Master's thesis

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

Simen Ullern & Vebjørn Przytula Fjeldberg

Predicting Final Intraday Electricity Prices in the Very Short Term Utilizing Artificial Neural Networks

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

NTNU
Norwegian University of Science and Technology

Abstract

With the growing inclusion of renewable energy sources, developing price models for intraday trading has become an essential task for many market participants in order to optimize the decision-making process. Yet the available literature on the topic has not been keeping up with the pace of increased intraday trading activity. We predict prices in the final hour prior to delivery on the German intraday market, utilizing Deep Learning techniques.

This thesis looks into the usage of feed-forward neural networks and recurrent neural networks (LSTM and GRU). Price information from earlier transactions for a given target contract is known to be strong variables for this predictive task. For this study, we also look at how well the models can forecast with merely exogenous effects. We make use of a rich feature set composed of weather forecasts, and their intraday updated errors, changes in available production capacities, imbalance volumes, and features engineered from the bid-offer curve. That makes this is a novel study when applying computational intelligence methods for intraday market research.

Prices in the final trading hour are known to be very noisy, caused by short-term dynamics. With careful training of the neural networks, we are able to outperform the statistical baselines on linear errors.

Preface

This Master's thesis was written in the spring of 2020 by Simen Ullern & Vebjoern Przytula Fjeldberg, two computer science students at the Norwegian University of Science and Technology (NTNU).

The project was done in collaboration with the Norwegian OpenAI Lab. We would like to thank our supervisors Massimiliano Ruocco from the Department of Computer Science at NTNU and Gabriele Martinelli from Refinitiv for giving us this research opportunity and receiving useful guidance during this project.

Simen Ullern & Vebjoern Przytula Fjeldberg

Trondheim, June 3, 2020

Contents

List of Figures

List of Tables

List of Acronyms

LASSO Least Absolute Shrinkage and Selection Operator

EPEX European Power Exchange

FFNN Feed-Forward Neural Network

LSTM Long Short-Term Memory

ANN Artificial Neural Network

EPF Electricity Price Forecasting

GRU Gated Recurrent Unit

OLS Ordinary Least Squares

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

VST Variance Stabilizing Transformation

DM Diebold-Mariano

ML Machine Learning

Chapter 1

Introduction

This chapter begins with an introduction to the electricity price forecasting domain and the motivation behind undertaking these studies. We further explain our research goals in Section 1.2 and how we plan to answer them in Section 1.3. Finally, Section 1.4 describes the structure of this thesis.

1.1 Background and Motivation

Electricity is an essential element in modern societies, both in advancing industries and supporting our personal lives at home. With the liberalization of electricity, many power markets have become deregulated, meaning that free competition is introduced to create more efficient and secure markets. As a result, power from a great variety of different sources can enter the grid, for instance, wind, solar, nuclear, coal, and hydro.

Market participants include energy generators, companies that distribute the energy further, large consumers on the demand side who buy power on exchanges or through bilateral contracts, as well as traders who can see economic profits by operating in these rapidly evolving markets. Accurately estimating future electricity prices is an important task for all producers, consumers, and retailers, as it contributes to a more secure and efficient market.

Electricity is an extraordinary commodity since it has limited storage capabilities. In order to maintain a stable frequency in the electric grids, electricity must be consumed as it is produced [Kaminski, 2012]. Though, it is impossible to know ahead of time the exact supply and demand levels for electricity due to

stochastic fluctuation in consumption and production. Of particular attention is the current shift in installed renewable capacity. As we progressively transition into greener energy markets, predicting the future situation is increasingly driven by weather data such as wind and solar production. The weather is, as we all know, stochastic in its very nature. Forecast errors of solar and wind from the day before are regularly above thousands of megawatts (MW) in countries such as Germany; mean absolute errors are rarely below 10 percent. Market participants make their decisions based on those forecasts, and misplanning leads to over- or undersupply in the electricity markets [Ziel, 2017].

European markets usually solve this problem through a set of cascading markets that are highly interrelated. The exact dynamics between these markets differ from country to country because the supply share of different power production technologies can vary greatly. In this thesis, we focus on the German market since it is commonly used amongst researchers. The reason for this is that Germany has been one of the leading nations transitioning into renewable energy. Over 25 percent of the current energy mix is powered through renewables, and this keeps on increasing [Murdock et al., 2019]. Wind and solar production alone accounted for more than a third of the net electricity generation in Germany in 2019[1]. It is the largest electricity market in Europe and has over one thousand active market participants [Germany Trade And Invest, 2018]. Furthermore, since Germany has been at the forefront of innovating modern energy market designs, certain results found here may be predictive of what may yet become relevant in other electricity markets [Goodarzi et al., 2019].

The main market for electricity trading is the spot market. This market is also called the day-ahead market because market participants trade power contracts for fixed time slots for the following day. The goal with such a market is to match the supply and demand to determine a market clearing price, which is the intersection between the (aggregated) supply offers and bids for different power volumes submitted by market participants [Weron, 2014]. This concept is shown in Figure 1.1. The spot market in Germany closes at 12 PM, and a market-clearing price is then obtained for each of the 24 hours for the next day.

The intraday market is the subsequent market that allows participants to continuously adjust to the evolving levels of demand and supply that was not foreseen from earlier. Intraday trading is therefore especially valuable for market participants with a high share of renewable energy sources such as wind and solar, and need to balance their short-term deviations due to weather forecast errors. The intraday market in Germany allows for trading of hourly and quarter-hourly products closer to the final hour of delivery. In Germany, it is possible to trade

[1] See www.energy-charts.de

until 30 minutes before delivery, with an exception being a special rule that allows for trading until 5 minutes before delivery if localized inside one the four regional control zones. Trading on the German continuous opens at 15:00 for hourly products and at 16:00 for quarter-hourly products of the following day. Figure 1.2 illustrates the timing between the day-ahead auction and the intraday market.

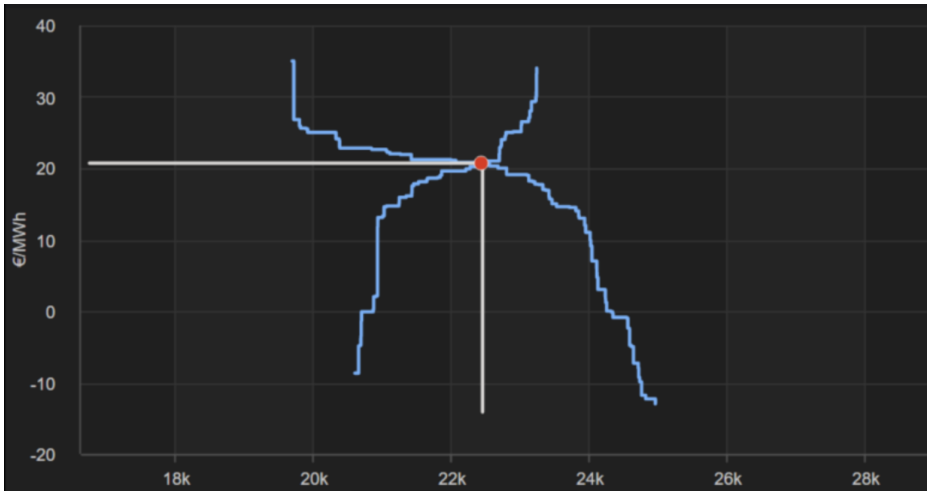

Figure 1.1: Supply offers and demand bids for the 10th of May 05:00 on the EPEX day-ahead exchange in Germany. The intersection between the supply offers and demand bids for different power volumes determines the market-clearing price. Figure from Refinitiv's visualization platform.

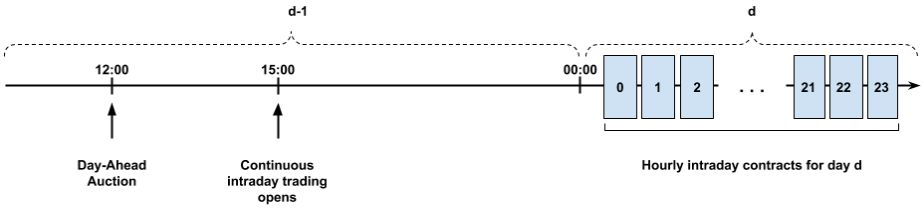

Figure 1.2: Timing between spot and intraday market.

If the supply has not matched demand after trading activities on the day-ahead and intraday market, imbalances are corrected shortly before delivery by the transmission system operators on a real-time regulating market to ensure final equilibrium. Imbalances are managed financially by imposing a settlement cost

that traders must pay if they diverge from their original binding plans to the real-time production or consumption levels. In many areas, this cost can be very high because system operators need to be able to hold back or call in extra production at very short notice [Weron, 2014].

To avoid punishing imbalance costs, market participants are encouraged to fix their position on the intraday market. With more intraday trading caused by stochastic generation levels and greater uncertainties, these costs have been rising. Market participants face in turn greater risk [Goodarzi et al., 2019]. The tendency is to trade more power intraday; total trading volumes from intraday trading sessions in Germany have increased from 26 TWh in 2014 to above 50 TWh in 2018 [Glas et al., 2020].

Traditional literature on the topic of Electricity Price Forecasting (EPF) is usually concerned with day-ahead prices, whereas the body of articles related to intraday EPF still is little, but now steadily growing. To this developing trend we wish to contribute.

It should be made clear that the motivation behind accurately predicting electricity prices is not just about making pure economic profits, but also ensuring system stability and reliability. As prices become more volatile from unpredictable conditions, the chance of severe imbalances occurs, and hence the risk of an unstable frequency in the grid increases. By accurately forecasting future electricity prices, system stability also improves [Lago et al., 2018].

From a research perspective, EPF makes for a fascinating study. Electricity prices have characteristics like seasonality in different frequencies (daily, weekly, yearly), as production and consumption depend on the weather situation and the business cycle [Weron, 2014]. Moreover, prices can change rapidly and unpredictably and cause sharp spikes, but with a mean-reverting process in the long run. Complex and dynamic non-linearities influence financial bidding strategies in the short term. In turn, prices are obscured with leptokurtic noise caused by many factors. [Weron, 2007].

In this thesis, we look into some of these factors, and study to what extent they can be used as features in Machine Learning (ML) models to predict the hourly prices on the European Power Exchange (EPEX) intraday market traded shortly before delivery. The justification for the latter is simple: Most market participants trade in the final hours before delivery because they want to be able to react to the latest information. We choose to focus on the prediction of prices for the final hour, as most transactions happen this hour (opposed to any other hour) [Narajewski and Ziel, 2019]. Our constructed target series is an approximation of the average price for hourly products traded in the final 60 minutes before delivery, described more thoroughly later in Section 4.1. Similarly

to [Marcjasz et al., 2020], we work under the assumption that a practitioner has one hour to make the forecast, take strategic decisions, and schedule bids. This aims to be as similar to real conditions as possible.

Deep Learning-based models have been a major success in many fields for dealing with unstructured, complex, and noisy time series data. We are motivated by the idea that using these techniques can be beneficial on this task as well, because of their known flexibility and efficiency in handling complexity and non-linearity [Zhang and Fleyeh, 2019] [LeCun et al., 2015].

Using Neural Networks is no new phenomenon in the EPF literature, and many have reported much success in using them [Ugurlu et al., 2018], [Lago et al., 2018] [Kuo and Huang, 2018]. But the literature employing computational intelligence methods is especially scarce when intraday market research is concerned. Day-ahead and intraday electricity markets are characterized by different data generating processes and therefore require different modeling approaches [Maciejowska and Weron, 2019]. There is a clear research need for exploring advanced computational intelligent methods within the intraday EPF literature.

We have a unique dataset composed of fundamental variables such as weather forecasts of wind and solar production and their errors since spot time, changes in grid availability, imbalance volumes, and features that are engineered from the bid-offer curve. In this thesis, we explore how our different proposed deep learning-based models tackle the task of predicting final intraday prices on the German continuous and compare them against state-of-the-art statistical methods utilizing data from June 2019 till March 2020.

1.2 Goals and Research Questions

The objective is to predict the final intraday prices using artificial neural networks. In that process, we want to explore and especially find out the following:

Research question 1 *To what extent does exogenous factors determine the final intraday prices that can be predicted?*

We define exogenous factors in this setting to be all data that is not directly tied to intraday market state variables. Specifically, we want to explore the difference in the proportion of variance that can be explained with and without involving prior intraday prices in the set of feature variables. This translates to comparing R^2 scores from two sets of models: The first set is built from models that do not include prior intraday transaction prices, while models in the second set have no such restriction.

We want to emphasize this because the price formation from earlier transactions

for a given target contract is already known to be influential variables for this predictive task. That is because fundamental information already is partly or fully incorporated into the observable transaction prices. From a research point of view, it can be interesting to build a more fundamentals-based model that only includes exogenous effects. Undertaking such a study gives a clearer picture of how fundamental data determine the final intraday prices.

In addition, we want to compare different architectures of neural networks and see how they apply in this domain.

Research question 2 *How do recurrent networks compare to feed-forward networks when predicting final intraday prices?*

This means that we want to compare model architectures that lack the concept of time with sequential models designed to deal with the serial dependence in the time series. We build a Feed-Forward Neural Network (FFNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) to help us answer these questions. The FFNN is fed only with the most recent updated values, whereas the recurrent models are fed with a history of prior values.

1.3 Research Method

Our research questions will be answered through data-driven design and experiment.

This research methodology builds on induction: We try to find theories from data, and seek conclusions from particular cases to the general case. Specifically, we build models as well as we possibly can, that might find patterns expressing certain regularities in the underlying data-generating process. On the contrary, we might not find any significant patterns at all. Machine learning is, to some extent, always stochastic, and we can give no definite answers. We will perform hypothesis tests to assess the statistical significance of different forecast results.

We further argue that the best way to answer our type of questions and put them to use, is to place ourselves in the shoes of a practitioner and exhibit having conditions similar to real conditions. This principle is kept in mind throughout this work.

1.4 Thesis Structure

First, Chapter 2 presents the background theory required to understand the baselines and proposed models, and how we statistically can assess the relative significance of different forecast accuracies obtained from them. We also say some

words about price determinants in the intraday market as these insights are required to know why and how we choose to include certain variables in the feature set.

Chapter 3 then moves into a review of published related work, where we put ourselves in the landscape of literature that concerns intraday EPF research.

Afterward, we turn to the dataset in Chapter 4 and explain precisely how the variables are timely constructed. Chapter 5 then introduces our baselines and proposed models.

Chapter 6 specifies the experimental setting. How we search for optimal network configurations is an integral part of this work, and explained here in the experimental setup.

Further, we report our results in Chapter 7 by evaluating the models' predictive capability and test the significance of the outperformance of different forecasts. From the results, we evaluate and give a thorough discussion.

In Chapter 8 the thesis ends with answers to our research questions. We also argue whether any contributions have been made to the field, and finally give some words about suggested future work.

Chapter 2

Background Theory

This chapter covers the theoretical background of this Master's thesis. Section 2.1 to 2.3 present the theory of neural networks and linear regression methods that is needed to understand how our models work. We then proceed to the Diebold-Mariano test in Section 2.4, which is the test we use to statistically identify out-performance between forecast series. Finally, Section 2.5 summarizes price determinants in the German intraday market. The latter is included to better understand the market dynamics and why we choose to train the models with certain input variables.

Relevant background material on neural networks were carried out in the project preceding this thesis [Ullern and Fjeldberg, 2019]. Section 2.1 and 2.2 are taken from the project report, with some additional background theory included.

2.1 Artificial Neural Networks

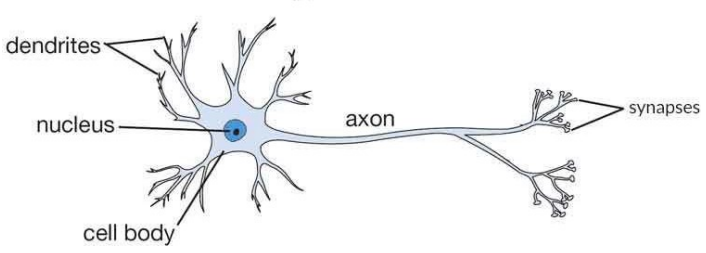

Figure 2.1: Illustration of a biological neuron

The human brain consists of nearly 10 billion basic information-processing units called neurons, interconnected through synapses. As Figure 2.1 illustrates, neurons have a simple structure. Each neuron consists of a cell body, soma, several fibers called dendrites, and a single long fiber called the axon. Although the structure is simple, together, the neurons constitute tremendous processing power [Negnevitsky, 2005, p. 165-166].

An Artificial Neural Network (ANN) is a computational model loosely inspired by the functionality of the human brain. An ANN is composed of simple interconnected units, also called neurons. The neurons pass signals to its connected neurons through their shared link. The functionality of the link is not only to propagate signals but also to determine the sign and strength. Each link, therefore, has an associated weight, w, where the magnitude and sign express the importance of the signal [Negnevitsky, 2005, p. 166-167]. Neurons also have a bias, b, used to shift its activation. Artificial neural networks are used to approximate generally unknown functions, including tasks like speech recognition, machine translation, playing board games, and predicting stock prices.

2.1.1 Feed-forward neural networks

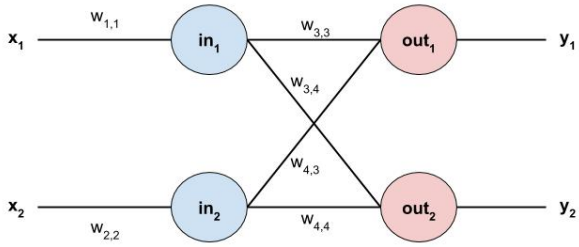

Figure 2.2: Feed-forward network with two input layers and two output layers.

The FFNN is the simplest type of neural networks, where information moves only in one direction, with no cycles. Figure 2.2 illustrates a simple FFNN with two input and two output neurons. The output signal of a neuron j is generated by first computing the weighted sum of its input signals, and subtract the value of the bias (Equation 2.1). Then the neuron applies an activation function g to derive the output (Equation 2.2) [Russell and Norvig, 2010, p. 728].

$$in_j = \sum_{i=0}^{n} w_{i,j} x_i - b_j \tag{2.1}$$

$$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j}x_i - b_j) \tag{2.2}$$

The network illustrated in Figure 2.2 is also a perceptron since all inputs are connected directly to the outputs. With such a network, only linearly separable functions can be learned [Russell and Norvig, 2010, p. 729].

