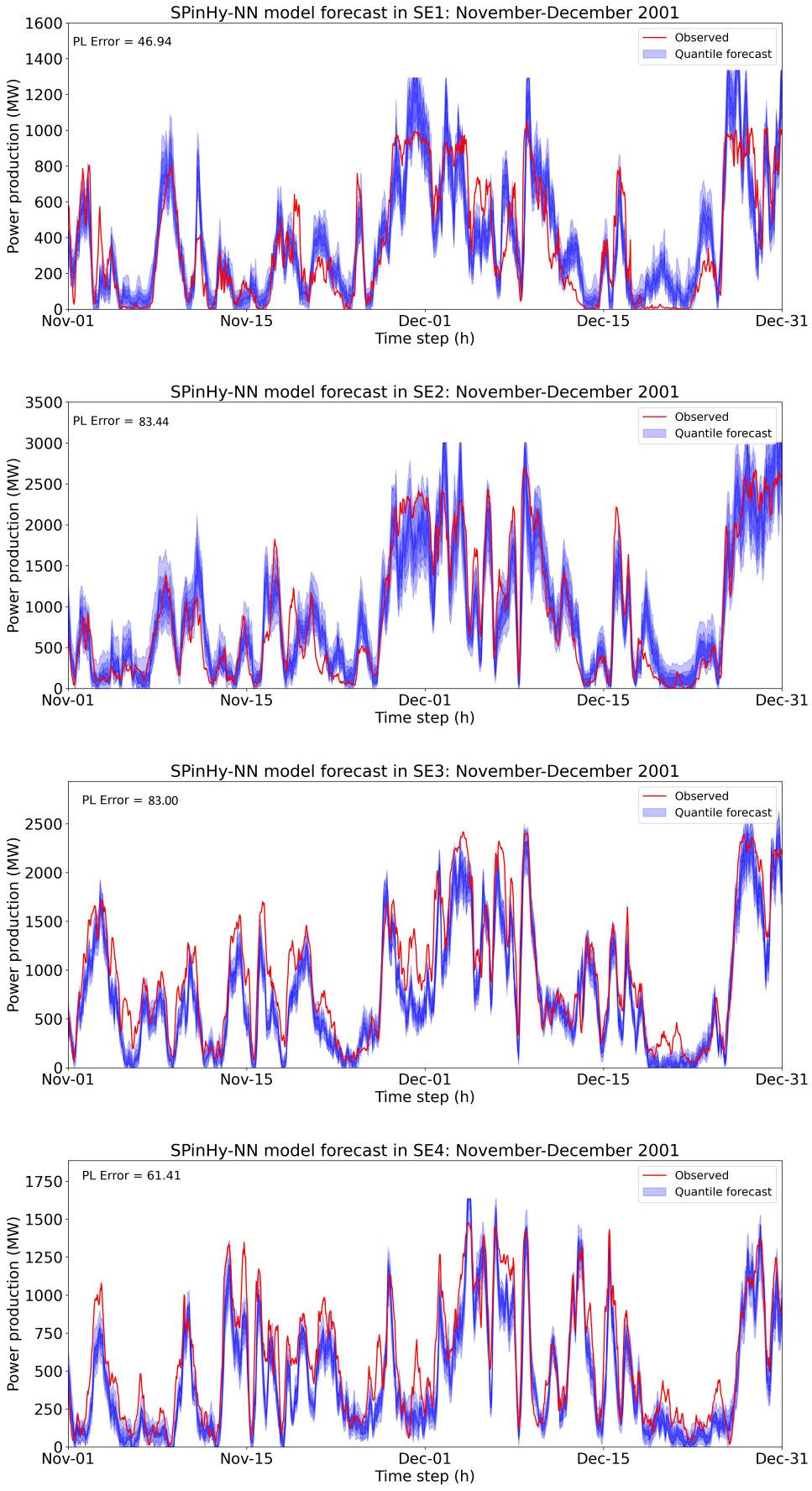


Figure 6.14: Resulting probabilistic wind power forecasts in Sweden for period November-December 2001.

Table 6.12: Improvement of the forecast round score based on clipping factor approach per round.

Round	R2	R2	R3	R4	R5	R6
Round score	5.70%	8.30%	0.37%	8.47%	0.00%	2.88%

6.3.2. Quantile cross-over

To assess the quality of the probabilistic framework, the CL metric is computed for every price region forecast. Table 6.13 shows CL and NC value for every round and at each price region. The SPinHy-NN probabilistic framework manages to deal with the quantile cross-over problem as stated in [49]. The total CL for each price region is 2657, 35818, 56295, and 42512, respectively. This validates the robustness of the non-parametric approach, as the average of CL/crossing [MW/crossing] at each price region is 4.4, 13.4, 12.1, and 8.8, respectively. Additionally, the NC represents 0.86%, 3.82%, 6.65%, and 6.91% of the predicted quantiles. In particular, SE1 has a lower CL and NC given the higher quantile margin (ϵ), showing that this parameter can enhance forecast consistency. Nevertheless, the predictive model hyperparameters were optimized to enhance accurate and sharp forecasts, considering the trade-off with the logical order of quantiles. In consequence, these crossings represent a minor inconsistency in a simple but robust methodology.

Table 6.13: Crossing loss (CL) and number of crossings (NC) results for every round at each price region.

Price region	Metric	R1	R2	R3	R4	R5	R6
SE1	CL	60	32	27	0	510	2028
	NC	34	23	9	0	154	383
SE2	CL	5079	4566	7065	8048	5695	5365
	NC	478	362	495	524	389	432
SE3	CL	8160	6888	8026	9649	10367	13205
	NC	719	625	740	747	802	1025
SE4	CL	6185	7297	6759	8279	6728	7264
	NC	731	787	771	871	817	864

6.3.3. Competition results

The post-competition model is compared to the top three teams of the competition. Figure 6.15 illustrates the results per round between the aforementioned models. The SPinHy-NN model exhibits reasonable results compared to other machine learning approaches except in round 4, where it was not able to capture the spatial patterns. The reasons for this behavior are not well understood, as the top teams improved their accuracy in this round. The final score for the top teams and the post-competition results obtained by the SPinHy-NN are: (1) MINES ParisTech: 44.92; (2) Univ. of Strathclyde: 47.93; (3) TU Delft, Turbulence: 51.52; (-) SPinHy-NN: 57.54. Refer to the conference paper in Appendix D.

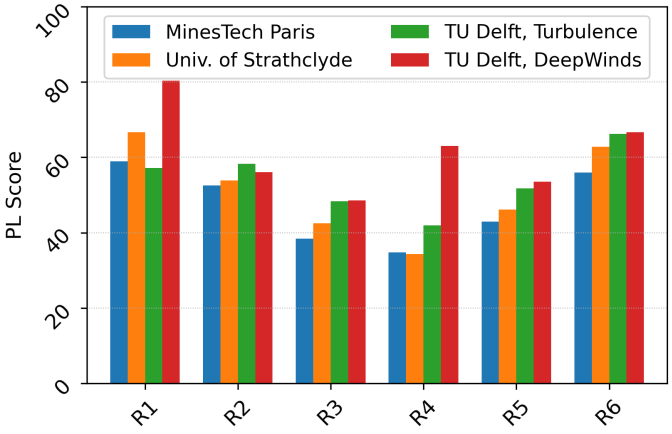


Figure 6.15: EEM20 forecasting post-competition results by round.

7

Conclusions & Recommendations

The problem statement at the beginning of the report stressed the need for accurate forecasting tool, given the increasing penetration of RES. The rise of variable generation in the power system can increase operational uncertainty. The objective of this project was to employ machine learning techniques to enhance wind power forecasting, applying them to different regions and promoting an alternative methodology, understanding its advantages and limitations.

The case study consisted of the wind turbine aggregated power hourly production of Sweden for its four price regions (SE1, SE2, SE3, SE4) in the period 2000-2001. The setting provided three main data sets: NWP data, a wind turbine record, and aggregated power output based on region. The approach comprised the use of data-driven models to create an alternative probabilistic forecasting framework. The main steps of this strategy consisted of evaluating different deep learning architectures in a deterministic approach, extensive feature engineering for optimal input values, and integration of a probabilistic-based predictive model. The outcome reached from this methodology was named **Smooth Pinball Hybrid Neural Network (SPinHy-NN)**.

The SPinHy-NN is a simple non-parametric deep learning-based approach for wind power probabilistic forecasting. It can deal with non-convex optimization problems, manages the quantile cross-over problem, and shows reasonable results in comparison to other methods. The difference between this architecture and the novel SP-NN is the concatenation of a CNN with a customized MLP architecture design. The parameters of this framework were tuned to achieve the most accurate forecasting results. Each price region climate was trained following this framework, using a particular hyperparameter tuning in each case.

The most suitable CNN is a simple LeNet-5 based architecture. More complex CNN architectures such as the VGG-16 and the AlexNet managed to capture intricate spatial patterns during the testing phase, but they did not perform better than the structure used for the final models. Moreover, the LeNet-5 is computationally more efficient, requiring less time to output the values of the predictive model. This represents a supplementary outcome of the original research objectives of this project.

The final results of this framework are highlighted in Table 7.1. It shows the pinball score for six groups of the testing data (2001), every group defined by two months of the year. These results correspond to the best tuned models developed post-competition.

The global results of this approach are compared with the best teams of the EEM20 forecasting competition. The top three teams, namely MINES ParisTech, University of Strathclyde, and TU Delft, scored 44.92, 47.93, and 51.52, respectively. Hence, the simple framework proposed in this project showed reasonable good predictions compared to other techniques for all periods, except for round 4, obtaining a post-competition final score of 57.54.

This framework tackled the main challenges of the competition, namely big data, wind turbine availability, and dynamic installed capacity. The simple yet robust SPinHy-NN probabilistic framework model

Table 7.1: Post-competition results of the SPinHy-NN framework in six subsets of 2001 (values in MW).

Round	R1	R2	R3	R4	R5	R6
SE1	31.90	33.57	34.55	43.88	33.19	44.29
SE2	139.30	73.06	69.16	94.63	72.06	79.18
SE3	92.08	62.36	55.55	73.01	62.08	82.33
SE4	58.07	55.16	34.75	40.39	46.39	61.10
Global score	80.34	56.04	48.51	62.98	53.43	66.72

can be used in future works based on grid-like topology input data and additional features to capture spatial patterns and operational features, respectively. Moreover, tuning these models separately for each price region turned out to be a good strategy. Finally, this proposal successfully managed to minimize the quantile cross-over problem, which is a typical issue in non-parametric approaches.

