Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Erik Liodden

A Deep Learning-Based Method for Regional Wind Power Production Volume Prediction

Master's thesis in Computer Science
Supervisor: Massimiliano Ruocco and Gabriele Martinelli
June 2020

NTNU
Norwegian University of Science and Technology

Erik Liodden

A Deep Learning-Based Method for Regional Wind Power Production Volume Prediction

Master's thesis in Computer Science
Supervisor: Massimiliano Ruocco and Gabriele Martinelli
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

The aim of this thesis was to predict the wind power production volume of a large geographical region given the Numerical Weather Prediction data (NWP) over the region using deep learning. Accurate production volume predictions is important for power grid balancing, production planning, and price estimation. Having an accurate forecast for the upcoming wind power production volume has become more and more important in the past years due to the fast increasing number of installed wind turbines and installed total production capacity. Due to the physical properties of wind turbines, wind power production has a strong correlation with the current weather system. This motivates a thorough analysis of the weather using the past, current, and upcoming weather as a basis for the volume prediction. However, the highly nonlinearity of the spatial and temporal characteristics of the weather system makes accurate power volume predictions difficult. To address this, this study designes and evaluate a deep learning architecture using techniques that have shown great success on other similar problems. Convolutional Neural Networks, CNNs, have had great success in image classification, and are able to extract spatial relations and information. An extension to CNNs, called 3D convolution, has had success in capturing temporal dependencies in sequences of image-like data. This study found that deep learning methods were able to directly predict the wind power production volume more accurately than other common machine learning methods in 13 out of 20 regions. A hybrid model combining the proposed deep learning architecture for feature generation and a tree-based learning algorithm, LightGBM, for the final power predictions, improved the prediction accuracy in 17 out of 20 regions compared to the LightGBM algorithm trained without these additional features. Future research in applying deep learning to wind power analysis is encouraged to further investigate the possibilities of capturing the spatio-temporal dependencies to improve predictions.

Sammendrag

Målet med denne studien var å estimere produksjonsvolumet av vindkraft i en stor geografisk region gitt de numeriske vær-datane (NWP) over regionen og metoder basert på dyp læring. Et presist estimat for fremtidig produksjonsvolum er viktig for balansering av strømnettet, produksjonsplanlegging og prisestimering. De siste årene har dette stadig blitt viktigere, og det har vært en hyppig økning i antall nyinstallerte vindmøller og dermed den totale produksjonskapasiteten. Vindmøller er konstruert for å generere energi basert på vind, noe som gir en sterk korrelasjon mellom værsystemet i et område og mengden strøm som produseres av vindmøller i det aktuelle området på et gitt tidspunkt. Denne sammenhengen motiverer en grundig analyse av historisk, øyeblikkelig og fremtidig værdata som grunnlag for estimering av produksjonsvolumet. Dette er en krevende oppgave gitt de ikkelineære romlige og temporale korrelasjonene i værsystemet. Denne studien designer og evaluerer en modellarkitektur basert på dyp læring som tar i bruk metoder som har vist gode resultater på andre lignende problemer. Konvolusjonelle nevrale nettverk, CNN, har hatt stor suksess innenfor bildeklassifisering og er i stand til å ekstrahere romlige korrelasjoner. En utvidelse av CNN kalt 3D konvolusjon, har tidligere vist å være i stand til å ekstrahere temporale korrelasjoner i en sekvens av bilde-lignende data. Denne studien fastslo at den foreslåtte modellen basert på dyp læring var i stand til å estimere produksjonsvolumet av vindkraft direkte med en høyere presisjon i 13 av 20 regioner, enn andre, mer vanlige maskinlæringsmetoder. En hybrid modell ble konstruert av den foreslåtte modellen basert på dyp læring for *feature engineering* med en valgtre-basert læringsmodell, LightGBM, for den endelige estimeringen. Den hybride modellen forbedret presisjonen på estimeringen av produksjonsvolumet i 17 av 20 regioner, sammenlignet med den samme valgtre-baserte algoritmen trent uten de ekstra genererte karakteristikkene. Videre undersøkelser av modellarkitekturer basert på dyp læring er oppfordret for å få bedre innsikt i hvordan de romlige og temporale relasjonene i værdata kan brukes til estimering av produksjonsvolum av vindkraft.

Preface

This master thesis was written by me, Erik Liodden, at NTNU Trondheim during Spring 2020. The thesis is based my specialization project from the course TDT4501 that was conducted during Fall 2019. The master thesis takes a different direction in terms of model architecture and aim, and should *not* be considered as a direct continuation or extension to the specialization project report.

I would like to thank everyone who has supported me during this work with numerous good discussions and reflections. A special thanks to my supervisors, Massimiliano Ruocco (NTNU) and Gabriele Martinelli (Refinitiv), for giving me the opportunity to write this thesis and provide guidance throughout the process. I would also like to thank the company Refinitiv for letting me use their datasets.

Erik Liodden

Trondheim, June 9, 2020

Contents

1 Introduction **1**
- 1.1 Background and motivation . . . 1
- 1.2 Goal and Research Questions . . . 3
- 1.3 Outline of the report . . . 4

2 Background and Theory **5**
- 2.1 Artificial Neural Networks . . . 6
- 2.2 Convolutional Neural Networks . . . 7
  - 2.2.1 Temporal dependencies with 3D convolution . . . 9
- 2.3 Tree-based models and ensemble models . . . 10
- 2.4 Generalization and regularization . . . 11

3 State of the Art and Related Works **15**
- 3.1 Wind power prediction . . . 15
  - 3.1.1 Models based on historic wind power production . . . 15
  - 3.1.2 Models based on wind speed . . . 17
  - 3.1.3 Data-driven approaches . . . 21
- 3.2 Similar spatio-temporal dependent problems . . . 24
- 3.3 Image analysis . . . 24

4 Methods **27**
- 4.1 Dataset . . . 27
  - 4.1.1 Weather data . . . 28
  - 4.1.2 Target series . . . 31
- 4.2 Data preprocessing . . . 32
  - 4.2.1 Weather data . . . 32
  - 4.2.2 Target series . . . 34
- 4.3 Artificial Neural Network . . . 37
- 4.4 Decision Tree and the hybrid model . . . 44

5 Experimental Setting **47**

5.1 Dataset . . . 47
5.2 Metrics . . . 48
5.3 Experimental plan . . . 51
5.3.1 Plan 1 - Create baselines . . . 51
5.3.2 Plan 2 - Artificial Neural Network . . . 52
5.3.3 Plan 3 - Hybrid model . . . 52

6 Results and Discussion **53**
6.1 Baselines . . . 54
6.1.1 Parameter search . . . 54
6.1.2 Baseline accuracy and load factor transformation . . . 55
6.2 Artificial Neural Network . . . 58
6.2.1 Load factor transformation and temporal window . . . 58
6.2.2 Normalization strategy . . . 60
6.2.3 Network output configuration . . . 62
6.2.4 Stationarity and ensemble model . . . 65
6.2.5 Accuracy relative to load factor . . . 67
6.2.6 Model accuracy comparison on the test set . . . 73
6.3 Hybrid model . . . 76
6.3.1 Choice of tree-based algorithm . . . 76
6.3.2 Accuracy compared to the CNN-only model . . . 78
6.3.3 Hybrid ensemble model . . . 79
6.3.4 Model accuracy over time and final results . . . 79

7 Concluding Remarks **85**
7.1 Summary . . . 85
7.1.1 The first aim of the study . . . 86
7.1.2 The second aim of the study . . . 88
7.1.3 The third aim of the study . . . 89
7.2 Contribution . . . 90
7.3 Future work . . . 91

Bibliography **93**

Appendices **99**

List of Figures

1.1 Installed wind power capacity in Europe 2

2.1 Fully Connected Neural Networks 6
2.2 Convolution operator 8
2.3 Visualization of overfitting and underfitting 11
2.4 Regularization 12
2.5 Dropout 13

3.1 Typical wind power curve 18
3.2 Illustration of the model proposed by Ding et al. [2019] 19

4.1 Geographical reordering of the NWP 29
4.2 Zero-padding of non-rectangular NWP 29
4.3 Temperature distribution 30
4.4 Pressure distribution 31
4.5 Wind speed distribution 32
4.6 Transformation of the load factor target series 36
4.7 Model data input structure 38
4.8 Convolutional Neural Network 39
4.9 Overview of Spatial Pyramid Pooling 41
4.10 Ordinal Classification 43
4.11 Ensemble regressor 44
4.12 Overview of the hybrid model 45

5.1 Map of regions 48

6.1 Parameter tuning of baseline models 55
6.2 Baseline comparisons 57
6.3 Auto-correlation in production 59
6.4 The effect of the local vs global normalization strategies 61
6.5 The effect of the mean/std vs min-max normalization strategies . . 63
6.6 Error distribution comparing output configurations 64

6.7 1 month moving average error for region SE4 67
6.8 1 month moving average error for region SE2 68
6.9 AAPE distribution given load factor for region SE4 69
6.10 AAPE distribution given load factor for region SE2 70
6.11 Example 1 - Good performance on low load factor 72
6.12 Example 2 - Good performance on low load factor 73
6.13 Example 3 - Good performance on high load factor 74
6.14 LightGBM and Random Forest model error distribution 77
6.15 MAAPE 1 month moving average for region DK1 (onshore) and FIN 80
6.16 Offshore production drop 83

List of Tables

4.1 Different NWP normalization strategies. 34

5.1 Dataset description for regions in Germany 49
5.2 Dataset description for regions in the Nordic countries 50

6.1 Effect of load factor transformation 56
6.2 CNN-based model compared to the LightGBM baseline based on load factor 71
6.3 Aggregated results for all regions in Germany 75
6.4 Aggregated results for all regions in the Nordics 75
6.5 Overview of final results for each region 82

Chapter 1

Introduction

In this chapter, I will give an overview of the background and motivation for the work done in this thesis and define my goal and research questions.

1.1 Background and motivation

The background and motivation from the specialization project, Liodden [2019], preceding this master thesis, is highly relevant and therefore reviewed. The presentation from the project report has been adopted to the current thesis and is included below.

Generating an accurate forecast for the wind power production volume in a large geographical region is a challenging task. This thesis explores a data-driven solution to the problem by utilizing deep learning algorithms that have shown success in similar, but not necessarily equal, problems. Many factors affect the total power production volume, such as the weather situation in the region, time of year, and the location and type of wind turbines. The large number of different factors involved, and the complexity of those factors, makes it unfeasible to create a good prediction model analytically, Liu et al. [2019]. The weather forecast in itself is difficult to predict accurately, and will not account for local fluctuations of the wind speed around the wind turbines. The locations of the wind turbines themselves are not fully known for every wind farm in the region, and it is therefore not practical to predict the power production of each wind turbine individually and summarize the result, which could be regarded as the naive approach.

In the new global energy market, wind power production forecasting has become an important issue. For the past few years, there has been a rapid increase in the amount of installed wind power capacity. In 2018 wind power had the second

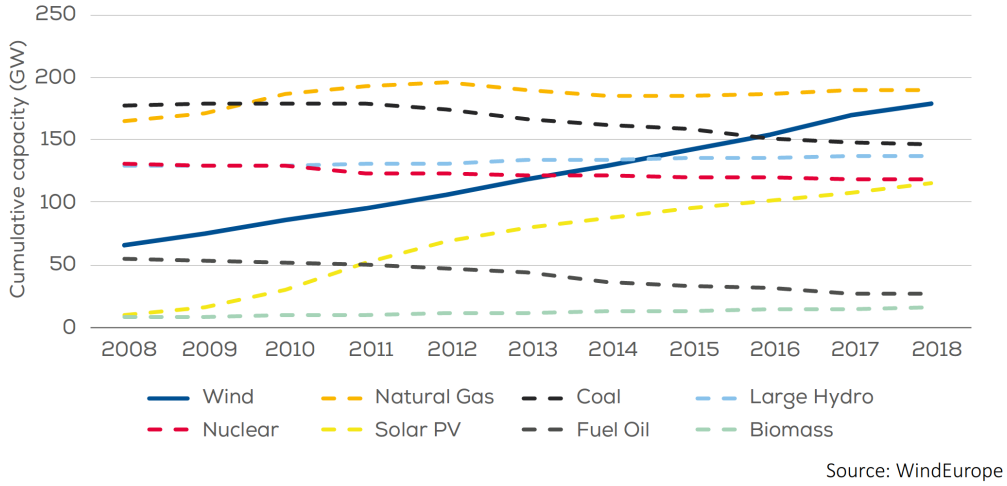


Figure 1.1: Total installed wind power capacity in Europe over the past 10 years compared to the total installed capacity of other energy sources, Komusanac et al. [2019].

largest power generation capacity in the EU-28 and was estimated to likely overtake natural gas power generation by the end of 2019, Komusanac et al. [2019], Figure 1.1. Each electricity producer on the power grid commits to producing a specific volume at a certain price one day ahead of delivery. This ensures predictability in the supply to the power grid. If it is expected to be a lot of wind tomorrow, it could be beneficial to wind down a nuclear power plant to prevent waste of nuclear fuel that day. On the other hand, if it is expected to be less wind tomorrow, it might be necessary to import electricity from another region or market to compensate for the low production of wind power. Having an accurate day-ahead forecast for the amount of electrical energy that will be produced is important for both the power grid operator and the power market traders. Mispredictions can lead to an over- or under-supply in electricity production which might be more expensive, as there will be increased costs when running power plants unnecessary.

A literature search revealed few studies which have discussed this exact problem. There have been several studies discussing the day-ahead wind power production volume prediction problem, but most of these studies have been concerned about a single wind farm and not an entire region or market. These studies have usually taken only the temporal information (the historic data) of the production and the weather system into account in their analysis. Liu et al. [2019] suggests that the spatial and temporal correlation between different wind farms should be considered in feature work to obtain better universality. This is the problem that will be explored and discussed in this thesis.

It is important to realize that the problem of predicting the total energy production volume in a region is more a financial problem rather than an engineering problem. The benefit of having a better forecasting model than the market is a competitive edge for traders which can open up the possibility to earn money. Several companies have most likely studied this problem in detail, but there exists very little published material on the topic. While prior studies have examined time series forecasting, the study in this thesis differs by including future weather information. A lot of articles about forecasting consider the problem: "We have the past information up until now. What comes next?". In this study the weather forecast for the next 6 hours will be considered as true values. Therefore, this is a spatio-temporal problem where some future information is already known which makes it a calibration problem rather than a forecasting problem.

1.2 Goal and Research Questions

The objective of this study is to determine whether the spatio-temporal information in the Numerical Weather Prediction, NWP, can be captured with a data-driven approach through deep learning and how this information can be used to predict the total wind power production volume in a geographical region. A deep learning architecture based on Convolutional Neural Networks, CNNs, will be designed and explored in this thesis. The CNN will predict the wind power production volume both directly and indirectly combined with a tree-based machine learning algorithm in a hybrid model. The research questions for this study are formulated as:

- **RQ1**: Is a deep CNN-based architecture able to capture the spatio-temporal dependencies of the NWP data and generate descriptive features for a given weather situation that can be used to predict the wind power production volume in a geographical region at that time?

- **RQ2**: Is a tree-based machine learning model combined with features generated by a deep learning-based model able to capture the spatio-temporal dependencies of the NWP data and predict the wind power production volume to a higher accuracy than any of those two alone?

- **RQ3**: How does the deep leaning-based models compare to more standard machine learning approaches on the wind power production volume prediction problem?

1.3 Outline of the report

This report is structured into six parts. Chapter 2 will give a short introduction to the background and theory used in this study. Chapter 3 will give an overview of the State of the Art and related work. Chapter 4 will describe the proposed deep learning model as well as the structure of the dataset. The experimental setup is outlined in Chapter 5. Chapter 6 presents the results of the experiments and a discussion of the results. And finally, a short summary of the work done in this thesis and suggestions for future work are presented in Chapter 7.

Chapter 2

Background and Theory

The past twenty years have seen increasingly rapid advances in the field of Machine Learning, and in particular deep learning with Artificial Neural Networks. The availability of powerful computers with fast and highly parallel computing power makes it now possible to explore models that were unfeasible to explore in the past due to limited resources. This chapter will contain the background and theory used in my thesis. The presentation of the background and theory from the work carried out in the project preceding this thesis, Liodden [2019], were reviewed. The section from Liodden [2019] is still relevant for this thesis for the most part, and the presentation from the project report has been adapted to the current thesis and is included below. Section 2.2.1, discussing the 3D convolution method, has been added and Section 2.4 has been expanded to reflect the additional regularization techniques that will be used in this thesis.

The main goal of machine learning is to create an algorithm that is able to generalize and capture an underlying structure or concept of a problem without being given explicit instructions. If the algorithm manages to generalize and capture the underlying structure of the data, it will be able to give useful information about situations or cases it has not encountered before. This is of great interest in many applications, as real-life analysis often encounter new variations of data that has not yet been explored. Instead of having domain experts or strict mathematical models tailored specifically for a particular problem, a general machine learning algorithm might give the same performance, or in some cases even better performance, than traditional approaches.

One major branch of machine learning is called *supervised learning*. In supervised learning, the machine learning algorithm "learns" the generalized structure by being exposed to training cases organized in a finite set of $\{(\mathbf{X}_i, y_i)\}_{i=1}^{N}$ pairs. $\mathbf{X}_i$

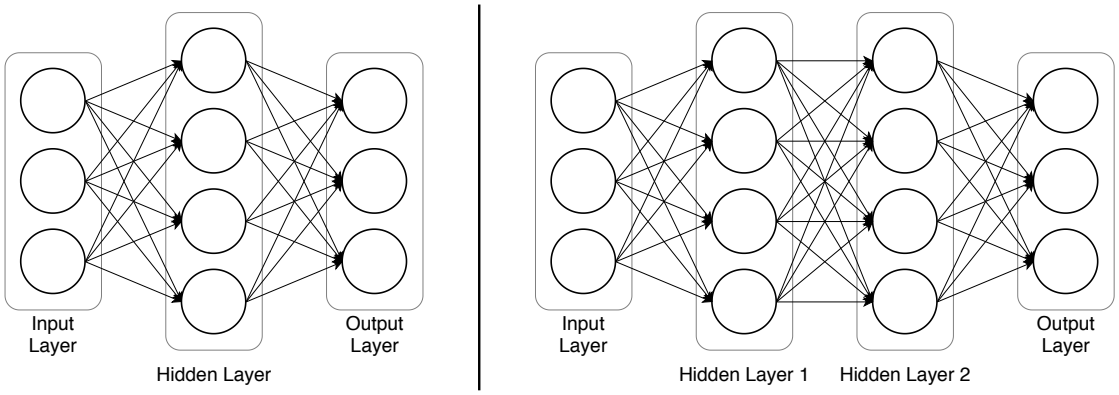


Figure 2.1: **Left**: A small neural network with one hidden layer. This network has three input nodes, four hidden nodes, and two output nodes. **Right**: A deep neural network with two hidden layers. This network has three input nodes, two hidden layers with four nodes each, and one output node.

is a feature vector drawn at random from the space of all possible feature vectors, D_X. y_i is the corresponding target value drawn from the space D_y. The samples are drawn according to some underlying probability distribution. The goal of the machine learning algorithm is to converge towards a function $f : D_X \rightarrow D_y$ such that f represents the mapping between the feature vectors and the target values.

2.1 Artificial Neural Networks

Artificial Neural Networks, or ANNs, is an old machine learning concept that has gained a lot of interest in the last few years. The structure of ANNs is inspired by the network of biological neurons in the brain. Such a network is composed of several nodes, or *neurons*, that is organized in one or several layers. The network process information by propagating the data from one side of the network (input) to the other side of the network (output) through a set of different layers in between. Each node in a layer takes as input one or several weighted inputs generated by nodes earlier in the network and the node uses a non-linear activation function as its output. Non-linearity of the activation function is necessary for the model to be able to approximate a non-linear target functions. These output values is then propagated to nodes later in the network. An ANN with many hidden layers of neurons between the input and output is typically called a Deep Neural Network, DNN. The most basic structure of an artificial neural network is the fully connected neural network, FCNN. In this architecture, each node in one layer is connected to each node in the next layer. Figure 2.1 shows two examples of this architecture.

The ANN can be considered as the representation of the function f that maps the samples from the feature domain, D_X, to the target domain, D_y. The network, and therefore the function f, is composed of a set of adjustable weights $\{W_i\}$ which are adjusted during training. The performance, or accuracy, of the network is measured by a *loss function*. The loss function is a measure of how right or wrong the predictions of the network are compared to the true values that are expected given the feature vector. The most common way of optimizing the network is to minimize the loss iteratively by changing the weights in the network through a gradient descent algorithm. The problem of finding the optimal set of weights $\{W_i\}$ can be considered a search problem in the multidimensional weight-space, given the loss function.

The choice of the loss function, network structure, and gradient descent algorithm are considered as parts of the hyperparameters of the model. Hyperparameters are not learnable during gradient decent and have to be specified in advance. To find the optimal hyperparameters are often a challenging task, as the hyperparameters are closely related to the dataset that the network is trying to generalize. Therefore it is difficult to find general recommendations for the hyperparameters and in most cases they are found by try and error.

2.2 Convolutional Neural Networks

Convolutional Neural Networks, CNN, is a type of an ANN architecture that is inspired by the biological system in humans and animals that process visual stimuli and visual information. These type of networks has demonstrated great success in numerous practical applications such as time series analysis and image analysis, Goodfellow et al. [2016]. CNNs utilize a mathematical operation called convolution. The convolution operator in its most general form is defined as

$$s(t) = (x * w)(t) = \int_{-\infty}^{\infty} x(\tau)w(t-\tau)d\tau.$$

x is referred to as the input, w is referred to as the kernel, and the result of the convolution, s, is often referred to as the feature map of the convolution operation. A discrete variant of the convolution operator over two variables can be defined as

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n),$$

and is similar to the convolution operation used in CNNs. A visualization of the convolution operator used in CNNs is shown in Figure 2.2.

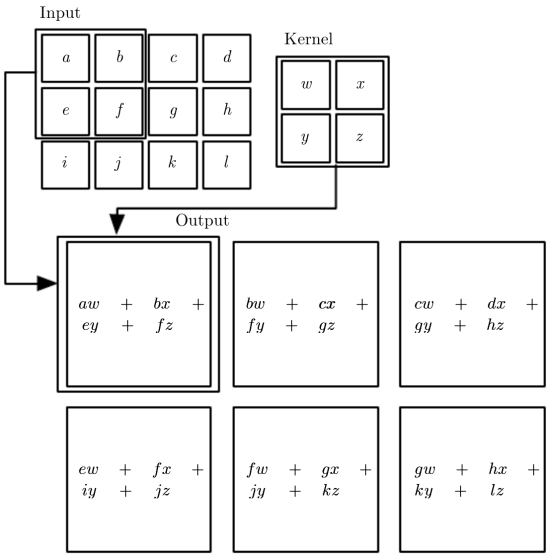


Figure 2.2: Illustration of the input, kernel and output of a convolution operator as it is often implemented in libraries. Note that the kernel is rotated. Illustration taken from Goodfellow et al. [2016]

Goodfellow et al. [2016] states that the convolution operator has three great advantages for machine learning: *sparse interactions*, *parameter sharing* and *equivalent representation*.

Sparse Interactions
The concept of connecting every node of the previous layer to every node in the next layer, as is done in FCNN, scales badly when the network dimensions get large. When the network size increase, the number of parameters to fit grows exponentially. When applying the convolution operation with a kernel that is smaller than the input size, each neuron in the layer is then connected to a local region in the previous layer and the number of parameters is greatly reduced. The kernel is usually several orders of magnitude smaller than the input size and the complexity is reduced accordingly, Goodfellow et al. [2016]. This kernel itself is a set of learnable parameters that are adjusted during training. The kernel used in CNNs is often referred to as a *filter*.

Parameter Sharing
The same filter is used across the entire input layer, and thus the parameters of the filter are shared across the input. This parameter sharing has the advantage of reducing the amount of parameters and complexity of the network.

Equivalent Representation
In convolutional neural networks parameter sharing causes a layer to have a prop-

erty called *equivariance* to translation, Goodfellow et al. [2016]. This means that if the input changes, the output will change in the same way. For instance, if the input is an image and the image is shifted slightly to the right, the activation of the convolutional layer is shifted slightly to the right compared to the non-shifted version of the input. This is of great advantage because the network now will become less sensitive to slight translational differences in the input.

The filter used in CNNs is commonly in size of 3×3, 5×5, 7×7 or 11×11. A filter always expands the whole depth dimension of the input. The depth of an input related to a CNN is often referred to as channels. The size of the filter defines the filter's receptive field.

2.2.1 Temporal dependencies with 3D convolution

For sequences of data with a temporal component, the network must be able to "understand" the temporal relation between data points in the sequence. This is particularly important in time series analysis such as language processing and forecasting problems.

The convolutional operator described above are only concerned of the spatial information in the data and are usually unaware of any temporal information in the dataset. The convolutional operator can be extended to be able to capture temporal information as well. This method is called *3D convolution* in literature and has shown promising results in the area of computer vision, in particular analyzing spatio-temporal data such as videos, Ji et al. [2013]; Tran et al. [2015]. A video is constructed of a sequence of frames (images) where each image contains spatial dependent information. The sequence of images in the video inhibits temporal correlations as the next frame in the sequence usually are correlated with the frame that came before in the temporal direction.

The input to a 3D convolutional layer are extended from one image (or image-like input) to a sequence of images stacked together in a cube. Each image in the input cube have identical size and number of channels. The convolution operator traverse the input tensor along all three axis: the width, height, and depth. This is different than the more common 2D convolution operator described above which only traverse the input over the width and height dimension. The 3D kernel connects features across the temporal dimension, and temporal dependencies such as motion can therefore be captured by the network.

2.3 Tree-based models and ensemble models

Tree-based models represents a different branch of machine learning. A decision tree is a tree-based model that is used in predictive tasks in machine learning, data mining, and statistics. The decision tree makes predictions by traversing a tree structure based on properties of the input (the branches) til it reaches a leaf node (the prediction). An ensemble model is a model that combines several models into one prediction model. The ensemble a collection of machine learning models with the underlying idea that many different weak classifiers or regressors perform better together than one would do alone.

Random Forest is an ensemble model that uses several decision trees as its core machine learning algorithm. Each decision tree is trained on a random sample of the dataset with replacement. This technique is called *bagging*. The Random Forest also applies a bagging-like algorithm to the different feature attributes that describe the data. Each decision tree is not only trained on a subset of the training data, but also on a subset of the data features. This makes each decision tree a week classifier or regressor, but a large collection of them together has shown to be more resilient to overfitting. The final predicted value of the Random Forest is usually the average predicted value of the underlying decision trees.

Boosting is another technique that is often used in ensemble methods. Instead of training each sub-algorithm on a random subset of the dataset, which is the case with the Random Forest, boosting ensures that the next classifier is likely to give more attention to the samples in the dataset that the previous sub-model predicted with a large error. Bagging is still applied, but each sub-model is now trained in sequence and the next sub-model in the sequence draws its training dataset with weighted probabilities based on the loss of the previous sub-model. This ensures that there is a higher probability of drawing the samples in the dataset that the previous sub-model predicted with high loss.

Gradient boosting is another ensemble method that has been proposed. Instead of training the next sub-model on a weighted subset of the dataset based on the loss of the previous sub-model, the next sub-model is trained to fit the difference between the target and the predictions of the previous sub-model. Given a sub-model M at training stage i, M_i, the next sub-model, M_{i+1}, can be expressed as

$$\begin{aligned} M_{i+1}(x) &= M_i(x) + h(x) = y \\ \implies h(x) &= y - M_i(x). \end{aligned}$$

Each step of the gradient boosting algorithm improves the previous step by modeling the residual, $h(x)$. XGBoost and LightGBM are two commonly used implementations based on this algorithm, Chen and Guestrin [2016]; Ke et al. [2017a].