Deep neural networks

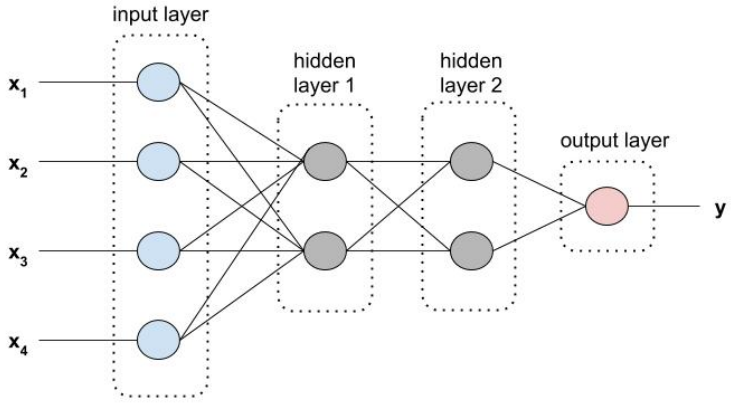

Figure 2.3: Deep neural network with two hidden layers.

Things get more interesting when we add hidden layers between the input and output layers. We then get a multilayer neural network that can solve quite complex tasks. Figure 2.3 shows a deep neural network with a four neurons input layer, two hidden layers with two neurons, and an output layer with one neuron. Each layer has its functionality. The input layer accepts inputs from the outside world and passes them to the hidden layer, usually without computation. Then the output layer gets information from the hidden layer and uses this to determine the output. Multiple processing layers learn representations of data with multiple levels of abstraction [LeCun et al., 2015]. According to [Touretzky and Pomerleau, 1989], the hidden layers should be called *learned-feature detectors* or *re-presentation units* because the activity pattern in the hidden layer is an encoding of what the network thinks are the significant features of the input. With one hidden layer, any continuous function of the input signal can be represented. With two layers, even discontinuous functions can be represented [Russell and Norvig, 2010, p. 731-732].

2.1.2 Backpropagation

The most commonly used learning algorithm for artificial neural networks is backpropagation. Based on a training set of input patterns with corresponding desired outputs, the algorithm repeatedly adjusts the network's weights, ensuring that the actual output of the model is the same as, or sufficiently close to, the desired output. [Rumelhart et al., 1986].

A network calculates its output by propagating the input to the output layer. This procedure is called the forward pass. When the forward pass is done, the backpropagation algorithm can adjust the weights. All neurons are a part of the computation and responsible for some of the errors. Therefore weights have to be adjusted accordingly. The first step is to compute the error between actual and desired output, based on the chosen loss function. Mean squared error is a commonly used loss function, calculating how close a regression line is to a set of points (Equation 2.3), where y is the actual value and $\hat{y}$ is the predicted value.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.3}$$

The goal then is to minimize the loss function by using gradient descent. It is necessary to compute the partial derivatives of the loss function with respect to each weight in the network. In the backward pass, these partial derivatives are propagated from the output layer to the input layer, and the weights get adjusted. Backpropagation is, therefore, a procedure of repeatedly using the chain rule to assign the blame for a particular error to specific weights in the network [Rumelhart et al., 1986].

Although such a learning rule lacks biological plausibility, networks trained with backpropagation (or slight variations of them) have been the most used neural network architecture since its invention. Significant successes have been achieved in real-world applications, meaning that it is one of the most impressive engineering feats of our time [Stork, 1989][Specht and Shapiro, 1991].

Optimization algorithms

Backpropagation is an iterative process. For each iteration, we feed the network with training data and modifies parameters based on the calculated gradient of the loss function, to slightly improve it. Different optimization algorithms have been presented, and they can be divided into three categories; batch gradient descent, Stochastic Gradient Descent (SGD), and mini-batch gradient descent. Batch gradient descent computes the gradient by processing all the training

records simultaneously in a large batch. This computation can be very slow as we need to calculate the gradients of all the records to perform one update but is guaranteed to converge to the global minimum for convex error surfaces and a local minimum for non-convex surfaces. SGD methods only use a single record at a time, which is usually much faster and can be used to learn online, meaning that records are drawn for a continually created stream of records, rather than from a fixed set of records. Mini-batch gradient descent takes the best of both worlds by performing an update based on a mini-batch of records. Standard mini-batch sizes range between 50 and 256 but is also an optimization problem as it can vary for different applications [Ruder, 2016][Goodfellow et al., 2016, p. 275-278]

Weight initialization

When training networks, local optimization techniques are usually employed, often leading the training algorithm to reach a local minimum. Therefore, the local minimum will determine the quality of the network, making it advantageous to get a local minimum close to the global minimum. A factor that influences the final local minimum is weight initialization. Several weight initialization methods exist, such as the uniform random initialization inside the interval [-0.05, 0.05]. Different weight initialization methods should be explored when training neural networks to find the initialization that best suits the network. Weight initialization is also influencing the speed of convergence, the probability of convergence, and the generalization capabilities [Fernández-Redondo and Hernández-Espinosa, 2001].

2.1.3 Regularization techniques

The goal of machine learning is to build models that learn from data to find patterns or give accurate predictions on new and unseen similar data. The concept of a model that can learn from some data, and apply the gained knowledge correctly on unseen data is called generalization. Neural networks are known to be difficult to generalize, and how well a network generalizes depends, among other things, on the number of samples available for training, the complexity of the underlying structure in the data, and the network architecture.

The process of training neural networks is demanding, and a trained network's predictive power can be classified into three types of fit. Figure 2.4 illustrates these types of fit. The blue dots are samples used to train the model, the green line is the real underlying data structure, and the red line is the model prediction. *Underfitting* is characterized by a model that can neither produce accurate predictions on the training data nor generalize to new data. The model struggles to capture the underlying trend and is often a result of a too simple architecture.

When a model fits the training data too well, it is known *overfitting*. Overfitting occurs when a model is too complicated relative to the complexity of the data. The model then starts to memorize the training data by capturing its noise and can't generalize to new data.

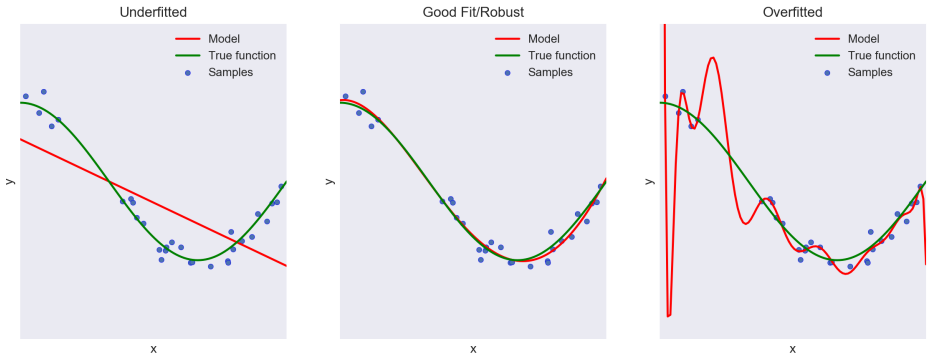

Figure 2.4: Underfitting vs. overfitting.

Both overfitting and underfitting lead to weak predictions on unseen data, and ideally, we want a model to be robust. Robustness can be achieved by applying regularization techniques. A regularization technique is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error [Goodfellow et al., 2016, p. 117].

L1 and L2 regularization

Many regularization techniques exist. L1 and L2 regularization are two approaches based on limiting the capacity of neural networks by adding a norm penalty $\Omega(\theta)$ to the loss function. Equation 2.4 and Equation 2.5 respectively describe the L1 and L2 norm penalties mathematically, where $\mathbf{w}$ refers to the weights of the network.

$$\Omega(\theta) = ||\mathbf{w}|| \tag{2.4}$$

$$\Omega(\theta) = ||\mathbf{w}||^2 \tag{2.5}$$

L2 is commonly known as weight decay and drives the weights in a network closer to zero by adding the sum of squared values of weights to the loss function while the L1 penalty is the sum of absolute values. One can look at L1 regularization as

a feature selection mechanism, as this norm penalty causes some of the weights to become zero. These zero-weights correspond to features not essential and can be discarded in a feature selection analysis. L2 doesn't force weights to be zero and performs better when all input features influence the output [Goodfellow et al., 2016, p. 227-231].

Dropout

Dropout is a well-used regularization technique, providing a computationally inexpensive but yet powerful method to regularize models. Dropout is similar to bagging methods in some way. Bagging is an ensemble method where we train multiple models and combine their results to achieve better performance on new unseen inputs [Goodfellow et al., 2016, p. 255]. It can be seen as a voting scenario where all the models in the ensemble vote for output. For example, the ensemble outputs the prediction with the most votes if it is a classification problem or the mean of all predictions if it is a regression problem.

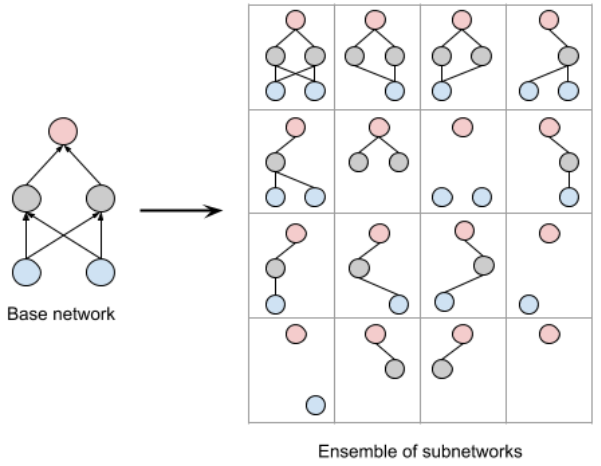

Figure 2.5: Base network with sub networks.

A naive approach to get the same behavior in a neural network is to train all subnetworks and use them in an ensemble. This is done by constructing different combinations of non-output neurons from the original network, as illustrated in Figure 2.5. But as we can see, a lot of the subnetworks have no input units or connected paths from input to output. As mentioned earlier, neural networks usually consist of several hidden layers with many neurons, so the problem escalates with the size of the network. Another issue with this approach is the lack of

shared parameters between subnetworks and the escalating memory usage with more sophisticated networks.

Dropout approximates the process of training an ensemble of networks with different architectures, but with shared parameters. For each iteration, we sample a randomly different binary mask applied to all the input and hidden neurons. The binary mask chooses which neurons to be dropped out during the forward pass, backward pass, and the parameter update. Usually, input neurons have a probability 0.2 of being dropped and hidden neurons 0.5 [Goodfellow et al., 2016, p. 256-257]. In Figure 2.6, neurons are being dropped out in each iteration, resulting in different architectures.

Conceptually the effect of dropout makes the training process noisy, forcing neurons to be more independent and take on more or less responsible for the inputs.

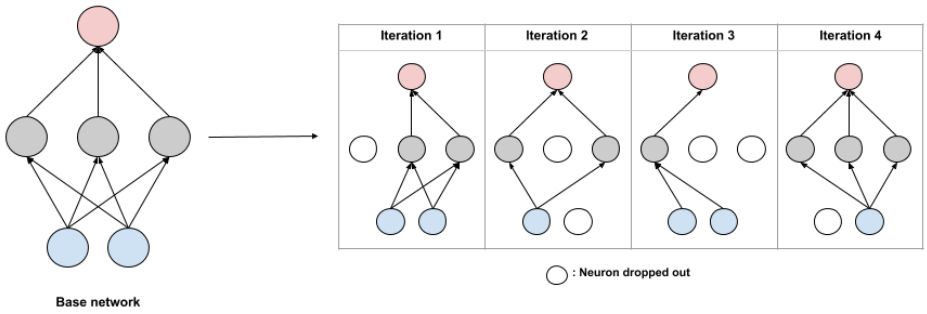

Figure 2.6: The effect of dropout in neural networks when training on different iterations.

2.2 Recurrent Neural Networks

Feed-forward networks offer a lot of useful properties and capabilities. Input-output mapping, with fixed input size, is one of them. In input-output mapping, an input vector is presented to the network, and the weights are then modified to minimize the error between the network's actual output and the input vector's desired output. This is repeated for all vectors in the train set over several epochs as to get the most optimal weights. The network learns from the examples by constructing an input-output mapping of the data [Svozil et al., 1997].

Many problems can be solved by using FFNNs with a fixed input size vector, but not all. The fixed input size limits their usage, especially affecting problems with

no predetermined input size. This is, for instance, the case in machine translation problems, where the length of sentences may vary.

Another limitation of FFNNs is their lack of the concept of time. Several problems require this as useful information can be found. When predicting a stock price, the price trend during the last hours may contain useful patterns that can give a more precise prediction of the future price. FFNNs have no built-in mechanisms to remember information back in time. They lack the expressivity to remember past information for arbitrarily large input sizes.

RNNs address the issue of remembering past information by having cycles in them. The structure of the RNN is similar to that of a standard feed-forward network, but through the cycles, the network can retain information about the past, making it possible to discover temporal relationships between information that are far away from each other in the input data [Pascanu et al., 2013]. Figure 2.7 visualizes an unfolded RNN.

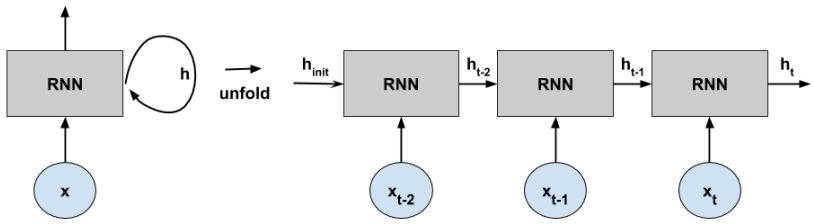

Figure 2.7: RNN architecture, both folded and unfolded.

From Figure 2.7 we see that the RNN consists of a hidden state , h, and operates on a variable sequence length $\mathbf{x} = (x_1, x_2, ..., x_t)$, ordered in time. At each time step, i, we feed the network with x_i, and the hidden state is updated. Hence, h_i is a vector representation of the i'th point in time. When we refer to the last hidden state in an RNN, what we mean is h_t, which is the hidden state produced after feeding the network with the final element, x_t, in the sequence $\mathbf{x}$. It is also possible to stack recurrent layers. This idea was first introduced in [Graves et al., 2013]. The idea of stacking recurrent layers is to feed the hidden state, h_i, through another recurrent layer, as visualized in Figure 2.8. This allows for more complex functions to be captured in the input sequence. Even though the additional power gained by stacking RNNs is not theoretically clear, some studies have shown indications of deep recurrent networks performing better compared to shallower ones [Sutskever et al., 2014] [Irsoy and Cardie, 2014].

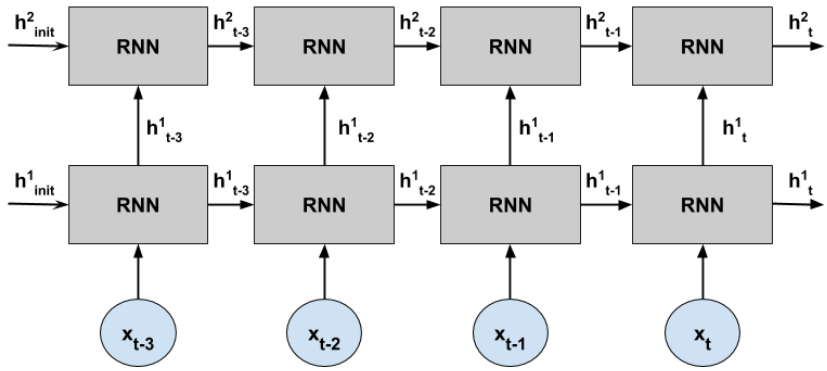

Figure 2.8: A stacked RNN, with two stacked RNN layers.

RNNs are configured differently based on the task. Figure 2.9 shows three configurations. If the problem is to translate sentences, a many-to-many structure can be used, as both the original sentence and the translated sentence contain several words. When predicting a stock's price exactly an hour in the future, a many-to-one structure is suitable based on its price history. Many-to-many may also be used if it is desirable to predict several future prices of the stock. A one-to-many configuration can be used if a neural network is trained to map a one-dimensional vectorized sentence to a whole sentence. The vectorized sentence is then fed into the RNN.

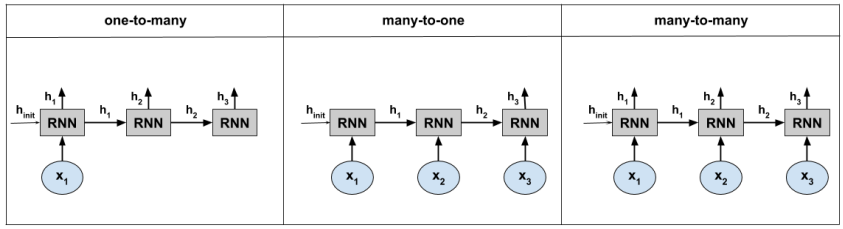

Figure 2.9: Different RNN configurations.

RNNs can also be trained with backpropagation, but the algorithm is called backpropagation through the time when dealing with recurrent networks [Pascanu et al., 2013]. Backpropagation through time is an extension of the standard backpropagation, allowing us to calculate the derivatives needed when optimizing neural networks with the capability of learning temporal patterns [Werbos, 1990]. It works by treating the recurrent model as a multi-layered model (with

the theoretical possibility of having an unbounded number of layers) with back-propagation performed on the unrolled model.

Even though we have algorithms for learning with recurrent networks, it is still considered to be a difficult process. Sequences fed into recurrent networks may often have long-range dependencies the network needs to extract, but it is too hard to capture due to exploding and vanishing gradients. This is when the norm of the gradient increases or decreases exponentially during training. To see why this happens, consider a large unrolled network: Computing the gradient of h_{init} involves many repeated factors of the weight matrix W. If the largest singular value in W is greater than 1, the weights will quickly intensify and grow large. The opposite happens when the largest singular value is less than 1. Exploding and vanishing gradients thus occur when backpropagating errors across long sequences [Lipton et al., 2015, p. 13-14].