On the other hand, the SPinHy-NN framework has limitations. First, it serves as a framework to capture only spatial patterns. As can be seen from the results, the CNN branch dominated the output of the forecast. Therefore, feature engineering is limited to selecting the most optimal meteorological images. Second, the probabilistic approach is purely based on the pinball loss objective function defined by the target quantiles. The architecture does not run stochastic simulations, reducing the reliability in particular cases (such as in round 4 of the forecasting competition). Moreover, the structure does not make use of ensemble members, which were available for this competition setting.

The recommendations for future work are summarized as follows:

1. Incorporation of satellite imaging as input data. During this research project, only NWP data was used from the models provided by MET Norway. Therefore, The substitution of simulated weather data by real images is a promising research direction for short-term wind power forecasting.

2. A basic approach for wind turbine availability was proposed. The time proxy did not seem to fully capture the operational constraints of the case study. Hence, it is suggested to complement this framework with another machine learning approach for anomaly detection.

3. The physical-based model was used as a benchmark, establishing elementary assumptions that have an impact on the final forecast. In particular, an α = 0.3 was assumed in all locations, neglecting the terrain height and additional effects. The competition allowed only to use the data provided by organizers, restricting the possibility of exploring external information. However, future work can enhance the physical-based model with additional data on the surface conditions surrounding every wind turbine.

4. The SPinHy-NN is a simple and reliable tool for probabilistic forecasting. However, based on the limitations, it is suggested to evaluate different probabilistic methodologies to enhance the accuracy of these forecasts.

5. In this project, the full grid size of the NWP data was used. It is encouraged to evaluate different grid sizes for every price region. This is supported by a correlation matrix between input-output, following mutual information theory.

6. Spatio-temporal models were not fully covered. The proposed tool focused on capturing spatial patterns. However, it is still not capable of capturing temporal patterns that govern atmospheric dynamics. Using novel convolution layers such as the *ConvLSTM* architecture –a combination of CNN and LSTM–, could further improve the accuracy of these forecasts. In this case, identifying the optimal window time is a relevant study.

7. Ensemble models were not part of the scope of this thesis. However, it is known that decision-makers work with various forecasts of different nature. Hence, the integration of multiple models to enhance the reliability of prediction is a possible next step.

A

Backpropagation algorithm intuition

The following derivation is obtained from Andrew Ng, lecturer of the Machine Learning course available in Coursera [37]. Also, acknowledgements to Aditya Saini for these demonstrations [62]. He provides an intuition of the backward propagation algorithm (backpropagation). It concerns the intuition behind the gradient computation $\nabla_{ij}^{(l)}$ of the objective (cost) function $\theta_{ij}^{(l)}$. Furthermore, this derivation is used following a conventional gradient descent technique to simplify the explanation and understand the reasoning of the errors ($\delta_i^{(l)}$). Furthermore, a classical non-linear activation function is assumed.

The starting point is the derivative. The gradient is defined as:

$$\nabla_{ij}^{(l)} = \frac{\partial C}{\partial \theta_{ij}^{(l)}} \tag{A.1}$$

This formula cannot be solved directly. Hence, it has to be modified using two methods to derive a formula that can be computed by the neural network model. This final applicable formula is:

$$\nabla_{ij}^{(l)} = \theta^{(l+1)^T} \delta^{(l+1)} \cdot * \left(a_i^{(l)} \left(1 - a_i^{(l)} \right) \right) * a_j^{(l-1)} \tag{A.2}$$

The first method is based on the idea that the gradient can be written using $\delta_i^{(l)}$:

$$\nabla_{ij}^{(l)} = \delta_i^{(l)} * a_j^{(l-1)} \tag{A.3}$$

Where $\delta_i^{(l)}$ is defined by the partial derivative of the cost function respect to every feature or variable, as follows:

$$\delta_i^{(L)} = \frac{\partial C}{\partial z_i^{(l)}} \tag{A.4}$$

The second method is based on the relation between $\delta_i^{(l)}$ and $\delta_i^{(l+1)}$, or adjacent errors in the network, following chain rule to communicate the updates and pass the information to other nodes:

$$\delta_i^{(l)} = \theta^{(l+1)^T} \delta^{(l+1)} \cdot * \left(a_i^{(l)} \left(1 - a_i^{(l)} \right) \right) \tag{A.5}$$

As a result, a derivation for both methods is obtained, adapting the general form of backpropagation shown in Chapter 2.

$$\nabla_{ij}^{(l)} = \theta^{(l+1)^T} \delta^{(l+1)} \cdot * \left(a_i^{(l)} \left(1 - a_i^{(l)} \right) \right) * a_j^{(l-1)} \tag{A.6}$$

Demonstration of Equation A.3

Initially, it was defined that:

$$\nabla_{ij}^{(l)} = \frac{\partial C}{\partial \theta_{ij}^{(l)}} \tag{A.7}$$

Using the chain rule for higher dimensions enables re-writting it to the following expression:

$$\nabla_{ij}^{(l)} = \sum_{k} \frac{\partial C}{\partial z_{k}^{(l)}} * \frac{\partial z_{k}^{(l)}}{\partial \theta_{ij}^{(l)}} \tag{A.8}$$

On the other hand:

$$z_{k}^{(l)} = \sum_{m} \theta_{km}^{(l)} * a_{m}^{(l-1)} \tag{A.9}$$

Thus, the following expression can be given:

$$\frac{\partial_{k}^{(l)}}{\partial \theta_{ij}^{(l)}} = \frac{\partial}{\partial \theta_{ij}^{(l)}} \sum_{m} \theta_{km}^{(l)} * a_{m}^{(l-1)} \tag{A.10}$$

By linearity of the differentiation [(u + v)' = u'+ v']:

$$\frac{\partial z_{k}^{(l)}}{\partial \theta_{ij}^{(l)}} = \sum_{m} \frac{\partial \theta_{km}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{m}^{(l-1)} \tag{A.11}$$

$$if k, m \neq i, j, \frac{\partial \theta_{km}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{m}^{(l-1)} = 0 \tag{A.12}$$

$$if k, m = i, j, \frac{\partial \theta_{km}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{m}^{(l-1)} = \frac{\partial \theta_{ij}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{j}^{(l-1)} = a_{j}^{(l-1)} \tag{A.13}$$

Then for k=i:

$$\frac{\partial z_{i}^{(l)}}{\partial \theta_{ij}^{(l)}} = \frac{\partial \theta_{ij}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{j}^{(l-1)} + \sum_{m \neq j} \frac{\partial \theta_{im}^{(l)}}{\partial \theta_{ij}^{(l)}} * a_{j}^{(l-1)} = a_{j}^{(l-1)} + 0 \tag{A.14}$$

Finally:

$$\frac{\partial z_{i}^{(l)}}{\partial \theta_{ij}^{(l)}} = a_{j}^{(l-1)} \tag{A.15}$$

The first expression of the gradient $\nabla_{ij}^{(l)}$ resuts in:

$$\nabla_{ij}^{(l)} = \frac{\partial C}{\partial z_{i}^{(l)}} * \frac{\partial z_{i}^{(l)}}{\partial \theta_{ij}^{(l)}} \tag{A.16}$$

Equivalent to:

$$\nabla_{ij}^{(l)} = \frac{\partial C}{\partial z_i^{(l)}} * a_j^{(l-1)} \tag{A.17}$$

Or, using A.4 and A.15:

$$\nabla_{ij}^{(l)} = \delta_i^{(l)} * a_j^{(l-1)} \tag{A.18}$$

Demonstration of Equation A.5

It was established that:

$$\delta_i^{(l)} = \frac{\partial C}{\partial z_i^{(l)}} \tag{A.19}$$

Again, using chain rule for higher dimensions:

$$\delta_i^{(l)} = \sum_k \frac{\partial C}{\partial z_k^{(l+1)}} \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} \tag{A.20}$$

Replacing $\frac{\partial C}{\partial z_k^{(l+1)}}$ by $\delta_k^{(l+1)}$:

$$\delta_i^{(l)} = \sum_k \delta_k^{(l+1)} \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} \tag{A.21}$$

Focusing on the $\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}}$ part:

$$z_k^{(l+1)} = \sum_j \theta_{kj}^{(l+1)} * a_j^{(l)} = \sum_j \theta_{kj}^{(l+1)} * g\left(z_j^{(l)}\right) \tag{A.22}$$

Then, an expression is derived based on $z_k^{(l)}$:

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = \frac{\partial \sum_j \theta_{kj}^{(l)} * g\left(z_j^{(l)}\right)}{\partial z_i^{(l)}} \tag{A.23}$$

By linearity of the derivation:

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = \sum_j \theta_{kj}^{(l)} * \frac{\partial g\left(z_j^{(l)}\right)}{\partial z_i^{(l)}} \tag{A.24}$$

If j ≠ i, then $\frac{\partial \theta_{kj}^{(l)} * g\left(z_j^{(l)}\right)}{\partial z_i^{(l)}} = 0$

As a consequence:

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = \frac{\theta_{ki}^{(l)} * \partial g\left(z_i^{(l)}\right)}{\partial z_i^{(l)}} \tag{A.25}$$

Then:

$$\delta_i^{(l)} = \sum_k \delta_k^{(l+1)} \theta_{ki}^{(l)} * \frac{\partial g\left(z_i^{(l)}\right)}{\partial z_i^{(l)}} \tag{A.26}$$

The activation function follows the following relationship, g'(z) = g(z)(1-g(z)), then:

$$\delta_i^{(l)} = \sum_k \delta_k^{(l+1)} \theta_{ki}^{(l)} * g\left(z_i^{(l)}\right)\left(1 - g\left(z_i^{(l)}\right)\right) \tag{A.27}$$

Finally, using vectorized notation:

$$\nabla_{ij}^{(l)} = \left[\theta^{(l+1)^T} \delta^{(l+1)} * \left(a_i^{(l)}\left(1 - a_i^{(l)}\right)\right)\right] * \left[a_j^{(l-1)}\right] \tag{A.28}$$

This final expression results in the same equation A.2

B

SPinHy-NN Python code implementation

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


#Import packages

import tensorflow as tf
import numpy as np
import random as rn
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

from netCDF4 import Dataset
from sklearn.preprocessing import StandardScaler
from keras.models import Model, Sequential
from keras.layers import Dense, Convolution2D, AveragePooling2D,
    MaxPooling2D, Dropout, Flatten, Input, concatenate
from keras.layers import SpatialDropout2D
from keras.layers import BatchNormalization
from keras import backend as K
K.set_floatx('float64')
from keras.callbacks import EarlyStopping, ModelCheckpoint