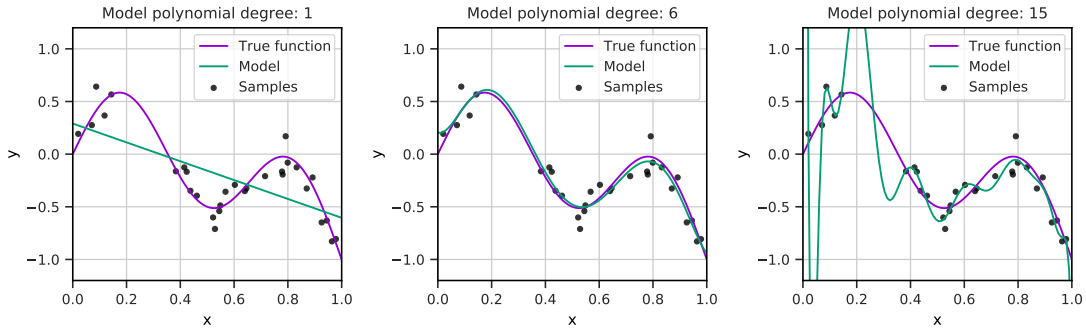


Figure 2.3: Three models of different complexity is fit to a set of data. The data points (black) are sampled from the function represented by the purple line with some noise. The purple line the *true* underlying structure of the data. **Left**: The model is too simple and is unable to correctly capture the structure of the data. The model is underfitting. **Center**: the model complexity is in line with the complexity of the data, and is able to converge towards a good generalization. **Right**: The model is too complex. Most of the data used for training is correctly labeled by the model, but the generalization is bad and the model would not perform well on new unseen data drawn from the same underlying distribution.

2.4 Generalization and regularization

As mentioned in Section 2.1, the goal of most, if not all, machine learning algorithms is to be able to generalize beyond the training data. If the model does not generalize well, the model complexity might not be right relative to the amount of data or the amount of data might not large enough to represent the underlying structure of the problem. A normal assumption is that the data is drawn at random from the space of all possible data points, and that a good model should be able to capture the underlying structure, if there is any, if we have enough data. The term *overfitting* and *underfitting* is usually used if there is an underlying structure in the data, but the model is unable to find it. An overfitted model is a model that contains more unknown parameters than what can be justified by the data, Everitt [1998]. Underfitted models are models where some parameters or terms that would appear in a correctly specified model are missing either by mistake or by design, Everitt [1998]. In practice, given the assumption that the data is drawn at random, overfitting usually happens if the model complexity is more complex than it should given the number of data points in the dataset used for training. Underfitting is the opposite case. Figure 2.3 shows a visual representation of overfitting and underfitting.

A possible solution to underfitting is to increase the model complexity. A feed forward neural network with one hidden layer that is sufficiently wide is able to approximate any continuous function with n parameters, Csáji [2001]. If the model underfits, it is possible to increase the complexity of the model by scaling up the size of the hidden layer.

Several regularization techniques can be employed to avoid overfitting. Different regularization techniques that can be used include L1, L2, dropout, and k-fold cross-validation. A common way of identifying overfitting is when the loss on a validation set during training starts to get worse or do not improve over time, Sarle [1996].

L1 and L2 regularization

L1 and L2 are regularization techniques that punish complex models in favor of simpler ones. One problem that can arise is that there are several sets of weights $\{W_i\}$ that gives the same predictions on the training data. By punishing a model with large weights, a model might converge towards a set of weights that is "simpler" and hopefully generalize better. The L1 regularization tries to minimize the sum of the weights and the L2 regularization tries to minimize the sum of the square of the weights. Figure 2.4 illustrates the concept of regularization

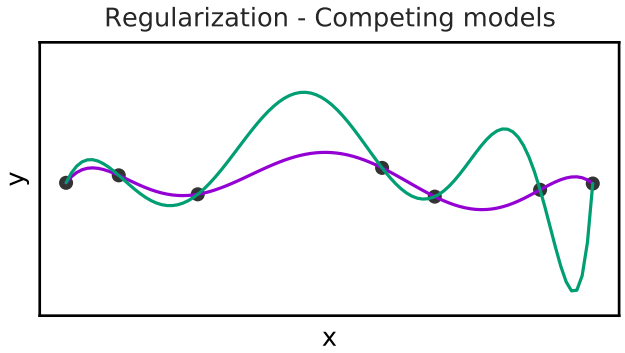
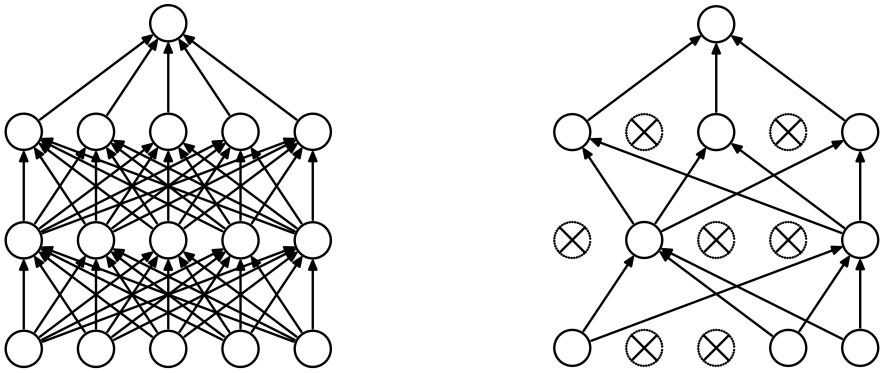


Figure 2.4: Two competing models, purple and green, that fits the dataset (black points) perfectly. With L1/L2 regularization the purple model is preferred to the green model as it is simpler and therefore more likely to generalize better given the underlying structure of the data.

Dropout regularization

Dropout is a recent regularization technique that has shown great success in preventing overfitting in ANNs, Srivastava et al. [2014]. This regularization technique



(a) ANN before dropout. (b) ANN after dropout.

Figure 2.5: Illustration of the dropout technique. **Right**: A plain neural network with two hidden layers that has all nodes and all connections available. **Left**: Dropout is applied to the network to the right and a thinned version of the network is produced and used. Nodes with crosses are deactivated. The illustration is taken from Srivastava et al. [2014].

tries to combat overfitting in a neural network by randomly deactivate nodes along with their connections during each training step. By deactivating the nodes in a layer, the network is prevented to create too large co-adaptations between different nodes. During training, the network that is chosen for a forward and backwards pass is sampled from an exponential large number of "thinned" networks with fewer nodes and connections. During testing, the final network reactivates all nodes along with their connections and scale down the weights to effectively create an average prediction of the thinned networks, Srivastava et al. [2014]. Figure 2.5 is taken from Srivastava et al. [2014] and illustrates the mechanism.

K-fold cross-validation

K-fold cross-validation is another technique to prevent overfitting. In k-fold cross-validation, the training dataset, D, is split into k different partitions, or *folds*, of approximately equal size, $D_1, D_2, \ldots, D_k$. During training, a network is trained and tested k different times, and each network, i, is trained on $D \setminus D_i$ and validated on D_i, $i = 1 \ldots k$. The final prediction is the average prediction of all the networks, Kohavi [1995].

Chapter 3

State of the Art and Related Works

As mentioned in Chapter 1, accurate regional wind power production volume prediction has become a central issue for power grid operators and traders in the new global energy market. Since a good model might give a competitive edge in the market, there is barley any published material on solutions to this exact problem. In this chapter, I will discuss the related work of this particular problem, similar problems in the same problem class, and discuss the state of the art of the methods that I will use.

3.1 Wind power prediction

Publications on the subject of wind power prediction more frequently choose to focus their research on power production at a single wind farm or at a specific location rather than the aggregate production of a large region. Several methods have been proposed, and the two main approaches that are most frequently discussed in the literature are to either make the prediction based on the historic power production volume or make the prediction based on the wind speed at the time of interest.

3.1.1 Models based on historic wind power production

Models that are based on the historic wind power production volume rely heavily on the assumption that the historic development of the wind power production inhabits some well behaved temporal patterns and information that strongly correlates with the production in the future. Kaya [2018] created and investigated

a hybrid model based on this assumption. The model combined *Empirical Mode Decomposition*, EMD, and a Random Forest regressor, RFR, for wind power forecasting at a specific location. The hybrid EMD-RFR model consisted of three main steps. First, the original wind power time series was decomposed into several intrinsic mode functions and a residual component using the EMD method. These components were more well behaved than the original wind power production time series, and were therefore easier to predict. The aggregate sum of these decomposed functions recreated the original wind power production time series. The different components was then used as input to a Random Forest regressor to forecast the next value of that particular component. Finally, the forecast value of each intrinsic mode function and residual component were summarized to create the final forecast for the wind power production.

Kaya [2018] trained the EMD-RFR model on wind power production data form a major energy company in Turkey with hourly temporal resolution from April 1 to April 30, 2015. The EMD-RFR model performed better than three other models that it was compared against: Support vector machine regressor, Random forest regression and a EMD-Support vector machine regressor.

Răzuşi and Eremia [2011] did a comparative study between ANNs and fuzzy inference system to predict the total wind power production in Romania. Similar to the study conducted by Kaya [2018], they only used historic wind power production data as basis for future predictions, and hence relied on the same assumption. Răzuşi and Eremia [2011] argued that because the total installed capacity is increasing over the years, it would be beneficial for the models to be trained using a sliding window approach through time of constant width over the data. The input to the ANN and the fuzzy inference system consisted of the last 10 production values of the target time series. Both models had a single target value as output and the target value corresponded to the wind power production volume at a particular time in the future ranging from 1 to 12 hours ahead. Răzuşi and Eremia [2011] concluded that having a separate network for each hour ahead in the forecast gave better results than one single model with multiple outputs. The different models was easier to train, and became more specialized. The neural network was constructed with one hidden layer with five neurons.

Răzuşi and Eremia [2011] trained the models on wind power production data of the entire Romanian power system with hourly resolution from June 12, 2010 to January 31, 2011. The sliding window was set to 1000 hours as a smaller window yielded sub-optimal results and a larger window provided little improvement. The experiments concluded that both models needed a large training data set in order to give good predictions. Both models provided better performance than the naive (persistence) model for a time horizon greater than 4 hours. The fuzzy inference

model performed better and required shorter training times an the ANN.

3.1.2 Models based on wind speed

As the name *wind power* suggest, wind turbines are designed to collect the kinetic energy in the wind and use it to generate electrical power. This means that there is a strong correlation between the wind speed at the location of the turbine and the electrical power that is generated at a specific point in time. Lydia et al. [2014] stated that the theoretical power, P, captured by a wind turbine can be estimated as

$$P_{\text{estimated}} = \frac{1}{2}\rho\pi R^2 C_p a^3, \tag{3.1}$$

where ρ is the air density, R is the radius of the rotor blades, C_p is a given power coefficient of the turbine, and a is the wind speed. Although this formula can be used to calculate the power output of a wind turbine, the conversion between the wind speed and the generated electrical power in a wind turbine is more commonly characterized by a function called the *wind power curve*. Calculating a reliable wind power curve is a challenging task due to the amount of different physical conditions that influence the power production at any given time, Wang et al. [2019]. However, rough estimations and generalizations can be made, and the wind power curve is usually divided into four different segments as shown in Figure 3.1. Before the *cut-in*, the wind speed is too low to generate any power and after the *cut-out* the wind speed is so strong that the turbine is shutdown to protect it from damages. In these two scenarios the power production is zero. In the segments between the *cut-in* and the *cut-out* wind speed the power production first increase at a cubic rate before it converges towards the manufacturers labeled optimal production rate. The cubic factor is in line with the theoretical power as described in equation (3.1).

Ding et al. [2019] developed and tested a model that estimated the wind power production based on the current wind speed at the location of the wind turbine. The model utilized bidirectional Gated Recurrent Units, GRU, to improve the wind speed forecast of a location based on Numerical Weather Predictions, NWP. They used the wind power curve to map wind speed to power production. Ding et al. [2019] observed that the measured wind speed at the location deviated significantly from the predicted wind speed from the NWP due to the local terrain. The large error in the predicted wind speed resulted in a large error in the estimated power production. To improve the wind speed predictions, the wind speed time series was divided into local windows of smaller length. For each window the wind speed time series was decomposed into intrinsic mode functions using Empirical Mode Decomposition, a similar approach as Kaya [2018]. This was done for both the

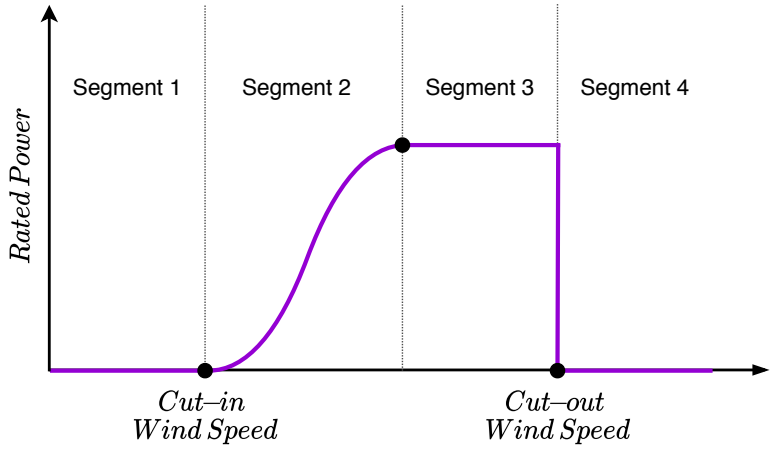


Figure 3.1: Typical wind power curve. Power production is 0 before the *cut-in* and after the *cut-out*. In segment 2, the wind power curve follows a cubic relation. In segment 3, the power production converges towards the manufacturers labeled optimal production rate. In the segments between the wind power curve has a sigmoid-like shape.

measured wind speed time series and for the wind speed time series provided by the NWP. An ANN based on the bidirectional GRU architecture was trained to map the time series provided by the NWP to the time series based on the measured wins speed at the location. The architecture is illustrated in Figure 3.2. The wind-power curve provided by the manufacturer was then used for wind speed to wind power conversion.

The data used for training and testing consisted of 1200 sample points with a 15 minutes temporal resolution from a wind farm located in the Sichuan Province, China, in 2016. Ding et al. [2019] concluded that the model performed very well when the NWP wind speed was much higher than the measured wind speed. Using the wind speed predicted by the proposed model was shown to be much better than using the NWP values directly, but it was only marginally better than compared models based on Support Vector Machine, Davò et al. [2016], and ANN, Buhan et al. [2016].

Lima et al. [2017] proposed a wind forecasting model based on NWP and statistical models. Similar to Ding et al. [2019], Lima et al. [2017] also observed that the wind speed forecast from the NWP deviated a lot from the measured wind speed at the location of the wind turbine. To solve this problem, Lima et al. [2017] used Kalman filtering techniques to reduce systematic errors in both the wind speed forecasting data and the predicted power production. In general, Kalman filtering techniques are used to estimate system states that can only be observed

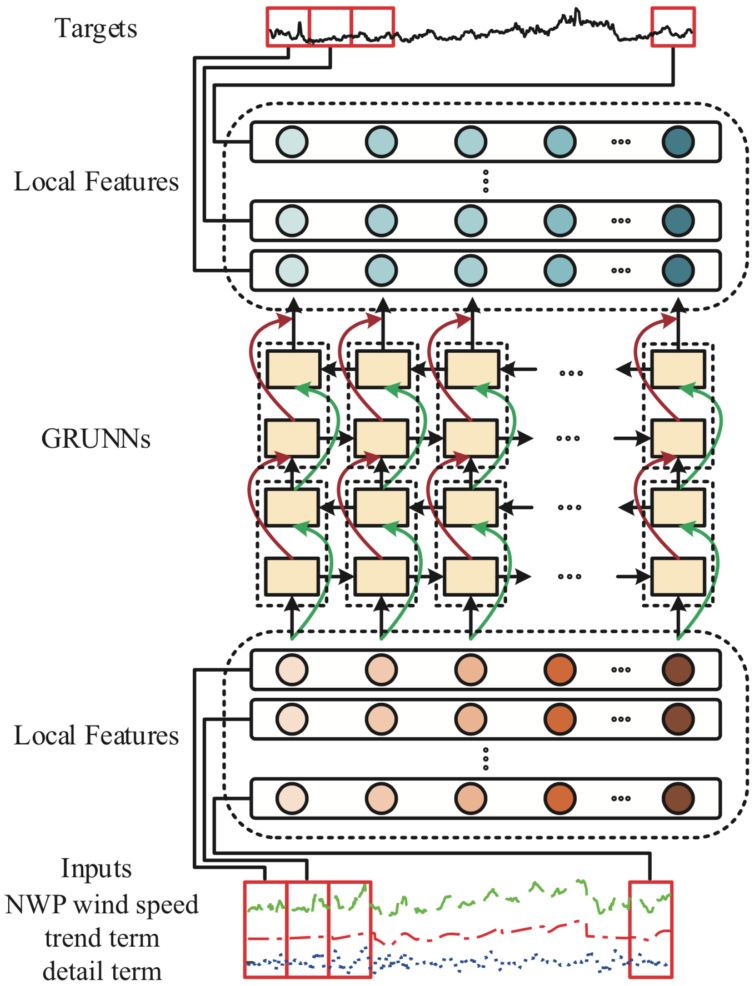


Figure 3.2: Illustration of the model proposed by Ding et al. [2019]. The input is the wind speed time series from the NWP, and the target is the wind speed time series based on measurements at the wind turbine location. Both series are partitioned with small temporal windows and the *Local Features* are found by decomposing the time series into intrinsic mode functions. The target series can be reconstructed as the sum of the *Local Features*. An ANN based on Bidirectional GRUs are used to map between the two time series. The illustration is taken from the original paper, Ding et al. [2019].

indirectly or inaccurately by the system itself. The estimates produced by the Kalman filtering method tends to be more accurate than the original estimates themselves, Kalman [1960].

The wind speed at two locations in Brazil was investigated by Lima et al. [2017], and in one of the locations the wind speed forecast was converted into wind power

forecast.

The first approach to the wind speed to wind power conversion used by Lima et al. [2017] was a polynomial regression approach originally proposed by Joensen et al. [1999]. The regression model was formulated as

$$p_{t+k} = aw_{t+k} + bw_{t+k}^2 + cw_{t+k}^3 + dp_t + l + e,$$

where p_{t+k} is the predicted wind power production at time $t + k$, w_{t+k} is the forecasted wind speed, p_t is the wind power production at the previous time step, a, b, c, d and l are corresponding weights, and e is the Gaussian systematic error. The second approach was to use the wind power curve to convert wind speed to wind power production directly. Using the wind power curve, the predicted wind power production can be calculated as

$$p_{t+k} = po(w_{t+k}) + e,$$

where $po : \mathbb{R} \to \mathbb{R}$ is the wind power curve similar to the one described in Figure 3.1, w_{t+k} is the wind speed at time $t + k$, and e is the Gaussian systematic error.

Lima et al. [2017] evaluated the model based on data from July 2012 to June 2013 and from December 2013 to June 2014 for the two different locations respectively. The time series had a temporal resolution of 10 minutes. They concluded that the estimated wind speed corrected by the Kalman filtering approach gave values that were closer to the real wind speed values than the NWP. Both the polynomial regression approach and the wind power curve approach provided similar results. Lima et al. [2017] suggested that the polynomial regression approach could be used when the wind power curve are not provided by the manufacturer or when the local geographical characteristics have a large influence on the relation between the wind speed and the power production.

Dolara et al. [2017] discussed the use of an Feed Forward Neural Network, FFNN, to predict the wind power production of a wind farm 24 hours ahead based on the NWP. Dolara et al. [2017] made a distinction between *direct power prediction* and *indirect power prediction*. In the direct power prediction, the FFNN made a power production prediction directly as output. In the indirect power prediction, the FFNN predicted the wind speed and a standard wind power curve was used to convert the wind speed to power production. Unfortunately, the details of the implementation and results are very limited and not provided in the paper, and the work by Dolara et al. [2017] is not reproducible given the lack of information regarding their model.

3.1.3 Data-driven approaches

Estimating the future wind power production to a satisfying level of accuracy with a purely data-driven approach is a relatively new area of interest. The vast amount of data available, and the availability of the computing power that is necessary to to analyze this data, has made computing-intensive methods appealing in recent years.

Gasparin et al. [2019] did a literature review of some data-driven deep learning methods that has been explored for the day-ahead power production prediction problem. Gasparin et al. [2019] identified that this problem was also referred to as *short term load forecasting*, STLF, in the literature. Deep learning methods, and in particular Temporal Convolutional Networks, showed promising performance. However, most of the previous work on this problem has been conducted on different datasets, which makes the models and approaches difficult to compare in an objective way. The limitations of these studies compared to the problem discussed in this thesis is that they usually discard spatial information in the data. Only the temporal information is considered, including the historical weather data and power production.

Liu et al. [2019] did a literature review of intelligent predictors and auxiliary methods that is used for deterministic wind power production prediction. Both shallow predictors and deep learning based predictors were compared. Liu et al. [2019] concluded that intelligent predictors had both high accuracy and effectiveness. Models that only relied on historical wind power production data was not convincible enough. Liu et al. [2019] identified that most of the existing literature had been constrained to look at one particular wind farm. They suggests that in order to obtain better universality, the forecasting model should not be limited to a fixed node, but target a large number of wind farms. The spatial and temporal correlation between different wind farms should be considered for feature work. This suggestion by Liu et al. [2019] is explored in may thesis in later chapters.

Tastu et al. [2014] investigated the problem of creating a probabilistic forecast of wind power production by accounting for geographically dispersed information. Tastu et al. [2014] focused their investigation on a single wind farm, but incorporated spatio-temporal information from 19 other wind farms with lead times from 15 minutes to 8 hours as basis for their model. They tested both parametric and non-parametric approaches to create a probabilistic forecast. The best performing approach was based on *adaptive quantile regression* using spatially corrected point forecasts as input. The results outperformed the compared models that used local information only by $1.5\% - 4.6\%$ depending on lead time. The quantile regression model was a non-parametric approach which meant that it did not rely on any

assumption of a known distribution of the data. The model was constructed by first finding the mean of the wind power predictions at any point in time and then find the uncertainty around these means. The data used for the wind power measurements had a temporal resolution of 15 minutes and the point forecast for the wind power production ranged from 0 to 48 hours with a temporal resolution of 15 minutes. The wind power forecast was generated with tool named *Wind Power Prediction Tool*, Nielsen et al. [2011]. Forecast for wind speed and direction at 10 m above ground was also included. A censored normal distribution without tails was used as model for the parametric approach, but based on overall skill, Tastu et al. [2014] concluded that the non-parametric adaptive quantile regression performed better.

Díaz et al. [2015] conducted a study where deep learning methods such as FCNNs and CNNs are used to predict wind power production by taking advantage of the spatial structure of the NWP patterns. In contrast to other models described in Section 3.1.2, Díaz et al. [2015] incorporated more parameters from the NWP data than just the wind speed alone. NWP parameters that was included for the model was: *pressure*, *temperature* at 2 m above ground level, and *wind velocity* at both surface level and 100 m above ground. The weather data was included for a whole region and structured in a grid-like pattern. Two different CNN architectures were tested. A standard CNN network with one convolution layer followed by two fully connected layers and LeNet-5, an architecture proposed by LeCun et al. [1998]. Two different fully connected networks were also tested. Díaz et al. [2015] concluded that although the models were undeniably powerful, the optimal architecture and best hyperparameters were difficult to set up and select. The CNN-based architectures outperformed the FCNN models as well as the baseline Gaussian SVR model. However, confidence intervals were not provided so the statistical significance of the presented results in the paper is unclear. Díaz et al. [2015] consider their work a first step in this particular research area, and encouraged further research on applying more complex convolutional models on this particular problem. They proposed that by running repeated experiments and selecting the M models with best validation score, the ensemble could potentially yield a better results. However, such experiments were not provided in the paper. Certain choices regarding the model developed and discussed in my thesis are motivated by the work done by Díaz et al. [2015]. Particularly including a larger range of numeric weather parameters in the data and creating an ensemble model based on the validation score.

Wilms et al. [2019] tried to exploit the spatio-temporal dependencies in in the NWP using a *Recurrent Neural Network* model for wind power prediction. Their work was focused on a single location containing one or several wind turbines. Wilms

et al. [2019] investigated the hypothesis that by including information about the wind speed and wind direction in the neighborhood of the location in question, the model might gain leverage on this information and give better predictions. The proposed model utilized a *Convolutional Long-Short Term Memory* Recurrent Neural Network denoted *convLSTM*. The advantage of this architecture was that it was be able to handle the temporal information in the wind speed and wind direction time series as well as the spatial dependencies from the geographically distributed locations. Wilms et al. [2019] experimented with different shapes of the input tensor. Their results showed that the structure of the dataset was of great importance to how well the model architecture was able to generalize. The models that were trained on a input tensor shaped and ordered according the the actual geographical shape of the region clearly outperform the models that were trained on a different ordering. This result indicates that the spatial correlations in the data is easiest captured when the different weather features are stacked along different channels of the input. The data structure used in this thesis is inspired by the finding of Wilms et al. [2019] and will also be used in my thesis.

Ju et al. [2019] developed a hybrid model combining a CNN and a tree-based learning algorithm, LightGBM, for *ultra-short-term wind power forecasting*. The combined model was trained over three separate steps. First, a feature set was generated by analyzing the wind power production time series for a particular wind turbine together with its adjacent wind turbines. Then, a CNN was used to extract information from the data by training the model with a loss function that compared the prediction done by the CNN with the the actual production volume. Finally, the features generated by the CNN after the convolution layer was flattened and used as input to a LightGBM model. The LightGBM model was then used to predict the final production volume. The time series data was organized in a set of time-order characters to better capture the temporal correlation in the target series. The time-order character consisted of n subsequent data points in the time dimension and was used to predict the production for data point $n + 1$. The data used in the model consisted of real sensor data from wind turbines located in a single wind farm in north China during 2013. The data had a temporal resolution of 5 minutes and included temperature, wind speed, various technical parameters for the operation of the wind turbine, the current production, and the production in the last 5 minutes. Ju et al. [2019] concluded that their combined model performed better than the CNN and LightGBM model was able to do by them self. However, a closer inspection of their results raise questions of the seemingly high auto-regressional behavior of the model. Ju et al. [2019] fails to address this issue. However, the idea of creating a combined model using a CNN for feature engineering and a tree-based learning algorithm for the final prediction will be explored in my thesis and will be discussed in later chapters.

3.2 Similar spatio-temporal dependent problems

The problem of wind power prediction can be classified as a problem that capture spatio-temporal characteristics of a collection of data. There exists published research on other problems in the same problem class with similar characteristics.

Ke et al. [2017b] looked at short term passenger demand forecasting for on-demand ride services. The paper considers both spatial, temporal and exogenous dependencies at the same time and proposed a novel deep ANN called FCL-Net. The FCL-Net architecture was built using ConvLSTM layers, standard LSTM layers, and Convolutional layers. The architecture was trained in one single end-to-end learning algorithm. The FCL-Net model utilized a Random Forest auxiliary model to identify the importance of independent variables that were later used for feature selection. Ke et al. [2017b] concluded that the fusion of convolutional techniques with the LSTM network architecture was able to capture spatio-temporal dependencies in their data to a higher degree than other models. Ke et al. [2017b] demonstrated in their experiments that the FCL-Net model achieved better performance than other commonly used machine learning algorithms such as XGBoost, LSTM, and CNN.

Ziat et al. [2017] discussed a novel model called Spatio-Temporal Neural Network, STNN, and explore forecasting problems that involve spatio-temporal dependent variables. They investigated the particular case of forecasting time series of spatial processes and uses wind speed forecasting and seawater temperature in a large geographical region for evaluation. The dynamics of the system was captured in the latent space, and a decoder was used to convert the prediction from the latent space to the real space. The STNN model and its variants presented in the paper performed well on the dataset they tested and were able to outperform other state of the art recurrent neural networks.