LSTM and GRU cells are configurations of the RNN, where sequences propagates through the cells the same way, but is designed to deal with the vanishing gradient problem.

2.2.1 LSTM

Vanishing and exploding gradients make it challenging for RNNs to learn long-range dependencies. Standard RNNs fail to learn in the presence of time lags greater than 5-10 discrete time steps between relevant input events and target signals [Gers et al., 1999]. In 1997, Hochreiter and Schmidhuber introduced the LSTM, a recurrent neural network, designed to overcome the problem of vanishing and exploding gradients [Hochreiter and Schmidhuber, 1997].

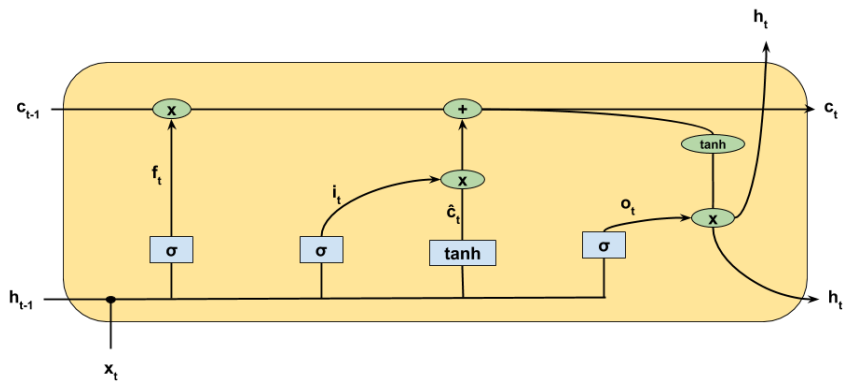

Figure 2.10: The architecture of an LSTM cell.

LSTM is a modification of the architecture of recurrent networks, with the idea of allowing error derivatives to flow better [Le et al., 2015]. LSTMs and RNNs are similar in the way they have a chain of repeating nodes, but the hidden layer is replaced by a more complex memory cell, illustrated in Figure 2.10.

In Figure 2.10, σ is the sigmoid function, *tanh* is the hyperbolic tangent function, $\times$ is element-wise multiplication and $+$ is element-wise addition

The key to LSTM is the cell state, illustrated as the horizontal line from C_{t-1} to C_t running through the network with minor interactions. The cell state is responsible for gathering and keeping information from earlier time steps so as not to lose dependencies between the target and earlier input signals.

With the inclusion of the cell state and the hidden state, new parameters are to be optimized. Standard ways to initialize the states are either by setting them to zero or a random value between -0.5 and 0.5. A third method is to learn a good representation for the initial state. The latter method requires enough state resets to be present in the training data. If that is the case, the model can learn a reasonable default initial state, making it more robust.

LSTM introduces gates to distinguish between essential and superficial information. A gate is a sigmoidal unit that takes activations from the hidden layer at time step h_{t-1} and the current input signal x_t and produces a value that is used to multiply the cell state to manipulate it [Lipton et al., 2015, p. 18]. LSTMs have three different gates; the forget gate, the input gate, and the output gate.

The forget gate, f_t, does the first manipulation of the cell state. [Gers et al., 1999] introduced the forget gate in 1999, learned to reset the cell state once its content is out of date and hence useless. Then the input gate, i_t, decides what new information is to be stored in the cell state. First, a sigmoid layer is fed with x_t and h_{t-1} and decides which values to update. Second, a tanh layer is also fed with x_t and h_{t-1} and creates a candidate vector, $\hat{c}_t$, that could be added to the state. These are then combined and used to update the cell state with new information. Finally, we have the output gate, o_t, where x_t and h_{t-1} run through a sigmoid. The cell state run through a tanh, and finally multiplied with o_t to get the output h_t, which the LSTM cell believes is the best representation of time step t. [Lipton et al., 2015, p. 17-19].

Many LSTM variations exist with minor differences in the implementation of the gates. In this thesis we use:

$$f_t = \sigma(W_{xf}x_t + b_{xf} + W_{hf}h_{t-1} + b_{hf})$$
$$i_t = \sigma(W_{xi}x_t + b_{ii} + W_{xi}h_{t-1} + b_{hi})$$
$$\hat{c}_t = tanh(W_{xc}x_t + b_{xc} + W_{hc}h_{t-1} + b_{hc})$$
$$o_t = \sigma(W_{xo}x_t + b_{xo} + W_{ho}h_{t-1} + b_{ho})$$
$$C_t = f_t \times C_{t-1} + i_t \times \hat{c}_t$$
$$h_t = o_t \times tanh(c_t)$$

where the W terms denote weight matrices and the b terms denote the bias vectors. For example, W_{xf}, is the weight matrix for the input, x_t, passing through the forget gate, f_t, while b_{xf} is the corresponding bias vector.

Variations of LSTMs exist, and the first LSTM to be introduced did not have any forget gate. An LSTM architecture with no forget gate has a weakness in situations of large input streams. The cell state often tends to grow linearly during a time series presentation, and without resets, it may break down the network. Forget gates were introduced as a solution to this [Gers et al., 1999].

2.2.2 GRU

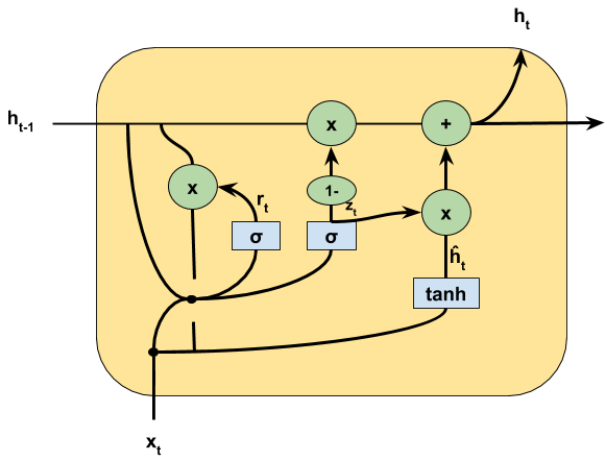

Figure 2.11: The architecture of a GRU cell

GRU is another recurrent layer modification and was proposed in [Cho et al., 2014]. GRU has been motivated by the LSTM but is less complex and, therefore, simpler to compute and implement. Both LSTM and GRU keep existing information and add new information, but are different as the GRU cannot control the exposure of the cell state. The output gate controls this in LSTM, and as shown in Figure 2.11, GRU does not have this gate and therefore exposes its full content. The separate cell state in the GRU is not distinguished with the hidden state, but instead merged. Therefore, the gates behave differently, and have different naming conventions; the reset (r_t), update (z_t), and new ($\hat{h}_t$) gates, computing the following functions:

$$
\begin{aligned}
r_t &= \sigma(W_{xr}x_t + b_{xr} + W_{hr}h_{t-1} + b_{hr}) \\
z_t &= \sigma(W_{xz}x_t + b_{xz} + W_{hz}h_{t-1} + b_{hz}) \\
\hat{h}_t &= tanh(W_{x\hat{h}}x_t + b_{x\hat{h}} + r_t \times (W_{h\hat{h}}h_{t-1} + b_{h\hat{h}}))
\end{aligned}
$$

The final oupout h_t is then:

$$
h_t = (1 - z_t) \times \hat{h} + z_t * h_{t-1}
$$

The performances of GRU and LSTM are often on par, but the GRU is said to be computationally more efficient due to its less complex structure [Chung et al., 2014].

2.3 Linear Regression Methods

2.3.1 OLS

Linear regression makes predictions based on a hyperplane fitted in feature space. For a target variable that can be indexed by day (d) and hour (h), the prediction can be formulated as

$$
\hat{y}^{d,h} = \beta_0 + \sum_{j=1}^{m} \beta_j x_j^{d,h} + \epsilon^{d,h}
$$

Where x_j are the input features, β_j are the fitted coefficients, ϵ is the error term, and β_0 is the intercept term (also better known as bias in Machine Learning literature).

The hyperplane is built with a loss-function that minimizes the residual sum of squares (RSS) between the observed targets and targets predicted by the model. Hence this model-type is often referenced as Ordinary Least Squares (OLS).

$$RSS(\beta) = \sum_{i=1}^{n} (y_i^{d,h} - \hat{y}_i^{d,h})^2$$

2.3.2 LASSO

OLS might be prone to overfitting. There is the option of increasing bias for a reduction in variance by adding a model complexity term to the loss function.

The Least Absolute Shrinkage and Selection Operator (LASSO) extends OLS built by miniziming the RSS while putting a linear penalty function on the coefficients. That is, find the coefficients β that minimize RSS while subject to $\sum_{j=1}^{m} |\beta_j| \leq \lambda$.

λ is a regularization parameter that must be carefully set. If $\lambda = 0$ we have ordinary least squares, and when $\lim_{\lambda \to \infty}$, the coefficients shrink towards zero. The parameters in LASSO are known to become sparse with many coefficients vanishing, and thus provides implicit feature selection when $\lambda > 0$.

2.4 Diebold-Mariano Test Statistics

The Diebold-Mariano (DM) test statistic is a way to assess the null hypothesis of equal forecast accuracy [Diebold and Mariano, 1995]. It can be used to formally assess the significance of the outperformance of different forecasts.

A DM test goes beyond just comparing forecast losses (such as with mean squared error or mean absolute errors), but also says something about how likely two different forecast accuracies are to be caused merely from chance. Note that this test is not intended for comparing models, i.e., it aims to assess the significance between forecasts, not the underlying forecasters [Diebold, 2015].

The test itself is pretty easy to understand. Say that model i outputs a forecast error $\epsilon_{i,t} = y_t - \hat{y}_{i,t}, \{t = 1, ..., T\}$, applied with a loss function $L(\epsilon_{i,t})$, such as the absolute $|\epsilon_{i,t}|$ or square error loss $\epsilon_{i,t}^2$. A loss differential series can be constructed between two series of forecasts, i.e., $d_t = L(\epsilon_{1,t}) - L(\epsilon_{2,t})$. The test is then an asymptotic test of the hypothesis that the mean of the loss differential series is zero.

Specifically, the test statistic is calculated as $DM = \frac{\bar{d}}{\hat{\sigma_d}}$, where $\bar{d}$ is the sample mean loss differential and $\hat{\sigma_d}$ is a consistent estimate of the standard deviation

of $\bar{d}$. Forecast errors may be serially correlated, DM therefore requires the loss differential to be covariance stationary and that $\hat{\sigma_d}$ is to be calculated robustly [Diebold, 2015].

The statistic is then usually conducted two-sided with critical values obtained from the standard normal distribution. However, in the presence of real-world applications where the size of the estimation sample remains finite, and errors occasionally can be huge with more heavy-tailed distributions than the normal, a possible extension is to use the student-t distribution with T-1 degrees of freedom to compare the critical values. This modification is done by [Harvey et al., 1997].

2.5 Price Determinants in the German Intraday Market

Effectively predicting electricity prices requires both domain knowledge of how to choose and structure the input variables, as well as sophisticated algorithms. Studying the intraday price drivers is therefore a necessary step to visit before trying to construct a sensible price model.

The intraday market is the subsequent market where participants trade after the spot market has closed. This market is used to balance the deviations of forecasted demand and supply from updated values. In Germany, you can trade intraday continuously just until a few minutes before delivery. Still, the actual errors in the hour of delivery must, in turn, be balanced on a regulating market in real-time by the transmission system operators.

As a consequence, intraday trading has seen increased activity with the growth of renewable energy sources. Between 2014 and 2018, annual wind production levels increased by ca. 55 percent and solar production levels by ca. 19 percent[1]. In the same period, intraday trading almost doubled [Glas et al., 2020].

But the spot market is still the place where the vast majority of power is traded. This is because it is financially advantageous for the system to aggregate liquidity. In the long run, prices are determined mainly by generation technologies and their relative capacities, fuel prices, and demand levels [Pelagatti, 2018]. As these macro factors are unlikely to change extremely from one day to the next, the intraday price for an hourly contract will, on average, be quite correlated with the spot-price for the corresponding hour.

After the spot price has been settled, supply and demand changes drive the direction of the new intraday price. Deviations can stem from numerous different

[1] See www.energy-charts.de

causes. How much an occurrence of one such cause affects the price, depends on the available technologies and their capacity constraints, and the flexibility and start-up costs of the power plants. In general, the closer the system works towards its capacity constraints, the higher the intraday prices may be, as the system is under pressure [Hagemann and Weber, 2013].

The electric grids always require a stable frequency, meaning that all the produced electricity must be consumed at every point in time. Consequently, it should be clear that even a small cause can result in a massive price spike if the system can recover from it only through expensive means. A well functioning electricity market thus requires the inclusion of flexible electricity technologies that can provide dispatch-ability to the power grid. These energy sources influence electricity prices differently than less flexible sources such as the volatile power generated from wind or solar.

Information about dispatchable energy sources might be obtainable from the merit-order. The merit-order system ensures that the first electricity sources accepted to meet the demand are those with the lowest marginal cost. This is done by arranging the available power plant capacities sorted in their short term variable costs, see Figure 2.12. This principle is the basis for setting day-ahead prices (also known as spot prices). The spot price then equals the marginal cost of the last operating power plant that is needed to satisfy the demand [Pape et al., 2016]. Assuming a perfectly competitive market, this ensures overall profit maximization [Hagemann, 2015].

It can also be assumed that the price formation in the intraday market may, to some extent, be explained through the same merit-order [Hagemann, 2015]. This is because the submitted generation technologies for the day-ahead are also likely the sources that are partly used in adjusting to the evolving equilibrium levels of supply and demand intraday.

With this in mind, we will briefly explain some of the most critical intraday price determinants after the day-ahead market has closed.

2.5.1 Solar and wind forecast errors

Wind and solar power are volatile in their production levels. Forecast errors of solar and wind from the day before have regularly absolute errors above 10 percent. Market participants make their decisions based on those forecasts, which means that over- or undersupply initially enter the electricity markets [Ziel, 2017]. There is a clear need to be able to balance the errors coming from old forecasts as new forecasts become available. Forecast errors from renewables are, therefore, one of the primary sources of intraday liquidity [Hagemann and Weber, 2013].

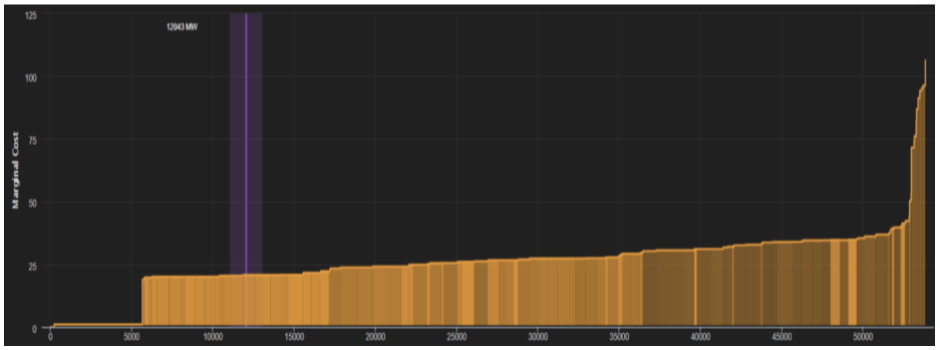

Figure 2.12: Merit-order for 16 May 14:00 in Germany, showing marginal cost [EUR/MWh] as a function of total accumulated generation capacity [MW] for the hour. In this figure, the yellow depicted area under the curve might be generation capacity coming from mostly gas plants, whereas when the curve steepens, and the color becomes more brown, lignite and coal plants are in succession involved. Wind and solar generation yields power that is submitted into the grid with priority, and its marginal costs are therefore deduced from consumption. The purple line is the expected system-wide load minus the generation coming from renewables, i.e., the residual load. This figure is from Refinitiv's visualization platform.

The effect of forecast errors from renewable energy sources on intraday prices is well documented in the literature. See for example [Würzburg et al., 2013], [Kiesel and Paraschiv, 2017], [Ziel, 2017] and [Gürtler and Paulsen, 2018]. There is conclusive evidence that an increase of power generation coming from renewables results in decreasing electricity prices, and vice versa. Electricity generation from renewable sources has low marginal costs and is, therefore, going into the grid with priority.

But the effect on prices is non-linear. [Kiesel and Paraschiv, 2017] argues that the impact on intraday prices coming from the forecast errors in renewables should not be judged in isolation, but dependent on the demand quote (that is to which extent the forecasted demand is covered by the traditional capacity already planned for in the day-ahead market). The impact of weather changes is reported to be more severe for mid-day delivery periods. This is because the demand is high (factories are running), and the merit-order is usually steeper. In these periods, market participants adjust their bids to the updated forecasts more quickly. In this case, the system is under pressure, operating closer towards its constraints, and imbalance costs can, as a consequence, be much higher if rapid

activation of expensive technologies is required. Likewise, [Kremer et al., 2019] finds that renewable forecast changes are more significant in the steep than in the flat merit-order regime and that renewable forecasts have a higher explanatory power at noon than in the morning and evening.

2.5.2 Consumption forecast errors

It is not only the supply side that determines the intraday price. The system load, or actual consumption values, is influenced by seasonal effects, smart appliances, weather, and random effects.

Consumption is systematically higher during day than night and higher on workdays than weekdays [Hagemann and Weber, 2013]. Moreover, some hours see a higher demand. For instance, in the morning times when people are waking up and industrial shifts begin, causes the grid load to quickly increase [Kath and Ziel, 2018]. In the evening, factories might shut their activities, but consumption increases in personal homes from the use of home appliances. Moreover, some smart devices can automatically be turned on when electricity prices are estimated to be at its lowest, e.g., recharging an electric vehicle during night time.

Consumption is also very related to the weather. Colder weather means that electricity must be turned up when it is used for heating. Additionally, one can observe a slight increase in demand during summer months through the use of air-condition [Maciejowska and Weron, 2019].

2.5.3 Unplanned power outages

The impact of sudden changes to the planned production can have enormous effects. This is typically caused by production failures, unanticipated power outages, or availability changes in the grid infrastructure.