# In[ ]:


#Check if GPU is working properly

tf.config.list_physical_devices('GPU')


# In[ ]:
```

```
#Fix random seed for reproducibility

seed = 2

np.random.seed(seed)
rn.seed(seed)
tf.compat.v1.set_random_seed(seed)


# In[ ]:


#Call local directory to load data

get_ipython().run_line_magic('cd', '"C:\\Users\\Eric Lacoa Arends\\
    Documents\\MSc SET\\MSc Thesis\\Data\\NC files 2000"')


# # The Smooth Pinball Function and  Quantile Cross-over Penalty

# In[ ]:


#Defining the smooth pinball loss function and the penalty for cross-over
    frequency

def pinball_loss(y, q, tau, alpha = 0.001, kappa = 0, margin = 0):

    """
    :param y: target
    :param q: predicted quantile
    :param tau: coverage level
    :param alpha: smoothing parameter #see paper to understand how it
       works
    :param kappa: penalty term #to avoid cross-over of quantiles
    :param margin: margin for quantile cross-over #define a suitable
       margin of quantiles
    :return: quantile loss
    """

    # Calculate smooth pinball loss function
    error = (y - q)
    quantile_loss = K.mean(tau * error + alpha * K.softplus(- error /
       alpha))

    # Calculate cross-over penalty
    diff = q[:, 1:] - q[:, :-1]
    penalty = kappa * K.mean(tf.square(K.maximum(tf.constant(0, shape = (1
       , 1), dtype = tf.float64), margin - diff)))

    return quantile_loss + penalty


# In[ ]:
```

```
#Defining CL function

def CL_score(q,tau):
    diff = q[:, 1:] - q[:, :-1]
    penalty = np.sum(np.maximum(0,-diff))
    penalty2 = np.sum(np.array(diff) < 0)
    return penalty, penalty2


# # Additional Functions

# Load NWP data function

# In[ ]:


def load_weather(begin, finish):

    """
    :param begin: string start-date (YYYYMMDD)
    :param finish: string end-date (YYYYMMDD)
    """

    WS10_data = []
    WD10_data = []
    Gust_data = []
    time_data = []

    file = list(pd.date_range(start = begin, end = finish, freq = 'D').
       strftime('%Y%m%d') + 'T00Z.nc')
    for day in file:
        if day == '20000514T00Z.nc' or day == '20000926T00Z.nc' or day ==
           '20010730T00Z.nc':
            continue

        ds = Dataset(day, 'r')

        #Create spatial data

        Wind_U = ds.variables['Wind_U'][:, :, :, :]
        #Wind_U[Wind_U <= 1e-3] = 1e-3
        #Wind_U[np.isnan(Wind_U)] = 1e-3
        Wind_V = ds.variables['Wind_V'][:, :, :, :]
        #Wind_V[Wind_V <= 1e-3] = 1e-3
        #Wind_V[np.isnan(Wind_V)] = 1e-3

        #WS10 = np.sqrt(Wind_U ** 2 + Wind_V ** 2)
        WS10 = np.median(np.sqrt(Wind_U ** 2 + Wind_V ** 2), axis = 1)
        #WD10 = np.median(np.arctan(Wind_V / Wind_U), axis = 1)

        WS10_data.append(WS10)
        #WD10_data.append(WD10)

        #Create temporal data
```

```
        NumberDay = (dt.date(int(day[:4]), int(day[4:6]), int(day[6:8])) -
            dt.date(2000, 1, 1)).days + 1
        NumberDay_vector = NumberDay * np.ones(24).reshape(24, 1)
        Hours = np.arange(24)

        hour_cos = np.cos((Hours / 24) * 2 * np.pi).reshape(24, 1)
        hour_sin = np.sin((Hours / 24) * 2 * np.pi).reshape(24, 1)
        day_cos = np.cos((NumberDay_vector / 366) * 2 * np.pi)
        day_sin = np.sin((NumberDay_vector / 366) * 2 * np.pi)

        time_values = np.hstack((hour_cos, hour_sin, day_cos, day_sin))
        time_data.append(time_values)

    WS10_data=np.array(WS10_data, dtype = "float64").reshape(-1, 169, 71)
    #WD10_data=np.array(WD10_data).reshape(-1, 169, 71)
    time_data=np.array(time_data, dtype = "float64").reshape(-1, 4)

    return WS10_data, time_data


# Load (output) power data

# In[ ]:


def load_power(begin, finish, SE):

    power_data = []
    IC_data = []
    file = list(pd.date_range(start = begin, end = finish, freq = 'D').
       strftime('%Y%m%d') + 'T00Z.nc')
    for day in file:
        if day == '20000514T00Z.nc' or day == '20000926T00Z.nc' or day ==
           '20010730T00Z.nc':
            continue

        NumberDay = (dt.date(int(day[:4]), int(day[4:6]), int(day[6:8])) -
            dt.date(2000, 1, 1)).days + 1
        power = pd.read_csv('windpower_updated.csv').values
        start = (24 * NumberDay) - 24

        #prod = power[start:start + 24, SE]
        CF = power[start:start + 24, SE + 8]
        IC = power[start:start + 24, SE + 4]

        power_data.append(CF)
        IC_data.append(IC)

    power_data = np.array(power_data, dtype = "float64").reshape(-1, 1)
    IC_data = np.array(IC_data, dtype = "float64").reshape(-1, 1)

    return power_data, IC_data


# Feature scaling function
```

```
# In[ ]:


#Feature scaling: based on node[i,j] respect to m—examples

def feature_scaling(df):

    sc = StandardScaler()
    data_array = sc.fit_transform(df.reshape(—1, df.shape[1] * df.shape
       [2])).reshape(df.shape)
    return data_array


# Evaluate test loss

# In[ ]:


def quantile_loss(y, q, tau):
    quantiles = []
    N_tau = len(tau)
    for i in range(N_tau):
         diff = y—q[:,i].reshape(q.shape[0],1)
         pinball = np.maximum(tau[i] * diff, (tau[i] — 1) * diff)
         quantiles.append(pinball.T)
    quantiles = np.average(np.array(quantiles).T, axis = 0).reshape(1,
       N_tau)
    return quantiles


# # Smooth Pinball Hybrid Neural Network (SPinHy—NN)

# Step 1: Define dates and price region

# In[ ]:


#Define price region to forecast

SE = 1

#EM = 0


# In[ ]:


#Training data

begin_training = '20000101'
finish_training = '20000102'

WS_train , time_train = load_weather(begin_training, finish_training)
power_train, IC_train = load_power(begin_training, finish_training, SE)
WSnorm_train = feature_scaling(WS_train)
```

```
# In[ ]:


#Test data

begin_test = '20011101'
finish_test = '20011231'

WS_test, time_test = load_weather(begin_test, finish_test)
power_test, IC_test = load_power(begin_test, finish_test, SE)
WSnorm_test = feature_scaling(WS_test)


# In[ ]:



#Reshape to according dimensions of the neural network

x_train = WSnorm_train
x_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.
    shape[2], 1)
t_train = time_train
y_train = power_train

x_test = WSnorm_test
x_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2],
     1)
t_test = time_test
y_test = power_test


# Step 2: Define features of the forecast and the SPinHy—NN

# In[ ]:


#Define features of the forecast

tau = np.arange(0.1, 1.0, 0.1) #Vector of quantiles
N_tau = len(tau)  #Number of quantiles
N_PI = int(N_tau / 2)  #Prediction intervals (for plots)

N_train = x_train.shape[0] #Number of examples in training set
N_test = x_test.shape[0] #Number of examples in test set

dim_in_space = (x_train.shape[1], x_train.shape[2], x_train.shape[3]) #
    Shape of spatial features


# In[ ]:


#Define SPinHy—NN features
```

```
Lambda = 0.0001 # L2 regularization

# loss function parameters (no need to modify)
loss_param = {
    'tau': tau,
    'alpha': 0.0001,
    'kappa': 1000,
    'margin': 0.000 #0.001
}


# Step 3: Build the NN architecture

# In[ ]:



def spinhy():

    #Inputs

    space = Input(shape = dim_in_space, name = 'space')
    time = Input(shape = (4,), name = 'time')

    #CNN (LeNet-5)-based architecture for spatial recognition

    conv = Sequential()
    conv.add(Convolution2D(6, (3, 3),
                           activation = 'relu',
                           input_shape = dim_in_space,
                           kernel_initializer = 'normal'))
    conv.add(MaxPooling2D())

    conv.add(Convolution2D(16, (3, 3),
                           activation = 'relu',
                           kernel_initializer = 'normal'))
    conv.add(MaxPooling2D())

    #Flatten to convert into a fully-connected layer

    conv.add(Flatten())
    conv.add(Dense(120, activation = 'relu',
                   kernel_initializer = 'normal'))

    conv.add(Dense(84))

    encoded_wind = conv(space)

    #MLP based architecture for temporal features

    mlp = Sequential()
    mlp.add(Dense(6, input_shape = (4,), activation = 'relu',
                  kernel_initializer = 'normal'))

    mlp.add(Dense(26))
```

```
    encoded_time = mlp(time)

    #Combine the outputs of both models to produce the quantiles of our
       probabilistic forecast

    merge = concatenate([encoded_wind, encoded_time])
    #merge = concatenate([conv.output, mlp.output], axis = -1)

    final = Dense(N_tau)(merge)

    #Define the resulting model SPinHy-NN

    model = Model(inputs = [space, time], outputs=[final])
    #model = Model([conv.input, mlp.input], [final])

    #Compile the model
    model.compile(loss=lambda Y, Q: pinball_loss(y = Y, q = Q, **
       loss_param), optimizer = 'Adam')

    return model


# Step 4: fit the model

# In[ ]:


#Callbacks

filepath = "bestweights-" + "SE" + str(SE) + ".hdf5"
es = EarlyStopping(monitor = 'val_loss', mode = 'min', patience = 5,
   verbose = 1)
cp = ModelCheckpoint(filepath, monitor = 'val_loss', verbose = 1,
   save_best_only = True, mode='min')


# In[ ]:


#Fit

model = spinhy()
model.summary()
#model.save("SpinHy")
#with tf.device('/cpu:0'):
#    model.fit([x_train, t_train], y_train, epochs = 10, verbose = 3,
#               batch_size = 64, shuffle = True, callbacks = [es, cp],
   validation_split = 0.25)