3.3 Image analysis

As mentioned in Section 2.2, CNNs have achieved great success with data that has a clear grid-structure topology such as the two-dimensional image topology [Goodfellow et al., 2016]. A great deal of previous research into image analysis with CNNs has focused on image classification, He et al. [2016]; Szegedy et al. [2017]; Tan and Le [2019]; Krizhevsky et al. [2012], object detection, Girshick [2015]; Ren et al. [2015]; Redmon and Farhadi [2018]; Liu et al. [2016]; Cai and Vasconcelos [2018], and image segmentation, Badrinarayanan et al. [2017]; Ronneberger et al. [2015]; He et al. [2017]; Kirillov et al. [2019]. Using the CNN architecture for image regression problems with a single continuous target value, which is more relevant in

the wind power production volume prediction problem, has been given less attention in the literature. Lathuilière et al. [2019] conducted a thorough systematic investigation of deep convolutional regression models on various computer vision tasks. Lathuilière et al. [2019] concluded that an adequately tuned vanilla deep CNN such as the VGG-16 or ResNet-50 with a linear regression layer on top could yield results close to other complex, ad-hoc regression models.

As the NWP data can be organized relative to the actual location of the data point in the geographical world, the NWP can be seen as a type of an image-like structure at a particular point in time over an area. The similarities between the data structure used in the NWP and the data structure used in images, motivates the use of a CNN as the main architecture for the deep learning based algorithm developed in my thesis. The developed deep learning architecture will be discussed in Chapter 4.

Chapter 4

Methods

The architecture of the deep learning-based model that is designed and evaluated in this thesis consists of several independent components. In this chapter, I will discuss the motivation and justification behind the different architectural choices, and give a detailed description of how each component works. The structure and preprocessing of the data has been important for various design choices and will be discussed in Section 4.1 and 4.2 respectively. The architecture of the artificial neural network that was developed in this thesis will be described and justified in Section 4.3. Section 4.4 will introduce the architecture for the hybrid model that combines the ANN for feature engineering with a tree-based learning algorithm for the final prediction.

4.1 Dataset

The data used in this study consists of 21 independent datasets, where each dataset corresponds to one of 21 different wind power production target series. In total, the datasets covered the total wind power production in Germany and in four Nordic European countries: Denmark, Norway, Sweden and Finland. Each dataset have identical structure and differed only in the geographical location of the source of production. The five countries is divided into 11 geographically separated, non-overlapping, regions resulting in two regions in Denmark, four in Germany, four in Norway, four in Sweden, and one in Finland.

Two regions in Denmark and two regions in Germany discriminates between offshore and onshore wind power production. For these regions, the wind power production have two different target series, one for onshore and one for offshore. In total there is therefore six regions in Denmark (including onshore, offshore and

combined series of both onshore and offshore), four in Germany (including onshore and offshore), four in Norway, four in Sweden, and one in Finland.

Each dataset, and therefore each region, have the same data structure. The data consists of three parts: weather data in the region, total wind power production volume in the region, and the total capacity in the region. All three have an hourly resolution. The capacity is an estimated number of how much electricity that can possibly be produced given optimal conditions. Because of the similarities between the different regions, the discussion about the datasets in this chapter will be a general discussion that applies to all regions which will be referred to as either *the region* or *the dataset*. The three different parts of the dataset are discussed below.

4.1.1 Weather data

The dataset contains the Numerical Weather Predictions, NWP, for the geographical region of interest. As mentioned, the NWP has an hourly temporal resolution, which means that the weather state over the region is known for every hour. The NWP is represented as an array of various weather parameter values at different geographical locations, where each location are separated from another by $0.125°$ in both longitude and latitude. The absolute equivalent representation of this separation varies depending on the geographical location if the data point. However, a reasonable approximation is that each numerical weather parameter data point represents a 10×10 km square in the geographical world. This approximation will be used throughout the rest of the thesis. The NWP data used in this thesis included three different weather characteristics averaged over the last 10 minutes before the timestamp of the data point. The weather characteristics are *temperature*, *atmospheric pressure*, and *wind velocity*.

In order to analyze the spatial correlations and relations in the weather information, the values of the NWP data at each timestamp t is restructured from n different arrays of data, corresponding to each of the n different weather parameters, to n different matrices. The mapping from a one-dimensional array to a two-dimensional matrix is applied by preserving the relative geographical location of the data points. This representation could be beneficial for the model accuracy according to the work done by Wilms et al. [2019]. The structure ensures that the data that belongs to geographical locations that are close together in the real world are organized close together in their numeric representation. In other words, the matrices are constructed such that if one 10×10 km square, S_a, is directly north of another 10×10 km square, S_b, then S_a is placed in the same column and in the row above S_b in the matrix, Figure 4.1. The same principle applies for west and east relations. Following this strategy, the weather data is now structured relative to the geographical location of the data and it resembled an overlay to a

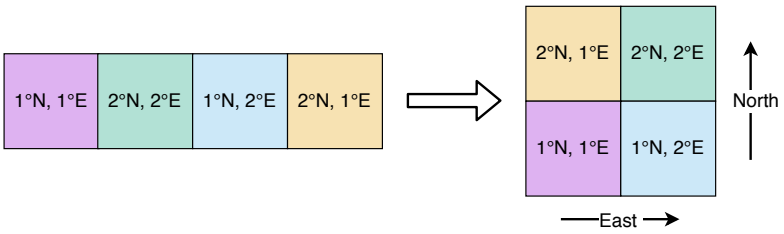


Figure 4.1: A simple example of the geographical reordering of the NWP. Each square represents a 10×10 km square with a corresponding geographical location represented with a longitude and latitude coordinates. The one-dimensional array is rearranged into a two-dimensional matrix where the relative ordering of the elements corresponds to the geographical location of the square. The simple coordinates are used for illustration purposes.

standard European map.

An issue with this mapping is that the borders of the electrical power regions are non-rectangular while the matrix containing the weather parameter values are of rectangular shape. For some of the regions the dataset includes the NWP data in a rectangular grid around the electrical power region that was large enough to contain the borders of the electrical power region itself, but not so large that it contained too much noisy and unimportant weather data that was outside of the region. In this scenario, the conversion to the n weather parameter matrices is a trivial task, as the shape of the matrix will be equal to the number of different latitudes and longitudes included in the NWP. Other regions in the dataset only included the relevant NWP locations for that particular region which resulted in a non-rectangular shape of 10×10 km squares. In these cases the elements of the n weather parameter matrices that was not included in the NWP was padded with zeros outside of the region borders to create a rectangular shape, Figure 4.2.

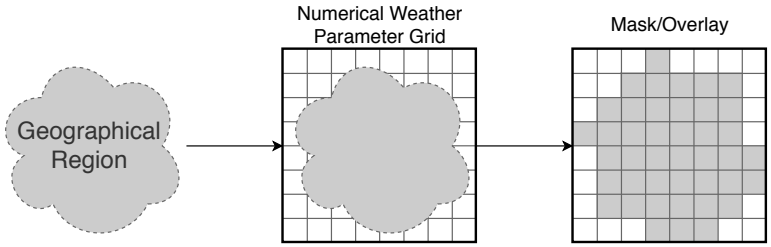


Figure 4.2: Zero-padding of non-rectangular NWP. The numeric weather prediction values are placed on top of a grid of zeros for padding.

A brief description of the three different weather parameters included in the datasets is described below.

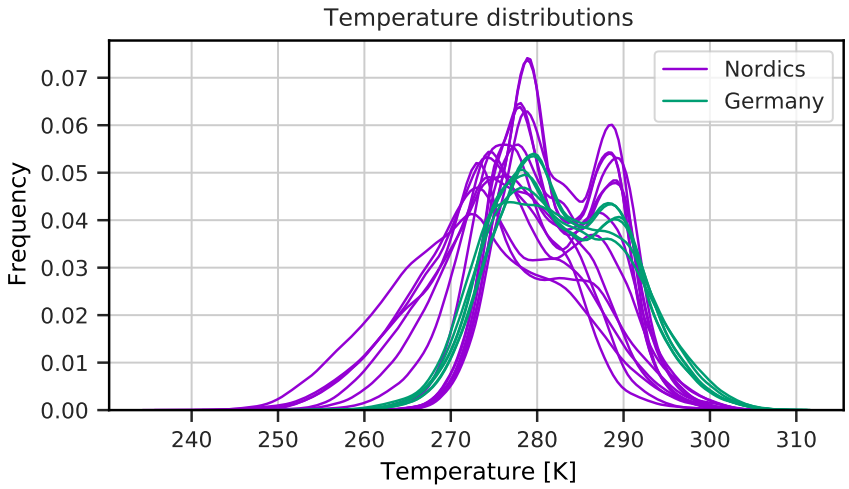


Figure 4.3: Temperature distribution of the different regions. Each line represents the temperature distribution of a region. The regions are listed in Table 5.1 and 5.2.

Temperature

The weather forecast of the temperature is estimated at 2 meters above ground. The temperature is given in the units of Kelvin. The distribution of temperature for the different regions are shown in Figure 4.3.

As can be seen in the figure, the regions located in Germany have a more consistent shape of the temperature distribution than the regions in the Nordics. This is not surprising, as the temperature distribution is expected to varies a lot based on latitude.

Atmospheric pressure

The weather forecast for atmospheric pressure is normalized to sea level. The actual pressure at the location of the wind turbines varies based on the geographical topology in the region. For the regions that are close to sea level, the forecasted value is expected to be similar to the atmospheric pressure at the actual location. For the regions that have a more changing terrain (e.g. the regions in Norway and south Germany), the forecast might deviate more from the actual atmospheric pressure. The pressure is measured in hPa. The distribution of atmospheric pressure for the different regions are shown in Figure 4.4.

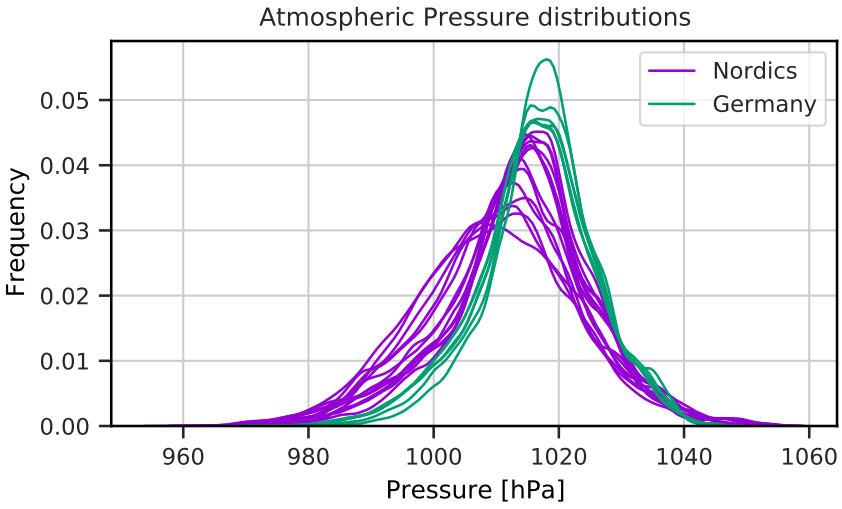


Figure 4.4: Atmospheric pressure distribution of the different regions. Each line represents the atmospheric pressure distribution of a region. The regions are listed in Table 5.1 and 5.2.

Wind velocity

The forecast for the wind velocity is given at 100 m above ground level. According to González-Aparicio and Monforti [2017], this height corresponds well with the average hub height of wind turbines installed in the European Union. The wind forecast is given in two components, an u-component and a v-component. These two components are a decomposition of the wind velocity vector along the west to east axis and the south to north axis respectively. The wind speed is measured in meters per second. The distribution of wind speeds for the different regions are shown in Figure 4.5.

Wind speed exceeding $25\,\mathrm{m\,s^{-1}}$ were set to 0. This was motivated by the shape of the wind power curve described in Figure 3.1 and the fact that most wind turbines do not produce power when the wind speed is over a *cut-out* threshold value, usually $25\,\mathrm{m\,s^{-1}}$.

4.1.2 Target series

The target series in the dataset consists of hourly aggregated wind power production volume in the relevant region. This is the total amount of electrical energy produced by wind turbines in the region during the last hour, measured in MWh/h. The instant production power is difficult to measure accurately, so the values in the

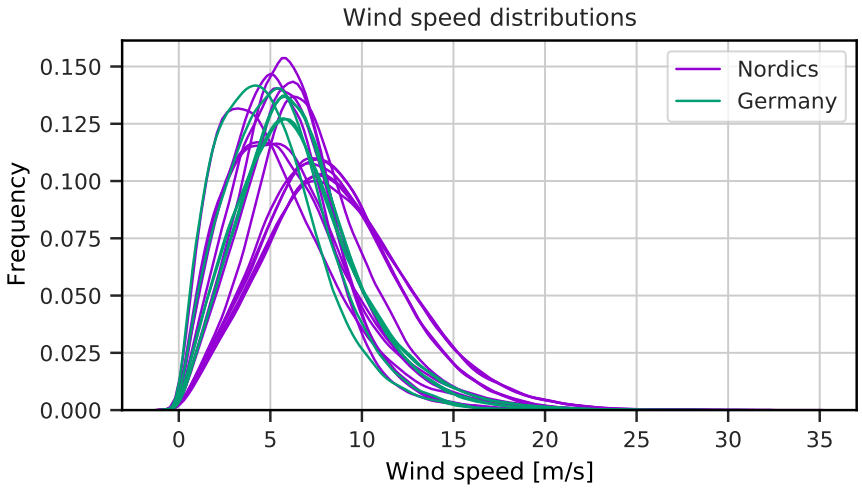


Figure 4.5: Wind speed distribution of the different regions. Each line represents the wind speed distribution of a region. The regions are listed in Table 5.1 and 5.2. The wind speed is measured in meters per second.

dataset are estimates averaged over the last hour. The dataset also contains the production capacity of each region. The capacity indicates how much electricity that can be produced at any given time if the weather conditions are optimal.

4.2 Data preprocessing

Due to the physical properties of the system, the weather state identified by the NWP is the strongest driving factor in this analysis. Careful prepossessing of the data is important to achieve good performance in the models.

4.2.1 Weather data

Most wind turbines are built with an automatic jaw control unit that enables the hub of the turbine to rotate such that the blades are directly facing the direction of the wind. This is favorable, as the production is reduced if the wind is blowing against the turbine blades from an angle. Because of this, it makes sense to convert the wind velocity from its (u, v)-component into polar coordinates, (r, θ), where r is the wind speed measured in meters per second and θ is the angle of the wind direction. The rotational control unit makes the angle θ a less important feature, and it is therefore removed from the model input to reduce the amount of unimportant input features.

As mentioned in Equation (3.1), the theoretical power, P, captured by a wind turbine can be calculated as

$$P = \frac{1}{2}\rho\pi R^2 C_p a^3,$$

where ρ is the air density, R is the radius of the rotor blades, C_p is a power coefficient and a is the wind speed, Lydia et al. [2014]. Since the wind turbine is locked in their location on the ground, the air pressure will change slowly and fluctuate little while the wind speed, on the other hand, will vary a lot. The relation between the different parameters of Equation (3.1) indicated that the wind speed a is the driving factor for the power production in a wind turbine, as one might expect. Representing the wind velocity with polar coordinates which separates out the absolute wind speed and angle is therefor expected to help the model predict more accurately.

Normalization strategies

The weather forecast as it is presented in its original form is unsuited for training a neural network. The true values for the pressure, temperature and wind speed are large and of a different order of magnitude compared to each other. The distribution of the weather parameter values as seen in Figure 4.3, 4.4, and 4.5, resembles a (slightly deformed) normal distribution. It would therefore be natural to try and normalize the data with a mean of zero and unit variance. This type of normalization is often referred to as *Standardization* or *Z-score Normalization* in literature and is defined as

$$x' = \frac{x - \bar{x}}{\sigma},$$

where x' is the normalized value, x is the original value, $\bar{x}$ is the average value of the variable, and σ is the corresponding standard deviation.

Another normalization strategy that is commonly used for feature scaling is *Rescaling*. This method is also referred to as *min-max normalization* in literature and is defined as

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)},$$

where x' is the normalized value and x is the original value.

In unordered data it is common to normalize a parameter x by using the global mean and standard deviation of x based on the dataset used during training. However, in the problem discussed in this thesis, the data are ordered and fixed in geographical space across the time dimension. This opens up different possibilities on how to normalize the data. Instead of normalizing a parameter using its global

Strategy	Description
NS1: Global mean/std	For each parameter, take the global mean and standard deviation and calculate the Z-score Normalization.
NS2: Local mean/std	For each grid location of each parameter, take the mean and standard deviation of that location and calculate Z-score Normalization at that location
NS3: Global min-max	For each parameter, rescale the range to the interval $[0, 1]$ using the min-max normalization based on the global distribution
NS4: Local min-max	For each grid location of each parameter, rescale the local range to the interval $[0, 1]$ using the min-max normalization based on the local distribution at that location

Table 4.1: Different NWP normalization strategies.

characteristics, the weather parameter can be normalized given the characteristics of the value distribution at the local geographical location of the data. Combining this with the two different normalization methods described above gives four different normalization strategies. The four different normalization strategies are described in Table 4.1.

4.2.2 Target series

As previously mentioned, the total amount of electrical energy that can potentially be produced in a region is increasing every year, Figure 1.1. New wind farms are built and new wind turbines are both larger and more efficient than the old ones. This means that the target series of total production volume has an increasing trend over time. For this reason, it is challenging to construct a model that targets the absolute production volume alone, as one particular weather situation a given year would produce less energy than the same weather situation a few years later. This property makes old data less valuable to use during training, and effectively creates a trade-off between the amount historical production data of a region and the amount of "good" historical data (with similar capacity) that could be used for training.

The challenge with ever-increasing capacity can be solved to some degree by making the target series unit-less. A unit-less target series can be constructed from the production target series and the capacity series by calculating the ratio:

$$\text{ratio}(t) = r(t) = \frac{\text{production}(t)}{\text{capacity}(t)}.$$

This new target series will have the property $r(t) \in [0, 1]$ because the production has a lower bound at 0% and an upper bound of 100% of total capacity. This ratio will be referred to as the *load factor*.

Predicting the load factor instead of the absolute production volume is preferable in two ways:

1. The target values are bounded between 0 and 1 for each region, and it is therefore a probability that the neural network will be able to generalize better.

2. The prediction will be independent of the capacity of the region. As the capacity of the region is likely to increase over time, a given weather situation a certain year will yield a different production volume than an identical weather situation a year later. Using the unit-less load factor as target will increase the information value in older data. Normalizing and targeting the load factor are beneficial under the assumption that new wind turbines are randomly distributed or constructed in areas that are known to be good. With that assumption, a given weather state will yield the same load factor independent on the capacity of the region at that time.

In an optimal situation, the total production volume is equal to the capacity of the region. However, this is unlikely to occur. Far more often than not, the weather conditions are not ideal and the production is only a fraction of the total capacity. This is typically the case for onshore regions. The distribution of the load factor over time are skewed toward the lower half of the range. To account for this imbalance, a transformation $T : \mathbb{R} \to \mathbb{R}$ can be applies on the target load factor which will increase the granularity of the lower values, while decreasing the granularity of the higher values. The transformation is formulated as

$$\hat{r} = T(r) = -r^2 + 2r, \tag{4.1}$$

where r is the real load factor and $\hat{r}$ is the transformed load factor. The inverses transformation can be formulated in a similar way

$$r = T^{-1}(\hat{r}) = 1 - \sqrt{1 - \hat{r}}. \tag{4.2}$$

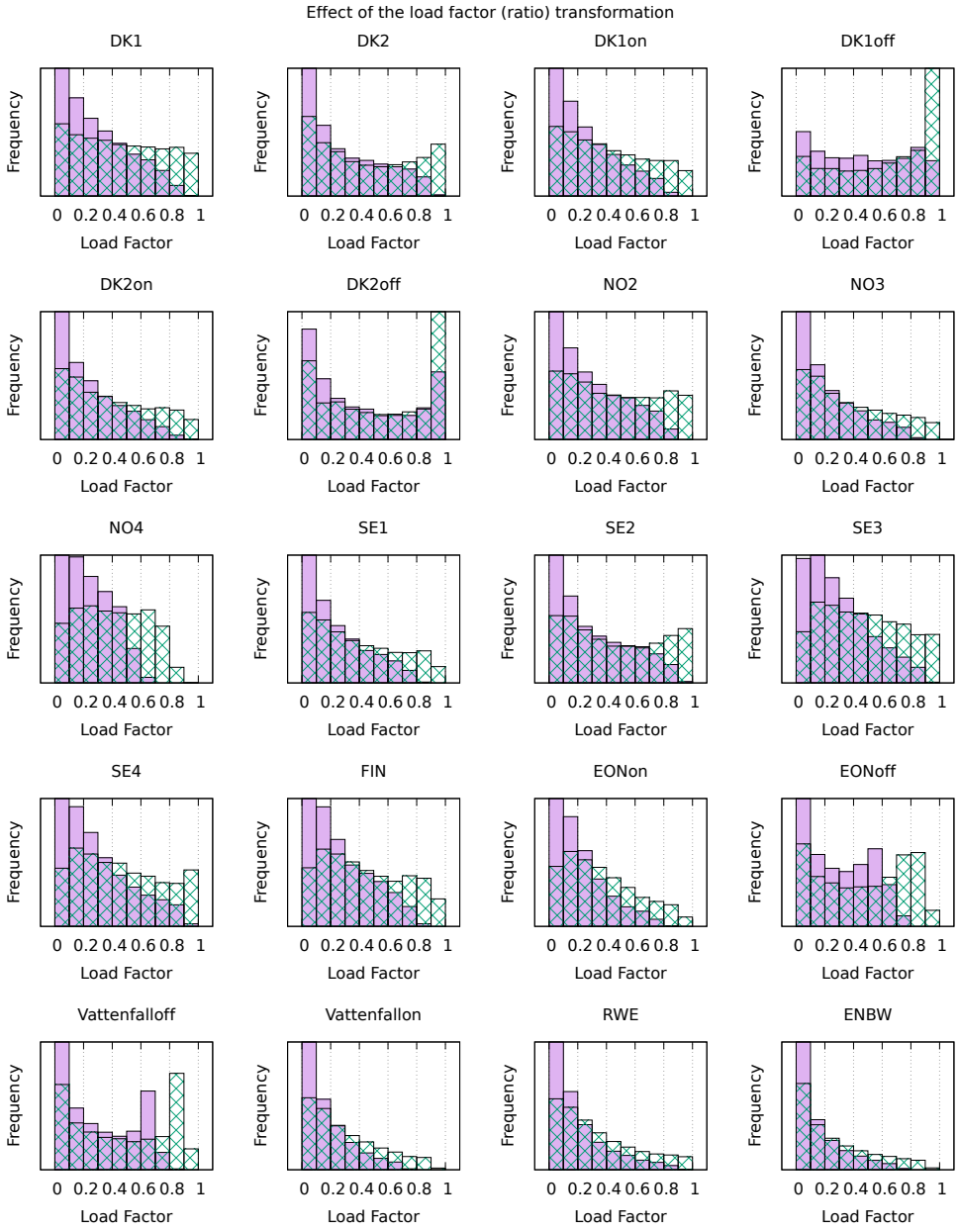


Figure 4.6: Transformation of the target series. The distribution of the load factor (ratio) values becomes more even at the expense of more coarse resolution when the load factor values close to 1. The solid bars is the distribution of the real target load factor (ratio). The textured bars are the distribution of the load factor after the transformation in Equation (4.1) is applied.

The distribution of the transformed and non-transformed (real) target load factor for all regions is shown in Figure 4.6. As can be seen in Figure 4.6, the transformation strategy makes the distribution more uniform across the different regions. This is desirable, as an uneven distribution might be more difficult to train due to bias on the lower half of the dataset. The regions that already have a distribution that is close to an uniform distribution does not seem to improve with the transformation strategy. This is particularly evident in region DK1 (offshore) and DK2 (offshore) where the transformation have made the distribution more unevenly distributed.

The load factor transformation comes at a cost. The resolution on the predictions at the higher load factor values decreases, which will lead to larger absolute errors in the predicted volume at situations where the target load factor is close to 1. The effect of the load factor transformation will be discussed in Chapter 6.

4.3 Artificial Neural Network

In order to get leverage on the spatial correlations in the weather data, a CNN is used as basis for the deep learning-based model. As mentioned in Section 3.3, CNNs have shown great success in other image analysis tasks and it is therefore possible that the CNN architecture will be able to extract important spatial information from the data in this problem as well. As discussed above, the restructured NWP data consists of multiple matrices containing numerical weather data for three relevant weather parameters. The matrices are stacked on top of each other to create a single tensor describing the weather state over a region at a particular hour. This is illustrated in Figure 4.7. This data structure is similar to an ordinary RGB image where each weather parameter over a region corresponds to one channel in the final "image" tensor describing the weather state of the region at that time.

The architecture of the CNN model developed in this thesis is illustrated in Figure 4.8. The motivation for the different components are outlined below. Batch Normalization, Ioffe and Szegedy [2015], is used for every convolutional layer. Dropout, Srivastava et al. [2014], is used in the final fully connected network with a probability of 0.2. The *Swish* activation function, Ramachandran et al. [2017], with $\beta = 1$ is used as activation functino for each layer throughout the network.

3D convolution

The spatio-temporal dependencies in hourly weather data over a fixed region motivates the use of 3D convolution as part of the model. The weather state at a given

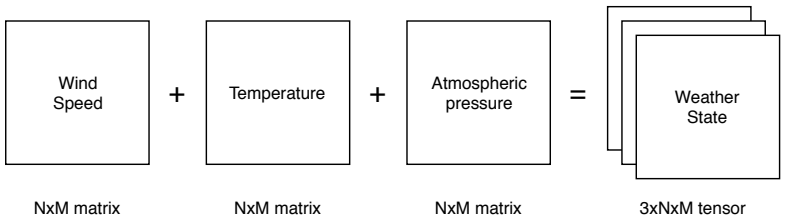


Figure 4.7: The structure of the input data. The values of each weather parameter are organized in a matrix relative to the real geographical location of the measured value of that parameter. If the region is not rectangular, the matrix is padded with zeros at the locations outside the region to make a rectangular shape. The different parameter matrices are stacked to a tensor.

location at time t is likely correlated with the weather state at the coordinates in close proximity along both spatial and temporal dimensions. Allowing the network to operate on weather data not only for the particular hour of interest, but also at a selection of hours before and after as well, might help the network achieve better generalization of the problem and therefore make better predictions. Mehrkanoon [2019] used a similar approach for weather forecasting with promising results which motivates the use of 3D convolution here as well.

The 3D convolution block in Figure 4.8 consists of two convolution layers with batch normalization.

2D convolution

As mentioned in Section 3.1.3, the spatial relation between wind farms is seldom considered and Liu et al. [2019] suggests that the spatial and temporal correlation between different wind farms should be considered for feature work to obtain better universality. To emphasize on the information in the spatial correlation that exists in the physical weather system, a separate set of 2D convolution layers are applied after the 3D convolution. The motivation for this part of the model is to be able to capture the local connections between the different 10×10 km squares similar to how other, more conventional, CNN models capture local spatial connections in a multi channel input image, as mentioned in Section 3.3.

The 2D convolution block in Figure 4.8 consists of three convolution layers with batch normalization.