If power plant owners experience unplanned outages, they still have to deliver the already scheduled electricity production [Hagemann, 2015]. This calls for trading activity. Market participants are legally required to minimize their use of the imbalance market [Garnier and Madlener, 2015], and imbalance prices are usually so punishing that it puts an economic incentive to use the intraday market to balance predictable deviations. When market participants are encouraged to fix their position on the intraday market, unplanned power outages influence intraday prices.

2.5.4 System imbalance

Almost all trades on the intraday market happens in the last 3 hours prior to delivery [Narajewski and Ziel, 2019]. The reason for this is not only to operate with the most recent weather forecasts. Fundamental weather drivers are sometimes known in advance, but not exploited until very close to delivery. Another reason to trade shortly before gate closure is that you can also better predict the direction of the system imbalance and its corresponding price.

There are price incentives for intraday imbalance optimization. At the time of writing, the volume-weighted average price is used as a reference price, with no particular emphasis on the market situation close to delivery time. [Koch, 2019] reports that utilizing such a reference price is problematic: Between 01/07/2017 and 30/06/2019, the imbalance price incentive was said to be too low in 13 percent of the quarter-hours. Taking intentional imbalance positions is forbidden. However, [Koch and Maskosa, 2019] argues that some practitioners might, to some extent, react to imbalance price expectations, based on their empirical analysis.

Actual imbalance prices are in Germany released to the public long after the settlement period. But imbalance volumes are released almost in real-time. All information regarding the regulating market can be expected to influence intraday prices. Moreover, this might contribute to a wait-and-see behavior among market participants. When the market tightens up shortly before delivery, prices are more volatile, and predicting them is harder for practitioners.

2.5.5 Imports and exports

The influence of neighboring markets can, in certain situations, be of importance. There is currently an increasing share of cross-border trades taking place between countries. The importance of considering market integration for intraday market research is, therefore, an emerging topic. [Hagemann, 2015] finds some empirical evidence that flows between France and Germany influence intraday prices, specifically that French trades affect German intraday prices only during the off-peak periods. [Kath, 2019] is, however, not able to measure any significant influence from the intraday cross-border project.

Transmission capacities between countries is another factor that then comes into play. Line congestions and bottlenecks in the grid cause additional imbalance. Although the transmission system operators might take this part of the bill, it promotes further intraday trading as market participants are legally bound to balance all predictable deviations.

2.5.6 Market power and strategic behavior

Some market participants might be able to influence prices through market power. The typical example is that a generator can offer power when the system is already operating towards its constraints for non-competitive prices that exceed marginal generation and ramping costs [Hagemann, 2015].

Furthermore, larger firms might be able to operate around the clock having offices in different parts of the world. In contrast, small producers are only able to clock-in during regular office hours. Traders sitting on larger shares might also have a richer set of trading strategies to utilize. For example, imbalances can be internally adjusted from within their portfolios before trading the difference in the market. In this way, market power can be used to darken the available market state variables for other participants.

This is very relevant because empirical results also show that traders tend to use present or past price information to forecast future prices [Pape et al., 2016]. If you are a price setter, this can be used as an advantage.

That was a brief introduction to some of the price determinants that affect the intraday market. It does not aim to be a complete study on this comprehensive topic (for that, please consult the referenced literature). Instead, this helps us understand which variables we should include in our feature set, how we should go about formatting them, and what kind of predictive ability we might expect to see.

Chapter 3

Related Work

This chapter studies the related literature. Following the review of [Weron, 2014], the various methods proposed within the EPF literature can be divided into five areas: Multi-agent methods, reduced-form methods, fundamental models, statistical models, and methods that utilize computational intelligence:

- Multi-agent methods create simulations where agents interact with each other in a game-theoretic fashion. Different agents have different utility functions and goals in the electricity market, so their interdependent behavior jointly builds the price estimating process.
- Reduced-form models are quantitative models that study the statistical properties of electricity prices over time to deduce optimal positions.
- Fundamental methods use the underlying assumption from standard economic theory that prices in competitive markets result from the equilibrium of demand and supply [Pape et al., 2016]. These methods then look at the physical and economic factors that drive the demand and supply levels for electricity prices.
- Statistical methods utilize traditional econometric models. These models search after an equation based on the statistical relationship of the variables.
- Computational intelligence is a broad term for all types of nature-inspired methodologies where dynamic learning is employed to accomplish a task, usually by finding non-linear and complex patterns in a dataset.

Regression methods (statistical) and neural networks (computational intelligence) are the two most widely used models [Weron, 2014]. A pitfall of statistical models is often said to be that they lack the expressive power required to effectively model the non-linear behavior in high-frequent data, as these methods often are linear forecasters [Lago et al., 2018] [Amjady and Hemmati, 2006]. Neural networks have greater expressive power but a higher variance in their performance, as they are dependent on a broad set of parameters that can be hard to tune effectively.

In this chapter, we review the literature which, to the best of our knowledge, exists for predicting intraday prices in the very short term on European electricity markets. Market designs from other parts of the world are often quite different from European and are outside the inclusion of this study. We furthermore emphasize that predicting prices in the very short term means predicting prices after the day-ahead market has closed and hence having full knowledge of the settled spot prices.

Section 3.1 covers studies on the German intraday market. Other European market studies are reviewed in 3.2. Finally, in Section 3.3, we summarize the findings.

3.1 The German Intraday Market

[Narajewski and Ziel, 2019] analyze the ID3-price (an index that records the volume-weighted average price in the final 3 hours before delivery in the German intraday market) for hourly and quarter-hourly products separately. Using data from 01/01/2015 to 29/09/2018, this study utilizes three statistical modeling techniques: Ordinary least squares, LASSO, and elastic net (elastic net linearly combines both L1 and L2 penalties on the size of the coefficients). In the set of regressors, they include results from the day-ahead and intraday market, as well as the imbalance volume from the balancing market. Among the proposed linear models, the LASSO performed the best.

By assuming full information and hence having access to the most recent transactions, they claim the market to be weak-form efficient when hourly products are concerned. That is, none of the proposed models performed better than the volume-weighted average price of the last 15 min before the forecast. For quarterly products, the most recent price was reported not to give satisfying results. One explanation for this might be the lower number of transactions by the time of forecasting compared to hourly products. They say this is because the distribution of transactions for quarter-hourly products is even more skewed close to delivery time.

Also [Uniejewski et al., 2019] use the LASSO estimator to predict an ID3-like

time series for the German intraday market, using data from 01/01/2015 to 30/04/2018. The LASSO model was reported to significantly outperform its competitors, namely an extremely naive benchmark that is just the corresponding spot price, as well as an autoregressive expert-model. Unsurprisingly, they also found the strongest regressor to be the latest available price. No fundamental variables were included in this study; only day-ahead prices and prior ID3-prices were used as external regressors.

In later work, the same authors showed that it is possible to beat the latest available ID3-price slightly, but significantly [Marcjasz et al., 2020]. This the case if the LASSO model is also augmented with timely predictions of fundamental variables for the coming hours. In that study, the system-wide load, solar and wind generation forecasts, and their corresponding ex-post errors are accounted for. Furthermore, balancing volumes and recent price information from neighboring products are included. The latter could say something about changes in the expectations about price levels over time, they argue.

Be aware that the latest available price, in this case, is the volume-weighted price between 4 hours and 4 hours and 15 minutes before delivery, i.e., they give the market participants one hour to make the forecast and make strategic decisions. This price is not the same as the one used by [Narajewski and Ziel, 2019], which assumes full information and is constructed in the final quarter before forecasting, giving only 15 minutes for an agent to schedule bids. The latter is harder to beat since many transactions are likely to happen within that differing 45-minute time window.

A technique that these authors make use of to boost performance is to ensemble different forecasts. The best result takes a simple average of the two predictions coming from the LASSO-estimated model and the most recently available ID3 price. The reason why this simple averaging scheme improves on linear errors is because of the massive price spikes that can occur in the electricity price time series. Because they happen rather infrequently, models trained on a limited sized dataset and designed to generalize will not be capable of predicting the full magnitudes of price spikes. In these cases, the autocorrelation coming from prior transaction prices are tremendously rewarding on aggregated error measurements.

An interesting extension made in this study is the inclusion of knowing future values of the exogenous variables for the day, i.e., having "perfect" forecasts at hand. In this case, they claim to be able to reduce the mean absolute error by over 2 percent, highlighting the importance of accounting for the latest forecasts when predicting electricity prices in the very short term.

A rather different approach is taken by [Janke and Steinke, 2019], who forecast the entire volume-weighted price distribution in the final 3 hours of German intraday

electricity market using data from 01/07/2017 to 31/03/2019. This work aims to estimate a much richer representation of the market behavior, which is important if different market participants value electricity differently.

More specifically, this study looks into forecasting the quantiles for the last three hours before delivery, using various linear regression models and an ensemble of neural networks with two hidden layers. Their forecasts report to not improve significantly over the naive benchmarks, except for the tail of the distribution. Among the proposed models, the LASSO again performs the best.

Note that both the proposed models and naive benchmarks utilize price information all the way up until 3 hours before delivery. From this, they find that including exogenous variables of load, wind, and solar power forecasts do not improve on the accuracy while considering time-series information from neighboring products and quantiles did. This finding is perhaps not surprising if market participants quickly adjust their bids to updated forecasts of renewables [Kiesel and Paraschiv, 2017]. Empirical studies of [Kremer et al., 2019] find that renewable forecast updates are reflected in intraday prices within one trading minute. Receiving an update on the weather forecasts every hour is thus likely not sufficient enough.

3.2 Other European Intraday Markets

[Monteiro et al., 2016] look into hourly price forecasting in the six intraday sessions of the Iberian electricity market (mainland of Portugal and Spain), having data from 2012 and 2013. In this work, each intraday session was modeled independently using shallow neural networks with a single hidden layer with $2n + 1$ neurons, where n is the number of input explanatory variables. For robustness, the outputs of 20 training processes of the same network were assembled to a single value.

Many networks were built differing only in the selected input variables, which were all a subset of recent prices, demands, generations from the previous day, forecasts of demand, forecasts of wind production, as well as calendar effects. The best results on the majority of the intraday sessions were obtained from only utilizing prior prices from the day-ahead and the intraday session, as well as accounting for calendar effects.

[Andrade et al., 2017] also find that simpler models tend to yield the best performance for Iberian intraday auctions. Using data from June 2015 to June 2017, linear quantile regression models for each intraday session were constructed. Despite having a broad set of fundamental predictors available, the models that only

incorporated price information from previous sessions performed the best for the probabilistic forecasts.

In [Kolberg and Waage, 2018], multiple Deep Learning-based models were designed to predict the volume-weighted average price for the last six hours of trading before each delivery hour on Elbas, the Nordic continuous. This study had an extensive dataset covering all intraday transactions from 02/11/2011 till 31/12/2017. Weather forecasts were included in the form of having separate spatio-temporal networks for pattern-finding in weather forecast maps. Other interesting features included in this work were grid transmission capacities and prices from the regulating market, as these are quickly publicly released in the Nordics.

Networks as deep as 18 layers were tried in this study. An LSTM network with residual connections was reported to perform the best. In this architecture, the weather forecast maps are first sent through a convolutional section before going through an LSTM layer. This new feature representation of the weather situation is further concatenated with the sequential market data features, which have also first been sent through two stacked LSTM layers. Finally, three residual blocks are piped, where extensive use of dropout and batch normalization is employed.

A more recent article from [Oksuz and Ugurlu, 2019] uses data from 01/01/2017 to 28/02/2019 to predict prices in the Turkish intraday market. In this setting, the target series was the quantity weighted average of all the intraday trades of the hourly contract. Five input variables were used: The day-ahead prices, balancing market prices, forecasts of the generation's ratio coming from renewables, forecasted demand over forecasted supply, and the volumes that have been traded for the delivery hour. The spot price was reported to be the most important variable because intraday prices were said to be very close to the day-ahead prices.

Among the proposed models, the recurrent neural networks outperformed the classical statistical methods. The GRU achieved the best results, which had a simple architecture of one GRU-layer with a hidden dimension of 50 blocks.

3.3 Summary

We see that the proposed models in the visited literature are all either statistical linear regression models or neural networks. Whereas neural networks have been reported to outperform linear methods outside of Germany, the latter methods have shown the most convincing results there.

One must keep in mind that the generalizability of the results coming from the

literature concerning intraday markets outside of Germany, translated over to our task, is very questionable. We note that:

- Although the Iberian market is characterized by having a tight integration of different renewable energy technologies in their supply share (with comparable wind and solar generation levels to Germany), the Iberian market with its 6 auction-based intraday sessions is a different market design compared to the continuous trading in Germany.
- The study by [Kolberg and Waage, 2018] in the Nordic intraday market experienced issues with liquidity. The intraday market had an average of only 23 intraday transactions per delivery hour in their dataset. The target series had to be calculated in a six-hour wide window because shorter windows would give sparsity problems. The problem of little intraday liquidity stems from having a lot of lucrative hydropower in the Nordics[1]. Having flexible hydropower to balance over- or undersupply might put on less of a burden for market participants to trade on the intraday market. This likely makes the settlement costs lower, and market participants are further not as encouraged or required to trade on the intraday platform. Low liquidity causes the benefits of trading renewable energy sources in the intraday market to remain unexploited [Skajaa et al., 2015].
- The Turkish intraday market is also not as liquid as the German market. Even though Germany and Turkey have similar population levels, Turkish intraday trading in 2018 only accounted for 2.93 TWh [Epias, 2018], whereas German intraday trading in 2018 surpassed 50 TWh [Glas et al., 2020]. One of the reasons for this is the fact that Turkey has a much lower share of wind and solar production levels[2].

It is thus more likely to be the other way around; as Germany has been at the forefront of innovating modern energy market designs, certain results found here may be predictive of what may yet become relevant elsewhere [Goodarzi et al., 2019].

In Germany, it is the LASSO augmented with fundamental variables that has shown the best results. We include a LASSO-model in our baselines later. This model provides both implicit feature selection and strict regularisation. Yet we have seen from section 2.5 that many non-linearities determine the intraday prices, so it is somewhat difficult to envision that the best one can do is drawing a straight line through feature space and make predictions from that. Evidently, research employing advanced computational intelligence methods should be further

[1] See, for example, https://www.iea.org/statistics
[2] See footnote 1.

explored, and so this is where we are.

Furthermore, all of the studies in Germany incorporate prior price information. The lagged transaction price for a specific target contract will alone be a powerful predictor because parts of the fundamental information is already incorporated into prior prices, coupled with other unobservable effects. It can be interesting to build a more fundamentals-based model that only includes exogenous effects and no prior price information. This extension will be included in this work.

Chapter 4

Data

This chapter describes the dataset in some detail.

In Section 4.1 we look into the properties of our target series, that is the average of the 4 latest traded prices, sampled every quarter for the hourly product. We further elaborate on how to scale and transform it in order to maximize generality and accelerate learning capabilities in Section 4.2.

4.3 covers the features. At the center are the weather forecasts made at spot time for renewable generation and the system wide consumption load, together with their respectively intradaily updated forecast errors. We then utilize the forecast errors in relation to the bid-offer curve that is available from the day-ahead in order to get data on how the supply side might value the change coming from updated values. We call this the merit-order response. Finally, we incorporate power outages, imbalance volumes and price formations from neighboring products.

We wrap it up with a summary of the features in Section 4.4.

Important to note is that all of these variables are very timely constructed. We work under the assumption that a practitioner has one hour to make the forecast, take strategic decisions and schedule bids for the final hour. This aims to be as similar to real conditions as possible.

Notation-wise, $V_q^{d,h}$ depicts variable V with its value recorded q quarters before delivery for the product that is delivered at hour h on day d.

All data is either given directly by, or engineered from data given by, Refinitiv.

4.1 Target Series

The German continuous intraday market is a pay-as-bid market, meaning that the exchange can match a buy order and a sell order if they are of identical prices. This results in a large number of transactions with different volumes and prices. Capturing the overall market price thus requires some form of aggregation scheme for point prediction tasks.

The volatility of the prices is said to depend on the volume in the transaction [Narajewski and Ziel, 2019]. Therefore, small volumes can result in quite different prices even though they are traded close in time. The leading reference price in the German intraday literature is, therefore, the ID3 or ID1 price index. This is, in essence, the volume-weighted price for all transactions happening 3 and 1 hours before delivery, respectively.

Unfortunately, we do not have volumes available to us. Instead, we have the latest traded price on the hourly products, for every quarter in the last 96 quarters (or 24 hours) prior to delivery. Most market participants trade in the final couple of hours before delivery because they want to be able to react to the latest information. In our work, we choose to focus on predicting the prices for the final hour, as this is the hour with the maximum number of transactions [Narajewski and Ziel, 2019]. Moreover, the traded volume from hourly contracts is more than five times larger than the volume of the quarter-hourly products [Janke and Steinke, 2019], so we focus on the hourly product.

We decide to take the average of the final 4 quarters and have that constitute our target series. We hope that this, overall, is a good representation of the final intraday prices. Formally, our target series can be described as

$$y^{d,h} \equiv \frac{p_1^{d,h} + p_2^{d,h} + p_3^{d,h} + p_4^{d,h}}{4}$$

Where $p_q^{d,h}$ is the difference between the latest traded price with delivery on hour h at day d, recorded q quarters before delivery and the corresponding spot price for hour h on day d.

From this definition, it should be clear that we do not predict the full magnitudes of actual intraday prices, but rather the spread between the intraday and day-ahead prices. The delta has a smaller solution space because intraday and day-ahead prices are pretty correlated to begin with, as previously explained in Section 2.5. Predicting the price spread is a better way to train the models and evaluate their actual predictive ability.

Figure 4.1 illustrates how the target series is calculated for 15 randomly sampled hours in the dataset. It shows that some "outliers" are bound to exist when we aggregate prices from individual transactions. To further highlight how prices differ within the final trading hour, some basic statistics of the differential series between the maximum and minimum price sampled in the final hour have been summarized in Table 4.1. However, it is also evident from Table 4.2 that the price spread between intraday and day-ahead prices are altogether quite correlated when being close in the temporal order.