# Step 5: predict the test data

# In[ ]:
```

```
#Predict

model.load_weights('bestweights-SE1_sub4.hdf5')
#model.load_weights('bestweights-SE2_sub4.hdf5')
#model.load_weights('bestweights-SE3_sub4.hdf5')
#model.load_weights('bestweights-SE4_sub4.hdf5')
with tf.device('/cpu:0'):
    forecast = model.predict([x_test,t_test])
    forecast[forecast > 1] = 1
    forecast[forecast < 0] = 0
    q_hat = forecast * IC_test

q_hat.shape


# Step 6: evaluate the test data (if observed values are available)

# In[ ]:


#Evaluate
evaluate = quantile_loss(IC_test*power_test, q_hat, tau)
crossloss, crossing = CL_score(q_hat,tau)
print(evaluate)
print('The average pinball error is: ', np.mean(evaluate))
print('The CL loss is: ', crossloss)
print('The number of crossing is: ', crossing)


#
# # Forecast Plot

# In[ ]:


plt.figure(figsize=(17,7))
plt.plot(y_test * IC_test, color = 'red', label = 'Observed')

months = 'November-December'
labels = ['Nov-01','Nov-15','Dec-01','Dec-15','Dec-31']
positions = [0,N_test/4,N_test/2,3*N_test/4,N_test]


for i in range(N_PI):
    if i == 1:
        lab = 'Quantile forecast'
    else:
        lab = str()
    y1 = q_hat[:, i]
    y2 = q_hat[:, - 1 - i]
    plt.fill_between(np.arange(N_test), y1, y2, color = 'blue', alpha = 1
       / N_PI, label = lab)
plt.title('SPinHy-NN model forecast in SE' + str(SE) +': '+ months +' 2001
   ', fontsize = 22)
plt.xlabel('Time step (h)', fontsize = 20)
plt.ylabel('Power production [MW]', fontsize = 20)
```

```
plt.xticks(positions, labels, fontsize =20)
plt.yticks(fontsize =20)
plt.ylim(ymin=0, ymax = np.max(q_hat) + 250)
plt.xlim(xmin=0,xmax=N_test)
plt.ylabel('Power production (MW)', fontsize = 20)
plt.figtext(.14, .83, "PL Error = " + str(round(np.mean(evaluate), 2)),
    fontsize = 16)
plt.legend(loc = 'BEST', fontsize = 15)
filetitle = 'spinhy_forecast_' + 'SE' + str(SE) + '_' + str(begin_test
    [4:6]) + str(finish_test[4:6])
plt.savefig(filetitle, dpi = 300)


# In[ ]:


#For SE1_wk2 0.66, wk3 [0.70,0.02(HIGH)], wk4 [0.88, 0.02], wk5 [0.90,
    0.005], wk6 sub2 0.65,0.02, wk7 sub2 0.70 0.002
#Fpr SE2_wk2 0.75, wk3 [0.80, 0.01(MID)], wk4 [0.88, 0.01], wk5 [0.90,
    0.005], wk6 sub4 0.73,0.01, wk7 sub4 0.75 0.02
#For SE3_wk2 0.82, wk3 [0.86. 0.005], wk4 [0.90, 0.005], wk5 [0.90, 0.005]
    , wk6 sub4 0.80,0.01, wk7 0.85 0.01 sub 4
#For SE4_wk2 0.88, wk3 [0.87, 0.005], wk4 [0.90, 0.005], wk5 [0.90, 0.005]
    , wk6 sub2 0.80,0.005

#The clipping should start at Q50, CF(Q50)+C*CF(Q50) [starting week 4]


# In[ ]:


plt.figure(figsize = (24, 15))

plt.xlim([0, 1464])
plt.ylim([0, 2200])
plt.ylabel('Wind Power (MW)')
plt.xlabel('Time (hour)')
positions = [0,360,720,1080,1440]
labels = ['May-01','May-15','June-01','Jun-15','June-30']
plt.xticks(positions, labels)
#period = 'Jan-Feb 2001'
title = 'SE' + str(SE) + ' Wind Power prediction in round 3 ' #+
    begin_test[4:6] + '/' + finish_test[4:6]
plt.title(title)
filetitle = 'SE' + str(SE) + '_' + begin_test[4:6] + finish_test[4:6] + '
    _final'
#plt.savefig(filetitle, dpi = 300)


# In[ ]:


#Corrections

#Power clipping based on experience
bound = 0.71
```

```
delta = 0.005

quant_CF = np.repeat(IC_test, N_tau, axis = 1)
limit_low = bound*np.ones((1,4)) #q10 to q50
limit_up = np.arange(bound, bound+delta*4.1, delta).reshape(1,5) #q50 to
    q90
limit = np.concatenate((limit_low, limit_up),axis = 1)

clipping = limit * quant_CF

#Clipping of the SpinHy forecast
forecast = np.minimum(q_hat, clipping)

#Negative value are physically impossible
forecast[forecast < 0] = 0

filecsv = 'SE' + str(SE) + '_' + begin_test[4:6] + finish_test[4:6] + '.
    csv'
#np.savetxt(filecsv, forecast, delimiter=",")
```

C

K-means clustering Python code implementation

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import datetime as dt
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import fetch_mldata
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.cluster import MiniBatchKMeans
from sklearn.metrics import homogeneity_score

import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

from netCDF4 import Dataset


# In[ ]:


#Call local directory to load data

get_ipython().run_line_magic('cd', '"C:\\Users\\Eric Lacoa Arends\\
    Documents\\MSc SET\\MSc Thesis\\Data\\NC files 2000"')


# # Defining functions

# In[ ]:
```

```
def load_weather(begin, finish):

    """
    :param begin: string start-date (YYYYMMDD)
    :param finish: string end-date (YYYYMMDD)
    """

    WS10_data = []
    WD10_data = []
    temp_data = []
    pres_data = []
    time_data = []

    file = list(pd.date_range(start = begin, end = finish, freq = 'D').
        strftime('%Y%m%d') + 'T00Z.nc')
    for day in file:
        if day == '20000514T00Z.nc' or day == '20000926T00Z.nc' or day ==
            '20010730T00Z.nc':
            continue

        ds = Dataset(day, 'r')

        #Create spatial data

        Wind_U = ds.variables['Wind_U'][:, :, :, :]
        #Wind_U[Wind_U <= 1e-3] = 1e-3
        #Wind_U[np.isnan(Wind_U)] = 1e-3
        Wind_V = ds.variables['Wind_V'][:, :, :, :]
        #Wind_V[Wind_V <= 1e-3] = 1e-3
        #Wind_V[np.isnan(Wind_V)] = 1e-3
        #Temp = np.median(ds.variables['Temperature'][:, :, :, :], axis =
            1)
        #Temp[np.isnan(Temp)] = 273.15
        #Pres = np.median(ds.variables['Pressure'][:, :, :, :], axis = 1)
        #Pres[np.isnan(Pres)] = 101325

        #WS10 = np.sqrt(Wind_U ** 2 + Wind_V ** 2)
        WS10 = np.median(np.sqrt(Wind_U ** 2 + Wind_V ** 2), axis = 1)
        #WD10 = np.median(np.arctan(Wind_V / Wind_U), axis = 1)

        WS10_data.append(WS10)
        #temp_data.append(Temp)
        #pres_data.append(Pres)
        #WD10_data.append(WD10)

        #Create temporal data

        #NumberDay = (dt.date(int(day[:4]), int(day[4:6]), int(day[6:8]))
            - dt.date(2000, 1, 1)).days + 1
        #NumberDay_vector = NumberDay * np.ones(24).reshape(24, 1)
        #Hours = np.arange(24)

        #hour_cos = np.cos((Hours / 24) * 2 * np.pi).reshape(24, 1)
        #hour_sin = np.sin((Hours / 24) * 2 * np.pi).reshape(24, 1)
```

```
        #day_cos = np.cos((NumberDay_vector / 366) * 2 * np.pi)
        #day_sin = np.sin((NumberDay_vector / 366) * 2 * np.pi)

        #time_values = np.hstack((hour_cos, hour_sin, day_cos, day_sin))
        #time_data.append(time_values)

    WS10_data=np.array(WS10_data, dtype = "float64").reshape(-1, 169, 71,
        1)
    #temp_data = np.array(temp_data, dtype ="float64").reshape(-1,169,71,
        1)
    #pres_data = np.array(pres_data, dtype ="float64").reshape(-1,169,71,
        1)
    #WD10_data=np.array(WD10_data).reshape(-1, 169, 71)
    #time_data=np.array(time_data, dtype = "float64").reshape(-1, 4)
    #weather_image = np.concatenate((WS10_data, temp_data, pres_data),
        axis = 3)

    return WS10_data


# In[ ]:


def load_power(begin, finish):

    power_data = []
    IC_data = []
    file = list(pd.date_range(start = begin, end = finish, freq = 'D').
        strftime('%Y%m%d') + 'T00Z.nc')
    for day in file:
        if day == '20000514T00Z.nc' or day == '20000926T00Z.nc' or day ==
            '20010730T00Z.nc':
            continue

        NumberDay = (dt.date(int(day[:4]), int(day[4:6]), int(day[6:8])) -
            dt.date(2000, 1, 1)).days + 1
        power = pd.read_csv('windpower_updated.csv').values
        start = (24 * NumberDay) - 24

        #prod = power[start:start + 24, SE]
        PP = power[start:start + 24, 1:5]
        IC = power[start:start + 24, 5:9]

        power_data.append(PP)
        IC_data.append(IC)

    power_data = np.array(power_data, dtype = "float64").reshape(-1, 4)
    IC_data = np.array(IC_data, dtype = "float64").reshape(-1, 4)

    return power_data, IC_data


# In[ ]:


#Feature scaling: based on node[i,j] respect to m-examples
```

```
def feature_scaling(df):

    sc = StandardScaler()
    data_array = sc.fit_transform(df.reshape(-1, df.shape[1] * df.shape
        [2])).reshape(df.shape)
    return data_array


# # Data set

# In[ ]:


#Load three datasets for training period
begin_training = '20000101'
finish_training = '20001231'

#NWP images
NWP_images= load_weather(begin_training, finish_training)

#Aggregated power production for training dataset labelling
PP2000, IC2000 = load_power(begin_training,finish_training)


# In[ ]:


#Feature scaling
NWP_images_norm = feature_scaling(NWP_images)


# In[ ]:


#Training data
x_train = NWP_images_norm
y_train = PP2000


# In[ ]:


#Reshaping input data
X_train = x_train.reshape(len(x_train),-1)
X_train.shape


# In[ ]:


#Load three datasets for test period
begin_test = '20010301'
finish_test = '20011231'

#NWP images test
```

```
NWP_images_test= load_weather(begin_test, finish_test)

#Aggregated power production for testing dataset labelling
PP_test, IC_test = load_power(begin_test,finish_test)


# In[ ]:


#Feature scaling
NWP_images_test_norm = feature_scaling(NWP_images_test)


# In[ ]:


#Testing data
x_test = NWP_images_test_norm
y_test = PP_test


# In[ ]:


#Reshaping input data
X_test = x_test.reshape(len(x_test),-1)
X_test.shape


# # Analog-based weather classification

# In[ ]:


total_clusters = 364
# Initialize the K-Means model
kmeans = MiniBatchKMeans(n_clusters = total_clusters, max_iter = 200,
    random_state =2)
# Fitting the model to training set
kmeans.fit(X_train)


# In[ ]:


weather_class = kmeans.labels_
weather_class = weather_class.reshape(weather_class.shape[0],1)


# In[ ]:


#Create Pandas dataframe of weather classes with the power production
matrix = np.concatenate((weather_class,y_train),axis = 1)
df = pd.DataFrame(matrix, columns = ['Weather Class','SE1','SE2','SE3','
    SE4'])
```

```
# In[ ]:


matrix = df.groupby('Weather Class').mean()
classification = np.array(matrix)


# In[ ]:


class_prediction = kmeans.predict(X_test)


# In[ ]:


power_prediction = classification[class_prediction,:]
power_prediction.shape


# In[ ]:


MAE = np.sum((np.abs(y_test-power_prediction)), axis =0)/power_prediction.
    shape[0]
```