Spatial Pyramid Pooling (SPP)

The parameters of a convolutional layer in a CNN is specified with a weight matrix for the kernel, the kernel size, stride and padding. These parameters are indepen-

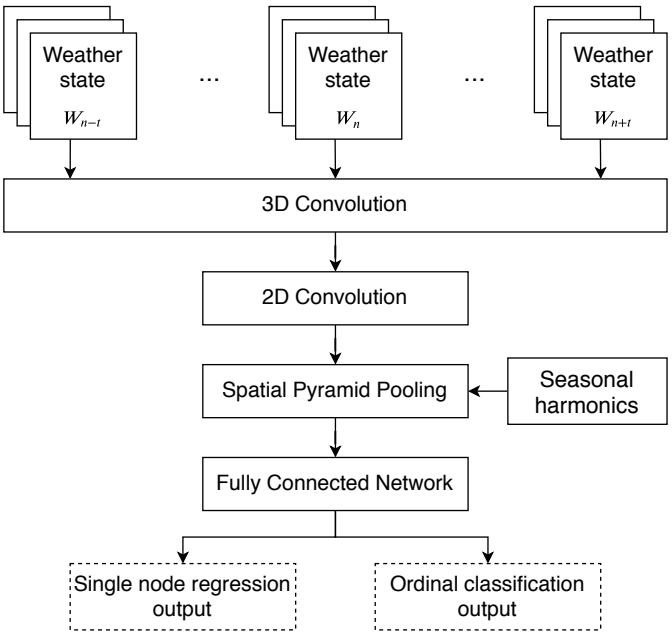


Figure 4.8: Overview of the architecture of the Convolutional Neural Network. The model accept a stack of weather states where each state is over a particular region at a fixed point in time. The different weather states that is used as input is separated along the temporal dimension. The Input a shape as described in Figure 4.7. Spatial Pyramid Pooling is used to down-sample the tensor to a flat array of fixed size independent of input size. 25 Seasonal harmonics are appended to the flattened array before the fully connected network. The output of the network can be configured to be either a Single Node Regression output or a Ordinal Classification output. Both variants are tested in this thesis.

dent of the width and height of the input as the convolutional operator is only concerned about the number of channels. This property means that the output size of the convolutional layer is dependent on the parameters of the layer as well as the size of the input. A fully connected layer requires a flat input vector of nodes. The achieve this, the output after a set of convolution layers is flattened before it is used as input to the fully connected network. This is usually done by just reshaping the tensor to one dimension.

An issue with this approach is that the number of nodes in the last last layer before the fully connected layer must be known in advance. This is because the size of the weight matrix in the fully connected layer is a hyperparameter that must be specified before run-time. If the size of the input is constant, this is not an issue as the size after the convolution operations can be calculated in advance. However, in this study the model must be able to accept inputs of different sizes as the

same model architecture is tested on different electrical power regions that varies in size. There are several strategies that can be used to solve this problem. One strategy that is commonly used in image classification is cropping and warping. An image can be cropped or warped to fit a predefined size that is accepted by the network while still preserve the relevant information in the image. I believe that this approach might not be beneficial in this problem, as each number in the data is a measurement of a weather parameters fixed to specific geographical location. A crop could potentially discard necessary information and warping the input tensor would require interpolation or other means of generating data that is not there. Another solution to the problem would be to construct N different model architectures for the N different regions. This could possibly be beneficial for the predictions as the model architecture is specifically made for the region of interest, but it would be difficult to give a fair evaluation of the model architecture performance across the different regions since the architecture would not be the same.

He et al. [2015] proposed another method called Spatial Pyramid Pooling (SPP) to solve this particular problem. SPP is a down-sampling method that is able to create a fixed size vector representation of the convolution activations independent on the size of the input tensor. He et al. [2015] conclude that the Spatial Pyramid Pooling should be able to improve any CNN-based image classification methods as the network can train on images of different sizes without the need for cropping or warping the input to conform to one predefined size and therefore preserve informaion in the input that would otherwise be lost.

The SPP layer is constructed with N number of different pooling layers consisting of bins of different sizes relative to the size of the input. The pooling layers applies a *maxpool* operation on each feature map bounded to the size of the bins of the pooling layer. For example, a 1×1 bin is a bin that contains the whole input and the *maxpool* operation on this bin results in a single value. A 2×2 bin divide the input into 2×2 chucks and apply the *maxpool* operation on each chunk resulting in four final values. Figure 4.9 illustrates a SPP layer with three layers of pooling with 1×1, 2×2, and 4×4 bins respectively.

The SPP block in Figure 4.8 consists of five bins of size 1×1 to 5×5.

Seasonal harmonics

Due to several seasonal variation in the weather (such as humidity, the presence of snow, etc.), the wind power production volume is correlated to the time of year as well as the current wind speed, temperature, and atmospheric pressure that is included in the NWP in the dataset. To force seasonality to the time series,

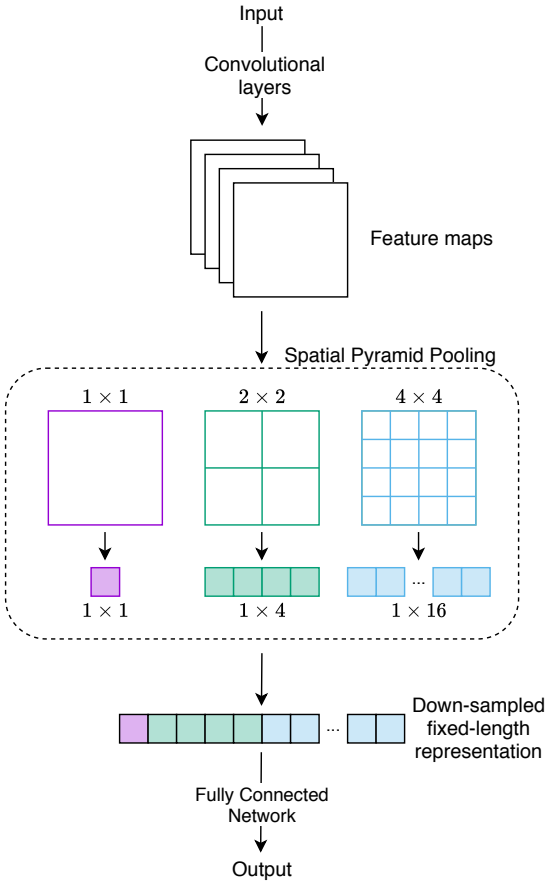


Figure 4.9: An illustration of a Spatial Pyramid Pooling block between the convolutional layers and the fully connected layers of a generic CNN. The SPP layer consists of 3 different pooling layers with 1×1, 2×2, and 4×4 bins. This configuration results in the one-dimensional representation of the feature maps is of length $(1 + 4 + 16) \times d$ where d is the number of channels.

the dataset includes a set of N artificially constructed harmonic functions with different periods, $\theta_i(t)$ with $0 < i \leq N$. Each of the harmonic functions have the form

$$\theta_i(t) = \cos(\omega_i t + \phi_i), \tag{4.3}$$

where ω_i is the frequency and ϕ_i is the initial phase. Each timestamp in the dataset will therefore have an encoding corresponding to a vector of N elements with numbers ranging from $[-1, 1]$ given by the N different harmonic functions.

The dataset includes 25 different harmonic functions. The periods of the different harmonic oscillators varies from 6 h at the minimum to 6 months at the maximum. The values of these functions are appended to the one-dimensional feature repre-

sentation right before the fully connected network. Using the harmonic functions as a representation of seasonality is preferred over simply using the timestamp at the relevant hour as the predictions will be independent on the absolute time of the prediction. The implication of this is discussed in more detail in Section 6.2.4.

Fully Connected Network (FC)

The fully connected network is used as the final predictor of the CNN model. The final prediction of the network can be expressed either as a *Single Node Regression output* or as a *Ordinal Classification output*. The Single Node Regression output consists of a single node with a sigmoid activation function,

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

This Single Node Regression output should give good results for a CNN-based regression network according to Lathuilière et al. [2019]. The target value of the network is the ratio between the produced volume and the total capacity of the region. The total capacity changes slowly and corresponds to the maximal amount of electrical energy that can be produced in the region given an optimal weather situation. The sigmoid activation at the end of the single node output ensures that the target values are always within $(0, 1)$, as expected.

Cheng et al. [2008] proposed a different learning strategy for models constructed to solve an ordinal regression problem using CNN. The method is called *ordinal regression*. This method combines regression with classification for the CNN. Given N different classes denoted $o_1, \ldots, o_N$, with order relation, $<$, then the classes can ordered such that $o_1 < o_2 < \cdots < o_N$ without loss of generality. If the target space is continuous, an order classification can be constructed by discretize the continuous target space into non-overlapping segments with an ordinal relation.

Given the target class o_i, many common image classification problems encodes this class as a one-hot vector of all zeros except for the element at position i which has value 1. The last layer of the network applies a softmax function,

$$softmax(z_i) = \frac{\exp(-z_i)}{\sum_{i=1}^{N} \exp(-z_i)},$$

that is applied to all N nodes (classes). The same target class encoded using Ordinal Classification is a vector where all classes of lower or equal ordinal value than the target value is set to 1 and all classes with higher ordinal value that the target class is set to 0. The sigmoid activation function is applied to each node in the final layer of the network instead of the softmax function. An illustration of

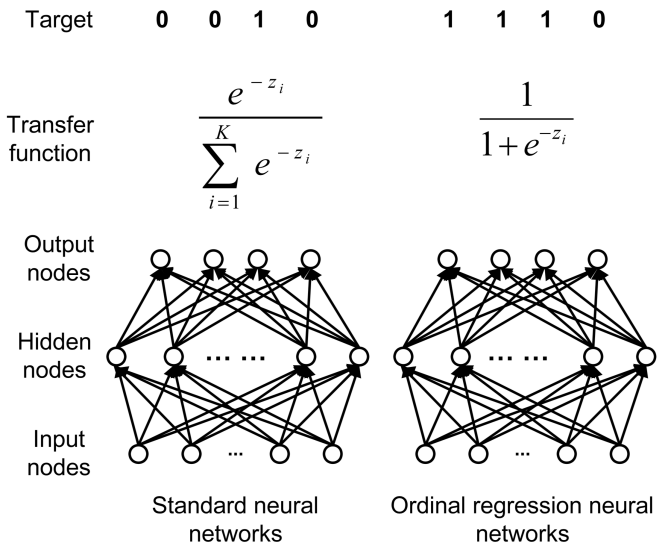


Figure 4.10: An illustration of how Ordinal Classification differs from "standard" classification. Assume four classes, $A - D$, where $A < B < C < D$. If the target category is C then the "standard" neural network would have target vector $(0, 0, 1, 0)$, while the target vector using Ordinal Classification would be $(1, 1, 1, 0)$. The final activation function of the "standard" neural network is the softmax function and the final activation function for the Ordinal Classification network is the standard sigmoid function applied to each node. The figure and example is taken from Cheng et al. [2008].

the method is shown in Figure 4.10. In this thesis I have chosen to refer to this method as *Ordinal Classification* to make the distinction from the *Single Node Regression* output more clear.

The output nodes of the network is not guaranteed to follow a monotonic relation, so the decoding of the predicted class is not trivial. To decode the predictions, the output nodes are scanned in order until one node has a value less than a predefined threshold. In the model evaluated in this thesis, the threshold is set to 0.5.

The strategy of categorizing a seemingly continuous target space has been used with success on other problems as well such as text-to-speech generation model WaveNet, Oord et al. [2016a], and image reconstruction with PixelRNN, Oord et al. [2016b].

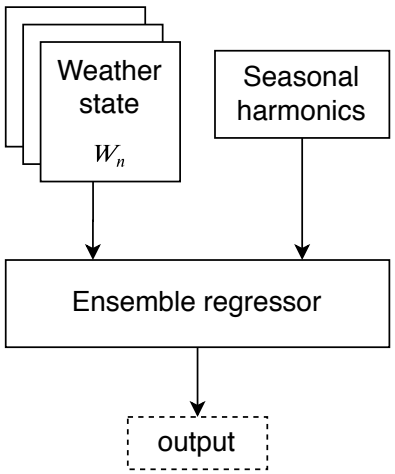


Figure 4.11: Architecture of the ensemble regressor. The ensemble regressor is a decision tree-based method such as Random Forest or LightGBM. The input is a combination of the numeric weather data at a given timestamp and the seasonal harmonics corresponding to that timestamp. The output is a single continuous value representing the production ratio at that timestamp.

4.4 Decision Tree and the hybrid model

The dataset used in this thesis inhibits structural and tabular data. Each point in the numerical weather data is ordered according to a fixed location in the geographical world and the wind turbines can be considered locked in place after they are constructed. This nature of the data motivates the use of a decision tree-based machine learning algorithm to extract the relevance of each individual feature and use that for prediction, Safavian and Landgrebe [1991].

An ensemble-based decision tree algorithm such as a Random Forest, Liaw and Wiener [2002], or LightGBM, Ke et al. [2017a], tends to generalize better than a standard decision tree alone. The decision tree-based ensemble algorithm can be trained on a tabular version of the dataset where each row corresponds to one particular hour, and the features on the columns correspond to the weather state at that hour. In addition to the weather data for each individual timestamp, each row also contains the the values of the harmonic functions for that timestamp (as described above). That way, the weather state is represented as a single one-dimensional array where each element in the array corresponds to a particular weather parameter at a constant geographical location. The location grid is striped every third block to improve training time and feature complexity. This will discard some weather information but will also reduce the complexity of the model. Figure 4.11 shows a simple schematic of this architecture.

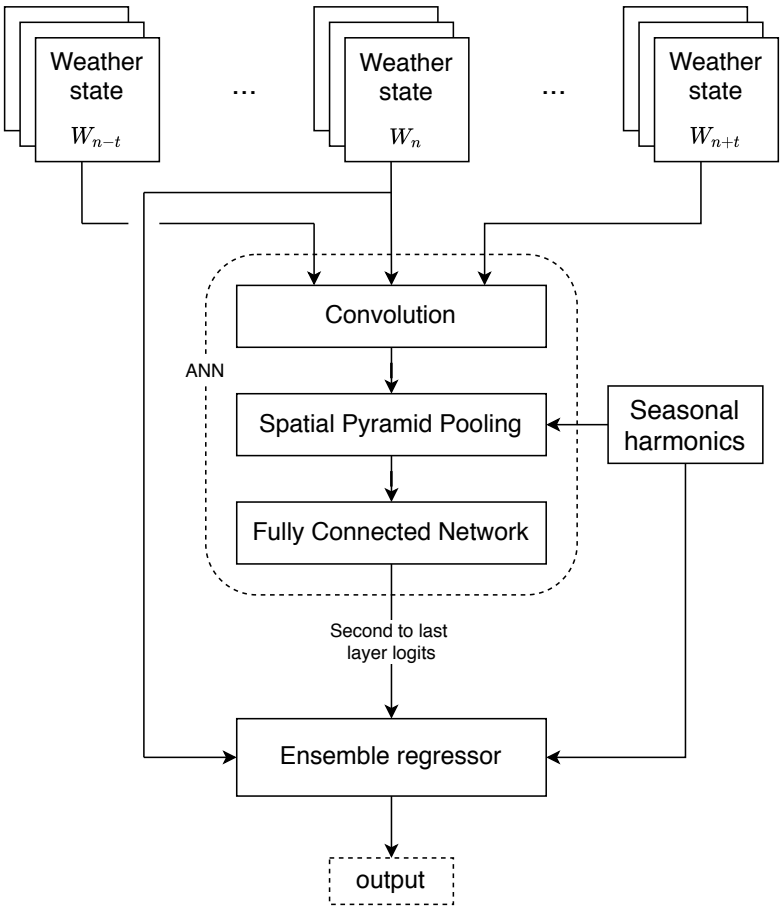


Figure 4.12: Overview of the hybrid model. This model is a combination of the CNN in Figure 4.8 and the ensemble regressor in Figure 4.11. The CNN extracts local spatial and temporal correlations from the input weather state and the activations from the second to last layer is used as additional input to the decision tree-based ensemble regressor.

In order to gain leverage on the spatial correlation in the data, a hybrid model is constructed that combines an ensemble-based decision tree algorithm with an artificial neural network for feature generation. The idea is that the neural network might be able to capture spatio-temporal correlation in the weather data and provide additional features that will improve the accuracy of the ensemble regressor. The raw activations of the second to last layer of the network will be used as generated features. This is, to the best of my knowledge, not been attempted in previous research on this particular problem. However, Ju et al. [2019] had success with this method in a similar wind power estimation problem. The architectural details of the hybrid model (CNN + tree-based learning algorithm) is shown in

Figure 4.12.

The numerical weather parameters as well as seasonal harmonics for a particular timestamp is used as input to the hybrid model. The weather state for a selection of hours before and after the particular timestamp of interest is used as input. The output of the hybrid model is a single continuous value representing the production ratio at that timestamp. This is similar to the Single Node Regression output for the ANN described above.

Chapter 5

Experimental Setting

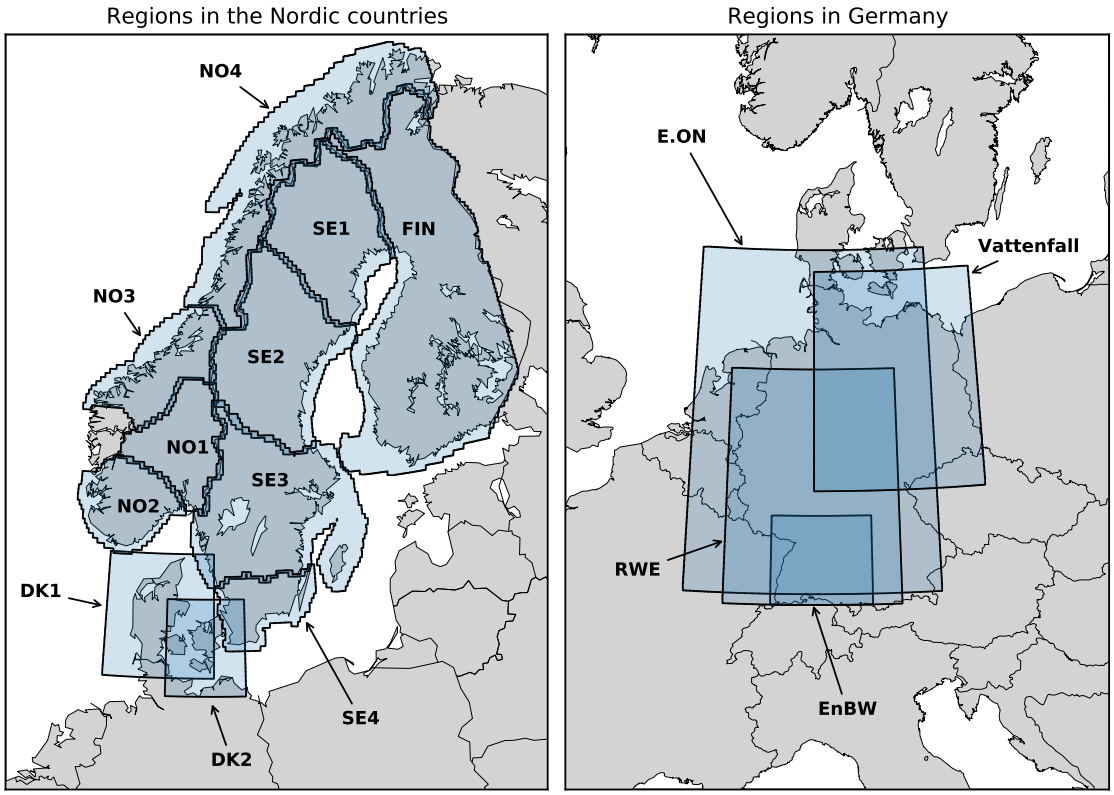
In this chapter I will discuss and describe the datasets that will be used in this study in more detail. Section 5.1 will outline specific information about the dataset related to the different geographical regions. Section 5.2 will discuss and define the metrics that will be used to measure the performance of the models. Finally, in Section 5.3 will outline the experimental plan that will be conducted to address the research questions.

5.1 Dataset

The architectures discussed in Section 4.3 and 4.4 were tested using several different datasets. The datasets had the same structure, but contained data from different power regions and energy markets. Although these regions are geographically separated and non-overlapping, some datasets included weather data for locations outside of the region. A visualization of the location of the different datasets and regions are shown in Figure 5.1.

Specific information of each region as well as the grid dimensions in Germany and the Nordic countries are shown in Table 5.1 and 5.2 respectively. Some of the datasets included invalid data in the target series, denoted as *NaN* in the tables. The invalid data accounted for a insignificant low proportion of the total dataset and were discarded for the relevant region.

The region *NO1* is quite new and the dataset included no production data before 2019. The total number of data points in this dataset was therefore an order of magnitude smaller than what it was in the other datasets. Therefore, this region was discarded in the evaluation the the total number of regions used for evaluation in this study is therefore 20 instead of 21.



(a) The Nordic countries (b) Germany

Figure 5.1: Map of regions in the Nordic countries (5.1a) and in Germany (5.1b). The regions are a little smaller than what is shown. The slightly enlarged regions is a side-effect of the visualization techniques of point data over an area. The regions are drawn based on the coordinates of the NWP data for that region. The dataset for *DK1*, *DK2*, and all regions in Germany include weather data for a rectangular area that includes the actual non-rectangular geographical region within the rectangle borders. See discussion in Section 4.1.1.

5.2 Metrics

It is important to have a good metric that can measure how well one model is performing compared to another. In the context of this thesis, a metric defines a measurement of how close the model predicted production values are to the real production values. Many different metrics can be used to measure the error in forecasting problems. A commonly used metric is the *Mean Absolute Percentage Error*, MAPE, discussed by Armstrong and Collopy [1992]. The interpretation of this metric is intuitive as it describes the mean relative deviation, or error, between the model and the real target values. Given a series of length N consisting of actual

Region	Grid	N	NaN	Start	End	Rect.
E.ON (onshore)	70×63	29 928	0	2016-Jan-01	2019-Jun-01	*True*
E.ON (offshore)	70×63	29 928	0	2016-Jan-01	2019-Jun-01	*True*
RWE	48×43	29 913	7	2016-Jan-01	2019-Jun-01	*True*
EnBW	27×16	29 909	19	2016-Jan-01	2019-Jun-01	*True*
Vattenfall (onshore)	48×40	29 928	0	2016-Jan-01	2019-Jun-01	*True*
Vattenfall (offshore)	48×40	29 928	0	2016-Jan-01	2019-Jun-01	*True*

Table 5.1: Dataset description for regions in Germany. Each grid element is 10×10 km. The dataset for E.ON and Vattenfall includes two target series. One for onshore and one for offshore. **N** denotes the number of valid data points and **NaN** denotes the number of invalid data points available in the time frame. The sum of valid and invalid data points is equal to the number of hours from start to end. **Rect.** is *True* if the NWP data included in the dataset is of rectangular shape, and *False* otherwise. See discussion in Section 4.1.1.

values $\{A_t\}_{t=1}^{N}$ and forecast values $\{F_t\}_{t=1}^{N}$, the MAPE metric of the forecast series compared to the actual series is defined as

$$\begin{aligned} \text{MAPE} &= \frac{1}{N}\sum_{t=0}^{N} \text{APE}_t, \text{ for } t = 1, 2, \ldots, N \\ \text{where} \quad \text{APE}_t &= \left|\frac{A_t - F_t}{A_t}\right|. \end{aligned} \tag{5.1}$$

Although the MAPE metric is commonly used, it has some issues that make it unsuitable as a metric in this thesis. One problem with the standard MAPE metric defined in Equation (5.1) arises when there are data points in the dataset has an actual value, A_t, of zero. This is the case for many of the regions, and will cause a division by zero calculation error. Another problem with the MAPE metric is that it is sensitive to small errors in predictions, F_t, at times with low actual values for production. The calculation of APE_t would blow up when $0 < A_t \ll 1$ and $F_t/A_t \gg 1$. This one APE_t term could potentially dominate the final score of the metric. Makridakis [1993] criticized the MAPE metric for placing a higher penalty on negative errors than positive errors. There are modifications to this metric that has been proposed to deal with these issues.

In an attempt to solve this issues, Kim and Kim [2016] proposed a new and improved metric called the *Mean Arctangent Absolute Percentage Error*, MAAPE. Instead of looking at slope as a ratio as is done in the MAPE metric, MAAPE

Region	Grid	N	NaN	Start	End	Rect.
DK1 (onshore)	41×27	21 144	0	2017-Jan-01	2019-Jun-01	*True*
DK1 (offshore)	41×27	21 090	54	2017-Jan-01	2019-Jun-01	*True*
DK1 (composite)	41×27	35 064	0	2015-Jan-01	2018-Dec-31	*True*
DK2 (onshore)	29×21	21 144	0	2017-Jan-01	2019-Jun-01	*True*
DK2 (offshore)	29×21	21 144	0	2017-Jan-01	2019-Jun-01	*True*
DK2 (composite)	29×21	35 064	0	2015-Jan-01	2018-Dec-31	*True*
NO1	47×30	2881	0	2019-Feb-01	2019-Jun-01	*False*
NO2	45×21	21 144	0	2017-Jan-01	2019-Jun-01	*False*
NO3	65×28	21 137	7	2017-Jan-01	2019-Jun-01	*False*
NO4	156×58	21 137	7	2017-Jan-01	2019-Jun-01	*False*
SE1	70×41	21 137	7	2017-Jan-01	2019-Jun-01	*False*
SE2	71×47	21 144	0	2017-Jan-01	2019-Jun-01	*False*
SE3	73×43	21 144	0	2017-Jan-01	2019-Jun-01	*False*
SE4	39×18	21 144	0	2017-Jan-01	2019-Jun-01	*False*
FIN	85×83	21 082	62	2017-Jan-01	2019-Jun-01	*False*

Table 5.2: Dataset description for regions in the Nordic countries: Denmark, Norway, Sweden, and Finland. Each grid element is 10×10 km. The dataset for DK1 and DK2 includes two target series. One for onshore and one for offshore. **N** denotes the number of valid data points and **NaN** denotes the number of invalid data points available in the time frame. The sum of valid and invalid data points is equal to the number of hours from start to end. **Rect.** is *True* if the NWP data included in the dataset is of rectangular shape, and *False* otherwise. See discussion in Section 4.1.1. *DK1 (composite)* and *DK2 (composite)* are datasets that do not discriminate on onshore and offshore, but contains more historical production data.

consider slope as an angle. MAAPE is defined as

$$\begin{aligned} \text{MAAPE} &= \frac{1}{N}\sum_{t=1}^{N} \text{AAPE}_t, \text{ for } t = 1, 2, \ldots, N, \\ \text{where} \quad \text{AAPE}_t &= \arctan\left(\left|\frac{A_t - F_t}{A_t}\right|\right). \end{aligned} \tag{5.2}$$

The MAAPE metric solves the problem of division by zero with the property of $\lim_{A_t \to 0} \arctan(|A_t - F_t/A_t|) = \pi/2$. The AAPE_t term is bounded in the domain $[0, \pi/2]$. Kim and Kim [2016] state that both MAPE and MAAPE shared some of the same characteristics and can be interpreted in similar ways, and concluded that MAAPE seemed more fair in general. In this thesis, I will mainly use MAAPE

as defined in Equation (5.2) to compare the accuracy of different models.

Occasionally, three other metrics will be used as well as auxiliary metrics. The first auxiliary metric is a modified version of the MAPE metric that I will denote MAPE*. MAPE* is defined by taking the mean absolute error and scale it by the in-sample mean of the series,

$$\text{MAPE*} = \frac{\sum_t |A_t - F_t|}{\sum_t |A_t|} \tag{5.3}$$

The second auxiliary metrics is the *Mean Absolute Error*, MAE. MAE can be defined as

$$\text{MAE} = \frac{1}{N}\sum_{t=0}^{N} |A_t - F_t|\,, \text{ for } t = 1, 2, \ldots, N. \tag{5.4}$$

The MAE metric indicates how far away the estimations are from the target values in absolute terms. The last auxiliary metric that will be used is the *Mean Error*, ME. This metric is defined as

$$\text{ME} = \frac{1}{N}\sum_{t=0}^{N}(A_t - F_t), \text{ for } t = 1, 2, \ldots, N, \tag{5.5}$$

and is simply the deviation from the target.