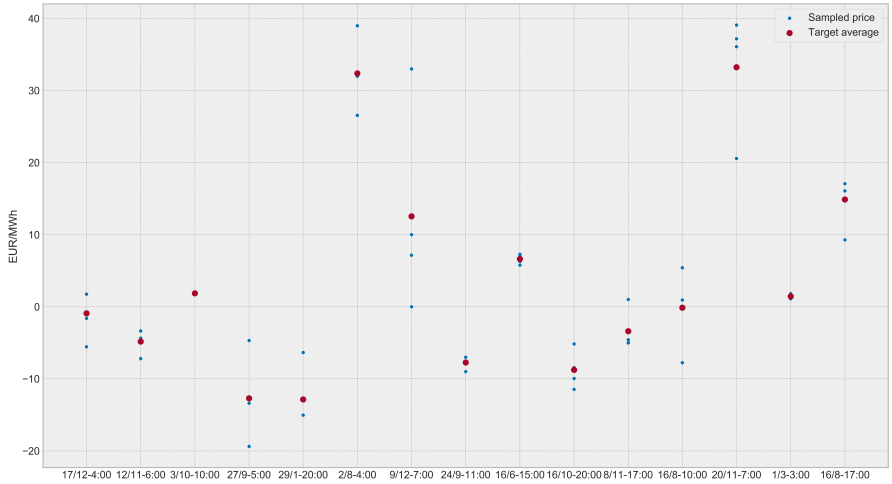

Figure 4.1: How the target series is constructed. Blue dots are sampled prices (spread between spot and intraday price) for every quarter in the final trading hour before delivery. The red circles denotes the average that constitute a point in the target series.

	mean	**std**	**min**	**25%**	**50%**	**75%**	**max**
$\max_{1\leq i\leq 4} p_i^{d,h} - \min_{1\leq i\leq 4} p_i^{d,h}$	8.53	13.75	0	3.04	6.00	10.41	498.00

Table 4.1: Descriptive statistics for the differential series between the maximum and minimum price observed during the 4 final quarters, i.e. statistics calculated from the series $\{\max_{1\leq i\leq 4} p_i^{d,h} - \min_{1\leq i\leq 4} p_i^{d,h}\}$.

	$\mathbf{p}_1$	$\mathbf{p}_2$	$\mathbf{p}_3$	$\mathbf{p}_4$
$\mathbf{p}_1$	1			
$\mathbf{p}_2$	0.85	1		
$\mathbf{p}_3$	0.77	0.92	1	
$\mathbf{p}_4$	0.71	0.83	0.83	1

Table 4.2: Correlation coefficient among the p_q sampled in the four final quarters before delivery.

To avoid too many outliers, we take several measures to enhance robustness. Firstly, in our target series, we only include those prices that are constructed from four non-missing values. This means that no imputation is done with regards to missing values in the variables that build the target series. This accounts for 572 hours or nearly 3 weeks of data, which is ca 7.9 percent of our original data that spans from 01 June 2019 till 27 March 2020. Losing this amount of data is a serious hit for any data-driven model, especially for ANNs with a lot of parameters (weights) that need to be finely tuned. We argue that it is better than the alternative of running the risk of including potential erroneous or (more) noisy targets.

Secondly, as will be described in more detail in the next section, we utilize variance stabilizing transformations on electricity price data. That means that no outlier can contribute with a loss that is many standard deviations higher than that of any other data point.

The constructed time series is illustrated in Figure 4.2. Descriptive statistics for it has been summarized in Table 4.3 below.

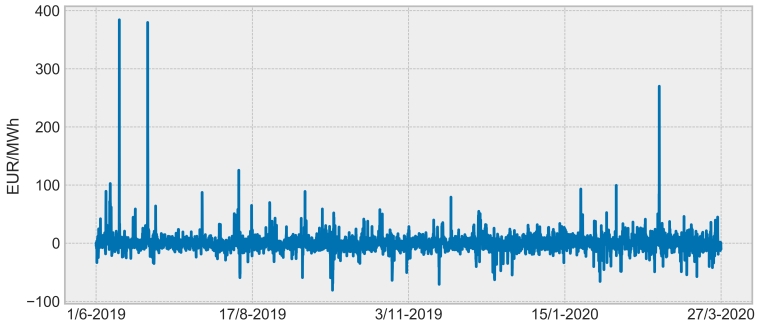

Figure 4.2: The constructed target series.

	mean	std	min	25%	50%	75%	max
$y^{d,h}$	0.49	15.15	-80.82	-5.68	-0.30	5.66	384.27

Table 4.3: Descriptive statistics for the target series.

4.2 Variance Stabilizing Transformation

Noticeable from Figure 4.2 is the price spikes in the target series. Such outliers should be dealt with somehow; otherwise, a model built by minimizing quadratic or absolute errors might severely overfit these occurrences.

The first option is to design a model with a separate price spike component built into it. A second option is to apply a Variance Stabilizing Transformation (VST) to reduce the variation in the data. Applying such a transformation reduces the impact of extreme outliers, and is well-studied within the EPF literature [Uniejewski et al., 2017].

When utilizing deep neural networks, one can hope that the network automatically finds an appropriate mapping from original values to a more useful representation in parameter space in the first couple of layers. But this is not given, and puts on additional complexity to the learning process. We argue that we should, from having a relatively short dataset, apply a VST to the price series beforehand.

A common way to transform a spiky time series is simply through the logarithm. This can seldom be used within EPF studies because electricity prices can be negative[1]. The same goes for the price spread between day-ahead and intraday prices. Instead, we use a well-known VST that works for negative values, the area hyperbolic sine (asinh) function. This transformation has not too long ago been used by, for example, [Marcjasz et al., 2020] and [Narajewski and Ziel, 2019], and shown convincing results. This function is illustrated in Figure 4.3, and calculated as:

$$asinh(x) = log(x + \sqrt{(x^2 + 1)})$$

And the back-transformation is simply the hyperbolic sine:

$$sinh(x) = \frac{e^x - e^{-x}}{2}$$

[1]Negative prices occur when the costs of shutting down and ramping up a power plant exceed the loss of accepting negative prices. This usually happens when demand is low, and production levels from renewable sources, in turn, are very high (mostly due to wind) [Weron, 2014].

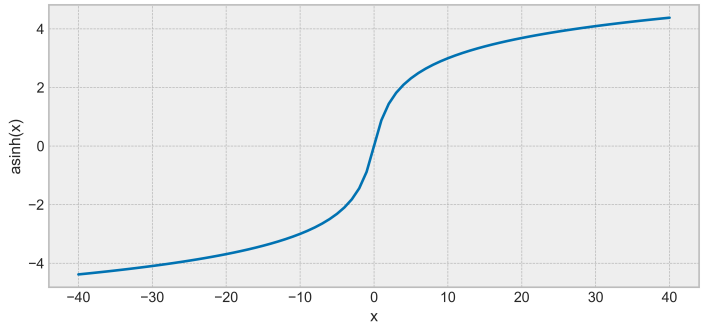

Figure 4.3: The area hyperbolic sine function.

Note that before applying this VST to a variable, we remove the median and scale it according to the interquartile range. This is an alternative to the more common standardization approach by removing the mean and scaling to unit variance. However, for small and spiky datasets where the sample mean and variance are sensitive to outliers, the median and the interquartile range is a more robust alternative.

Figure 4.4 depicts the original target series scaled and transformed. This is the format of the prices in which we train our models to predict. But when we later report and plot our results, we do the back-transformation such that the outputs still appear in a realistic form.

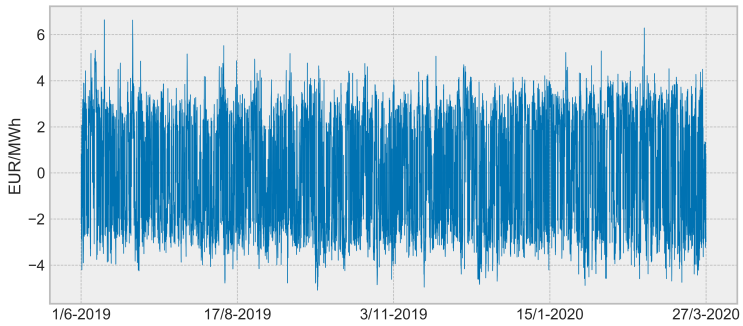

Figure 4.4: The target series transformed.

4.3 Input Features

4.3.1 Wind and solar forecasts made at spot time

For each target hour, $y^{d,h}$, we have data on the forecasted total wind power generation, $\hat{WI}^{d,h}$, and solar power generation, $\hat{SO}^{d,h}$, made at day *d-1*. These are estimates produced by Refinitiv. They, in turn, utilize forecasts from solar radiation, wind speed, temperature, etc., made by The European Center for Medium-Range Weather Forecast (ECMWF). These power generation forecasts are therefore built from proprietary models using public inputs. ECMWF runs four forecasts a day, which is every six hours. The first distributed forecast is made between 05.40 and 06.55 UTC, known conventionally as the spot forecast. This is the forecast we include in our feature set.

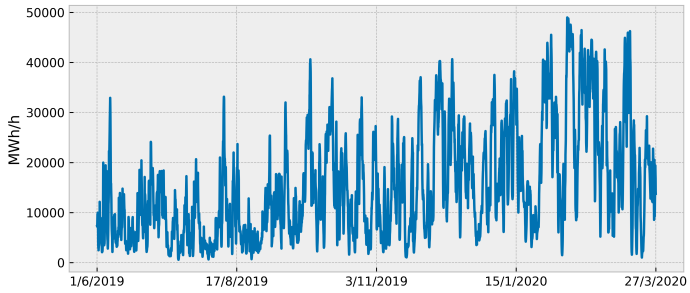

Figure 4.5: Spot time forecasts for wind power production, $\hat{WI}^{d,h}$.

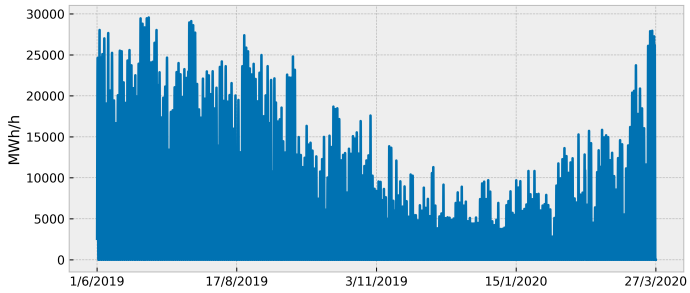

Figure 4.6: Spot time forecasts for solar power production, $\hat{SO}^{d,h}$.

Figure 4.5 and 4.6 visualizes the $\hat{WI}^{d,h}$ and $\hat{SO}^{d,h}$ time series from 01/06/2019, to 27/03/2020, in an hourly resolution. For $\hat{SO}^{d,h}$ we see a natural pattern. Estimated solar power is low at night, increases in the morning and into the afternoon, and decreases towards the evening. Sunny months have higher forecasts, measured in magnitude, compared to cooler months. This is not unexpected as the sun is stronger and lasts longer in the spring and summer. By studying Figure 4.7, this effect becomes more evident, where we plot the forecasted solar power for the first two weeks of August and December.

For the $\hat{WI}^{d,h}$ feature, we see that the forecasted wind power is, on average, lower in the summer with a steady escalation in the following months.

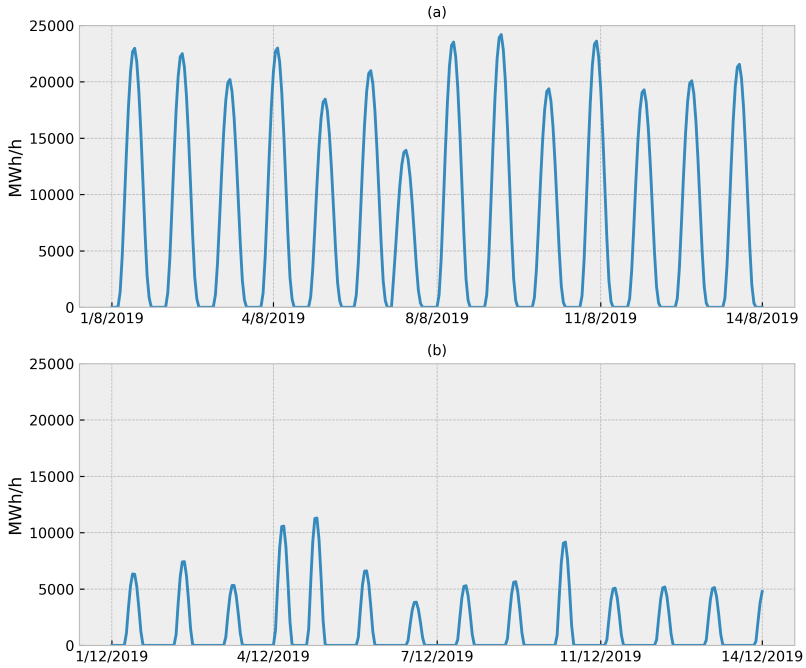

Figure 4.7: $\hat{SO}^{d,h}$ first two weeks of (a) August and (b) December.

4.3.2 Wind and solar forecast errors since spot time

Updated forecasts closer to the delivery hour have higher precision. With new forecasts, we can estimate the error relative to the one made at spot time, which we refer to as deltas. Refinitiv supplements us with this data in two variations. The first one is the deltas between the spot forecast and the forecast made by Refinitiv that builds on the data most recently published by the ECMWF. In the second variation, live production data (aggregated by the system operators) is utilized by interpolating the latest forecast against the latest production data. The resolution of the data is quarterly. The two variations have been illustrated in Figure 4.8.

A positive delta leads to a power shortage in the system, and a negative value leads to a power surplus in the system, compared to what was assumed at spot time.

We define solar power deltas and wind power deltas to be $\Delta SO_q^{d,h}$ and $\Delta WI_q^{d,h}$. The calculated deltas represent the forecast errors made for the day d and hour h, recorded q quarters before delivery. As we give an hour in advance to make the forecast for the final hour, q=8, is the latest delta that we can use.

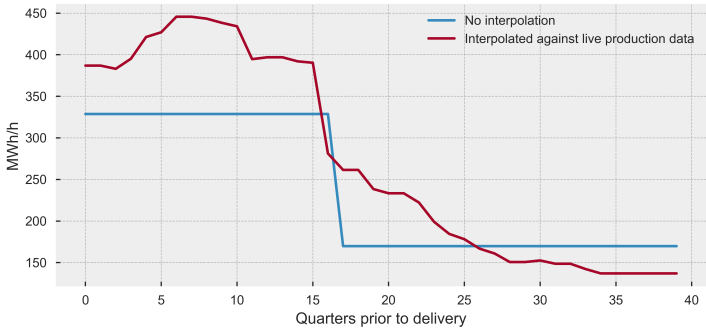

Figure 4.8: $\Delta WI_q^{d,h}$ illustrated in the 10 final hours before delivery for the product with delivery 2020-03-19 16:00 UTC with (red) and without (blue) the inclusion of interpolation against live production data.

If we miss a value for a certain quarter, we forward-fill with the most recently observed delta for that hour. If we have no prior observed deltas, we assume the forecast error is 0. In total we miss 186 hours or ca 2.8 percent of $\Delta SO_8^{d,h}$ and $\Delta WI_q^{d,h}$ in the dataset. In 23 of those 186 hours, we have no observation history for that hour, and thus have to assume that the forecast error is 0.

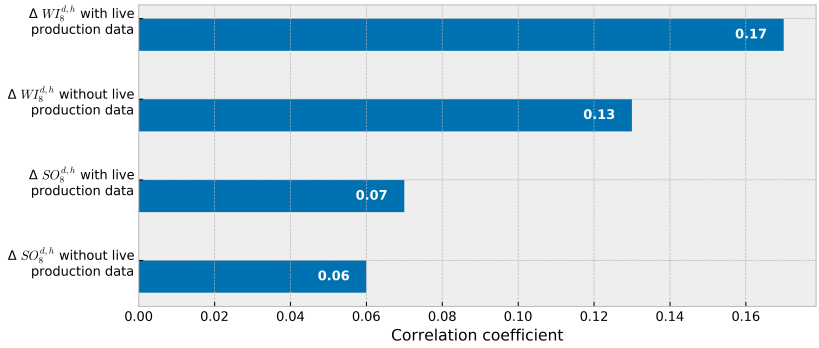

Figure 4.9: $\Delta WI_8^{d,h}$ and$\Delta SO_8^{d,h}$ correlations with $y^{d,h}$, both with and without the inclusion of interpolation against live production data.

Figure 4.9 shows a diagram of the correlation between the target series and the forecast deltas eight quarters before delivery, both with and without the inclusion of live production data. We see a higher correlation when the latest production data is included in the error calculation for both $\Delta SO_8^{d,h}$ and $\Delta WI_8^{d,h}$, so we choose to include only them in our feature set.

The $\Delta SO_8^{d,h}$ and $\Delta WI_8^{d,h}$ deltas are represented in Figure 4.10 and 4.11. We see that weather forecast errors from the day-ahead often are large. The solar deltas show a mean absolute error of 157 MWh/h and a maximum absolute error of 4.8 GWh/h. Extreme changes are also seen in the wind deltas with a mean absolute error of 853 MWh/h and a maximum absolute error of 7.7 GWh/h.

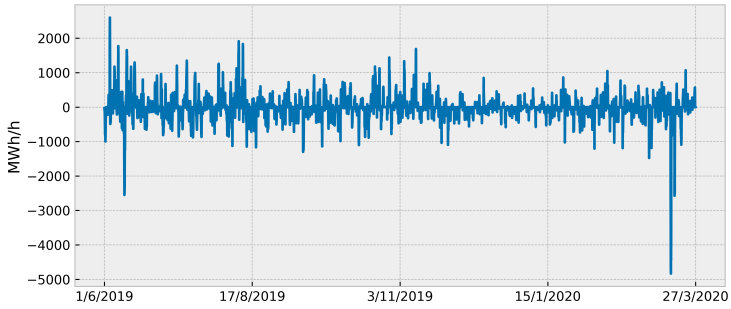

Figure 4.10: Solar forecast errors, $\Delta SO_8^{d,h}$, since spot time.