D

EEM20 Conference Paper

The following section refers to the conference paper published for the European Energy Markets 2020 Conference. The event took place online (originally in Stockholm, Sweden), September 16-18 2020. The outcome of this research project was presented on Thursday 17 September 2020, during the Forecasting Competition session (17:00-18:30). The top three teams, namely, MINES ParisTech, University of Strathclyde, and TU Delft (this team was led by Sukanta Basu), presented their results in this session. The following conference paper is referred in the bibliography, under numbering [63].

Probabilistic wind power forecasting combining deep learning architectures

Eric Lacoa Arends*, Simon J. Watson†, Member *IEEE*, Sukanta Basu‡, and Bedassa Cheneka†
*Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology
Delft, The Netherlands
†*Faculty of Aerospace Engineering, Wind Energy Section*
‡*Faculty of Civil Engineering and Geosciences*
Corresponding author: s.j.watson@tudelft.nl

***Abstract*—A series of probabilistic models were bench-marked during the European Energy Markets forecasting Competition 2020 to assess their relative accuracy in predicting aggregated Swedish wind power generation using as input historic weather forecasts from a numerical weather prediction model. In this paper, we report the results of one of these models which uses a deep learning approach integrating two architectures: (a) Convolutional Neural Network (CNN) LeNet-5 based architectrure; (b) Multi-Layer Perceptron (MLP) architecture –with two hidden layers–. These are concatenated into the Smooth Pinball Neural Network (SPNN) framework for quantile regression. Hyperparameters were optimised to produce the best model for every region. When tuned, the re-forecasts from the model performed favorably compared to other machine learning approaches and showed significant improvement on the original competition results, though failed to fully capture spatial patterns in certain cases when compared to other methods.**

***Index Terms*—wind power forecasting, convolutional neural network, smooth pinball neural network, multilayer perceptron, numerical weather prediction**

I. Introduction

System operators face the challenge of integrating variable wind power into the grid and avoiding possible power imbalances by scheduling other dispatchable generation units and calling on reserve mechanisms [3]. Wind power forecasting serves as a means to facilitate the decision-making of these operators, providing a tool for risk management in electricity markets [4]. This has stimulated research into new methodologies to to make best use of Numerical Weather Prediction (NWP) products [5].

Machine learning and Artificial Intelligence (AI) have shown promise in the energy sector to assist data-driven decision making [3]. As the performance of computers improves and algorithms become more efficient, society is shifting to an era of energy digitalisation [6], where the use of Information and Communication Technology (ICT) plays a key role in the energy transition. AI has also become a favored tool to provide probabilistic wind power forecasts [1]. The use of a Convolutional Neural Network (CNN) model is described in [8] for wind power generation forecasting using NWP data, capturing spatial patterns from relevant meteorological variables. Moreover, a Smooth Pinball Neural Network (SPNN) model is presented in [8], where an alternative is proposed to the traditional quantile deep regression model. Both architectures motivated the development of a deep neural network model described in this paper, developed by *DeepWinds*, Team 18 of the European Energy Markets (EEM) 2020 forecasting competition.

This paper describes the *DeepWinds* model, how it was implemented and its accuracy during the various rounds of the EEM 2020 competition. Furthermore, there is discussion of the challenges in developing the model, including feature selection, application to multiple price regions (climates), and how to apply the model when installed capacity is changing over time.

The structure of the remaining part of this paper is as follows: in Section II, the competition is described and an overview of the measured and forecast data is given. The methodology for the deep neural network model is explained in Section III. Section IV presents an analysis of the competition results, highlighting the performance of the *DeepWinds* model with respect to other models in the competition. The conclusions are given in Section V.

II. Competition Setting

The EEM organizers hosted a day-ahead market forecasting competition, in which teams were asked to predict the aggregated wind power of four price regions in Sweden, using a probabilistic methodology. The competition was divided into six submission rounds with every round focusing on two months of onshore wind power output during 2001 for which day-ahead forecasts were to be produced. The data provided to the competitors in order to make the forecasts consisted of three elements:

- NWP data of seven different meteorological variables.
- Aggregated wind power from the four price regions in Sweden.
- A record of the wind turbines installed in Sweden over the period of interest.

978-1-7281-6919-4/20$31.00

The EEM 2020 edition started on May 5, 2020 and ended on June 9, 2020. The data were released the day after every round submission, giving one week to train the models and produce new results for the following round. The ranking was published three days after the submission deadline for every round.

The data corresponding to the year 2000 were made available in advance of the competition proper, allowing participants to train their initial models and develop their forecast strategy, depending on their approach.

This first part of the data were daily netCDF format NWP model output consisting of 24 hourly values of seven meteorological variables (2m temperature, 10 m zonal and meridional wind speed components, 10 m wind gust speed, mean sea level pressure, relative humidity and total cloud cover) for ten ensemble members on a 71 $\times$ 169 grid covering Sweden with a spatial resolution of 10 km $\times$ 10 km. The forecasts were generated by MET Norway and archived by Greenlytics.

The first challenge was how best to utilize the multi-dimensional dataset which contained 83,993 variables per hour, accounting for 8736 hours in year 2000 (May 14, 2000 and September 26, 2000 were missing). Data cleansing and dimensionality reduction strategies were necessary to develop a model which was not computationally prohibitive.

The second part of the data consisted of the aggregated power production for the four price regions in Sweden, defined as: SE1, SE2, SE3 and SE4. The competition required quantile day-ahead forecasts to be produced for these data. Therefore, a further challenge was to derive quantile forecasts from a trained single-value output. Furthermore, a decision was required between training a single model for all price regions or training separate models for each region.

The third part of the data corresponded to a record over time of installed wind turbines in Sweden as the power capacity increased significantly during the training and forecasting time horizon of the competition (2000–2001). Therefore, there was a challenge in capturing the dynamics of the changing wind power installed capacity. Moreover, this record contained 4004 turbines accounting for 8640 MW, while in reality 4099 wind turbines were installed by that time, accounting for 8984 MW. As a consequence, the record did not entirely represent the actual conditions in which the aggregated power output was based for every price region.

The share of the installed capacity by the end of 2001 was 15.4 %, 34.8%, 30.8% and 18.9%, for the four price regions SE1–SE4, respectively. At the same time, the share of the number of turbines was 11.8%, 27.8%, 36.2% and 24.2%, respectively. Furthermore, the average terrain height in each region was 348 m, 476 m, 170 m and 73 m, respectively. It is important to note that the highest terrain location is 1003 m in SE2. This information is relevant to understand the diversity of conditions in every price region which represented a challenge to produce accurate forecasts with the limited data provided.

The accuracy of the models was measured using the pinball loss function. In this edition of the competition, the prediction output was required to consist of nine deciles from Q10 to Q90. Equation (1) shows the formula for the pinball loss function, ρ_k^i, evaluated for each price region:

$$\rho_k^i\left(q_k^i, y_k\right) = \begin{cases} (i/100)\left(y_k - q_k^i\right), & y_k \geq q_k^i \\ (1 - i/100)\left(q_k^i - y_k\right), & q_k^i < y_k \end{cases} \quad (1)$$

where i represents the percentile to be assessed (between 10 and 90), q_k^i is the predicted power value and y_k is the observed power value at time step k. Note that this formula only gives positive values. One of the purposes of this loss function is to properly penalize over- and under-estimates [8]. The final forecast score is calculated by averaging the pinball loss function over all percentiles, price regions and time steps for the particular two-month period.

III. Methodology

The deep learning-based (*DeepWinds*) forecasting model was developed in several stages: (a) Data cleansing; (b) Feature engineering; (c) Target engineering; (d) Development of a probabilistic framework; (e) Development of the deep learning framework. These stages are explained below:

A. Data cleansing

Identifying incorrect or missing data may be necessary to avoid any bias when training a model. Using a deep learning approach can facilitate and partly automate this time-consuming task [9]. In the case of the NWP data, the cleansing approach consisted of substituting missing or incorrect values with zeros. Outliers were not filtered out as a deep CNN approach is relatively robust to a small number of such data points. In fact, it was found that only a relatively small number of forecast values needed to substituted.

B. Feature engineering

Wind power forecasting models developed for one location may not be representative of other locations for a variety of reasons, e.g. the effects of varying terrain height, localised wind speed patterns, differences in local temperature, pressure and humidity, *etc*, [10]. Forecast bias and accuracy is a function of these and other variables. The challenge is to decide what input variables add value to a probabilistic forecast and how to develop a model which is both accurate and parsimonious.

The first approach was to reduce the dimensionality of the input data. This done by using the median value of the ten ensembles, as suggested in [11]. The full size grid NWP data was used to model every price region. The aim was thus for the model to use deep learning to output quantile forecasts directly.

According to [12], the accuracy of wind power predictions is seasonally dependent. In order to capture seasonal and diurnal dependencies, a time proxy was use, namely the day of year and time of day, following [14], in the form of periodic functions.

The main feature in deep learning-based models related to power production is wind speed as shown in [13], [14]. This variable is derived from the 10-meter zonal (U10) and meridional (V10) wind speed components. Table I summarizes the correlation between the input forecast variables and the power data for the four price regions.