All four metrics gives valuable insight into the models performance. The MAAPE and MAPE* metric gives a measure of how much the model prediction deviates from the true values in relative terms and the MAE and ME metric gives an indication of whether the model undershoots or overshoots the target value on average. The MAE metric is similar to MAPE*, but is not scaled on the in-sample mean of the production. The scalar difference between MAPE* and MAE will result in them behave in similar ways in certain analysis.

5.3 Experimental plan

The experimental plan is divided into three different parts. Before any experiment was conducted, the data was prepared by restructured as described in Section 4.1.1 and 4.2.

5.3.1 Plan 1 - Create baselines

The first experiment is to create baseline models trained directly on the dataset. The data is striped every third element to reduce complexity. The weather information at time t as well as the 25 harmonic function values at time t is used

as training features. The target values are the production load factor (ratio) at time t. The algorithms that are used as baselines are LightGBM, Random Forest, kNN and AdaBoost. A hyperparameter search is performed to identify the suitable parameters for the algorithms. All four baseline models are trained on both the transformed and non-transformed target load factor (as described in Section 4.2.2). The effect of the load factor transformation will be discussed. The normalization strategies does not apply here, as the models are not sensitive to the different magnitudes of the feature set values.

5.3.2 Plan 2 - Artificial Neural Network

The second experiment implements and evaluate the CNN-based architecture described in Section 4.3. The network is trained on all four normalization strategies outlined in Table 4.1 independently. This is done to identify whether the normalization strategies that exploits the fixed geographical locations of the data-points performs better than more conventional normalization strategies. The results from Experimental Plan 1 will be used to determine whether the transformed ratio or the non-transformed ratio will be used during training of the CNN. After the best normalization strategy is identified, both the Single Node Regression output and the Ordinal Classification output will be tested. Each model configuration is trained 30 independent times on the same region and with the same hyperparameters to be able to extract useful statistics about the performance of a particular configuration compare to other competing configurations.

5.3.3 Plan 3 - Hybrid model

The final experiment will explore the hybrid model described in Section 4.4. The training of the hybrid model will be done in three separate steps. First, the CNN model will be trained on the training dataset. Then, after the training is completed, the CNN model is used to generate features for each hour by running the complete dataset through the network in a forward pass. No training of the network will be done in this step and the parameters for the batch normalization layer and the weights used throughout the network will be fixed at the values found during training. The raw activation values of the second to last layer is taken out as an auxiliary output and appended to the dataset, matching the input hour with the output hour. Lastly, the extended dataset (with the features generated by the CNN as well as the striped weather data of the region and the seasonal harmonic values) are used to train the tree-based ensemble learner. Similar to Experimental Plan 2, both the Single Node Regression output and the Ordinal Classification output configuration will be tested and the model is trained and evaluated 30 independent times to generate useful statistics about its accuracy.

Chapter 6

Results and Discussion

In this chapter I will present and discuss the results from the different experiments outlined in Section 5.3. The structure of the sections in this chapter follows the same structure as the three experimental plans in the previous chapter.

Section 6.1 focus on the first experimental plan and establish the baseline models and compare them to each other. The effect of the proposed load factor transformation described in Section 4.2.2 will be discussed in Section 6.1.2.

Section 6.2 will focus on the second experimental plan, and will analyze and discuss the different aspects of the proposed CNN architecture. Section 6.2.1 and 6.2.2 will give a justification of the data-specific choices. This includes the 3D convolution time window span, the transformation of the load factor target series, as well as a thorough analysis of the four different normalization strategies outlined in Section 4.2.1. In Section 6.2.3, the different CNN model output configurations will be compared. Section 6.2.4 identify the stationary properties of the target series and the time-independence in the model input and exploit this in an ensemble model. Section 6.2.5 analyze the performance of the CNN relative to the target load factor in more detail. Finally, in Section 6.2.6 the different CNN models are compared for each region as well as the two larger aggregated regions, Germany and the Nordic countries.

Section 6.3 focus on the hybrid model that combines features generated by the CNN model with a tree-based learning model for the final prediction as outlined in the third experimental plan. Section 6.3.1 begins with justifying the choice of the decision tree-based learning algorithm that will be used in the hybrid model. Then, Section 6.3.2 will compare the accuracy of the hybrid model to the CNN-only model for each region. Section 6.3.3 discuss the construction of a hybrid ensemble model. Section 6.3.4 take a closer look at the hybrid model accuracy over time

and provide a summary of the findings in of the three experiments conducted in this chapter. Finally, Section 6.3.4 offer a possible explanation to the observation that the offshore regions have much lower accuracy as their corresponding onshore regions.

6.1 Baselines

The datasets used in this thesis is not public available. This means that it is necessary to create my own set of baseline models that the results from the CNN and the hybrid model can be compared against. The models that will be used as baselines are LightGBM, Random Forest, kNN and AdaBoost as mentioned in the experimental plan. These models are reasonable choices to be use as baselines as they are well known and frequently used in a large variety of problems.

All four baseline algorithms was trained on the raw values of the datasets including the seasonal harmonics as described in Section 4.3. The data for each region was tabulated such that each row corresponded to a particular hour and each column corresponded to a single data point at that time for that region. The geographical grid used for the weather information was flattened and striped every third element to make the training time of the model more manageable. As discussed in Section 4.1.1, wind speed, temperature and pressure was used from the NWP. The wind speed from the previous hour was included as well, as the production of a particular hour is calculated based on the average production of the past hour up until the measurement timestamp. This is discussed further in Section 6.2.1. For all baseline models, the first 70% of the data points in the dataset was used for training, the next 15% were set aside for validation purposes, and the last 15% of the data points in the dataset was used for testing. The dataset was intentionally not shuffled before this split, which resulted in the testing set to be completely separated in time from the training set.

6.1.1 Parameter search

A hyperparameter search for a selection of the baseline models was conducted in the project preceding this thesis, Liodden [2019]. The relevant section from the project report was reviewed and reused below.

The hyperparameters of the baseline models were mostly left as the defined default values in their implementation in `sklearn`[1] and `LightGBM`[2]. The hyperparameters for the kNN algorithm and the Random Forest algorithm were explored a bit more

[1] `https://scikit-learn.org`
[2] `https://github.com/microsoft/LightGBM`

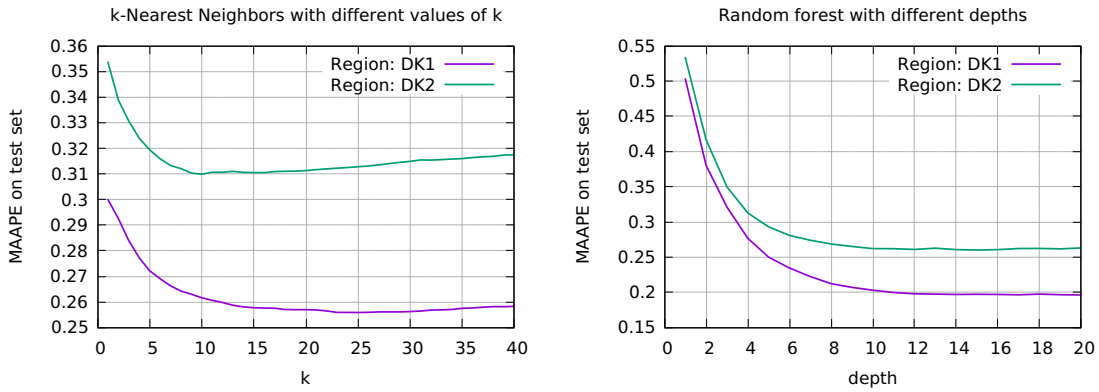


Figure 6.1: **Left**: k-Nearest Neighbors trained on the dataset with different values of the hyperparameter k. **Right**: Random forest regressor trained on the dataset with different max allowed depth of a tree. Number of regressors is 10. DK1 (composite) and DK2 (composite) were used in both experiments.

in depth and fine-tuned. The kNN algorithm was trained multiple times with a different k parameter at each iteration, where k corresponds to the number of nearest neighbors. The Random Forest algorithm was trained multiple times with a different depth of a tree at each iteration. The region DK1 (composite) and DK2 (composite) was used for this search. The total error on the test dataset during this experiment is shown in Figure 6.1.

The hyperparameter search indicates that it is beneficial to consider a rather large k-value for this problem for the kNN algorithm and rather deep trees for the Random Forest algorithm. The error converges at around $k = 15$ and a depth of 10 which indicates the limits of how good these algorithms can get on this dataset. Although only two regions were used in this test, similar conclusions were reached in both regions. I therefore assume that these hyperparameters are reasonable for the other regions as well.

6.1.2 Baseline accuracy and load factor transformation

The target value for the baseline models was the ratio between the actual production and the capacity of the area. This ratio is also known as the *load factor* as discussed in Section 4.2.2. During the discussion in this chapter I will refer to this quantity as both the *ratio* and the *load factor*. Each baseline model was trained on all regions/datasets separately with two independent configurations; one with the real load factor as target value and one with the transformed load factor, as described in Section 4.2.2, as target value. The final prediction was calculated by multiplying the predicted load factor with the capacity of the region at the time

Metric	LightGBM		Random Forest		kNN		AdaBoost	
	diff	*p-value*	*diff*	*p-value*	*diff*	*p-value*	*diff*	*p-value*
MAAPE	**-0.003**	0.002	-0.001	0.164	**-0.001**	0.047	**-0.033**	0.000
MAPE*	0.001	0.203	0.002	0.194	**0.002**	0.030	**0.016**	0.011
MAE	1.097	0.102	1.095	0.087	**2.403**	0.032	**13.106**	0.026
ME	**6.407**	0.047	**11.304**	0.042	**13.443**	0.025	72.669	0.060

Table 6.1: A paired sampled t-test was performed to measure the effect of the load factor transformation using the four different baselines. Negative difference means the transformation improved the accuracy of the model. Significant values are highlighted with bold ($p < 0.05$). Although the differences are small, three out of four baselines models performed significantly better on the main metric, MAAPE, when the transformation function was used on the target load factor. For other auxiliary metrics, the baseline models that was trained on the non-transformed load factor tended to perform better.

of interest. If the transformed load factor was used during training, the inverse transformation function was applied before the final prediction was made. All the baseline model algorithms are deterministic given the training data, so the training was not repeated multiple times. Each baseline model was evaluated using the four different metrics described in Section 5.2. The distribution of the evaluation metric value on the test set over all regions are visualized in Figure 6.2. The *True/False* separation indicates weather the load factor was transformed (*True*) or not (*False*).

A paired sampled Student t-test can be performed on the results obtained on the 20 different regions listed in Table 5.1 and 5.2 to identify whether the transformation strategy had any significant effect on the accuracy of the model predictions. The results was paired based on the region such that each region had one metric evaluation measurement when using the transformed load factor and one metric evaluation measurement when using the non-transformed load factor. Such a pair existed for each of the four metrics. Table 6.1 shows the average difference in the metric evaluation score with the corresponding p-value for all baseline models.

As can be seen from the results of the paired sample t-test in Table 6.1, using the transformed load factor as target is likely to improve the accuracy given the main metric, MAAPE. However, using the non-transformed load factor tends to do better with the auxiliary metrics. The conclusion that can be drawn from this is that although the load factor transformation strategy performed better than the non-transformed version given the main metric, the results should be interpreted with caution giving the contradicting conclusion from the auxiliary metrics.

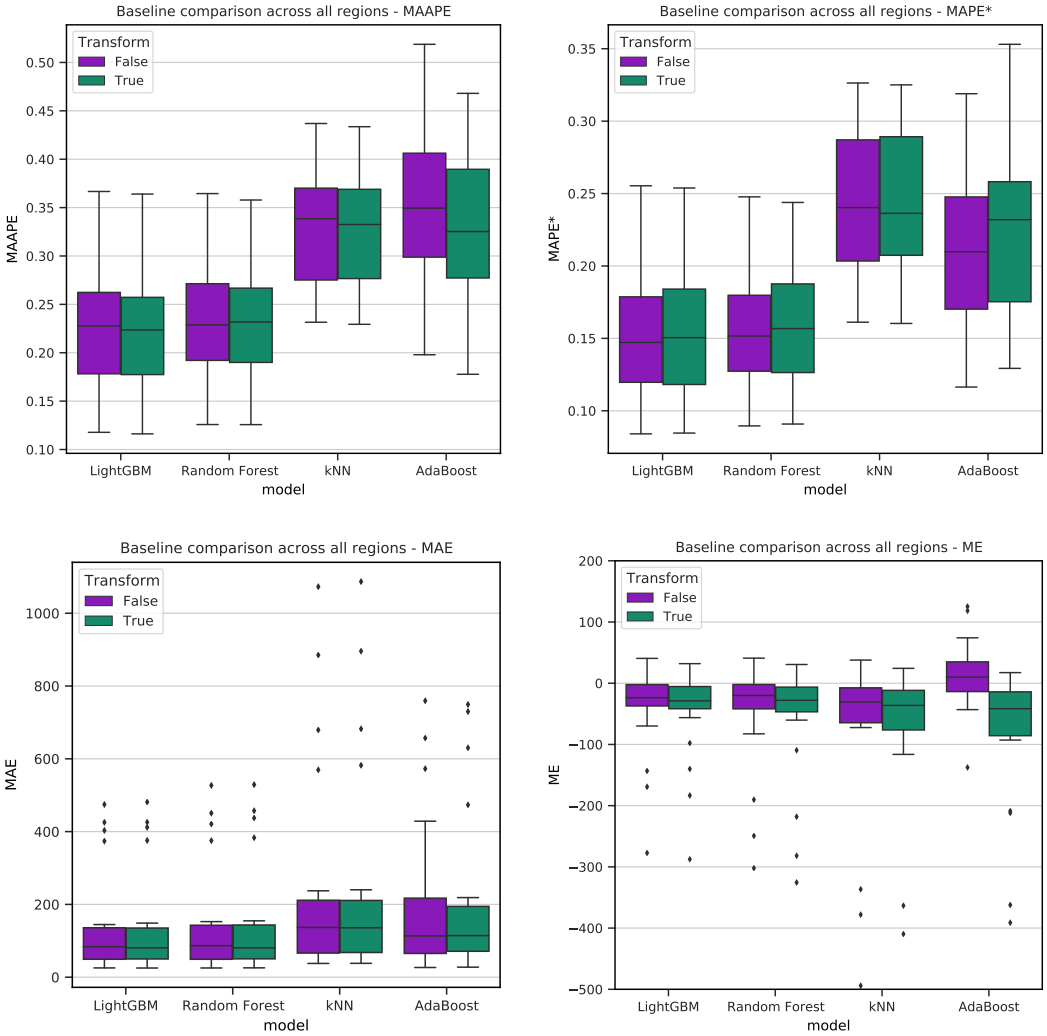


Figure 6.2: Comparison of the distribution of the evaluation score of the different baseline models across all regions using the four metrics. Purple bars represents the results obtained when the real, non-transformed, load factor (ratio) was used as target series, and green bars represents the results obtained when the transformed load factor was used as target series. The distribution of values for each box are the evaluation error of the test set of every region. Each quadrant in the figure corresponds to the evaluation of each of the four metrics. **Upper Left**: *Mean Arctangent Absolute Percentage Error*, MAAPE. **Upper Right**: Modified *Mean Absolute Percentage Error*, MAPE*. **Lower Left**: *Mean Absolute Error*, MAE. **Lower Right**: *Mean Error*, ME.

6.2 Artificial Neural Network

The proposed artificial neural network architecture described in Section 4.3 were trained, tested and evaluated on each region independently. The CNN model architecture in Figure 4.8 was implemented using `PyTorch`[3]. The Spatial Pyramid Pooling layer between the convolution layers and the fully connected layers in the network ensured that the same model architecture could be used on regions of different size, as discussed in Section 4.3. In order to generate useful statistics of the accuracy of the architecture, each relevant configuration of hyperparameters were trained and tested repeatedly 30 independent times. This ensured that the only difference between each run was the random seed that was used for weight initialization and shuffling of the training minibatches. The minibatch size was held fixed at 64 for all regions and the number of epochs was 50. The other hyperparameters was as described in Section 4.3.

The dataset was split in the same way as for the baseline models. the first 70% of the data points in the dataset was used for training, the next 15% were set aside for validation purposes, and the last 15% of the data points in the dataset was used for testing. The dataset was ordered in ascending order in the temporal dimension before the split. As the different datasets had different amount of temporal data, the absolute size of the train, validation and test sets (as well as the first and last timestamps) varied slightly from region to region. However, the test set for most regions contained data from the first five months of 2019. This was with the exception of the DK1 (composite) and DK2 (composite) region which did not have any data for 2019. The test set for these two regions contained data from the last half of 2018.

6.2.1 Load factor transformation and temporal window

There were a large number of possible model configurations of interest to evaluate and test for each region. It was infeasible to test all different configuration options to a satisfying degree, so some of the configuration options had to be decided in advance.

Target transformation of the load factor

A comparison of the transformed an non-transformed target value strategy was done with the baseline models in Section 6.1.2. Tree out of four baselines had achieved statistically significant better accuracy when the model was trained using the transformed target load factor values rather than the non-transformed target

[3] `https://pytorch.org/`

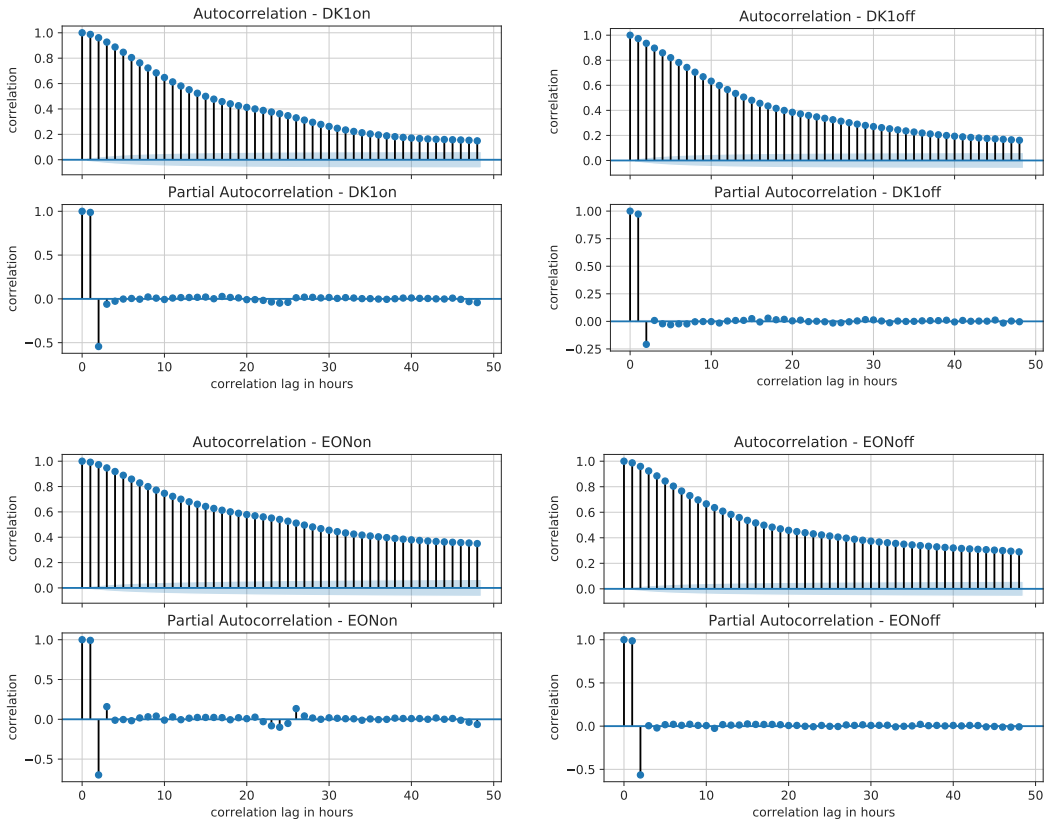


Figure 6.3: Auto-correlation and partial auto-correlation in the production volume with hourly resolution. The lag window is set to 48 h. The auto-correlation pattern in the DK1 (on/offshore) and E.ON (on/offshore) regions are representative for every region and are presented in the figure. The other regions show similar patterns and can be found in Appendix.

load factor as shown in Table 6.1. Therefore, I chose to use the transformed target load factor on all regions when evaluating the CNN model architecture.

3D convolution and time window

The CNN model described in Section 4.3 accept a range of weather states in the temporal dimension as input. This range input to the 3D convolution layers is motivated by the hypothesis that weather states close to each other in the temporal dimension are correlated to each other and that the corresponding production volumes are correlated. The auto-correlation and partial auto-correlation of the production volumes are calculated for all regions with a 48 h lag window. The results are similar for all regions, and the auto-correlation and partial auto-correlation for a representative selection of regions are shown in Figure 6.3. Similar figures for

the other regions are included in Appendix.

The partial auto-correlation is the auto-correlation at lag k after removing the effect of any prior lags. The single most interesting observation to emerge from the the partial auto-correlation comparison across the regions, was that the production values at time t_0 was strongly correlated with the production values at time t_{-1}. For many regions, the partial auto-correlation calculations also showed a significant negative correlation between the hour of interest, t_0, and the production value two hours before, t_{-2}. The correlation is symmetric at the time of interest which means that the same observation holds for t_{+1} and t_{+2}. Based on this observation, the CNN model was trained using weather data from two hours before and til two hours after the time of interest as input to the 3D-convolution block of the network. In other words, the input to the model was the five weather states $W_{\{t_{-2},t_{-1},t_0,t_1,t_2\}}$ around the time of interest $t = t_0$.

6.2.2 Normalization strategy

The four different normalization strategies outlined in Table 4.1 in Section 4.2.1 were tested and evaluated. The CNN was trained 30 independent times with each of the different normalization strategies for every region. To compare the normalization strategies, the first task of interest was to determine whether the fixed geographical location of the data could be exploited by normalizing on the local statistics at a given location instead of the global statistics of the weather parameters.

An independent Student's t-test is conducted for each region comparing the global and local mean/std normalization strategies (NS1 and NS2) and the global and local min-max normalization strategies (NS3 and NS4). Given that each region is trained 30 times with each normalization strategy, the number of degrees of freedom is $df = (N_1-1)+(N_2-1) = 58$. In order to reject the null hypothesis (that there is no difference between the local and the global normalization strategies) with a confidence of 95% ($p < 0.05$), the required t-value with 58 degrees of freedom is ± 2. The resulting t-values from the independent Student t-test for the MAAPE, MAE, and ME metrics across all regions can be seen in Figure 6.4. The MAPE* metric and MAE metric only differs by a scalar constant, and would have identical results in this test. The MAE metric is therefore not included in the figure.

Based on the results presented in Figure 6.4, it is possible to conclude that the local normalization strategies do improve the model accuracy for some regions, while the global normalization strategy improve the model accuracy for other regions. For most regions, however, the choice of whether to use local or global normalization strategy did not show any significant difference in the model accuracy and the

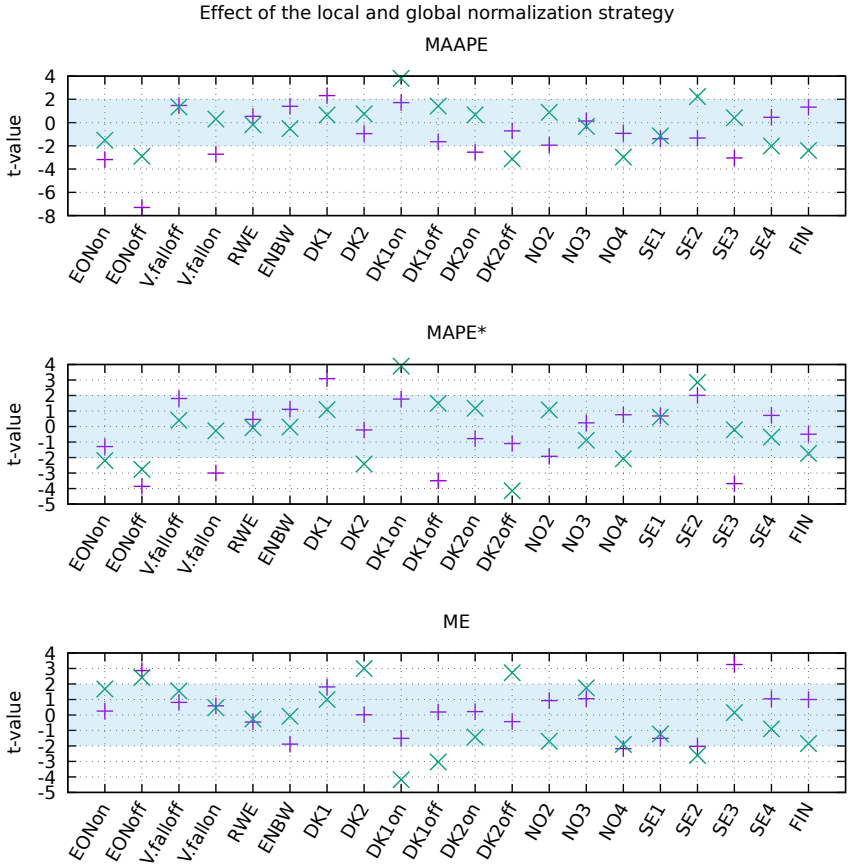


Figure 6.4: The effect of the local vs global normalization strategies. Each point in the scatter plot represents the t-value associated with a comparison between the global and the corresponding local normalization strategy for one region. For the local normalization strategy to be significantly better than the global normalization strategy ($p < 0.05$), the t-value must be greater than +2. On the other hand, for the global normalization strategy to be significantly better than the local normalization strategy ($p < 0.05$), the t-value must be less than -2. $t = \pm 2$ is indicated with the shaded region. The MAE metric would give identical results as the MAPE* metric in this test as they only differ by a scalar constant. The MAE metric is therefore excluded in this figure. The two different types of normalization is made distinct with purple for the mean/std normalization and green for the min-max normalization.

null hypothesis cannot be rejected in general. To my knowledge, there is not any particular feature of the dataset that separates the regions that benefited from a local or global normalization strategy from the regions that did not.

If the impact of the choice normalization strategy was only dependent on whether

the normalization was global or local, the outcome of the independent Student t-test would be independent on the choice of mean/std or min-max normalization strategy type. The t-values for a given region in Figure 6.4 would have been close to each other because the performance difference would be highly dependent on the choice of local or global strategy, not on the mean/std or min-max strategy. It is apparent from Figure 6.4 that this is not the case as the separation between the green and purple points of the figure for each region are far apart. The impact of the normalization strategy is therefore not only based on whether the strategy is local or global based, but also whether it is based on normalizing with a mean/std strategy or min-max strategy. A similar figure can be created to show the comparison between the mean/std and min-max normalization strategy. This comparison is showed in Figure 6.5.

Similar to the conclusion that there were significant differences in the model performance for some regions based on the choice of a global or local normalization strategy, Figure 6.5 shows that the same conclusion holds for the choice of mean/std or min-max normalization strategy. Although neither the mean/std nor min-max strategy was superior in all regions, more regions showed a significant difference in the accuracy based on the choice of mean/std or min-max strategy method compared to the choice of local or global normalization strategy. Based on the average MAAPE evaluation, 12 regions benefited from the choice of mean/std or min-max strategy compared to only 6 regions that benefited from the choice of a local or a global based strategy. This result indicates that it is difficult to gain leverage on the fixed geographical structure from a normalization point of view.

6.2.3 Network output configuration

In order to assess the effect of the Single Node Regression output and the Ordinal Classification output, repeated tests of each output type were conducted on each region. The normalization strategy that performed best on a given region was used for that region. Similar to the testing of the different normalization strategies, 30 independent training sessions with each output format was used. The distribution of the evaluation on the test set based on the MAAPE metric for each region is shown in Figure 6.6. Similar figures for the other three auxiliary metrics are included in Appendix.