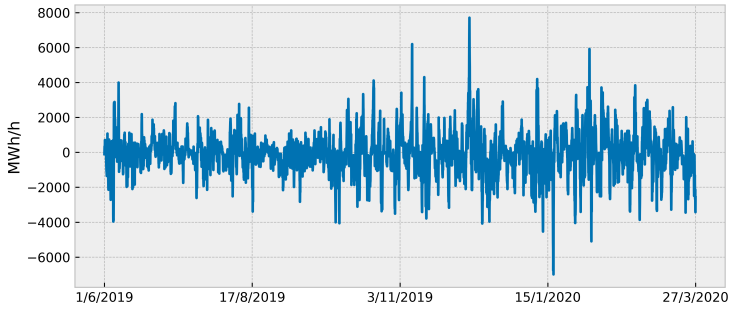

Figure 4.11: Wind forecast errors, $\Delta WI_8^{d,h}$, since spot time

4.3.3 Consumption forecast made at spot time

The total estimated power consumption (or the system-wide load) at spot time for a given hour h, on day d, is captured in the feature that we will refer to as $\hat{CO}^{d,h}$. Figure 4.12 visualizes it for our dataset, and we notice a few things. Firstly, in cooler periods, forecasted consumption increases, which is expected as more energy is needed for electrical heating. Secondly, consumption has a daily pattern. This is captured in Figure 4.13, showing a plot of 4 consecutive days of the forecast. The consumption is usually lowest in the middle of the night and has a steady increase in the morning, reaching its daily peak around midday. After that, consumption decreases as industrial shifts end, but one can observe a slight plateau when people return from work and cook their dinners, and possibly turning up the electrical heating.

We can also see the effect holidays have on consumption forecasts. In periods with low economic activity, power consumption is also, in turn, much lower. Hence, information about holidays, weekends, time of year, etc., is implicitly captured in the $\hat{CO}^{d,h}$ feature. The weekend effect of estimated consumption is visualized in Figure 4.14, where 4 weeks of data is plotted.

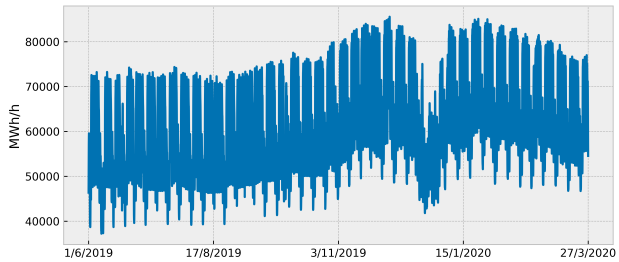

Figure 4.12: Forecasted consumption, $\hat{CO}^{d,h}$, made at spot time.

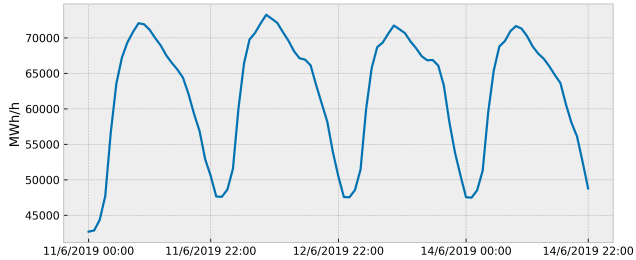

Figure 4.13: Four consecutive days of $\hat{CO}^{d,h}$.

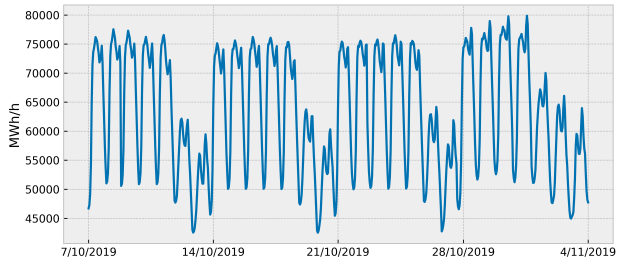

Figure 4.14: Weekend effect of consumption forecasts. On the November 1st is the All Saints' Day, also this public holiday in Germany is captured in the consumption forecast.

4.3.4 Consumption forecast errors since spot time

We refer to consumption forecast errors since spot time as $\Delta CO_q^{d,h}$. In this case, the delta is not calculated against live data (as it is bad from the source), but rather the most recent forecast relative to the spot forecast.

We forward-fill with the most recently observed delta for that hour if we miss a value for a certain quarter. If we have no prior observed deltas, we assume the forecast error is 0. In total, we miss 199 hours or ca 3 percent of $\Delta CO_8^{d,h}$ in the dataset. In 23 of those 199 hours, we have no prior observed history, and thus have to assume that there is no forecast error.

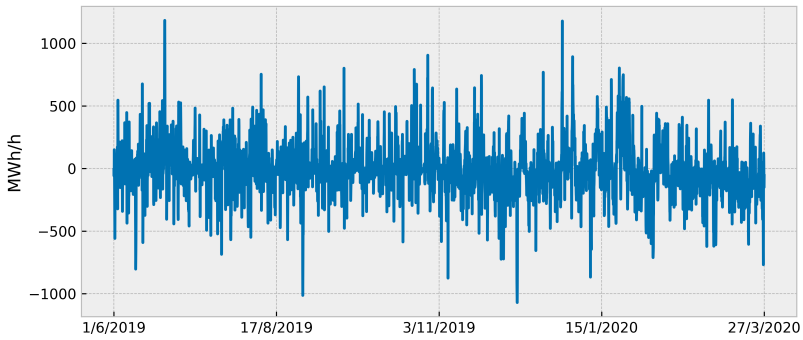

Figure 4.15: Consumption forecast errors, $CO_8^{d,h}$.

The consumption forecast errors for $q = 8$ are plotted in Figure 4.15. We see that the consumption errors have a smaller mean absolute error (128 MWh/h) and maximum absolute error (1.2 GWh/h) compared to the wind and solar forecast errors. This makes sense because the factors determining estimated consumption are likely less volatile than factors determining solar power production, and especially wind power production.

4.3.5 Changes in available production capacities since spot time

Power plant owners might experience unplanned outages after spot time, which leads to a reduced quantity of power offered in the market, affecting the prices. Refintiv supplements us with data representing changes in available production capacities. By parsing Urgent Market Messages, they assume whether this infor-

mation was known at spot time, and estimates how much power this change in availability corresponds to.

We define the feature to be $\Delta AV_q^{d,h}$, which is the changes in availability q quarters before hour h on day d since spot time.

We forward-fill with the most recently observed delta for that hour if we miss a value for a certain quarter. If we have no prior observed deltas, we assume there is no change in available production. In total, we miss 186 hours or ca 2.8 percent of $\Delta AV_8^{d,h}$ in the dataset. In 23 of those 186 hours, we have no prior observed history, so we must assume that there is no change in available production.

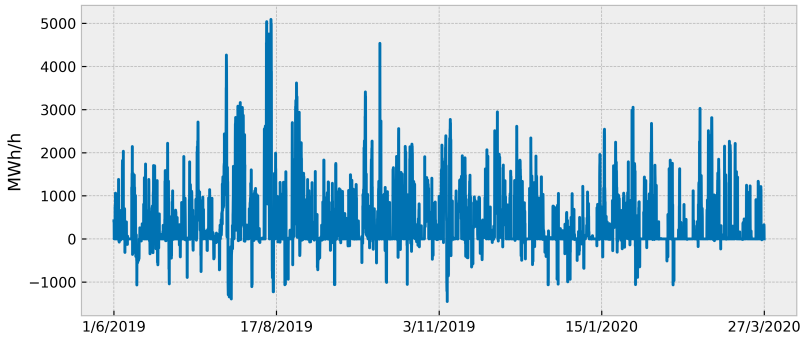

Figure 4.16: Changes in available production capacities, $\Delta AV_8^{d,h}$, since spot time.

We have plotted $\Delta AV_8^{d,h}$ for our entire dataset in Figure 4.16. Outages are common occurrences. In some cases, levels are negative, meaning that power plant owners reported on having available production capacities, previously assumed not to be available at spot time. This can, for example, be explained by delaying planned maintenance work. However, positive deltas are the most usual, with a mean value of 370 MWh/h and a max value of 5.1 GWh/h. The correlation coefficient with the target series is just slightly positive, showing a value of 0.06.

4.3.6 Sum of errors since spot time

Two simple features are engineered from the errors since spot time. The first one is the sum of the wind, solar, and consumption forecast errors, and says something about the net forecast errors for weather-dependent variables. Hence we refer to this as $\Delta WE_q^{d,h}$. In the second engineered feature, we also include the

change in availability to the aggregation, which we refer to as $\Delta AE_q^{d,h}$. Formally they can be described as

$$\Delta WE_q^{d,h} = \Delta SO_q^{d,h} + \Delta WI_q^{d,h} + \Delta CO_q^{d,h}$$

$$\Delta AE_q^{d,h} = \Delta SO_q^{d,h} + \Delta WI_q^{d,h} + \Delta CO_q^{d,h} + \Delta AV_q^{d,h}$$

As can be seen in Figure 4.17, these constructions have a slightly higher correlation with the target series. Therefore, we include them in the feature set.

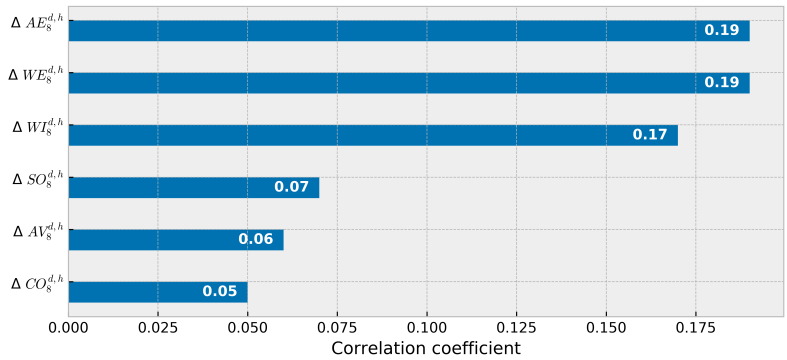

Figure 4.17: Deltas correlations with $y^{d,h}$.

4.3.7 Spot price

Spot prices are settled at 12 PM on the day ahead, and we refer to them as $DA^{d,h}$. Figure 4.18 shows plots of both the spot prices and the full intraday prices in the final hour. Spot prices have a lower standard deviation and a lower number of extreme levels. The time series have a high correlation coefficient of 0.75, hence telling us that the linear relationship between them already is strong.

Although we are not predicting the magnitudes, but rather the spread between day-ahead and intraday prices, it is reasonable to believe that the spot price is an important feature for this predictive task. Theoretically speaking, high spot prices arise from the fact that many power plants, including those with high marginal costs, are needed to equalize the demand in the day-ahead market. This leads to less available flexibility in the total system, and small changes can

cause major price changes because we are in a steep part of the merit-order. On the contrary, a low spot price probably means that the system is far from operating towards its constraints, but might then require ramping effects with a rapid increase in marginal costs to equalize demand. This is the convex property of the merit-order.

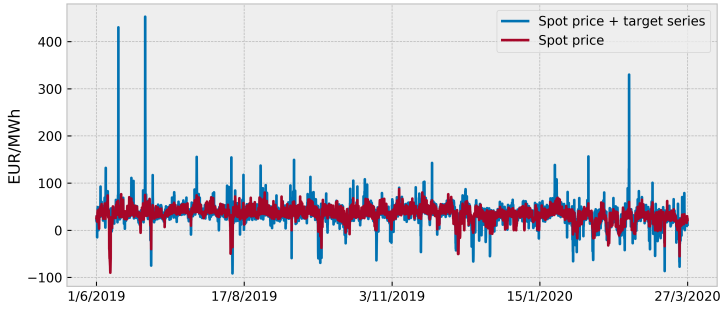

Figure 4.18: Spot price, $DA^{d,h}$, in red vs. intraday prices, $DA^{d,h} + y^{d,h}$, in blue.

We can empirically validate this by plotting in Figure 4.19 the average absolute spread for every 10-quantile in the spot price. More extreme spot prices at the outer quantile levels tend to show, on average, more extreme price spreads too.

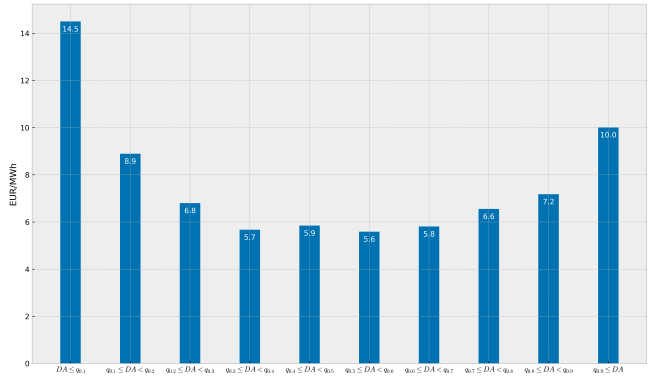

Figure 4.19: Average absolute spread for every deciquantile in the spot prices.

4.3.8 Merit-order response

Recall from section 2.5 that the aggregated generation capacities from the day-ahead market are transparent data from the submitted bid-offers. We can engineer two simple price features based on sensitivity analysis on the submitted bid-offer curve for a specific hour. The general idea is that the merit-order effect determines the electricity price, not just the spot price, but also the intraday price when there is an over- or undersupply of power compared to the forecast made at spot time. One way to model the merit-order is through the supply side from the submitted bid-offers.

Originally the market-clearing price is determined by the intersection between the supply bid-offers and demand for different power volumes. We can traverse the bid-offer curve using the sum of errors since spot, i.e., utilize $AE_q^{d,h}$ to hypothetically determine a slightly different market clearing price (assuming that the market price is sensitive to this change of supply). Moreover, we assume that only half of $AE_q^{d,h}$ is traded on exchanges.

The latter assumption is made because not all power is traded on exchanges, but rather through bilateral contracts. The share of power traded bilaterally varies greatly, and finding the exact amount is a complex and non-linear problem that likely depends on many latent factors. Ideally, this share should rather be dynamically determined, but that is outside the scope of this study. Our choice of picking half of the sum of errors stems from the fact that the prices of unobserved bilateral agreements are not deviating from exchange prices systematically because market participants have the option to trade either on the exchange or bilaterally. If the prices were very different, it would be possible to profit from pure arbitrage [Pape et al., 2016]. As a consequence, if prices on either platform are reasonably similar, then the simplest assumption we possibly can make is to think that half of the power volumes are traded on exchanges (or more specifically, the EPEX day-ahead exchange in Germany).

Note that we only have prices in the granularity of 50 MWh/h increments. So, for example, if there is a net error of 70 MWh/h, we linearly calculate with a factor of 0.4 times the first price increment and a factor of 0.6 times the second price increment.

The concept of following the bid-offer curve is best illustrated as in Figure 4.20. Say the market-clearing price for hour h on day d from the day-ahead market is $DA^{d,h}$ EUR/MWh for a total power supply offering of X MWh/h. Now assume that $AE_8^{d,h} = V$, i.e., a net undersupply of V MWh/h 8 quarters before delivery compared to what was forecasted since spot. We traverse the bid-offer curve with V/2 MWh/h to find a new price, $DA^{d,h} + t$.

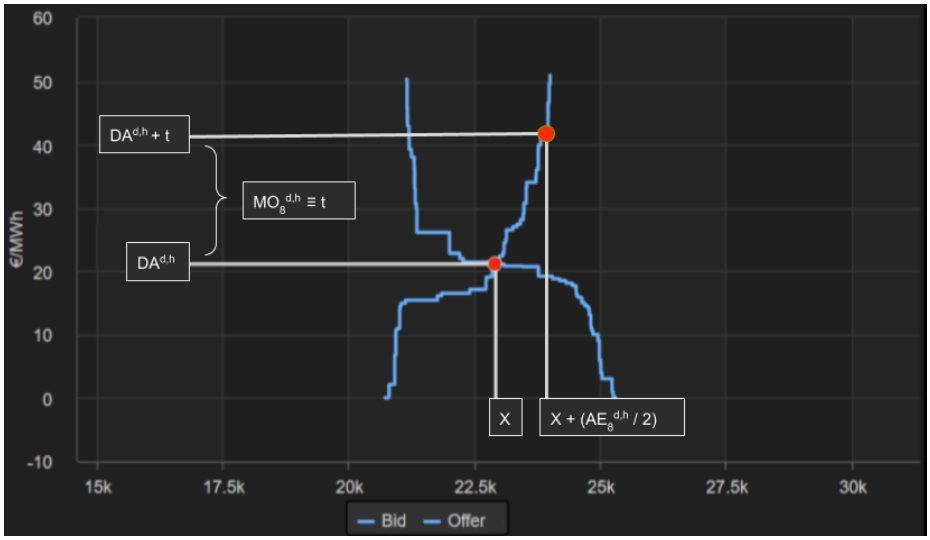

Figure 4.20: Traversing the bid-offers with half of the sum of errors since spot in order to construct the $MO_q^{d,h}$ feature.

Our first feature is the price difference between the actual settled spot price $DA^{d,h}$ and the traversed bid-offer price $DA^{d,h} + t$. The goal with this is to incorporate the response of merit-order effect, and we hence refer to this variable as $MO_q^{d,h}$:

$$MO_q^{d,h} = t$$

This feature quantifies how the supply side might value the change coming from updated values, relative to the original market-clearing price. We also capture the response from the merit-order effect between two consecutive hours:

$$\Delta MO_q^{d,h} = MO_q^{d,h} - MO_q^{d,h-1}$$

The justification for including the latter is as follows: if the net errors since spot are low, but $MO^{d,h}$ is high, this signals that the system likely is in a steep part of the merit-order curve. A rapid increase in price terrain indicates, for example, ramping effects from the involvement of new power plants, and is an important determinant for the intraday price.

There are 49 hours, that is, ca 0.7 percent of the total data, where we have no bid-offer curve. Imputing merit-order effects is difficult. To avoid this problem,

we choose to remove the corresponding rows altogether in the dataset.