TABLE I
ABSOLUTE CORRELATION BETWEEN FORECAST VARIABLES AND WIND POWER BY PRICE REGION - HIGHEST CORRELATIONS ARE SHOWN IN BOLD

Feature variable	SE1	SE2	SE3	SE4
Wind speed (WS)	**0.523**	**0.659**	**0.696**	**0.560**
Wind direction (WD)	0.060	0.060	0.016	0.164
Zonal 10-meter wind (V10M)	0.336	0.453	0.545	0.527
Meridional 10-meter wind (V10M)	0.416	0.512	0.511	0.338
Wind gust	**0.521**	**0.657**	**0.687**	**0.548**
Mean sea level pressure (MLSP)	0.063	0.240	0.315	0.234
Screen level rel. humidity (RH2M)	0.050	0.017	0.012	0.051
Surface temperature (T2M)	0.034	0.002	0.103	0.199
Total cloud cover (TCC)	0.065	0.149	0.244	0.197

Wind speed resulted to have the greatest correlation with wind power, as expected. On the other hand, wind direction showed almost no correlation. Although wind gust also shows a good correlation with power output, inclusion as an input to the model did not improve the forecast beyond using only the wind speed magnitude as they are already highly correlated between them (>0.97). The relative humidity shows little correlation. Pressure shows a small degree of correlation but this varies significantly by region. Consequently, only the wind speed magnitude was used as a forecast input variable in the model. The wind speed input was scaled by subtracting the mean and dividing by the variance of the training dataset to promote an efficient optimization process of the model [15].

C. Target engineering

As mentioned earlier, installed capacity per region changed over time. In order for the model to adequately cope with this variation, the power values for each price region were divided by the current installed capacity for that respective hour. This has the effect of normalizing the output values in the form of a capacity factor in order to train the model. Predicted capacity factors are then converted back into power production values for producing the final forecasts, assuming that wind turbine availability is 100%.

D. Development of a probabilistic framework

Traditional Bayesian statistics are used when more information about a forecast is required, extending the model from a deterministic to a probabilistic nature, by inferring the distribution over the dataset. Nevertheless. Bayesian methods have a high computational cost when used with large datasets and thus are unsuitable for this application [16].

In order to build a model that could be generalized for all price regions, a non-parametric approach was followed. No assumptions about the shape of the wind speed distributions were made, as this can vary in time for a given location [17]. However, the non-parametric approach requires the tuning of additional parameters and consequently, brings additional computational costs.

The Smooth Pinball Neural Network (SPNN) architecture was used to develop a deep learning model for quantile regression that uses a non-parametric approach [18], [19] and is a combination between a Huber loss (smooth L1-loss) and a pinball loss using an objective function S_j for each j^{th} decile ($j = i/10$) with quantile value τ=*i*/100, of the form:

$$S_j = \tau \cdot u_j + \alpha \cdot \log(1 + \exp(-\frac{u_j}{\alpha})) \quad (2)$$

The difference between the observed and predicted value for each decile, u_j, was smoothed using the parameter, α, to promote a non-convex optimization algorithm, facilitating the convergence of the model. The advantage of this approach was that optimization was based directly on the forecast evaluation metric used in the competition.

Non-parametric deep learning models can have difficulties interpreting the ranked order of quantiles. Hence, another advantage of the framework was dealing with the quantile cross-over problem: it occurs when the prediction output values for higher quantiles is smaller than lower quantiles (e.g. Q20 $<$ Q10). This behaviour becomes common when explanatory variables are heteroscedastic [8]. Eventually, the estimations do not follow the nature of a probability distribution function, reducing the reliability of the forecast. As a consequence, a penalty factor was applied in the objective function such that it stimulates particular local minima, similar to the idea of reinforcement learning [20]. The penalty function P is given by:

$$P = \kappa \cdot \max[0, \epsilon - (q^{<\tau-1>} - q^{<\tau>})]^2 \quad (3)$$

The margin, ϵ, expresses the desired spacing between two consecutive quantile forecast values, q^{τ}, and $q^{\tau-1}$, while the penalty factor, κ, indicates the severity of the cross-over error. This penalty term was added to the smoothed pinball function in (2), meaning that three additional parameters had to be tuned in the model.

E. Development of the deep learning framework

The final stage consisted of expanding the SPNN framework for quantile regression, concatenating CNN and MLP architectures. Hence, the input layer consists of two branches: (a) the NWP grid data, introduced in the CNN architecture; (b) the time proxy, introduced in the MLP architecture.

Three CNN architectures were considered, namely LeNet-5 [21], AlexNet [22] and the VGG-16, the latter used to win the Imagenet competition in 2014 [23]. The motivation to introduce a CNN was to capture spatial patterns in the NWP data. On the other hand, to simplify the MLP branch, a simple architecture with two hidden layers was used (6 and 26 nodes).

Quantile forecasts were made simultaneously following the methodology in [8]. The concatenation between the CNN and the MLP takes place at the last hidden layer of both architectures. This structure is integrated in the framework of the traditional SPNN, in which the nodes of the output layer

represent the quantiles to be forecasted. Hence, the output layer corresponds to nine nodes, each having a customized optimization function based on their respective decile target (τ), given in (2). The final quantile regression loss function (QRLF) can be expanded to the sum of every unit of the dense output layer representing the nine deciles:

$$QRLF = \sum_{j=1}^{9} S_j \tag{4}$$

Figure 1 illustrates the framework of the SPNN, including the concatenation of both branches.

The models were trained separately for each price region. However, the entire NWP grid of data was used to train a single model. As a consequence, the deep learning algorithm was able to capture the relationships between the input data, without introducing a spatial subset of the meteorological variables. Figure 1 shows an schematic representation of the final concatenated model, based on the SPNN quantile regression framework.

The training phase consisted of using 10 months of shuffled data from year 2000, while the testing data consisted of the remaining two months. Furthermore, a validation split was performed in the training set to evaluate both bias and variance: indicators of under- and over-fitting. To simplify the tuning process, the SPNN parameters, α and κ, were replicated from [18], and only the margin parameter, ϵ, was used to calibrate the forecasts. Moreover, the kernel random initialization used a normal distribution, while the regularization term, λ = 0.0001, was used to avoid over-fitting [18]. Regarding the activation function, the Rectified Linear Unit (ReLU) function was used. Finally, an early stopping criterion was employed instead of defining a fixed number of epochs. In this manner, once the validation error showed no further improvement, the model terminated the optimization algorithm to avoid memorizing the training dataset.

IV. Results

A. Model training

The selection of the CNN architecture was performed prior to the first round of the competition. Once the competition started, the deep learning architecture was not altered. Table II shows the results of the different architectures considered comparing the predicted and observed wind power values for the two-month testing period and determining the best mean absolute percentage error (MAPE) after model tuning.

It can be seen that the simple LeNet-5 architecture clearly showed the best results for this application. The more complex architectures (i.e. AlexNet and VGG-16) generalized the data, failing to predict well the power values during the testing period. Therefore, it was decided to concatenate the LeNet-5 architecture with the MLP architecture.

Four elements (hyper-parameters) were tuned to produce the best model for every price region: the type of sub-sampling layer (maximum or average pooling), the degree of spatial dropout, the batch size and the margin to manage quantile cross-over. Table IV-A shows the results of the pinball loss metric as a function of four different models with different hyper-parameter settings. The best (lowest) values for each price region are shown in bold.

TABLE II
Comparison of results between different CNN architectures; MAPE = Mean Absolute Percentage Error.

Architecture	Overview	Best MAPE
LeNet-5	Conv. layer: [6, 16] Fully-connected layer: [120, 84] Kernel: 3x3 Padding: no; stride: 1	8.5%
VGG-16	Conv. layer: [16, 16, 64, 64, 128, 128] Fully-connected layer: [4096, 4096] Kernel: 3x3 Padding: no; stride: 1	18.7%
AlexNet	Conv. layer: [96, 256, 384, 384, 384] Fully-connected layer: [4096, 4096] Kernel: variable from 3x3 to 11x11 Padding: 1; stride: variable from 1 to 4	16.2%

TABLE III
Pinball loss function as a function of hyper-parameter values for each price region. The best choice of parameters for each price region is shown in bold

Parameters	SE1	SE2	SE3	SE4
Model A a. Max Pooling b. Spatial Dropout = 0.25 c. Batch size = 64. d. ϵ = 0.002	38.9	184.2	117.4	77.3
Model B a. Max Pooling b. Spatial Dropout = 0 c. Batch size = 64. d. ϵ = 0.001	34.6	**139.6**	**92.2**	60.5
Model C a. Average Pooling b. Spatial Dropout = 0 c. Batch size = 64. d. ϵ = 0.001	34.9	143.3	92.4	59.6
Model D a. Max Pooling b. Spatial Dropout = 0 c. Batch size = 32. d. ϵ = 0.001	**32.2**	148.3	106.0	**58.2**

Based on the sensitivity analysis, two models were considered, namely Model B and Model D. Both models performed best with a margin, ϵ = 0.001, while the best sub-sampling layer approach was Max Pooling to reduce the shape of the input data. Moreover, Spatial Dropout did not improve the score, hence only the λ term was used for regularizing the weights of the kernels. In Model B, a batch size of 64 performed best for price regions SE2 and SE3. In contrast, the best fit for price regions SE1 and SE4 was achieved with a batch size of 32, corresponding to Model D. Finally, to avoid forecast quantile values exceeding the installed capacity factor at each time step, a clipping factor between 65% to 88% was applied based on the historical training data starting from the median quantile.

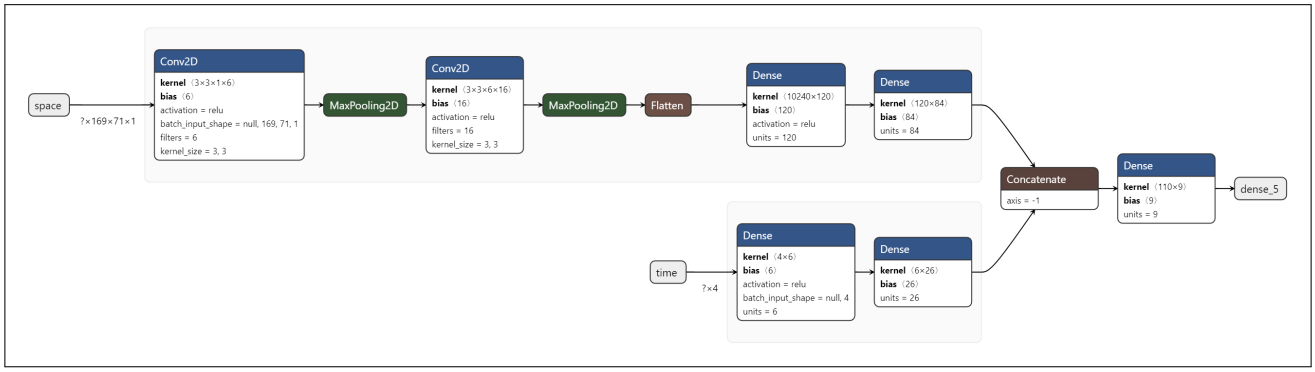


Fig. 1. A visual representation of the final model; (a) a LeNet-5 based CNN architecture for the NWP data; (b) a simple MLP for the time proxy. Both architectures are concatenated in their final hidden layer to compute the output quantiles.