To compare the model accuracy based on the choice of a Single Node Regression output or Ordinal Classification output, an independent Student t-test is conducted to determine if the average measured MAAPE accuracy for the two different output modes for a given region are drawn from the same distribution. Given a threshold of $p = 0.05$ for significance, the test concluded that the regions that performed significantly better with the Single Node Regression output was RWE

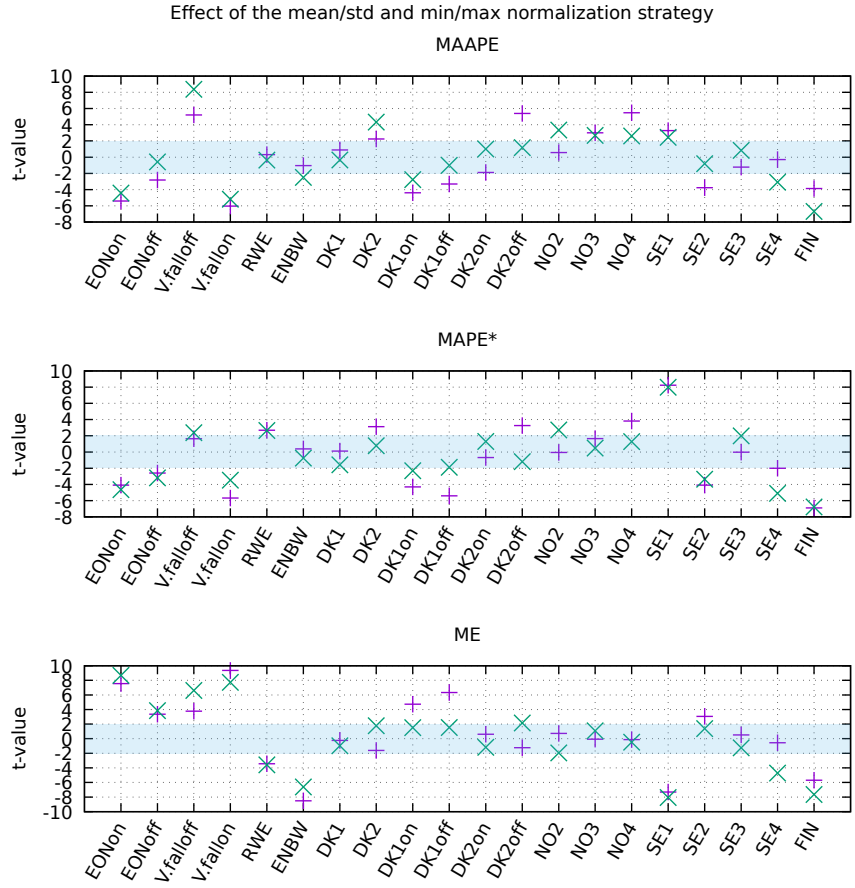


Figure 6.5: The effect of the mean/std vs min-max normalization strategies. Each point in the scatter plot represents the t-value associated with a comparison between the mean/std and the corresponding min-max normalization strategy for one region. For the mean/std normalization strategy to be significantly better than the min-max normalization strategy ($p < 0.05$), the t-value must be less than -2. On the other hand, for the min-max normalization strategy to be significantly better than the mean/std normalization strategy ($p < 0.05$), the t-value must be greater than +2. $t = \pm 2$ is indicated with the shaded region. The MAE metric would give identical results as the MAPE* metric in this case as they only differ by a scalar constant. The MAE metric is therefore excluded in this figure. The local and global variants of the normalization strategies is made distinct with purple for global normalization and green for local normalization.

and SE3. The regions that had insignificant differences between the Single Node Regression output and the Ordinal Classification output were E.ON (onshore), E.ON (offshore), Vattenfall (onshore), DK1 (onshore), NO2, and NO4. For the remaining 12 regions, the Ordinal Classification output was on average significantly

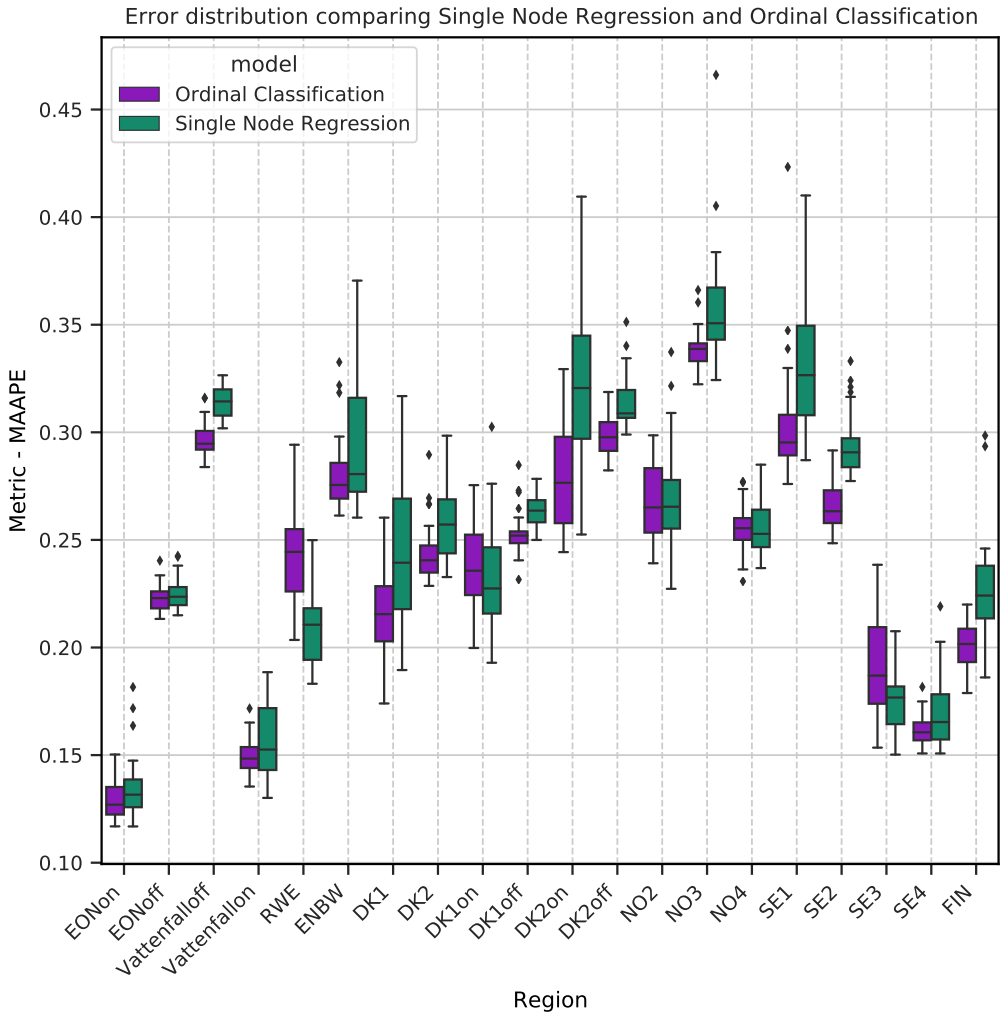


Figure 6.6: Error distribution using the MAAPE metric on the test set comparing Single Node Regression and Ordinal Classification output configuration of the CNN model. The sample size for each configuration is 30 independent runs.

better than Single Node Regression output consistently with $p < 0.05$. It has been suggested by Lathuilière et al. [2019] that a Single Node Regression output on a CNN-based regression network would give as good results as any complex ad-hoc networks. This does not appear to be the case on this problem. These results are in agreement with Cheng et al. [2008] findings that a CNN-based regression network that predicts ordinal data generalize better when using Ordinal Classification. A detailed overview of the findings for each region is shown in Table 6.5 later in this chapter.

6.2.4 Stationarity and ensemble model

Although the final prediction of the model is the absolute production value at time $t = t_0$, the network itself is trained on the load factor of the region at time $t = t_0$. The benefit of this is that even though the capacity of the relevant region changes over time, usually monotonically increasing, the range of target values does not. As previously mentioned, the input to the model are the five weather states $W_{\{t_{-2},t_{-1},t_0,t_1,t_2\}}$ around $t = t_0$, as well as the 25 values given by the seasonal harmonic functions at time t_0. Neither the absolute time t itself nor the capacity of the region at that time, C_t, is included as parts of the input to the model. This property makes the model independent on the absolute time corresponding to the input.

Given the large geographical size of the regions, the wind power load factor time series is assumed to be stationary. This assumption are based on the hypothesis that the power capacity in a region is increased by building new wind turbines in areas that are either already known to be well suited for wind power production or locations that are randomly selected within the region borders. Although this is a reasonable assumption, this might not always be the case. For instance, if new wind farms are built in a completely different part of the region, the model might not be able to account for that. Manero et al. [2018] suggested that smaller regions or regions with few wind park locations are more prone to non-stationary behavior in the wind power load factor time series.

Based on the assumption that the wind power time series is stationary, and the fact that the models do not have any absolute time information in the input, I expect that the models that do well on the validation set also will do well on the testing set. This is also suggested by Díaz et al. [2015]. As mentioned at the beginning of Section 6.2, both the validation set and the testing set are completely separated in time from the training set. The testing and validation set consists of the last 30% of the time series in the temporal dimension. The first half of the last 30% of the dataset are set aside for validation and the last half is set aside for testing. The validation set is used during training to indicate overfitting behavior, but is not used to adjust the set of weights in the network during back propagation. A new simple ensemble model is constructed by combining the five best models based on the evaluation score on the validation set. Each model in the ensemble makes a prediction based on the input and the ensemble takes the equally weighted average prediction as the final prediction of the model. This model will be referred to as the *CNN-based validation CNN top 5 ensemble* model, or *CNN top 5 ensemble* model for short. A one sample Student's t-test concluded that the CNN top 5 ensemble model performed significantly better than the 30 independent CNN models for all regions with $p < 0.01$ independent on the choice

of the Single Node Regression output or Ordinal Classification output.

In order to get more insight in how well the different models perform compared to each other, the one month running mean of the *Arctangent Absolute Percentage Error*, AAPE, partial metric was calculated for each region on both the validation and testing set. The models that was compared was the each of the 30 independent training runs, the CNN top 5 ensemble model, and the two best performing baseline models. I have chosen to present the finding for regions that illustrate the good and bad performance of the CNN top 5 ensemble model most consistently. Figure 6.7 and 6.8 shows this running mean for the region SE4 and SE2 respectively. The data points to the left of the vertical bar in the figures represents the running mean over the validation set, and the data points to the right of the vertical bar represents the running mean of testing set. The light shaded region on the right side of the bar represents data points that are in the first month of the testing data set where the one month running mean are calculated based on some of the values from the last part of the validation set. Because of this, the running mean of the testing set should not be considered until one month into the test set and therefore one month after the vertical bar in the figure. The LightGBM and Random Forest baseline models are included as reference. The other baseline models had low accuracy and is not included in these figures. The running mean of all CNN models trained with Ordinal Classification on the relevant region is shown in light gray. The CNN model that performed best on the validation set is highlighted as *validation rank 1*. The same model is also highlighted in testing set. Note that this is not necessarily the model that performed best on the test set.

I have chosen to present the finding for region SE4 in Figure 6.7 as it can clearly be seen that the top 5 ensemble model based on the validation set metric scores outperforms the baseline models on the test set consistently. The top 5 ensemble model also perform better than most single CNN models. On the other hand, the findings for region SE2 in Figure 6.8 shows the opposite. Although a bit difficult to see in the figure, the baseline models did outperform the CNN top 5 ensemble model on both the validation set and the test set. It is also apparent from the figure that the CNN model that did perform best on the validation set, did not necessarily perform best on the testing set. For the other regions, the different performance between the models was not as consistent. The one month running mean for the other regions can be found in Appendix.

As can be seen in both figure 6.7 and 6.8, the 1 month running mean of the AAPE partial metric varies a lot. The figures include the running mean of the target load factor, and it is possible to see that there is a symmetric relationship to some extent between the target load factor and the MAAPE metric value. This

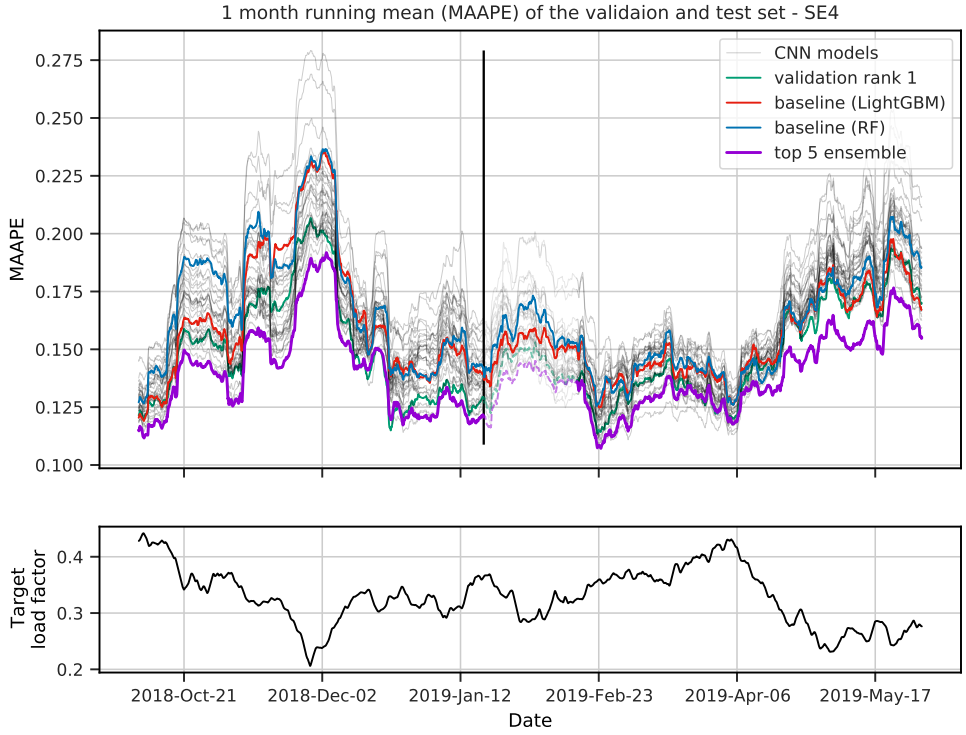


Figure 6.7: **Upper**: The 1 month moving average error for region SE4 using the MAAPE metric. The region to the left of the black vertical bar are part of the validation set and the region to the right are part of the testing set. **Lower**: The 1 month running mean of the target load factor. In this figure it is clear to see that the top 5 ensemble model based on the validation set also performed consistently well on the testing set. It is also possible to see a symmetric behavior between load factor and the MAAPE metric evaluation.

observation motivates a closer look at the model accuracy relative to the target load factor.

6.2.5 Accuracy relative to load factor

The results of the CNN-based top 5 ensemble model on the two regions SE4 and SE2 is further investigated. Figure 6.9 and Figure 6.10 show the distribution of the intermediate metric value, AAPE, for different ranges of the target load factor for region SE4 and SE2 respectively. For the other regions, similar figures are found in Appendix. A paired sample t-test is conducted to identify if the differences

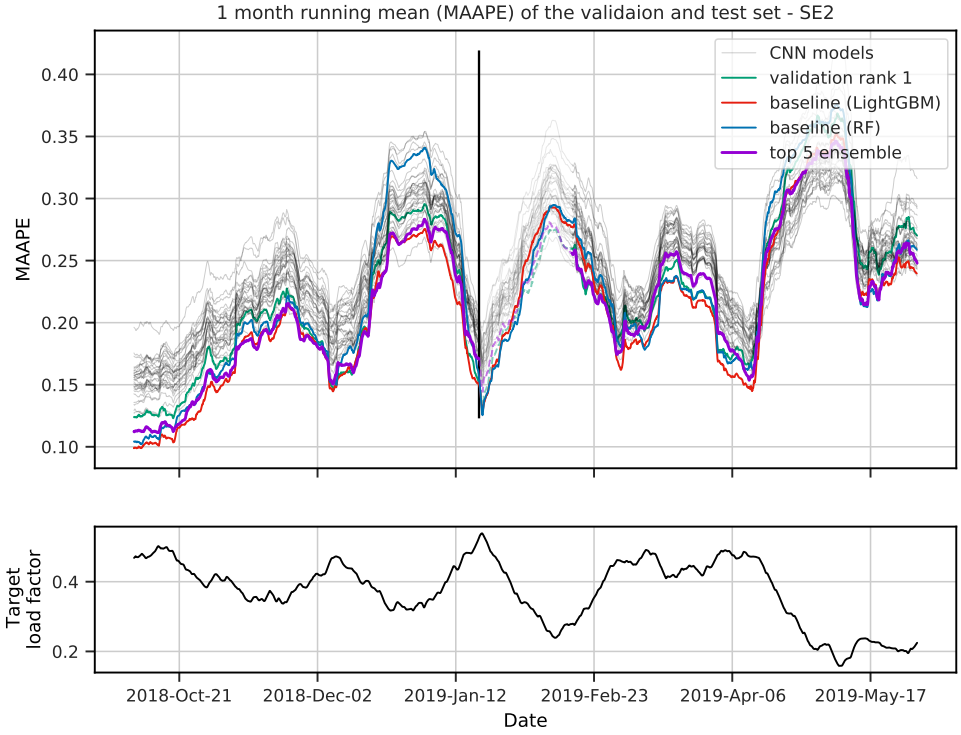


Figure 6.8: **Upper**: The 1 month moving average for region SE2 using the MAAPE metric. The region to the left of the black vertical bar are part of the validation set and the region to the right are part of the testing set. **Lower**: The 1 month running mean of the target load factor. Although a bit difficult to see in this figure, the baseline models did outperform the CNN top 5 ensemble model on both the validation set and the test set in this region. Similar to Figure 6.7, it is possible to see a symmetric behavior between load factor and the MAAPE metric evaluation here as well.

between the models within the load factor interval are significant. The samples are paired on the timestamp of the target production and model estimation of the region. For the SE4 region, the analysis conclude that the CNN-based model performed better than the LightGBM baseline for all load factors ($p < 0.05$) except when the target load factor was in range 60-80%. In that range the difference was insignificant. This was true for both the Single Node Regression output and the Ordinal Classification output.

On the other hand, for the SE2 region, the CNN-based models did not perform as well compared to the baseline. When Ordinal Classification output was used, the

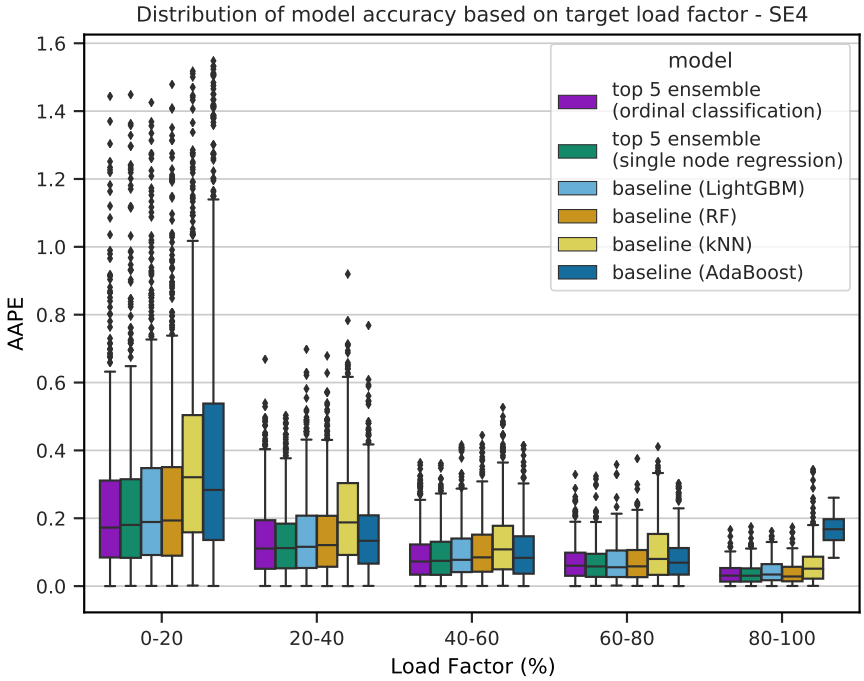


Figure 6.9: Distribution of model accuracy in terms of *Arctangent Absolute Percentage Error*, AAPE, based on the target load factor for region SE4. The CNN top 5 ensemble model outperform the LightGBM baseline model 4 out of 5 segments. When the load factor is 60-80%, the difference in AAPE is insignificant. This is true for both the Single Node Regression output and the Ordinal Classification output.

LightGBM baseline model outperform the CNN-based model significantly ($p < 0.05$) when the target load factor was in the range 40-80%. The CNN-based model performed better than the LightGBM baseline when the target load factor was in the range 0-20%. For the remaining target ranges the difference was insignificant. If the CNN-based model used the Single Node Regression output, the results were different. In this case the CNN-based model was outperformed by the LightGBM baseline on 4 out of 5 segments. The difference was insignificant when the target load factor was in range 20-40%.

To be able to see if there is a general pattern across all regions, the same statistical test was done for the other regions as well. By comparing the *Arctangent Absolute Percentage Error* for each point in time, a paired sample t-test was conducted to identify whether one model had a smaller error on average than another throughout the test set relative to the target load factor. The production and estimations

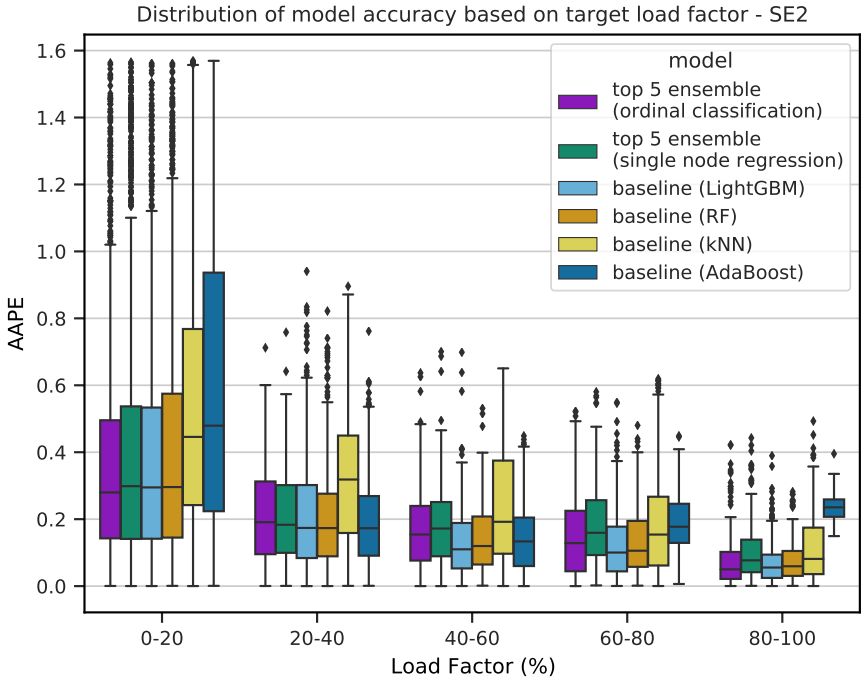


Figure 6.10: Distribution of model accuracy in terms of *Arctangent Absolute Percentage Error*, AAPE, based on the target load factor for region SE2. *Ordinal Classification*: The LightGBM baseline model outperform the CNN-based model significantly ($p < 0.05$) in the range 40-80%. The CNN-based model performs better than the LightGBM baseline in the range 0-20%. For the remaining target ranges the difference is insignificant. *Single Node Regression*: The CNN model are outperformed on 4 out of 5 segments except when the load factor range 20-40% where the difference is insignificant.

for each region were categorized according to the target load factor. The target load factor was divided into bins of 20% width and a paired sample t-test was conducted on each bin for every region. Table 6.2 shows an overview of the results where each model gets a "point" if it is significantly better than the other model for the relevant target load factor range. The findings in Table 6.2c suggest that the CNN-based model with Ordinal Classification output perform better than the CNN-based model with Single Node Regression output in the extreme cases (where the load factor is very low or very high). On the other hand, the Single Node Regression output mode seems to perform better when the target load factor are in the middle. The results in Table 6.2a also suggest that the CNN-based validation top 5 ensemble model with the Ordinal Classification output performs better than

Load Factor	Ord. classification. is better	LightGBM is better	No significant difference
0 – 20%	13	4	3
20 – 40%	6	5	9
40 – 60%	6	6	8
60 – 80%	9	5	6
80 – 100%	7	8	5

(a) CNN with Ordinal Classification output compared to the best performing baseline, LightGBM.

Load Factor	Single node reg. is better	LightGBM is better	No significant difference
0 – 20%	6	6	8
20 – 40%	8	5	7
40 – 60%	9	7	4
60 – 80%	9	6	5
80 – 100%	7	9	4

(b) CNN with Single Node Regression output compared to the best performing baseline, LightGBM.

Load Factor	Ord. classification is better	Single node reg. is better	No significant difference
0 – 20%	14	2	4
20 – 40%	2	10	8
40 – 60%	5	8	7
60 – 80%	8	8	4
80 – 100%	11	6	3

(c) CNN with Ordinal Classification output compared to Single Node Regression output

Table 6.2: The different output methods of the CNN top 5 ensemble model is compared against each other and against the best performing baseline model, LightGBM. The *Arctangent Absolute Percentage Error*, AAPE, is calculated for each point in time over the test set. The production is categorized according to the target load factor and the AAPE corresponding to each hour is categorized in the same way. This is done for each region. A paired sample t-test is conducted among the hours corresponding to each load factor category for each region and the results is displayed in the tables above. A "point" is given to the column that represents a significant result of the test. The numbers in each row corresponds to the number of regions, 20. Threshold of significance is $p = 0.05$.

the LightGBM baseline model, in particular when the target load factor is low.

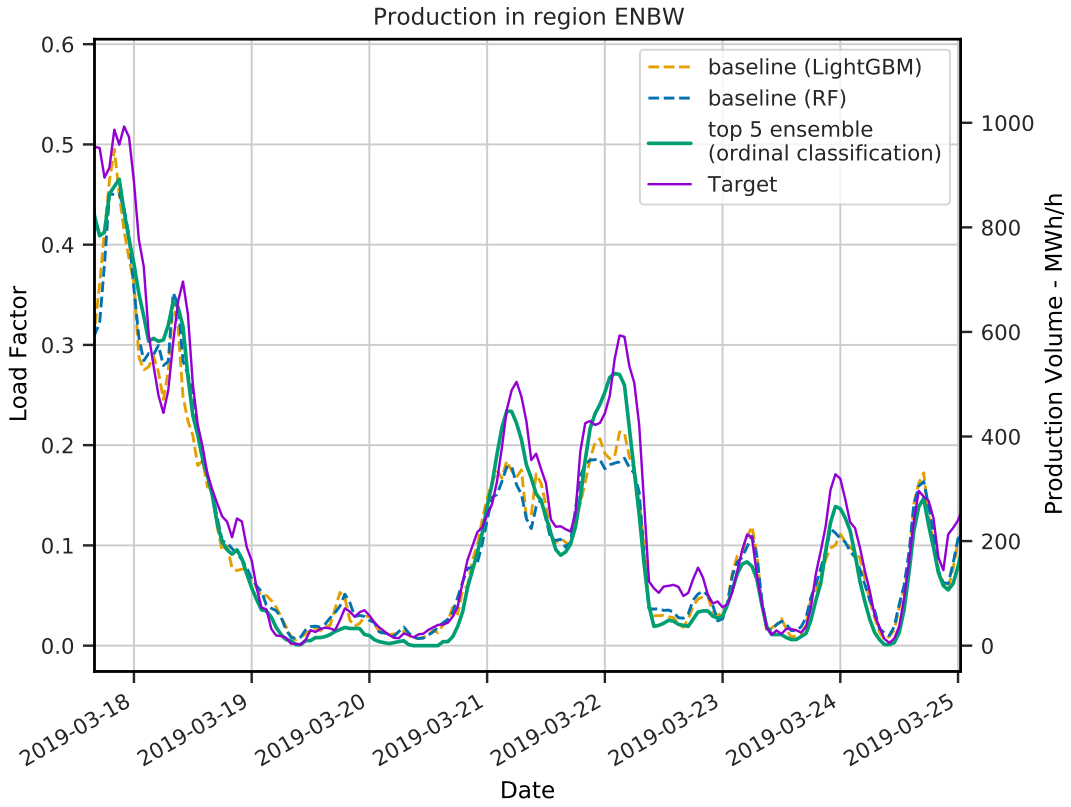


Figure 6.11: A section of the test dataset for the EnBW region where the CNN-based validation top 5 ensemble model (with Ordinal Classification) performs better than the best performing baselines for a *low* target load factor.