4.3.9 System imbalance

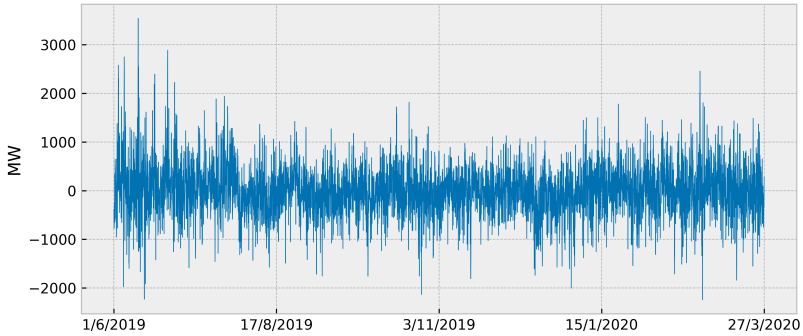

Figure 4.21: System imbalances, $SI_9^{d,h}$.

The deviation between the scheduled and the actual physically transmitted power in real-time are called imbalances. Some market participants might produce more than scheduled, while others might produce less than scheduled. Positive and negative imbalances offset each other, and the transmission system operators have to compensate for the net position, which is the system imbalance [Koch, 2019].

In Germany, the imbalance settlement period is 15 minutes. The system imbalance is released to the public after some time; [Maskos, 2017] finds that German system operators regularly publish the past system balance ca 10 minutes after settlement. For our task, in order to give a practitioner one hour to make the forecast and decide upon strategies for the final hour, we take the conservative approach being that the latest imbalance, $SI_q^{d,h}$, that we can use is the one concerning the settlement period 2 hours and 15 minutes prior to delivery: $SI_9^{d,h}$. $SI_9^{d,h}$ have a correlation coefficient of 0.25 with the target series and is visualized in Figure 4.21.

This lagged value is not available for the first 3 hours in our dataset. Instead of trying to impute them, we remove the corresponding rows altogether in the dataset.

4.3.10 Latest available transaction price

The latest available price for a product, which we refer to as the naïve transaction price, is simply $p_8^{d,h}$. As can be seen from Figure 4.22, the naive transaction price is regularly in the same direction as the target series, but often less extreme. The correlation coefficient between the two series is 0.495.

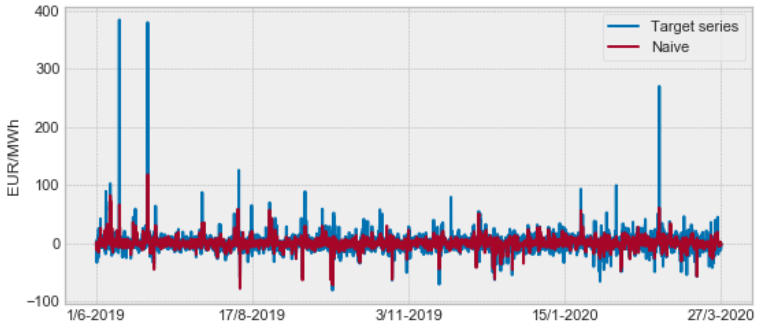

Figure 4.22: Illustration of the target series $y^{d,h}$ relative to the naive transaction price $p_8^{d,h}$.

4.3.11 Neighboring products

Some studies have shown that considering time-series information from neighboring products increases predictive abilities [Janke and Steinke, 2019] [Marcjasz et al., 2020]. This is motivated by the fact that similar market information drives adjacent contracts [Kremer et al., 2019].

When our target price is $y^{d,h} \equiv \frac{p_1^{d,h}+p_2^{d,h}+p_3^{d,h}+p_4^{d,h}}{4}$, it is very feasible to include in the feature set, not only the lags of prior targets, or prior transaction prices (e.g. naïve) for the same target, but also for example

$$
\begin{aligned}
{}_{-1}NP_8^{d,h} &\equiv \frac{p_5^{d,h-1} + p_6^{d,h-1} + p_7^{d,h-1} + p_8^{d,h-1}}{4} \\
{}_{+1}NP_8^{d,h} &\equiv \frac{p_{13}^{d,h+1} + p_{14}^{d,h+1} + p_{15}^{d,h+1} + p_{16}^{d,h+1}}{4}
\end{aligned}
$$

With this notation, ${}_iNP_q^{d,h}$ is the average price spread sampled quarterly in the latest available trading hour for the neighboring product i hours before the

referenced product with delivery at hour h on day d, q quarters before delivery.

In simpler terms, these two features are the average price between hours 1 and 2 prior to delivery for the previous product and the average price between hours 3 and 4 prior to delivery for the next product.

The very first hour in the dataset does not have a previous product (but this case is already removed from missing the lagged system imbalance). Furthermore, the latest hour in the dataset does not have a next product. In this case, we patch with the prices from the previous hour.

4.4 Summary of Features

Here we summarize the variables with a simplified description, report on elementary statistics and imputation methods.

	Simplified description
$\hat{WI}^{d,h}$	Wind production forecast made at spot time
$\hat{SO}^{d,h}$	Solar production forecast made at spot time
$\Delta WI_8^{d,h}$	Wind production forecast error 8 quarters prior to delivery
$\Delta SO_8^{d,h}$	Solar production forecast error 8 quarters prior to delivery
$\hat{CO}^{d,h}$	System wide consumption forecast made at spot time
$\Delta CO_8^{d,h}$	System wide consumption forecast error 8 quarters prior to delivery
$\Delta AV_8^{d,h}$	Change in available production capacity 8 quarters prior to delivery
$\Delta WE_8^{d,h}$	Sum of wind, solar and consumption forecast errors 8 quarters prior to delivery
$\Delta AE_8^{d,h}$	Sum of wind, solar, consumption forecast errors and changes in available production capacity 8 quarters prior to delivery
$DA^{d,h}$	Spot price from the day ahead
$MO_8^{d,h}$	Traversed bid-offer curve with intradaily updated values to obtain merit-order response
$\Delta MO_8^{d,h}$	Change in merit-order response from one hour to the next
$SI_9^{d,h}$	System imbalance 9 quarters prior to delivery
$p_8^{d,h}$	Transaction price recorded 8 quarters before delivery
${}_{-1}NP_8^{d,h}$	Average price between hour 1 and 2 prior to delivery for the previous product
${}_{+1}NP_8^{d,h}$	Average price between hour 3 and 4 prior to delivery for the next product

Table 4.4: Simplified description of the input features.

	mean	**std**	**min**	**25%**	**50%**	**75%**	**max**
$\hat{WI}^{d,h}$	15358.2	10946.5	566.6	6436.6	12618.8	22212.2	48993.3
$\hat{SO}^{d,h}$	4555.4	6819.5	0.0	0.0	209.0	7286.0	29562.0
$\Delta WI_8^{d,h}$	-73.1	1183.0	-6990.0	-698.7	-61.8	519.7	7709.2
$\Delta SO_8^{d,h}$	-8.7	323.3	-4832.7	-55.6	0.0	4.6	2600.5
$\hat{CO}^{d,h}$	63365.9	10990.3	37223.0	54263.0	63449.0	72092.0	85553.0
$\Delta CO_8^{d,h}$	-5.8	185.7	-1070.0	-95.0	-7.0	70.0	1183.0
$\Delta AV_8^{d,h}$	370.4	688.0	-1452.0	0.0	20.0	667.1	5091.5
$\Delta WE_8^{d,h}$	-103.0	1301.9	-6936.9	-812.5	-83.3	585.5	7522.0
$\Delta AE_8^{d,h}$	267.4	1484.3	-6939.9	-596.1	208.3	1097.2	8225.0
$DA^{d,h}$	33.9	15.5	-90.0	27.5	35.2	43.1	87.1
$MO_8^{d,h}$	1.0	5.1	-32.6	-1.0	0.2	2.7	89.5
$\Delta MO_8^{d,h}$	0.0	1.6	-26.7	-0.3	0.0	0.3	21.7
$SI_9^{d,h}$	0.4	513.1	-2239.9	-300.7	-3.8	292.2	3542.4
$p_8^{d,h}$	0.3	8.8	-78.0	-3.2	0.1	3.4	118.1
${}_{-1}NP_8^{d,h}$	0.3	9.5	-84.8	-3.3	0.0	3.4	233.3
${}_{+1}NP_8^{d,h}$	0.4	6.8	-57.0	-2.5	0.1	2.9	76.1

Table 4.5: Descriptive statistics for the input features.

	Missing values	**Imputation method**
$\hat{WI}^{d,h}$	0	
$\hat{SO}^{d,h}$	0	
$\Delta WI_8^{d,h}$	2.8 %	Forward fill
$\Delta SO_8^{d,h}$	2.8 %	Forward fill
$\hat{CO}^{d,h}$	0	
$\Delta CO_8^{d,h}$	3%	Forward fill
$\Delta AV_8^{d,h}$	2.8 %	Forward fill
$\Delta WE_8^{d,h}$	0	
$\Delta AE_8^{d,h}$	0	
$DA^{d,h}$	0	
$MO_8^{d,h}$	0.7 %	Remove rows
$\Delta MO_8^{d,h}$	0.7 %	Remove rows
$SI_9^{d,h}$	0.05 %	Remove rows
$p_8^{d,h}$	0	
${}_{-1}NP_8^{d,h}$	0.01 %	Remove rows
${}_{+1}NP_8^{d,h}$	0.01 %	Forward fill

Table 4.6: Missing values in the features.

Chapter 5

Models

This chapter covers the baselines and proposed models. The baselines are defined in Section 5.1, and are either statistical linear regression-based models or single variable values with no model built on top of it. Further, the proposed neural networks are described in Section 5.2. Finally, Section 5.3 explains how the study is extended by averaging different forecasts.

5.1 Baseline Models

5.1.1 Naïve

This baseline simply outputs the latest available transaction price for the target product, that is the naïve price:

$$\hat{y}^{d,h} = p_8^{d,h}$$

5.1.2 MOR

The second baseline predicts the target price by traversing the bid-offer curve with the sum of the latest available forecast errors and uses this Merit-Order Response (MOR) as the suggested target price. In other words:

$$\hat{y}^{d,h} = MO_8^{d,h}$$

Again, the fundamental idea behind this is that the merit-order-effect determine the electricity price, not just the spot price, but also the intraday price when there

is an over- or undersupply of power compared to what was originally forecasted at spot time.

5.1.3 LASSO

With this, we try to create a similar[1] model to the so-called "full model" in [Marcjasz et al., 2020], that perhaps has shown the most convincing results in the visited literature. It utilizes a rich set of input variables:

$$\begin{aligned}
\hat{y}^{d,h} = &\underbrace{\beta_0}_{\text{bias}} + \underbrace{\sum_{i=2}^{25} \beta_{i-1} \cdot y^{d,h-i}}_{\text{lagged targets}} + \underbrace{\sum_{i=1}^{24} \beta_{24+i} \cdot DA^{d,i}}_{\text{spot prices for today}} + \mathbb{1}_{h \geq 14} \underbrace{\sum_{i=1}^{24} \beta_{48+i} \cdot DA^{d+1,i}}_{\text{spot prices for tomorrow}} \\
&+ \underbrace{\sum_{i=1}^{7} \beta_{72+i} \cdot D^{d}}_{\text{day-of-week dummies}} + \underbrace{\sum_{i=1}^{24} \beta_{79+i} \cdot \hat{CO}^{d,i}}_{\text{consumption forecasts}} + \underbrace{\sum_{i=1}^{24} \beta_{103+i} \cdot \hat{WI}^{d,i}}_{\text{wind forecasts}} + \underbrace{\sum_{i=1}^{24} \beta_{127+i} \cdot \hat{SO}^{d,i}}_{\text{solar forecasts}} \\
&+ \underbrace{\sum_{i=2}^{23} \beta_{151+i} \cdot \frac{\sum_{j=4i}^{4i+4} \Delta CO_j^{d,h}}{4}}_{\text{hourly consumption forecast errors}} + \underbrace{\sum_{i=2}^{23} \beta_{173+i} \cdot \frac{\sum_{j=4i}^{4i+4} \Delta WI_j^{d,h}}{4}}_{\text{hourly wind forecast errors}} \\
&+ \underbrace{\sum_{i=2}^{23} \beta_{195+i} \cdot \frac{\sum_{j=4i}^{4n+i} \Delta SO_j^{d,h}}{4}}_{\text{hourly solar forecast errors}} + \underbrace{\sum_{i=9}^{11} \beta_{210+i} \cdot SI_i^{d,h}}_{\text{imbalance volumes}} \\
&+ \underbrace{\beta_{221} \cdot_{-1} NP_8^{d,h}}_{\text{prior product price info}} + \underbrace{\beta_{222} \cdot_{+1} NP_8^{d,h}}_{\text{next product price info}} + \underbrace{\beta_{223} \cdot p_8^{d,h}}_{\text{naïve}} + \underbrace{\epsilon^{d,h}}_{\text{error term}}
\end{aligned}$$

This model incorporates a few new things. Firstly, spot prices for tomorrow are included in the set of regressors when forecasting hours 14 to 24. This is possible because spot prices for the next day should be available at 12 PM, and we make the forecast 2 hours prior to delivery. Secondly, dummy variables per weekday are included to directly account for weekly seasonality. Also, note that this model takes as input not only the forecasts for the designated target hour but for every hour on that day. Additionally, hourly forecast errors are accounted for as long back in time as we have data, that is back until 24 hours before delivery.

[1] It is not identical. Note that [Marcjasz et al., 2020] predict the ID3 price and we predict a target series constructed from the prices in the final hour before delivery. Other slight differences in the variables exists.

5.1.4 OLS

This baseline is also a linear regression model, but is built only by minimizing the residual sum of squares. Less input variables are used compared to the LASSO. The input variables are the features previously summarized in Table 4.4 plus the target price for the previous hour:

$$\begin{aligned}\hat{y}^{d,h} = &\underbrace{\beta_0}_{\text{bias}} + \underbrace{\beta_1 \cdot y^{d,h-1}}_{\text{lagged target}} + \underbrace{\beta_2 \cdot DA^{d,h}}_{\text{spot price}} + \underbrace{\beta_3 \cdot \hat{CO}^{d,h}}_{\text{consumption forecast}} + \underbrace{\beta_4 \cdot \hat{WI}^{d,h}}_{\text{wind forecast}} \\ &+ \underbrace{\beta_5 \cdot \hat{SO}^{d,h}}_{\text{solar forecast}} + \underbrace{\beta_6 \cdot \Delta CO_8^{d,h}}_{\text{consumption forecast error}} + \underbrace{\beta_7 \cdot \Delta WI_8^{d,h}}_{\text{wind forecast error}} + \underbrace{\beta_8 \cdot \Delta SO_8^{d,h}}_{\text{solar forecast error}} \\ &+ \underbrace{\beta_9 \cdot \Delta WE_8^{d,h}}_{\text{sum of errors from weather-related variables}} + \underbrace{\beta_{10} \cdot \Delta AE_8^{d,h}}_{\text{Same as } \Delta WE_8^{d,h}\text{, plus changes in availability}} \\ &+ \underbrace{\beta_{11} \cdot SI_9^{d,h}}_{\text{imbalance volume}} + \underbrace{\beta_{12} \cdot_{-1} NP_8^{d,h}}_{\text{prior product price info}} + \underbrace{\beta_{13} \cdot_{+1} NP_8^{d,h}}_{\text{next product price info}} + \underbrace{\beta_{14} \cdot p_8^{d,h}}_{\text{naïve}} \\ &+ \underbrace{\beta_{15} \cdot MO_8^{d,h}}_{\text{Merit order response}} + \underbrace{\beta_{16} \cdot \Delta MO_8^{d,h}}_{\text{Change in merit order response}} + \underbrace{\epsilon^{d,h}}_{\text{error term}}\end{aligned}$$

5.2 Proposed Models

5.2.1 FFNN

The first proposal is a standard fully-connected FFNN. The model is visualized in Figure 5.1, with an input layer, In, of size S_{in}, which is equal to the number of input features. It is then followed by n number of dense layers, D_i for $i \in [1, n]$ with size S_{D_i}, all with the same activation function a and dropout rate d. The final output layer, O, is of size 1 and represents $\hat{y}^{d,h}$. This is a general explanation of the model, and Table 5.1 summarizes the hyperparameters that need to be specified. The exact model specification will later be found through excessive validation search, see Section 6.2. This model takes as input the same features as the OLS, also described in Table 4.4.

Hyperparameter	Description
S_{in}	Size of the input layer, In
n	Number of dense layers
S_{D_i}	Size of dense layer D_i
a	Activation function all dense layers
d	Dropout rate in all dense layers

Table 5.1: Description of hyperparameters that need to be specified for the proposed FFNN

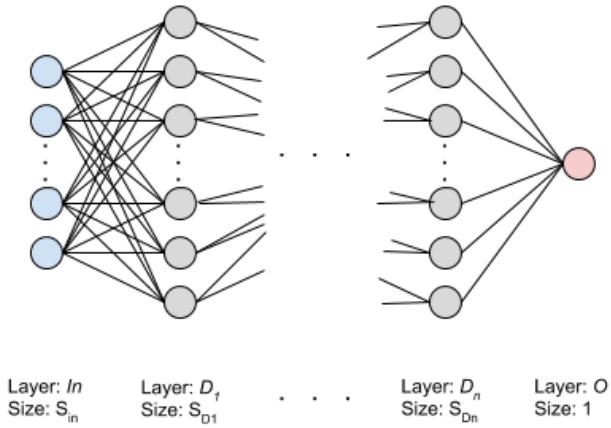

Figure 5.1: Proposed FFNN

5.2.2 LSTM and GRU

We also propose models with two variations of recurrent layers, namely LSTM and GRU. The models are fed with the same conceptual features as the FFNN but differ as more historical data will propagate through the recurrent layer(s), and not just the most recent value.

The sequential models are designed to better deal with the serial dependence that is in the time series of fundamentals that have evolved between spot time and the delivery hour. It is not apparent if only the most recent fundamental values or, if also old values from many hours ago to some extent, dictate future prices. If any trends since spot time determine the final intraday prices, these types of patterns are likely captured in these sequential models. For example,

cases where wind forecast errors gradually increase since spot time could affect prices differently than if the error appears more suddenly before gate closure.