B. Competition and re-forecast results

The pinball score results for the top four teams and the *DeepWinds* model are shown by round in Figure 2. For comparison, the *DeepWinds* model scores are also shown for re-forecasts using the final tuned version of the model in Round 6 and post-competition. Note that the model architecture and the feature engineering have *not* been changed.

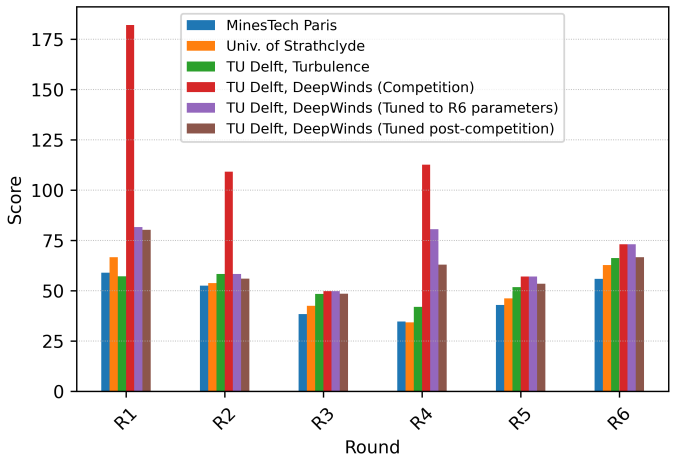


Fig. 2. Pinball scores by round for the top teams in the forecasting competition, compared with the *DeepWinds* model in three cases; (a) Official competition; (b) Tuned until round 6 parameters; (c) Tuned post-competition.

As the final score of the teams was determined by the best five submission rounds, the standings at the end of the competition are shown in Table IV, along with model type.

The *DeepWinds* model re-forecasts show competitive results with respect to the top performing teams and are a significant improvement on the model used during Rounds 1–5. However, the model was not able to capture the spatial patterns sufficiently well in rounds 1 and 4 when compared with other models by using the full size NWP grid data. Note that a separate team from TU Delft (*Turbulence*), which came third, also used a model incorporating a CNN and MLP, but the architecture was quite different to the *DeepWinds* model, and incorporated additional feature engineering.

TABLE IV
FINAL RESULTS OF THE TOP FOUR TEAMS IN THE FORECASTING COMPETITION COMPARED WITH THE *DeepWinds* MODEL; QRF = QUANTILE REGRESSION FOREST; QGAM = QUANTILE GENERALIZED ADDITIVE MODEL; GBM = GRADIENT BOOSTING MACHINE; HCM = HYBRID CNN-MLP

Rank	Team	Final Score	Model Type
1	MinesTech Paris	44.92	QRF
2	Univ. of Strathclyde	47.93	QGAM & GBM
3	TU Delft, Turbulence	51.52	HCM
-	TU Delft, DeepWinds (Post-competition)	57.55	SPNN, CNN & MLP
-	TU Delft, DeepWinds (until R6 parameters)	63.78	SPNN, CNN & MLP
-	TU Delft, DeepWinds (Competition)	80.38	SPNN, CNN & MLP

The post-competition tuned version of the model followed a re-evaluation of Model B and Model D. This showed that the accuracy of the models displayed a seasonal dependence despite the time proxy, i.e. Model D performed better in winter for price regions SE1 and SE4 but also performed better than Model B in summer for regions SE2 and SE3, and vice versa for Model B. As a consequence, better performance was achieved by having models tuned with hyper-parameters appropriate for both price region and season.

V. CONCLUSIONS

This paper has summarized the approach followed by the *DeepWinds* model to predict wind power production in Sweden using a probabilistic framework for the EEM 20 forecasting competition. The model was based on a deep learning method using the novel Smooth Pinball Neural Network (SPNN), concatenating CNN and MLP architectures.

The resulting framework provided a simple way to output quantile values from NWP input data. The non-parametric approach allowed a generalization of the model to different datasets, allowing it to be trained for different price regions separately. Moreover, ways to reduce data dimensionality and changes in installed capacity were proposed. The application of this deep learning model is suitable for mid- and long-term forecasting and can be used as a benchmark tool for other similar models.

ACKNOWLEDGMENT

The authors would like to thank the team at KTH, Sweden for organising the forecasting competition and providing help in interpreting the data. We would also like to thank Robbert Eggermont for making the High-Performance Computing (HPC) cluster from TU Delft available, and by providing his technical support.

REFERENCES

[1] Maldonado-Correa, Jorge and Solano, J. C. and Rojas-Moncayo, Marco. "Wind power forecasting: A systematic literature review", Wind Engineering, Early Access, pp 1–14, 2019, doi: 10.1177/0309524X1989167.

[2] "Global energy review 2020 – analysis", International Energy Agency, apr. 2020. https://www.iea.org/reports/global-energy-review-2020 (consulted jul. 15, 2020).

[3] S. Mujeeb, T. A. Alghamdi, S. Ullah, A. Fatima, N. Javaid, and T. Saba, "Exploiting Deep Learning for Wind Power Forecasting Based on Big Data Analytics", Applied Sciences, vol. 9, nr. 20, p. 4417, oct. 2019, doi: 10.3390/app9204417.

[4] Morales, Juan M., Antonio J. Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno. "Integrating renewables in electricity markets: operational problems", Springer Science & Business Media, vol. 205, 2013.

[5] G. Kariniotakis and P. Pinson, "Data science for renewable energy prediction", presented by Smart4RES, Online, jun. 04, 2020, consulted: jun. 04, 2020. [Online]. Available in: https://www.slideshare.net/sustenergy/smart4res-data-science-for-renewable-energy-prediction.

[6] S. Kr. Jha, J. Bilalovic, A. Jha, N. Patel, and H. Zhang, "Renewable energy: Present research and future scope of Artificial Intelligence", Renewable and Sustainable Energy Reviews, vol. 77, pp. 297–317, sep. 2017, doi: 10.1016/j.rser.2017.04.018.

[7] "Forecasting Competition", European Energy Markets 2020. https://eem20.eu/forecasting-competition/ (consulted jul. 16, 2020).

[8] F. Rodrigues en F. C. Pereira, "Beyond Expectation: Deep Joint Mean and Quantile Regression for Spatiotemporal Problems", IEEE Trans. Neural Netw. Learning Syst., pp. 1–13, 2020, doi: 10.1109/TNNLS.2020.2966745.

[9] T. Rozario, T. Long, M. Chen, W. Lu, en S. Jiang, "Towards automated patient data cleaning using deep learning: A feasibility study on the standardization of organ labeling", arXiv:1801.00096 [physics], dec. 2017, Consulted: jul. 18, 2020. [Online]. Available from: http://arxiv.org/abs/1801.00096.

[10] S. S. Soman, H. Zareipour, O. Malik and P. Mandal, "A review of wind power and wind speed forecasting methods with different time horizons," North American Power Symposium 2010, Arlington, TX, 2010, pp. 1-8, doi: 10.1109/NAPS.2010.5619586.

[11] Wallach, D. "When and why to predict using the mean or median of a crop multi-model ensemble." FACCE MACSUR Reports 10.S (2017): 37.

[12] Foley, A. M., P. G. Leahy, and E. J. McKeogh. "Wind power forecasting & prediction methods." 2010 9th International Conference on Environment and Electrical Engineering. IEEE, 2010.

[13] Pinson, Pierre, and Henrik Madsen. "Ensemble-based probabilistic forecasting at Horns Rev." Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology 12.2 (2009): 137-155.

[14] Torres, J. M., R. M. Aguilar, and K. V. Zuñiga-Meneses. "Deep learning to predict the generation of a wind farm." Journal of Renewable and Sustainable Energy 10.1 (2018): 013305.

[15] Ng, Andrew. "Machine learning. coursera." Standford University,[Online]. Available: https://www. coursera. org/learn/machine-learning.[Accessed 15 February 2020] (2016).

[16] S. Abeywardana, "Deep Quantile Regression", Medium, mar. 20, 2019. https://towardsdatascience.com/deep-quantile-regression-c85481548b5a (consulted jul. 27, 2020).

[17] Pinson, Pierre, and George Kariniotakis. "Conditional prediction intervals of wind power generation." IEEE Transactions on Power Systems 25.4 (2010): 1845-1856.

[18] K. Hatalis, A. J. Lamadrid, K. Scheinberg, en S. Kishore, "Smooth Pinball Neural Network for Probabilistic Forecasting of Wind Power", arXiv:1710.01720 [stat], oct. 2017, Consulted: jul. 18, 2020. [Online]. Available from: http://arxiv.org/abs/1710.01720.

[19] K. Hatalis, A. J. Lamadrid, K. Scheinberg, en S. Kishore, "A Novel Smoothed Loss and Penalty Function for Noncrossing Composite Quantile Estimation via Deep Neural Networks", arXiv:1909.12122 [cs, eess], sep. 2019, Consulted: jul. 18, 2020. [Online]. Available from: http://arxiv.org/abs/1909.12122.

[20] Sutton, Richard S., and Andrew G. Barto. "Reinforcement learning: An introduction." MIT press, 2018.

[21] LeCun, Yann. "LeNet-5, convolutional neural networks." URL: http://yann. lecun. com/exdb/lenet 20.5 (2015): 14.

[22] Krizhevsky, A., I. Sutskever, and G. E. Hinton. "2012 AlexNet." Adv. Neural Inf. Process. Syst. (2012): 1-9.

[23] R. Thakur, "Step by step VGG16 implementation in Keras for beginners", Medium, aug. 20, 2019. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c (consulted jul. 18, 2020).

Bibliography

[1] International Energy Agency. Global energy review 2020 - analysis and key findings. URL https://www.iea.org/reports/global-energy-review-2020/renewables.