Figure 6.11 and 6.12 display the target production and the model estimations of the CNN-based validation top 5 ensemble model with the Ordinal Classification output together with the two best performing baseline models, LightGBM and Random Forest. Figure 6.11 shows a good example from the EnBW region where the CNN-based model are more accurate than the baseline model when the load factor is low. Figure 6.12 shows an example from the NO4 region with the same property. Figure 6.13 shows a good example from the FIN region where the CNN-based model perform better than the baseline models when the target load factor is high. Although the CNN-based model does not perform better than the best performing baselines when the target load factor is high for the majority of regions, Table 6.2a, Figure 6.13 is a good example of a region where it does.

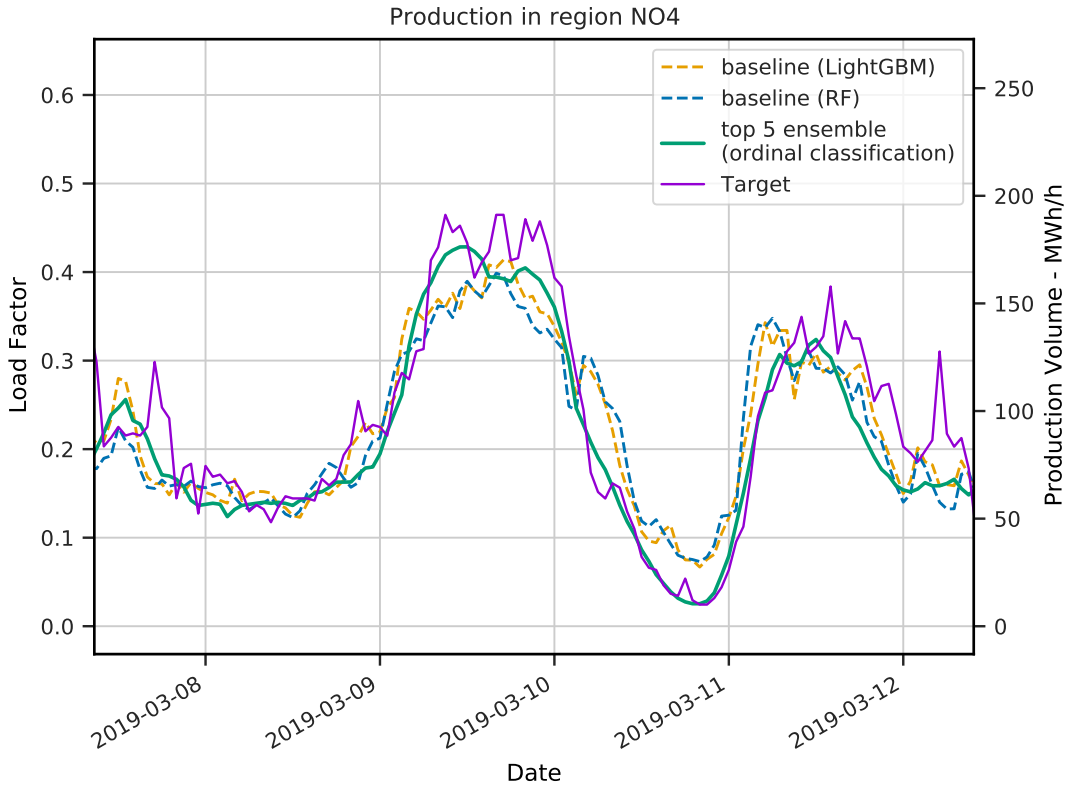


Figure 6.12: A section of the test dataset for the NO4 region where the CNN-based validation top 5 ensemble model (with Ordinal Classification) performs better than the best performing baselines for a *low* target load factor.

6.2.6 Model accuracy comparison on the test set

By comparing the AAPE for each point in time, a paired sample t-test can be conducted to identify whether one model have a smaller error on average than another throughout the whole test set. The threshold for significance was set to $p < 0.05$ throughout this test. Comparing the CNN-based validation top 5 ensemble model to the two best performing baseline model, concludes that the CNN-based model with the Single Node Regression output perform significantly better than the LightGBM baseline model in 10 regions and significantly better than the Random Forest model in 15 regions. In 5 regions the difference in accuracy between the CNN-based model and LightGBM was insignificant and in the remaining 5 regions the LightGBM model performed significantly better than the CNN-based model. Similarly, in two regions the difference in accuracy between the CNN and the Random Forest was insignificant, while in the reminding three regions the Random

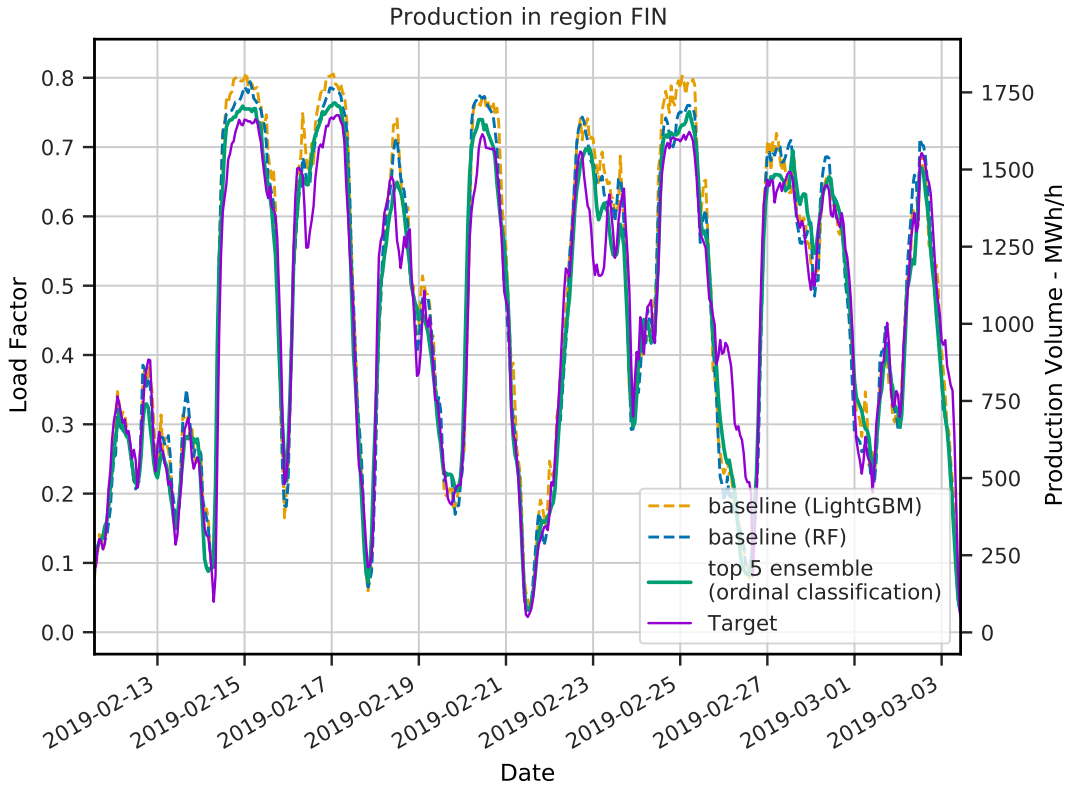


Figure 6.13: A section of the test dataset for the FIN region where the CNN-based validation top 5 ensemble model (with Ordinal Classification) performs better than the best performing baselines for a *high* target load factor.

Forest significantly outperformed the CNN-based model.

Training the CNN model with the Ordinal Classification output yield a slightly improved results over the Single Node Regression output. The CNN-based model outperformed the LightGBM model in 13 regions with this configuration. In 5 regions the LightGBM model outperformed the CNN-based model while in two regions the differences were insignificant. Compared to the Random Forest baseline, the CNN-based model perform better with the MAAPE metric in 17 out of 20 regions. The Random Forest performed significantly better in only one region, while in the remaining two regions the differences were insignificant. The region where the CNN-based model was outperformed consistently by both LightGBM and the Random Forest was DK2 (onshore). I am unable to identify any particular characteristics of the DK2 (onshore) region that explaines this deviation.

The CNN-based validation top 5 ensemble model outperformed both the kNN

baseline algorithm and the AdaBoost baseline algorithm significantly ($p < 0.05$) for all regions regardless of the output configuration of the CNN model.

Germany and the Nordic countries aggregated production

The 20 different regions can be clustered into two larger regions; one representing Germany and one representing the Nordic countries. The predicted production volume for these two larger regions was found by predicting the production volume for each smaller region individually and then aggregate the results. The final prediction was compared to the total production in the larger region, and the MAAPE, MAPE*, MAE, ME metric was applied to the estimated series. The metric evaluation on the test set for Germany are found in Table 6.3, and the metric evaluation on the test set for the Nordic countries are found in Table 6.4.

The average score for all individual CNN-based models was calculated by selecting a random model out of the 30 different models that was trained for each region and use those as a representation for the total estimator in the larger region. Because the model performance varies, this measurement was repeated 10000 times to form a distribution of accuracy. The mean of the distribution as well as the error, given by two standard deviations, is shown as numeric values in Table 6.3 for Germany and 6.3 for the Nordic countries.

	Ord. classification		**Single node reg.**		**LightGBM**	**RF**	**kNN**	**AdaBoost**
	avg.	*val. top 5*	*avg.*	*val. top 5*	*(baseline)*	*(baseline)*	*(baseline)*	*(baseline)*
MAAPE	0.105(19)	0.086	0.094(20)	0.080	0.087	0.095	0.165	0.151
MAPE*	0.081(16)	0.067	0.078(24)	0.062	0.071	0.080	0.139	0.123
MAE	1351(272)	1111.181	1289(397)	1027.839	1182.674	1330.677	2313.332	2038.554
ME	−1022(455)	-731.332	−823(800)	-412.587	-749.723	-974.575	-1501.942	-1230.548

Table 6.3: Aggregated results for all regions in Germany

	Ord. classification		**Single node reg.**		**LightGBM**	**RF**	**kNN**	**AdaBoost**
	avg.	*val. top 5*	*avg.*	*val. top 5*	*(baseline)*	*(baseline)*	*(baseline)*	*(baseline)*
MAAPE	0.104(17)	0.091	0.103(19)	0.093	0.070	0.074	0.115	0.120
MAPE*	0.095(16)	0.083	0.098(20)	0.087	0.064	0.067	0.107	0.120
MAE	524(86)	460.363	544(110)	479.399	353.073	369.317	590.887	661.130
ME	−458(120)	-392.867	−483(147)	-420.379	-161.688	-178.574	-326.093	-441.649

Table 6.4: Aggregated results for all regions in the Nordics

Most regions in Germany, with the exception of RWE, performed better with the CNN-based validation top 5 ensemble model trained the Ordinal Classification output compared to the baselines, Table 6.5. It is therefore not surprising that the metric scores for the CNN-based validation top 5 ensemble model on the larger

aggregated region (Germany) outperform the baselines. However, the good performance on the smaller individual regions in the Nordic countries are canceled out when the regions are aggregated into one larger region. This is a surprising result, as the majority of the region in the Nordic countries did perform significantly better with the CNN-based model (7 out of 12). For the Nordic countries the LightGBM and Random Forest baseline models performed better than the CNN-based models.

The poor total performance of the CNN-based models in the larger Nordic region was a surprising result, as the majority of the region in that area did perform significantly better with the CNN-based model compared to the baselines. A closer inspection of the regions in the Nordic countries shows that the 7 regions (out of 12) that performed significantly better with the CNN-based model with the Ordinal Classification output only accounted for approximately 40% of the total capacity in the large aggregated region. Likewise, the 5 regions (out of 12) that performed significantly better with the CNN-based model with the Single Node Regression output only accounted for approximately 22% of the total capacity. As can be seen in Table 6.2a, it can be seen that the CNN-based model with Ordinal Classification is better than the LightGBM baseline when the target load factor is low. As the regions are located in different geographic locations, the weather situation will vary a lot from region to region. It is possible, therefore, that the target load factor varies from region to region at the same point in time. It can thus be suggested that the low performance of the CNN-based model on the aggregated Nordic region is due to a combination of these factors. Even though the model accuracy is improved in some regions, if the contribution of that region turns out to be insignificant or low at that particular time, the improvement in individual regions are small and is not propagated well in general.

6.3 Hybrid model

As mentioned in Section 4.4, the hybrid model is a tree-based regression model that is trained with features generated by a CNN in addition to the striped weather data at time t and the corresponding seasonal harmonic values for that timestamp.

6.3.1 Choice of tree-based algorithm

The results from the testing of the different baseline models in Section 6.1.2 indicated that both the LightGBM algorithm and the Random Forest algorithm with depth 10 perform well across the different regions. Both algorithms are reasonable candidates to be used as the tree-based algorithm in the hybrid model. A thorough comparison of the performance differences between those two three-based models

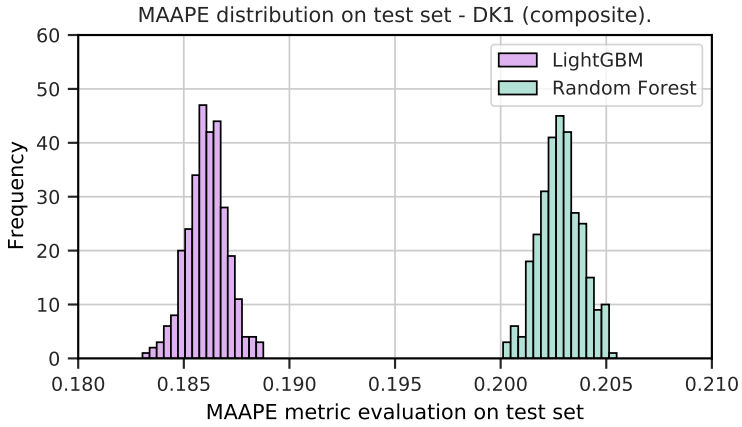


Figure 6.14: Evaluation error, MAAPE, of the model evaluated on the test set (2018) after training. The LightGBM and Random Forest model was trained and tested 300 independent times with different random shuffle of the training dataset and with a k-fold cross validation. The result of the models are fitted to a normal distribution with mean $\mu_{\text{LightGBM}} = 0.186$ and a standard deviation $\sigma_{\text{LightGBM}} = 0.001$, and $\mu_{\text{RF}} = 0.203$ and a standard deviation $\sigma_{\text{RF}} = 0.001$.

were conducted in the project preceding this thesis, Liodden [2019]. Liodden [2019] compared the algorithms on the DK1 (composite) dataset by running 300 independent training and testing sessions of each model on the dataset. The models was trained with a k-fold cross validation with $k = 3$, and the selection of the validation sets were changed at random for each of the 300 sessions. The data from 2018 was set aside for testing. The distribution of errors measured on the testing dataset is shown in Figure 6.14. The error of the models resembles a normal distribution with mean $\mu_{\text{LightGBM}} = 0.186$ and a standard deviation $\sigma_{\text{LightGBM}} = 0.001$ for LightGBM, and $\mu_{\text{RF}} = 0.203$ and a standard deviation $\sigma_{\text{RF}} = 0.001$ for the Random Forest. The results clearly show that the LightGBM algorithm performs better than Random Forest for this region given the MAAPE metric.

The good performance of LightGBM algorithm on the DK1 (composite) dataset compared to the Random Forest algorithm, makes this algorithm a good candidate to be used as the tree-based learner in the hybrid model. The average MAAPE error of the baseline models used in this thesis across the regions is MAAPE(LightGBM) = 0.225 for LightGBM and MAAPE(RF) = 0.233 for the Random Forest algorithm. A paired sample t-test concludes that the difference between the LightGBM and Random Forest model accuracy in general across the regions is significant, $t(38) = -4.76$, $p < 0.01$. The LightGBM algorithm is therefore chosen to be used as the tree-based regression algorithm in the hybrid model.

6.3.2 Accuracy compared to the CNN-only model

The hybrid model was trained in the three different steps outlined in the third experimental plan, Section 5.3.3. First, the CNN was trained on the training dataset. Then, the complete dataset was propagated through the CNN network in a forward pass using the weight and parameters that was found during the previous step. Finally, the raw activations from the 50 nodes in the second to last layer of the CNN was taken out as engineered features for the given timestamp. The dataset was tabulated including the striped NWP, the 25 seasonal harmonics, and the 50 additional generated features for each timestamp in each region. A LightGBM model was trained on this new dataset. The hybrid model was evaluated using both the Single Node Regression output and the Ordinal Classification output configurations of the CNN independently. Similarly to the CNN-only model, the training and testing sessions was conducted 30 independent times with the same hyperparemeters to be able to generate useful statistics of the performance of the model.

A paired sample t-test was used to determine if the hybrid model consisting of the CNN model with indirect predicted features combined with a LightGBM regression model for the final load factor prediction performed significantly better than the CNN model with direct load factor predictions. When the CNN model was trained using the Ordinal Classification output configuration, 12 out of 20 regions showed significant improvement in accuracy in favor for the hybrid model compared to the CNN-only model that was used to generate the features for the hybrid model ($p < 0.05$). 4 out of 20 showed a significant setback in accuracy in the hybrid model compared to the CNN-only model, and the remaining 4 regions did not show any significant difference between the CNN-only model and the hybrid model that was trained using the features generated by the CNN. When the CNN model was trained using a Single Node Regression output configuration, the introduction of a hybrid model showed even greater improvement. With this configuration, 16 out of 20 regions improved significantly ($p < 0.05$) from the CNN-only model to the hybrid model. The remaining 4 regions did not show any significant difference between the two.

Thus, comparing the hybrid model extension to the CNN-only model showed strong evidence that the model accuracy improved when expanding the model from a CNN-only direct prediction model to a hybrid model with a CNN for feature engineering and a LightGBM regression model for the final prediction.

6.3.3 Hybrid ensemble model

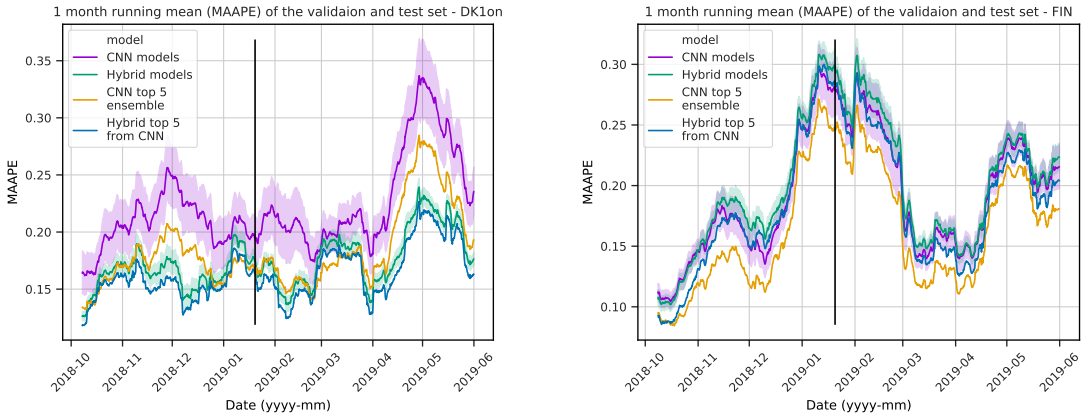
Although these numbers indicate a promising result, they paint an inaccurate picture. The distribution of the CNN-only model accuracy given the MAAPE metric had a large standard deviation and varied a lot around its mean as can be seen in Figure 6.6. Therefore, a more interesting comparison would be to let the five best CNN-models based on the validation set, the same models as in the CNN top 5 ensemble, generate features and use these features to create five hybrid models that is combined to an ensemble model similar to the CNN top 5 ensemble. This ensemble will be referred to as the *Hybrid top 5 ensemble* in this thesis. The motivation was based on the assumption that the CNN models with highest accuracy also generated the most distinct features. The Hybrid top 5 ensemble was compared to the CNN top 5 ensemble as well as the best performing baseline model with the MAAPE metric. The Ordinal Classification output configuration was used for the CNN as it generalized better than the Single Node Regression based on the results obtained in experiment in Section 6.2.3.

Comparing the three models on the test set for each region, the Hybrid top 5 ensemble model performed better, with lower MAAPE value, than the best performing baseline model (LightGBM) on 17 out of 20 regions. It performed worse than the LightGBM baseline model on the remaining 3 regions. However, the Hybrid top 5 ensemble model only performed better than the CNN top 5 ensemble model on 7 out of 20 regions. The CNN top 5 ensemble model performed better than the hybrid model on the remaining 13 regions. Based on the experiments conducted here, it can therefore be concluded that the introduction of the CNN generated features does indeed improve the LightGBM model accuracy for most regions, but in general the hybrid model does not improve the predictions done by the CNN-based validation top 5 ensemble model.

Compared on the main metric, MAAPE, the Hybrid top 5 ensemble model outperformed the Random Forest, kNN, and the AdaBoost baseline algorithm on the test set in all regions.

6.3.4 Model accuracy over time and final results

In order to get a visualization of the model accuracy over time, the running mean of the AAPE partial metric was calculated for the different models. I have chosen to focus on the regions that displayed the most consistent preference to a particular architecture throughout the test set. In the region DK1 (onshore), the hybrid model perform consistently better than the CNN-only model. The running mean of the AAPE partial metric for this region is shown in Figure 6.15a. The one month running mean of the different models are a bit difficult to distinguish from



(a) Region: DK1 (onshore)

(b) Region: FIN

Figure 6.15: MAAPE 1 month moving average for regions DK1 (onshore) (a) and FIN (b). The colored shaded band around CNN models and Hybrid models represent the standard deviation of the MAAPE 1 month moving average for the models at that time. The two different top 5 ensemble models are included as well. The baseline models are not included in the figure as it is not important for this comparison. The DK1 (onshore) region (a) illustrates a region where the hybrid models perform better than the CNN-based models. The FIN region (b) illustrates a region where the CNN-based top 5 ensemble model performed consistently better than the hybrid models across the testing dataset. The solid black vertical line separates the validation dataset (to the left) and the test dataset (to the right). High resolution of these figures can be found in Appendix.

each other in the figure, but note that the running mean of the AAPE partial metric is consistently better on for the hybrid top 5 ensemble model than all the CNN-only models. On the other hand, the region FIN is an example of a region where the CNN-based top 5 ensemble model performed consistently better than the hybrid models. The running mean of the AAPE partial metric for this region is shown in Figure 6.15b. Again, the different models are difficult to distinguish from each other in the figure, but note that the running mean AAPE partial metric score is consistently better for the CNN-based top 5 ensemble model compared to the hybrid models throughout the test set. Similar figures for the other regions is included in the Appendix.

Figure 6.15a and 6.15b show a selection of two regions and visualize the comparison of the running mean of four different models:

1. *CNN-only models*: The running mean of the 30 different CNN-only models together with the standard deviation at each point in time visualized as a shaded are of the same color around the mean of the distribution.

2. *Hybrid models*: Same as for the CNN-only models. The running mean of the 30 different hybrid models with corresponding standard deviation as a shaded color around the mean.

3. *CNN top 5 ensemble*: A single model created by combining the five best CNN models based on the MAAPE accuracy of the validation set into an ensemble. The final prediction of this model is the average prediction of the five models included in the ensemble.

4. *Hybrid top 5 ensemble*: The model is constructed by creating five hybrid models based on the features generated by the five best performing CNN models based on the MAAPE accuracy of the validation set. The five hybrid models are combined to an ensemble. Similar to the CNN top 5 ensemble, the final prediction of this model is the average prediction of the five models included in the ensemble.

As illustrated by the running mean error in Figure 6.15 it is important to bear in mind that the error change rapidly over time for each model that was tested and that it is not always clear which model perform best at any given time. Table 6.5 gives an overview of the final results for each region comparing the CNN-based model and Hybrid model with both the Ordinal Classification output and Single Node Regression output mode. The average performance of the 30 different models that was trained for each region with identical hyperparameters as well as the uncertainty in terms of two standard deviations are included in the table. The performance of the corresponding validation top 5 ensemble variants as well as the best performing baseline (LightGBM) is included in the table. The other baselines are not included as they performed significantly worse on most regions. The main metric (MAAPE) is used to compare the models.

Germany and the Nordic countries aggregated production

As has been done earlier in this chapter, the 20 different regions can be clustered into to larger regions representing Germany and the Nordic countries. The total production in these two larger regions can be calculated by aggregating the estimated production at each individual smaller regions at a given time. Given the main metric, MAAPE, the hybrid model performed better than both the best performing baseline model (LightGBM) as well as the CNN-based model in both larger areas. This result is shown in Table 6.5. When considering the average production volume of the regions that had better accuracy with the hybrid top 5 ensemble model compared to the CNN top 5 ensemble model, the results for Germany are reasonable. The three out of six regions in Germany that had better accuracy with the hybrid top 5 ensemble model accounts for 78% of the total pro-

Region	Ord. classification		Single node reg.		Hybrid Ord. classification		Hybrid Single node reg.		LightGBM
	average	*val. top 5*	*average*	*val. top 5*	*average*	*val. top 5*	*average*	*val. top 5*	*(baseline)*
E.ON (onshore)	0.129(18)	0.108	0.135(29)	0.115	0.118(4)	0.109	0.119(4)	0.109	0.116
E.ON (offshore)	0.224(13)	0.202	0.225(15)	0.207	0.218(6)	0.204	0.214(6)	0.204	0.208
Vattenfall (onshore)	0.150(17)	0.129	0.156(35)	0.125	0.134(5)	0.126	0.133(3)	0.124	0.133
Vattenfall (offshore)	0.297(16)	0.285	0.314(13)	0.299	0.309(7)	0.304	0.313(7)	0.306	0.306
RWE	0.242(40)	0.210	0.209(32)	0.189	0.199(6)	0.191	0.201(5)	0.196	0.196
EnBW	0.281(35)	0.247	0.298(67)	0.257	0.280(6)	0.267	0.277(6)	0.268	0.274
DK1 (composite)	0.217(39)	0.173	0.245(72)	0.191	0.169(6)	0.159	0.166(5)	0.155	0.161
DK2 (composite)	0.244(28)	0.217	0.256(32)	0.226	0.237(6)	0.226	0.236(6)	0.229	0.234
DK1 (onshore)	0.238(37)	0.197	0.234(49)	0.196	0.184(7)	0.172	0.184(7)	0.175	0.183
DK1 (offshore)	0.253(20)	0.239	0.264(15)	0.248	0.256(6)	0.250	0.252(6)	0.248	0.250
DK2 (onshore)	0.280(48)	0.232	0.320(74)	0.239	0.219(7)	0.209	0.213(6)	0.203	0.213
DK2 (offshore)	0.298(19)	0.278	0.313(24)	0.291	0.316(9)	0.308	0.316(7)	0.305	0.318
NO2	0.268(34)	0.225	0.269(47)	0.224	0.252(10)	0.233	0.252(11)	0.240	0.243
NO3	0.339(18)	0.326	0.358(54)	0.334	0.361(10)	0.357	0.365(8)	0.363	0.364
NO4	0.255(22)	0.228	0.255(25)	0.234	0.254(10)	0.242	0.252(11)	0.241	0.253
SE1	0.305(55)	0.274	0.334(69)	0.285	0.289(10)	0.280	0.284(9)	0.277	0.272
SE2	0.265(19)	0.250	0.295(30)	0.275	0.260(14)	0.247	0.259(12)	0.249	0.247
SE3	0.191(44)	0.159	0.174(28)	0.150	0.157(6)	0.150	0.157(6)	0.150	0.154
SE4	0.162(15)	0.143	0.170(34)	0.144	0.160(5)	0.150	0.160(5)	0.150	0.158
FIN	0.201(22)	0.174	0.226(49)	0.183	0.208(9)	0.193	0.204(12)	0.193	0.187
Germany	0.105(19)	0.086	0.094(20)	0.080	0.084(2)	0.080	0.083(2)	0.079	0.087
Nordics	0.104(17)	0.091	0.103(19)	0.093	0.068(2)	0.064	0.069(2)	0.066	0.070

Table 6.5: Overview of final results for each region evaluated and compared on the main metric, MAAPE. The average score for each region is measured over a set of 30 independently trained models with the same hyperparameters. The uncertainty is calculated as two standard deviations from the mean.

duction in Germany. On the other hand, the regions that had better accuracy with the CNN top 5 ensemble model accounts for only 22% of the total production.