[2] Wind Europe. *Offshore Wind in Europe*. Feb 2020. URL https://windeurope.org/data-and-analysis/product/.

[3] Ministerie van Algemene Zaken. Dutch national government - offshore wind energy - renewable energy, Jul 2017. URL https://www.government.nl/topics/renewable-energy/offshore-wind-energy.

[4] Sana Mujeeb, Turki Ali Alghamdi, Sameeh Ullah, Aisha Fatima, Nadeem Javaid, and Tanzila Saba. Exploiting deep learning for wind power forecasting based on big data analytics. *Applied Sciences*, 9(20):4417, 2019.

[5] Juan Morales, Antonio Conejo, Henrik Madsen, Pierre Pinson, and Marco Zugno. *Integrating renewables in electricity markets: operational problems*, volume 205. Springer Science & Business Media, 2013.

[6] H Nielsen, T Nielsen, and Henrik Madsen. An overview of wind power forecasts types and their use in large-scale integration of wind power. pages 25–26, 2011.

[7] Kariniotakis, Georges and Pinson, Pierre. Data science for renewable energy prediction, 2020. URL https://www.slideshare.net/sustenergy/smart4res-data-science-for-renewable-energy-prediction.

[8] Aoife Foley, PG Leahy, and EJ McKeogh. Wind power forecasting & prediction methods. pages 61–64, 2010.

[9] Sunil Kr Jha, Jasmin Bilalovic, Anju Jha, Nilesh Patel, and Han Zhang. Renewable energy: Present research and future scope of artificial intelligence. *Renewable and Sustainable Energy Reviews*, 77:297–317, 2017.

[10] Gregor Giebel, Richard Brownsword, George Kariniotakis, Michael Denhard, and Caroline Draxl. The state-of-the-art in short-term prediction of wind power: A literature overview. 2011.

[11] Aoife M Foley, Paul G Leahy, Antonino Marvuglia, and Eamon J McKeogh. Current methods and advances in forecasting of wind power generation. *Renewable Energy*, 37(1):1–8, 2012.

[12] Lars Landberg. Short-term prediction of local wind conditions. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(3-4):235–245, 2001.

[13] Yao Zhang, Jianxue Wang, and Xifan Wang. Review on probabilistic forecasting of wind power generation. *Renewable and Sustainable Energy Reviews*, 32:255–270, 2014.

[14] Georges Kariniotakis, Hans-Peter Waldl, Ignacio Marti, Gregor Giebel, Torben S. Nielsen, Jens Tambke, Julio Usaola, F Dierich, Alexis Bocquet, and Silvère Virlot. Next generation forecasting tools for the optimal management of wind generation. pages 1–6, 2006.

[15] Mohammad Bannayan and Gerrit Hoogenboom. Weather analogue: A tool for real-time prediction of daily weather data realizations based on a modified k-nearest neighbor approach. *Environmental Modelling & Software*, 23(6):703 – 713, 2008. ISSN 1364-8152. doi: https://doi.org/10.1016/j.envsoft.2007.09.011. URL http://www.sciencedirect.com/science/article/pii/S1364815207001764.

[16] You Lin, Ming Yang, Can Wan, Jianhui Wang, and Yonghua Song. A multi-model combination approach for probabilistic wind power forecasting. *IEEE Transactions on Sustainable Energy*, 10 (1):226–237, 2018.

[17] Nick Ellis, Robert Davy, and Alberto Troccoli. Predicting wind power variability events using different statistical methods driven by regional atmospheric model output. *Wind Energy*, 18(9):1611–1628, 2015.

[18] Song Li, Peng Wang, and Lalit Goel. Wind power forecasting using neural network ensembles with feature selection. *IEEE Transactions on sustainable energy*, 6(4):1447–1456, 2015.

[19] Duehee Lee and Ross Baldick. Short-term wind power ensemble prediction based on gaussian processes and neural networks. *IEEE Transactions on Smart Grid*, 5(1):501–510, 2013.

[20] Wenbin Wu and Mugen Peng. A data mining approach combining k-means clustering with bagging neural network for short-term wind power forecasting. *IEEE Internet of Things Journal*, 4(4):979–986, 2017.

[21] Aqsa Saeed Qureshi, Asifullah Khan, Aneela Zameer, and Anila Usman. Wind power prediction using deep neural network based meta regression and transfer learning. *Applied Soft Computing*, 58:742–755, 2017.

[22] Jesús Torres, Rosa María Aguilar, and K. Zuñiga-Meneses. Deep learning to predict the generation of a wind farm. *Journal of Renewable and Sustainable Energy*, 10(1):013305, 2018.

[23] Jesús Torres and Rosa María Aguilar. Using deep learning to predict complex systems: a case study in wind farm generation. *Complexity*, 2018, 2018.

[24] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2): 243–268, 2007.

[25] Pierre Pinson and George Kariniotakis. Conditional prediction intervals of wind power generation. *IEEE Transactions on Power Systems*, 25(4):1845–1856, 2010.

[26] Tao Hong, Pierre Pinson, Shu Fan, Hamidreza Zareipour, Alberto Troccoli, and Rob J Hyndman. Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond. 2016.

[27] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[29] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. pages 802–810, 2015.

[30] Filipe Rodrigues and Francisco C Pereira. Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE transactions on neural networks and learning systems*, 2020.

[31] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[32] Andrew Ng. Neural networks and deep learning, 2017. URL https://www.coursera.org/learn/neuralnetworks-deep-learning.

[33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[34] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[35] Dana Hughes and Nikolaus Correll. Distributed machine learning in materials that couple sensing, actuation, computation and communication. *arXiv preprint arXiv:1606.03508*, 2016.

[36] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6 (02):107–116, 1998.

[37] Andrew Ng. Machine learning, 2014. URL https://www.coursera.org/learn/machine-learning.

[38] Romain Thalineau. An introduction to backpropagation, 2018. URL https://www.qwertee.io/blog/an-introduction-to-backpropagation/.

[39] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[40] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12 (1):145–151, 1999.

[41] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] Belajar Pembelajaran Mesin Indonesia. Student notes to convolutional neural networks: an introduction, Jul 2018. URL https://indoml.com/2018/03/07/student-notesconvolutional-neural-networks-cnn-introduction.

[44] Fu Jie Huang and Yann LeCun. Large-scale learning with svm and convolutional for generic object categorization. 1:284–291, 2006.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[46] Imad Dabbura. K-means clustering: Algorithm, applications, evaluation methods, and drawbacks, Aug 2020. URL https://towardsdatascience.com.

[47] Marco Dena. K-means incoherent behaviour choosing k with elbow method, bic, variance explained and silhouette, Aug 2015. URL https://datascience.stackexchange.com.

[48] Kostas Hatalis, Alberto Lamadrid, Katya Scheinberg, and Shalinee Kishore. Smooth pinball neural network for probabilistic forecasting of wind power. *arXiv preprint arXiv:1710.01720*, 2017.

[49] Kostas Hatalis, Alberto Lamadrid, Katya Scheinberg, and Shalinee Kishore. A novel smoothed loss and penalty function for noncrossing composite quantile estimation via deep neural networks. *arXiv preprint arXiv:1909.12122*, 2019.

[50] Lars Herre, Mikhail Skalyga, and Priyanka Shinde. European energy markets 2020 forecasting competition, Mar 2020. URL https://eem20.eu/forecasting-competition/.

[51] Lisa Bengtsson, Ulf Andrae, Trygve Aspelien, Yurii Batrak, Javier Calvo, Wim de Rooy, Emily Gleeson, Bent Hansen-Sass, Mariken Homleid, Mariano Hortal, and et al. The harmonie–arome model configuration in the aladin–hirlam nwp system. *Monthly Weather Review*, 145(5):1919–1935, May 2017. ISSN 0027-0644. doi: 10.1175/MWR-D-16-0417.1.

[52] Inger-Lise Frogner, Andrew T. Singleton, Morten Ø Køltzow, and Ulf Andrae. Convection-permitting ensembles: Challenges related to their design and use. *Quarterly Journal of the Royal Meteorological Society*, 145(S1):90–106, 2019. ISSN 1477-870X. doi: 10.1002/qj.3525.

[53] Eugenia Kalnay. Historical perspective: earlier ensembles and forecasting forecast skill. *Quarterly Journal of the Royal Meteorological Society*, 145(S1):25–34, 2019. ISSN 1477-870X. doi: 10.1002/qj.3595.

[54] Gilles Notton and Cyril Voyant. Chapter 3 - forecasting of intermittent solar energy resource. In Imene Yahyaoui, editor, *Advances in Renewable Energies and Power Technologies*, pages 77 – 114. Elsevier, 2018. ISBN 978-0-12-812959-3. doi: https://doi.org/10.1016/B978-0-12-812959-3.00003-4. URL http://www.sciencedirect.com/science/article/pii/B9780128129593000034.

[55] Sukanta Basu, Simon J. Watson, Eric Lacoa Arends, and Bedassa Cheneka. Day-ahead wind power predictions at regional scales: Post-processing operational weather forecasts with a hybrid neural network. IEEE, Sep 2020.

[56] Kevin Bellinguer, Valentin Mahler, Simon Camal, and Georges Kariniotakis. Probabilistic forecasting of regional wind power generation for the EEM20 competition: a physics-oriented machine learning approach. IEEE, Sep 2020.

[57] Kevin Bellinguer, Robin Girard, Guillaume Bontron, and Georges Kariniotakis. Short-term photovoltaic generation forecasting using multiple heterogenous sources of data based on an analog approach. *EGU General Assembly 2020*, 2020. URL https://doi.org/10.5194/egusphere-egu2020-13790.

[58] Jethro Browell, Ciaran Gilbert, Rosemary Tawn, and Leo May. Quantile combination for the EEM20 wind power forecasting competition. IEEE, Sep 2020.

[59] Keras guide, 2020. URL https://www.keras.io.

[60] Statistics Sweden. Number of persons with foreign or swedish background (rough division) by region, age and sex. year 2002 - 2019, 2019. URL http://www.statistikdatabasen.scb.se.

[61] Christian Schumacher and Florian Weber. How to extend the lifetime of wind turbines, Sep 2019. URL https://www.renewableenergyworld.com/2019/09/20/how-to-extend-the-lifetime-of-wind-turbines/.

[62] Aditya Saini. Backpropagation algorithm intuition, 2014. URL https://stats.stackexchange.com/questions/94387/how-to-derive-errors-in-neural-net.

[63] Eric Lacoa Arends, Simon J. Watson, Sukanta Basu, and Bedassa Cheneka. Probabilistic wind power forecasting combining deep learning architectures. IEEE, Sep 2020.