For the Nordics, the explanation for the better accuracy with the hybrid model compared to the CNN-only model is not as clear. The experiments conducted in this thesis show that the CNN-based model perform better on a higher number of regions compared to the corresponding hybrid and baseline model. As discussed in Section 6.2.6, the contribution of the different regions to the total aggregated production volume must be analyzed to understand the result. As mentioned earlier in this chapter, the 7 out of 12 regions in the Nordics that was more accurately estimated by the CNN-based top 5 ensemble model compared to the best performing baseline model only accounted for approximately 40% of the total production capacity in the Nordic region. Although none of these 7 regions was improved with the hybrid model, the hybrid model did improve 4 out of the 5 regions that accounted the remaining 60% of the total capacity in the Nordics (SE2, SE2, DK1 (onshore), and DK2 (onshore) with the exception of SE1). This offer an explanation to why the hybrid model performed better with higher accuracy than the CNN-based model in the large regions, in particular the Nordics, even thought more regions was more accurately predicted using the CNN-based model alone.

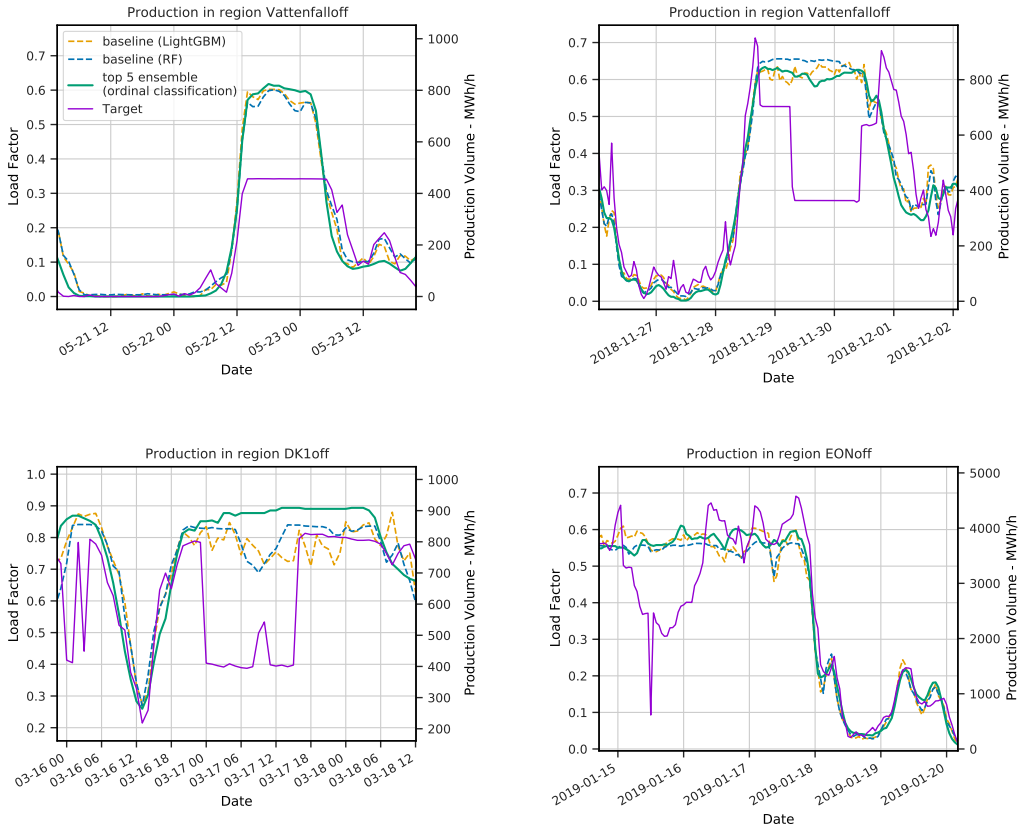


Figure 6.16: The figure shows a segment of the actual production volume and the estimated production volume in three different offshore regions. All the models predicts a high production volume based on the NWP and seasonal harmonics, but the actual production is low. This is not an uncommon phenomena in the offshore regions. All sub-figures share the same legend.

Offshore production drop

Closer inspection of Table 6.5 shows that the model accuracy in the offshore regions are a lot worse than the corresponding onshore regions. An explanation for this observation might be that the offshore regions contains large production farms which are occasionally shut down for maintenance. These large farms accounts for a relatively large proportion of the total production in the region. This information is unavailable to the model and are not possible to extract given the information that are used to make predictions, weather state and seasonal harmonics. In such situations the models will wrongly predict a high production volume given the weather data even though the actual production is low at that time. In such a case, the error will be large.

Figure 6.16 show this phenomena for three different offshore regions, E.ON (offshore), Vattenfall (offshore), and DK1 (offshore). In Figure 6.16 the relative large production drops can be seen as steep slopes that are quickly changed.

Chapter 7

Concluding Remarks

In this thesis, I have designed and evaluated a deep learning-based model architecture to solve the wind power production volume prediction problem of a large geographical region. This chapter provides a short summary of the work that has been done in this thesis, results of the experiments that has been conducted, a summary of the contribution to the field of wind power production volume prediction, as well as a few suggestions for future work. The summary in this chapter relates to the research questions stated at the beginning of the thesis in Section 1.2. The summary of the findings related to the three research questions are addressed in Section 7.1.1, 7.1.2, and 7.1.3 respectively.

7.1 Summary

This project was undertaken with a goal of designing and evaluating a model to predict the wind power production volume in a large geographical region at a given time. The designed model was based on deep learning methods, in particular Convolutional Neural Networks. The prediction of the wind power production volume was based on the weather state in the region at the relevant time. The underlying assumption for such a model was the strong correlation between the weather in a region at time t and the production of electrical power from wind turbines in the region at the same time. This assumption is reasonable given the physical properties of the system. The power output of a wind turbine is determined by many factors, with the wind speed as the most important. In total, 20 different non-overlapping regions representing different parts of Norway, Sweden, Finland, Denmark, and Germany were used for the analysis in this thesis.

The distribution of wind farms and wind turbines in the regions that were analyzed

and tested were unknown. Only the weather state at a time t, W_t, the time t itself, and the capacity of the region at that time, C_t, was known and available for the model as input. The weather state W_t consisted of Numeric Weather Prediction data, NWP, including wind velocity, temperature, and atmospheric pressure for the whole geographical region. The time t was encoded using 25 seasonal harmonic functions with periods ranging from 6 h to 6 months. The capacity of a region at time t represents the total power production that can be produced at that time given optimal weather conditions. This number changes over time, and is usually monotonically increasing because of the rising popularity of wind power as a power source and the increased construction of new wind turbines.

The representation of the weather state in a region is geographically fixed in a grid-like structure of cells relative to the location in the real world. The size of the cells varies, but they all have a size around 10×10 km. The weather is a continuous system, and therefore the weather in one cell will influence the weather in a neighboring cell. The grid structure of the input data, combined with the spatial correlation between the data points, motivated the use of a CNN-based architecture as a basis for the deep learning-based model. The power production of a wind turbine is directly dependent on the instant weather situation and does not carry any temporal dependencies in itself. However, the relevant target value is not the instant production, but the average production volume per hour. This quantity carries a temporal dependency which motivated the use of a 3D convolution segment in the proposed model. There is barely any published research on this particular problem, and previous research on the topic, Liu et al. [2019], has suggested that further research should be conducted to address the spatial and temporal correlation between different wind farms in a larger region.

7.1.1 The first aim of the study

The first aim of the study was to determine if the proposed deep CNN-based architecture[1] was able to capture the spatio-temporal dependencies of the data, and able to generate descriptive features for a given weather situation. The network's ability to capture the spatio-temporal dependencies was measured by letting the network estimate the production volume in a region for a set of timestamps that was completely separated from the timestamps that was used during training. The *Mean Arctangent Absolute Percentage Error*, MAAPE, metric was chosen as the main metric used in this thesis. The metric value of the test dataset was used to measure the accuracy of the model and was therefore used as a method to measure the network's ability to generate good descriptive features for a given weather situation.

[1] Also referred to as the *CNN* in this chapter

The 20 different regions were of different geographical sizes and therefore contained a different number of NWP data points, or cells, for each hour. In order to have a single architecture that could be utilized on all the different regions, the proposed CNN-based architecture was constructed to handle inputs of different sizes using Spatial Pyramid Pooling. Spatial Pyramid Pooling is a down-sampling method proposed by He et al. [2015], which ensures that a fixed number of output nodes is generated independent of the size of the input. The down-sampling is located between the last convolution layer, and before the first fully connected layer, in the network.

The target value of the network was the load factor of the region at the time of interest. The load factor is a measurement of the ratio between the actual production and the capacity of the region at a given time. Predicting the load factor instead of the absolute production volume was preferable for two reasons. First, the target values are bound between 0 and 1 independent of the region. Second, predicting the load factor instead of the actual production volume increase the value of older data as the predictions are independent of the capacity of the region at the time. Normalizing and targeting the load factor is beneficial under the assumption that new wind turbines are randomly distributed or constructed in areas with conditions that is already known to be suitable for wind turbines.

The target load factor was transformed by a simple quadratic polynomial equation to make the distribution of load factor more even throughout the dataset. This was done to prevent overfitting on an uneven dataset. The transformation created a more even dataset at the cost of slightly lower resolution when the load factor is high. Two different strategies were used as the final output layer of the CNN: Single Node Regression and Ordinal Classification. Single Node Regression is a single node output with a sigmoid activation function to keep the target value in range $(0, 1)$. Ordinal Classification, or Ordinal Regression, is a method proposed by Cheng et al. [2008] constructed to solve regression problems of ordinal data using CNN. Ordinal Classification combines regression and classification methods. The target space is divided into discrete non-overlapping segments or classes. The classes are ordered, and all classes with lower ordinal value than the target class are activated for prediction.

The NWP data was normalized and then used as input to the CNN. Given the fixed geographical location of the input elements, a local normalization strategy was tested alongside the more common global normalization strategy. The motivation was that the local variations could potentially be better represented when normalizing based on the parameter distribution at that location. In total, four different normalization strategies was tested, Table 4.1. Each region was trained and evaluated 30 different times with each normalization strategy. Although some

regions showed significant improvement with one strategy compared to the other, no significant difference was found between the local and global normalization strategies in general across the 20 regions.

The two output configurations, Ordinal Classification and Single Node Regression, were tested in a similar way. The CNN was trained and evaluated 30 independent times with both output configurations while keeping all hyperparameters constant across the 30 different sessions. Given a threshold of $p = 0.05$ for significance, a Student t-test (with 58 degrees of freedom) concluded that a CNN with the Single Node Regression output performed significantly better than the Ordinal Classification on 2 out of 20 regions. A CNN with the Ordinal Classification output performed significantly better than the Single Node Regression output on 12 out of 20 regions. For the remaining 6 regions, the difference was insignificant. This result is in agreement with the findings of Cheng et al. [2008], that a CNN-based regression network that predicts ordinal data generalize better when using Ordinal Classification. It has been suggested by Lathuilière et al. [2019] that a Single Node Regression output on a CNN-based regression network would give as good results as any complex ad-hoc networks. This does not appear to be the case in this study.

The wind power load factor time series is assumed to be stationary. As the CNN model did not have any absolute time information as input, it is expected that the defining set of model weights and parameters that performed well on the validation set also performs well on the test set. This has also been suggested by Díaz et al. [2015]. A simple ensemble model was constructed by combining the five CNN models with the best sets of weights found during training. The final prediction of this ensemble model was the average prediction of the five individual models. A one sample t-test concluded that the top 5 ensemble method performed significantly better on average than the 30 independent CNNs for all regions with $p < 0.01$ independent on output configuration.

7.1.2 The second aim of the study

The second aim of the study was to determine if a tree-based machine learning algorithm combined with features generated by a deep neural network would be able to capture the spatio-temporal dependencies in this problem to a higher accuracy than the two models alone. The architecture that was developed for this purpose has been referred to as the *hybrid model* in the previous chapters. The training of the hybrid model was threefold. First, the proposed CNN-based model architecture discussed earlier was trained on the training dataset. Then, the complete dataset was propagated through the CNN in a forward pass using the weights and parameters learned during training. The activations from the 50 nodes in the sec-

ond to last layer of the CNN was taken out from the network during the forward pass and appended to the dataset as additional features for each timestamp. The LightGBM implementation of a tree-based machine learning algorithm was chosen as the basis for the hybrid model, motivated by its good performance compared to the Random Forest algorithm. The LightGBM model was trained on a tabular version of the extended dataset including the striped NWP data, 25 seasonal harmonics, and 50 features generated by the second layer of the CNN for each timestamp. The MAAPE metric value on the test set was used for evaluation. A paired sample t-test concluded that the hybrid model significantly improved the accuracy of the predictions ($p < 0.05$) in 12 out of 20 regions compared to the CNN model that was used to generate features. In 4 out of 20 regions, the hybrid model performed significantly worse than the CNN model and the difference was insignificant in the remaining 4 regions. The CNN was trained using Ordinal Classification. The hybrid model improved accuracy in 17 out of 20 regions compared to the LightGBM model that was trained without the additional features generated by the CNN.

7.1.3 The third aim of the study

The third and final aim of this study was to compare how well the deep learning-based models used in this thesis perform against more standard machine learning approaches on this problem. Four different commonly used machine learning algorithms were chosen as baselines: LightGBM, Random Forest, kNN, and AdaBoost. These four algorithms are all suited given the structure of the wind power production problem. A hyperparameter search was conducted for the Random Forest and kNN to find the best suited parameters for these algorithms. The parameters used for LightGBM and AdaBoost were close to their default values. A paired sample t-test concluded that the LightGBM algorithm performed significantly better than the other three across the regions based on the MAAPE evaluation metric on the test set ($p < 0.05$). The LightGBM baseline was therefore compared against most frequently during this thesis. All the baseline models were trained on the same dataset as the deep learning model presented.

The different models was evaluated on the test set using the MAAPE metric. The threshold for significance was set to $p = 0.05$. A one sample t-test concluded that the CNN-based model performed significantly better than the kNN and AdaBoost baseline algorithms on all regions. The CNN-based model significantly outperformed the LightGBM model on average in 13 regions. In 5 regions the LightGBM model outperformed the CNN-based model while in two regions the differences were insignificant. Compared to the Random Forest baseline, the CNN-based model performed better in 17 out of 20 regions. The Random Forest baseline

performed significantly better than the CNN-based model in only one region, while in the remaining two regions the difference was insignificant. A closer inspection of the distribution of the MAAPE metric values over time suggests that the greatest improvement with the CNN-based model over the best performing baseline model, LightGBM, happen when the target load factor is low.

To test the overall performance of the models, the 20 different regions was clustered into to larger regions representing Germany and the Nordic countries. The total production in these two larger regions at time t can be calculated by aggregating the estimated production at each individual smaller regions at that time. The hybrid model improved the overall accuracy in the two larger regions compared to the baseline models. In Germany, the CNN-only model improved the overall accuracy compared to the baseline models.

In the aggregated Nordic region, however, the CNN-based model was not able to improve over the baselines. This was a bit surprising, as the CNN-based model was able predict the power production volume more accurately than the baselines in the majority of regions in the Nordics. A closer inspection of the results shows that the regions that had significant improvement with the CNN-based model in the Nordic countries only accounted for 40% of the total production in the Nordic region. This explain the observation that even though more regions performed better on with the CNN-only model compared to the baselines in the Nordic region, the total estimated production in the larger region did not improve compared to the baselines.

7.2 Contribution

The proposed deep CNN-based architecture were able to predict the wind power production volume more accurately than other common machine learning methods the majority of the regions that were tested. The contributions of this study to the field of wind power production volume prediction is outlined below.

1. *Direct prediction*: Out of the 20 regions that were tested, the proposed architecture predicted the wind power production volume more accurately than the four established baseline models in 13 or more regions depending on the baseline model.

2. *Feature Engineering*: The proposed architecture was able to generate features representing the spatio-temporal information in the weather state over a region in a specified time frame. By including these generated features as an addition to the original dataset, a state-of-the-art tree-based learning algorithm, LightGBM, was able to improve predictions of the wind power

production volume in 17 out of 20 regions.

3. *Novel architecture*: The proposed architecture utilize a 3D convolution section to take advantage of the the temporal correlations in the weather data. This is a different approach than other related works in literature, and is a contribution to the field of wind power production volume prediction.

4. *Flexible Architecture*: The proposed architecture is not bound to a specific geographical location or region due to a spatial pyramid pooling layer, as proposed by He et al. [2015], and can be utilized on any region independent of size and capacity without modifications.

The proposed architecture shows promising potential, and should be further considered as a way to improve the power grid balancing, production planning, and price estimation.

7.3 Future work

Several questions still remain to be answered. A natural progression of this work is to analyze the large variation of performance of the CNN models trained with identical hyperparameters. The large standard deviation of the distribution of accuracy for a single CNN evaluated on the test set after training indicates that the neural network is likely to converge at a sub-optimal local minimum in this problem. Only a few CNN models converge towards a better minimum of the loss function during training. As can be seen from the running mean of the AAPE metric across the validation and testing dataset, these models tend to perform well in general over time. This observation strengthens the general idea that certain combinations of weights and parameters in the neural network are able to capture the underlying stationary structure of the problem to a higher degree than others. Escaping sub-optimal local minima during the training of the neural network was challenging with the implementation used in this thesis.

General knowledge about the convergence of ANNs suggests that a lower learning rate, or a dynamic learning rate defined by a scheduler, combined with longer training sessions (more epochs) might help to avoid sub-optimal local minima and should be considered for further work. More data in terms of a longer historic time series might also improve the model accuracy but the unknown ever-changing distribution of the location of wind turbines might not make this approach as effective as one might expect. Kawaguchi and Kaelbling [2020] propose a more complex network architecture designed to combat the issue of local minima, however, this architecture has not been tested or explored in this thesis.

The architecture of the CNN model designed in this thesis is constructed such that it allows input of different sizes in the spatial dimension. This is possible due to the Spatial Pyramid Pooling down-sampling layer between the convolutional layers at the beginning of the network and the fully connected layers at the end of the network. Further research might explore the strategy of training a single network using the data available from all regions simultaneously. He et al. [2015] and Liu et al. [2019] suggests this learning strategy, and it would be interesting to explore in future work on this problem. This learning strategy would greatly increase the amount of data available to the model during training. Although the distribution of the wind turbines varies from region to region, there is a possibility that a network trained on data from all regions simultaneously could be able to improve the identification of general patterns in the weather data and therefore generate better descriptive features and provide predictions with higher accuracy.

Bibliography

Armstrong, J. S. and Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting*, 8(1):69–80.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

Buhan, S., Özkazanç, Y., and Çadırcı, I. (2016). Wind pattern recognition and reference wind mast data correlations with NWP for improved wind-electric power forecasts. *IEEE Transactions on Industrial Informatics*, 12(3):991–1004.

Cai, Z. and Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.

Chen, T. and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, San Francisco, California, USA. Association for Computing Machinery.

Cheng, J., Wang, Z., and Pollastri, G. (2008). A neural network approach to ordinal regression. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1279–1284. IEEE.

Csáji, B. C. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48.

Davò, F., Alessandrini, S., Sperati, S., Delle Monache, L., Airoldi, D., and Vespucci, M. T. (2016). Post-processing techniques and principal component analysis for regional wind power and solar irradiance forecasting. *Solar Energy*, 134:327–338.

Ding, M., Zhou, H., Xie, H., Wu, M., Nakanishi, Y., and Yokoyama, R. (2019). A gated recurrent unit neural networks based wind speed error correction model for short-term wind power forecasting. *Neurocomputing*, 365:54–61.

Dolara, A., Gandelli, A., Grimaccia, F., Leva, S., and Mussetta, M. (2017). Weather-based machine learning technique for Day-Ahead wind power forecasting. In *2017 IEEE 6th International Conference on Renewable Energy Research and Applications (ICRERA)*, pages 206–209. ISSN: 2572-6013.

Díaz, D., Torres, A., and Dorronsoro, J. R. (2015). Deep Neural Networks for Wind Energy Prediction. In Rojas, I., Joya, G., and Catala, A., editors, *Advances in Computational Intelligence*, Lecture Notes in Computer Science, pages 430–443, Cham. Springer International Publishing.

Everitt, B. S. (1998). The cambridge dictionary of statistics cambridge university press. *Cambridge, UK Google Scholar*.

Gasparin, A., Lukovic, S., and Alippi, C. (2019). Deep Learning for Time Series Forecasting: The Electric Load Case. *arXiv preprint arXiv:1907.09207*.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

González-Aparicio, I. and Monforti, F. (2017). Comparing the impact of uncertainties on technical and meteorological parameters in wind power time series modelling in the European Union. *Applied Energy*, 206.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. arXiv: 1502.03167.

Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Joensen, A., Giebel, G., Landberg, L., Madsen, H., and Neilsen, H. A. (1999). Model output statistics applied to wind power prediction. In *EWEC-CONFERENCE-*, pages 1177–1180.

Ju, Y., Sun, G., Chen, Q., Zhang, M., Zhu, H., and Rehman, M. U. (2019). A Model Combining Convolutional Neural Network and LightGBM Algorithm for Ultra-Short-Term Wind Power Forecasting. *IEEE Access*, 7:28309–28318.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45. Publisher: American Society of Mechanical Engineers Digital Collection.

Kawaguchi, K. and Kaelbling, L. P. (2020). Elimination of All Bad Local Minima in Deep Learning. *arXiv:1901.00279 [cs, math, stat]*. arXiv: 1901.00279.

Kaya, G. O. (2018). A Hybrid Method Based on Empirical Mode Decomposition and Random Forest Regression for Wind Power Forecasting. *Journal of Multiple-Valued Logic & Soft Computing*, 31.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017a). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

Ke, J., Zheng, H., Yang, H., and Chen, X. M. (2017b). Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies*, 85:591–608.

Kim, S. and Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, 32(3):669–679.

Kirillov, A., Girshick, R., He, K., and Dollár, P. (2019). Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

Komusanac, I., Fraile, D., Brindley, G., Walsh, C., and Pineda, I. (2019). Wind energy in Europe in 2018, Trends and Statistics. Technical report, Wind Europe.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lathuilière, S., Mesejo, P., Alameda-Pineda, X., and Horaud, R. (2019). A Comprehensive Analysis of Deep Regression. *arXiv:1803.08450 [cs]*. arXiv: 1803.08450.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Publisher: Ieee.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomForest. *R news*, 2(3):18–22.

Lima, J. M., Guetter, A. K., Freitas, S. R., Panetta, J., and de Mattos, J. G. Z. (2017). A Meteorological–Statistic Model for Short-Term Wind Power Forecasting. *Journal of Control, Automation and Electrical Systems*, 28(5):679–691.

Liodden, E. (2019). Wind power volume prediction. Project Report in {TDT4501}, IDI, NTNU.

Liu, H., Chen, C., Lv, X., Wu, X., and Liu, M. (2019). Deterministic wind energy forecasting: A review of intelligent predictors and auxiliary methods. *Energy Conversion and Management*, 195:328–345.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Lydia, M., Kumar, S. S., Selvakumar, A. I., and Prem Kumar, G. E. (2014). A comprehensive review on wind turbine power curve modeling techniques. *Renewable and Sustainable Energy Reviews*, 30:452–460.

Makridakis, S. (1993). Accuracy measures: theoretical and practical concerns. *International Journal of Forecasting*, 9(4):527–529.

Manero, J., Béjar, J., and Cortés, U. (2018). Wind Energy Forecasting with Neural Networks: A Literature Review. *Computación y Sistemas*, 22(4).

Mehrkanoon, S. (2019). Deep shared representation learning for weather elements forecasting. *Knowledge-Based Systems*, 179:120–128.

Nielsen, H., Nielsen, T., and Madsen, H. (2011). An overview of wind power forecasts types and their use in large-scale integration of wind power. In *Proceedings of the 10th international workshop on large-scale integration of wind power into power systems*, pages 25–26.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016a). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016b). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for Activation Functions. *arXiv:1710.05941 [cs]*. arXiv: 1710.05941.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Răzuşi, P. and Eremia, M. (2011). Prediction of wind power by artificial intelligence techniques. In *2011 16th International Conference on Intelligent System Applications to Power Systems*, pages 1–6. ISSN: null.

Safavian, S. R. and Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674. Publisher: IEEE.

Sarle, W. S. (1996). Stopped training and other remedies for overfitting. *Computing science and statistics*, pages 352–360.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, page 7.

Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946*.

Tastu, J., Pinson, P., Trombe, P.-J., and Madsen, H. (2014). Probabilistic Forecasts of Wind Power Generation Accounting for Geographically Dispersed Information. *IEEE Transactions on Smart Grid*, 5(1):480–489.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. (2015). Learning Spatiotemporal Features with 3D Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497. ISSN: 2380-7504.

Wang, Y., Hu, Q., Li, L., Foley, A. M., and Srinivasan, D. (2019). Approaches to wind power curve modeling: A review and discussion. *Renewable and Sustainable Energy Reviews*, 116:109422.

Wilms, H., Cupelli, M., Monti, A., and Gross, T. (2019). Exploiting Spatio-Temporal Dependencies for RNN-based Wind Power Forecasts. In *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*, pages 921–926. ISSN: null.

Ziat, A., Delasalles, E., Denoyer, L., and Gallinari, P. (2017). Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 705–714. ISSN: 2374-8486.

Appendices

Additional Figures & Tables

In this thesis, the designed deep learning-based model was tested and evaluated on a set of 20 different geographically separated regions. A selection of interesting figures was chosen to be presented in the text and the rest of the figures are included here in the Appendix. As the number of additional figures for each individual region grew large, I decided to make the figures available online. The additional figures and tables are available at `https://folk.ntnu.no/eriklio/master/appendix/`. The complete set of figures and tables included in the appendix is also available in a tarball. The tarball is available at `https://folk.ntnu.no/eriklio/master/appendix.tar`.

Source Code

The source code used in the thesis are available online at `https://github.com/kapteinstein/wind-power-prediction`. The source code contains the complete implementation of the designed deep learning-based model, the data preprocessing and preparation step (including organization, normalization, and transformation), the learning framework with a database storage interface, as well as the evaluation functions used to generate the figures and tables in this thesis. The datasets that has been used in this thesis belongs to the company *Refinitiv* (Dronning Eufemias gate 16, 0191 Oslo), and is not publicly available.

Documentation for the source code is available online at `https://folk.ntnu.no/eriklio/master/docs/`.

Erik Liodden

A Deep Learning-Based Method for Regional Wind Power Production Volume Prediction

NTNU
Norwegian University of
Science and Technology