We, therefore, split the features into sequential features, SF, and other features, OF, as described in Table 5.2. SF then contains the features where we also add historical data. We use 12 hours of historical data. Since the resolution of historical data is quarterly, the number of added values in the history is 48. For a feature $V_q^{d,h}$, the recorded values then correspond to $V_{56}^{d,h}$ to $V_8^{d,h}$. The only exception is $SI_q^{d,h}$, where $SI_9^{d,h}$ is the latest value we use and is thus shifted a quarter earlier. Hence, the size of SF is $(48, N_{SF})$, where N_{SF} is the number of sequential features used. N_{OF} represents the number of other features.

Sequential features	**Other features**
$\Delta WI_q^{d,h}$	$\hat{WI}^{d,h}$
$\Delta SO_q^{d,h}$	$\hat{SO}^{d,h}$
$\Delta CO_q^{d,h}$	$\hat{CO}^{d,h}$
$\Delta AV_q^{d,h}$	$DA^{d,h}$
$\Delta WE_q^{d,h}$	${}_{-1}NP_8^{d,h}$
$\Delta AE_q^{d,h}$	${}_{+1}NP_8^{d,h}$
$MO_q^{d,h}$	$y^{d,h-1}$
$\Delta MO_q^{d,h}$	
$SI_q^{d,h}$	
$p_q^{d,h}$	

Table 5.2: Sequential features and other features.

The model architecture is presented in Figure 5.2. SF and OF are fed into two different input layers, producing O_{SF} and O_{OF}, where both output sizes are determined by the value of the latent vector l. The function of the input layers is to give the model the chance to project the data into a parameter space that is easier for a network to find patterns in. O_{SF} then propagates through the recurrent layer, where the output (O_{rnn}) corresponds to the final hidden state, as illustrated in Figure 5.3 (a). This behavior assumes that N_{layers} equals 1. If recurrent layers are stacked ($N_{layers} >1$), the final output is then the sum of each layer's last hidden state. This is visualized in Figure 5.3 (b). Recall from Section 2.2 that the final hidden state is the vector representation after the last value in the sequence has propagated through the recurrent layer.

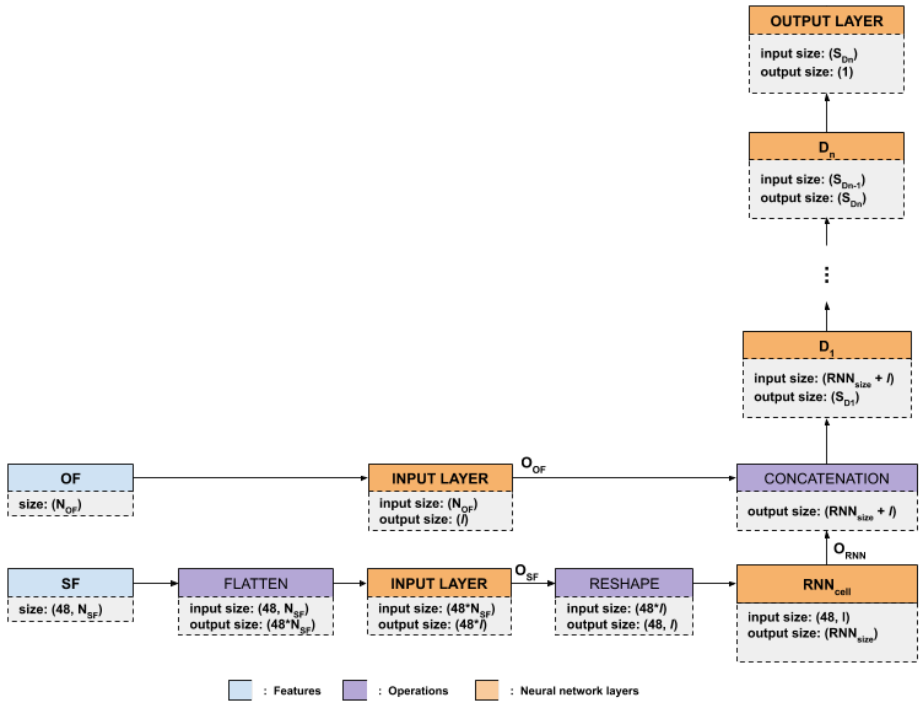

Figure 5.2: Proposed RNN

Further, we concatenate O_{rnn} and O_{OF} and feed it through a specified number of dense layers activated with a. The goal with this part is to find any non-linearities in the combinations of sequential and other features. Processing the data through multiple layers can be necessary if the concatenated features require embedding into some feature space where patterns can get learned more effectively. A complex multi-layered function might also be required if the non-linearities are very complex. The network should automatically find the appropriate number of layers and is thus among the hyperparameters that can be specified.

Finally the data is propagated to the output layer to get $\hat{y}^{d,h}$. Hyperparameters that need to be specified are summarized in Table 5.3.

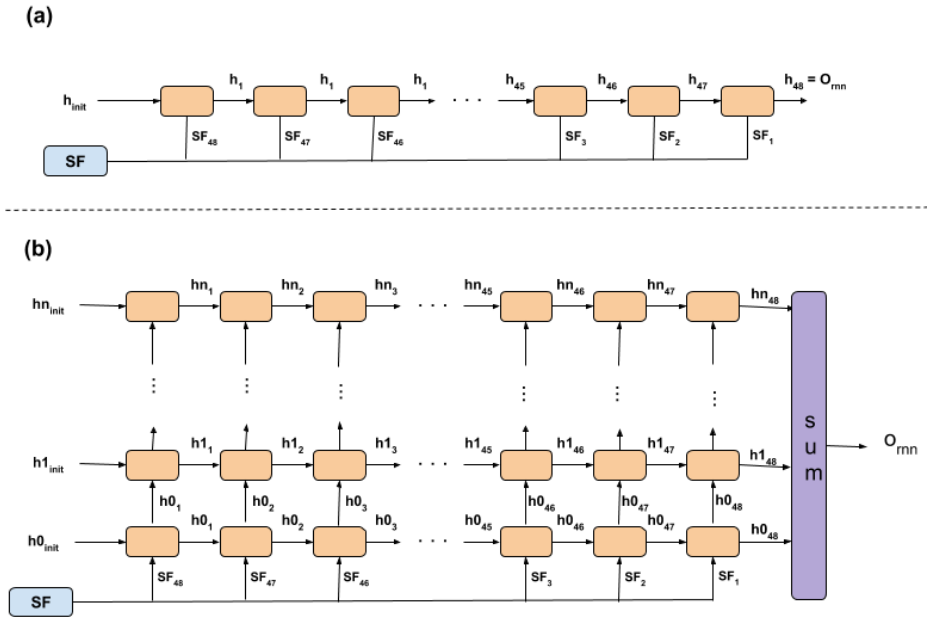

Figure 5.3: Data propagation through RNN. If $N_{layers} = 1$ then (a) describes the final output. If $N_{layers} > 1$ then (b) describes the final output.

Hyperparameter	**Description**
l	Size of latent vector
n	Number of dense layers
S_{Di}	Size of dense layer D_i
a	Activation function in input and dense layers
d	Dropout rate in dense layers
N_{layers}	Number of stacked recurrent layers
RNN_{cell}	Type of recurrent layer. Either LSTM or GRU
RNN_{init}	Cell/hidden state initialization method
RNN_{size}	Size of the cell/hidden state

Table 5.3: Description of hyperparameters that need to be specified for proposed GRU/LSTM

5.3 Averaging Forecasts

From Section 2.5, we know that many factors jointly impact the final intraday prices and that our dataset only has systematized a subset of the many determinants. The most recently available transaction price is thus expected to carry some latent information that otherwise is not available elsewhere in our data. A straightforward extension to our work is to take a simple average of the naïve transaction price and the predicted $\hat{y}^{d,h}$ outputted from the models:

$$ens(m(X^{d,h})) = \frac{1}{2}m(X^{d,h}) + \frac{1}{2}p_8^{d,h}$$

With this notation, $m(X^{d,h})$ is the $\hat{y}^{d,h}$ outputted from either the MOR, LASSO, OLS, FFNN, LSTM or GRU model.

Averaging forecasts' possible effects on accuracy gains for spot and intraday price prediction is known in the EPF literature [Nowotarski and Weron, 2015] [Marcjasz et al., 2020]. Because minimal extra engineering effort is required to incorporate this, we include it in our work.

Chapter 6

Experimental Setting

This chapter begins by explaining how we plan to go about the experiments and what research questions the experiments aim to answer. Section 6.2 then covers the setup for how we thoroughly train, validate and determine the final configurations of the models.

6.1 Experimental Plan

The objective is now to predict the final intraday prices. With that, we especially want to research the effect that exogenous factors have on the predictable prices, and how recurrent neural networks compare to feed-forward networks.

To study the effect of exogenous factors, we choose to make the forecasts in two separate stages:

Stage 1 No intraday transaction prices are used in the feature set. That accounts for prior targets, naïve transaction prices, available prices from neighboring products, or any other variables constructed from preceding intraday prices.

Stage 2 All features can be included.

By comparing the R^2 scores from the two stages on the test data, we will be able to explore the difference in proportion of variance that can be predicted with and without the involvement of prior intraday prices. R^2 is a metric that statistically shows the proportion of variance in the dependent variable that is predictable

from the independent variables. It is calculated as the accuracy of the predicted values relative to a horizontal line through the mean of the target series:

$$R^2 \equiv 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

However, to better assess the differences between the models and their outputted forecasts, comparing R^2 scores is not sufficient. Because of the spikes that can occur in electricity price time series, metrics that squares the residuals will be very sensitive to what the model outputs for these spiky points, and conservative predictions with price spreads close 0 are often rewarded. Thus, we also report the mean absolute errors (MAE) and the percentage of times the prediction has the same sign as the true observation, as these metrics are more robust to outliers. The correct sign metric is simply the accuracy of times the predicted value has the same sign as the actual observation, and MAE is calculated as:

$$MAE \equiv \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Note that we predict the price spread between the day-ahead prices and intraday prices. Therefore we report the results on these deltas. Evaluation of the full back-transformed prices (that is the sum of the spot price and predicted deltas) is available in Appendix D.

The test covers data from February and March 2020.

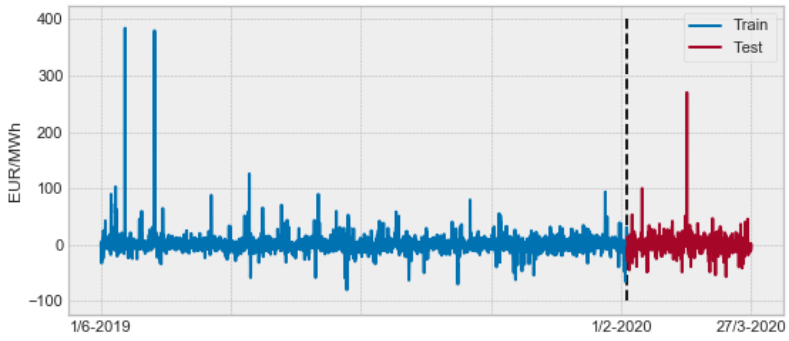

Figure 6.1: Initial train-test split.

The second research question is about comparing recurrent networks to feed-forward neural networks for this predictive task. In order to answer that, we apply the metrics as mentioned earlier to judge model performance. Moreover, we evaluate the significance of outperformance of different forecasts obtained with the Diebold-Mariano test, as described in Section 2.4. This test is not designed to compare model designs, but says something about the significance of different forecast results. We use the modifications of the test proposed by [Harvey et al., 1997], formulated as a two-step ahead forecast and mean absolute deviation as the chosen loss criterion[1].

Furthermore, to obtain an even better understanding of the models' true performance under different conditions, the metrics are reported when examining only specific subsamples of the test data. Concretely, we filter the test-data not to include spikes (we define spikes to be when the target lies outside the 5 and 95 percentile levels). Likewise, we look into the predictive skills of the models when fundamental inputs are at its outer percentile levels. The results for outer percentile levels in stage 1 and stage 2 are placed in Appendix C, because they are many.

Further discussion of model complexity, learning behavior, and statistical properties of the forecasted series is included in the comparison of models to assess their relative strengths and weaknesses for the learning task at hand.

6.2 Experimental Setup

This section explains how the models will be trained, how we search for optimal hyperparameters, and other implementational details that are required for reproducibility purposes.

In modern ML in practice, arguably more important than the exact model choice is how the parameters of the model are tuned. Sophisticated models have many tunable parameters, i.e., hyperparameters, and the default ones seldom work optimally for the specific task at hand. Instead, they must be optimized for the current task.

As we are proposing three models (FFNN, GRU, and LSTM) and are doing the experiments in two stages, we optimize hyperparameters for six models in total. In Section 5.2, we presented the tunable hyperparameters for each model. In addition to these, some hyperparameters that are not model-specific also require specification: The optimizer, batch size, learning rate, and the loss function.

[1]We use the open-source code in Python made available by John Tsang: https://github.com/johntwk/Diebold-Mariano-Test

We have directed the strategy for finding the best hyperparameters in two steps; first a grid search, followed by a forward chained validation. In the grid search, a large space of hyperparameter values is systematically explored on a single validation set with the objective of evaluating the effect of different loss functions, dropout rates, batch sizes, and model structures. The grid search builds a model for each parameter combination possible. Based on the findings from this, we proceed with the forward chained validation. The search space will be reduced with approximately an order of magnitude after the grid search, and the final hyperparameters for each model type are determined by validating more thoroughly on more of the data in the forward chain.

However, since this type of optimization is a very time-consuming process[2], and the hyperparameter value space is incredibly large, some hyperparameters will have to be set to a fixed value and not taken into account during the two steps.

All neural networks are implemented in Pytorch, an open-source machine learning framework [Paszke et al., 2019]. Our networks are composed of Linear (Dense), GRU, and LSTM layers with weights initialized with default Pytorch parameters.

6.2.1 Grid search

In the grid search, we first split the training data into a separate training and validation set. The training set consists of the data from 01/06/2019 to 31/12/2019, while the validation set consists of the data from 01/01/2020 to 31/01/2020. Then, for each hyperparameter combination, we train our models on 100 epochs for the FFNN and 70 epochs for the GRU and LSTM. The recurrent models are trained on fewer epochs because we see that they more quickly overfit due to model complexity, i.e., from having a richer set of model parameters. Then, for each epoch, we track the MAE on the validation set, and the lowest MAE gained during the training is defined as the score of the hyperparameter combination. This concept is visualized in Figure 6.2. By evaluating the different combinations and compare their scores against others, we can get useful insights into the hyperparameters that perform better on the chosen evaluation metric.

Minimizing MAE is chosen as the optimization criterion because it, from trial and error, seems to be more robust than other alternatives.

Table 6.1 presents hyperparameters included when grid searching the FFNN, while Table 6.2 presents the hyperparameters tested in the GRU and LSTM grid search. Since GRU and LSTM have similar architectures they are also tested with the same values. For the RNN_{init} we are testing three initialization methods, all described in Table 6.3.

[2]We train with a Tesla P100-PCIE-16GB GPU

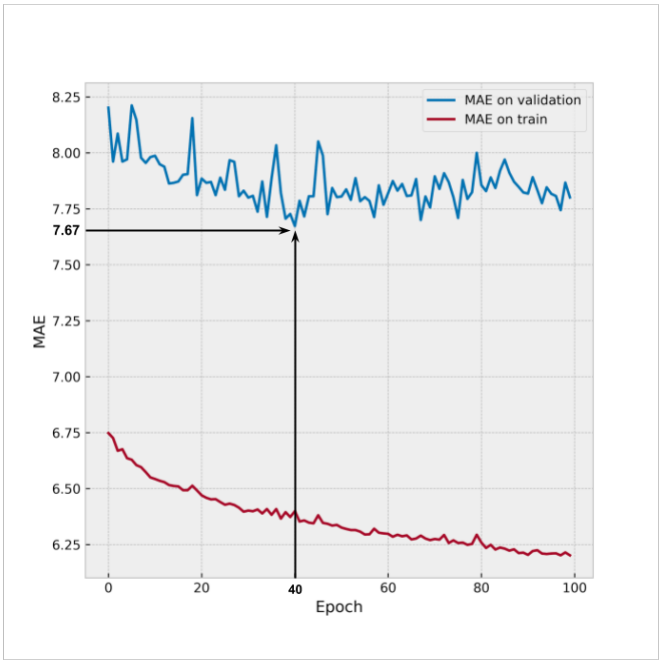

Figure 6.2: Example on how hyperparameters are evaluated. In this case, the model got the lowest MAE on epoch 40, with a value of 7.67, which is then set to be the score of the hyperparameter combination.

Batch Size	**Loss Function**	**Learning rate**	**d**	**$S_{D1};...;S_{Dn}$**
32	SmoothL1Loss	0.001	0	16
64	L1Loss	0.0001	0.25	64
128	MSE		0.50	128
				16;16
				64;16
				64;64
				128;128
				16;16;16
				64;64;64
				12;64;16
				128;128;128

Table 6.1: Tested hyperparameters in the FFNN grid search.

Batch size	**Loss function**	**Learning rate**	**d**	**l**
32	SmoothL1Loss	0.001	0	5
64	L1Loss	0.0001	0.25	25
	MSE		0.50	

$\mathbf{N_{layers}}$	$\mathbf{RNN_{size}}$	$\mathbf{RNN_{init}}$	$\mathbf{S_{Di};...;S_{Dn}}$
1	64	zero	128
2		random	128; 128
		learn	128; 128; 128

Table 6.2: Tested hyperparameters in the GRU/LSTM grid search.

Initialization method	**Description**
zero	Parameters in the hidden/cell state are initialized to zero distribution
random	Parameters in the hidden/cell state are initialized to random numbers sampled from the normal distribution
learn	Parameters in the hidden/cell state are first xavier initialized, then updated during backpropagation

Table 6.3: Initialization methods in the GRU/LSTM.

Findings from the grid search

Overall, it was not trivial to distinguish between the results in the grid search. The differences were small, due to the fact that the dataset undoubtedly is noisy and complex. However, there were some indications that certain hyperparameters produced more robust results than others.

For the FFNN, the effect of dropout is hard to evaluate solely by looking at the MAE scores. Models without dropout produced just as good results as other models, but models with no dropout showed a more unstable learning behavior throughout the epochs. In Figure 6.3 (a) and (b), we are plotting the train and validation learning of two models with the same hyperparameters, except for the dropout rate. We can see that the learning results seem to be more robust by adding dropout. When it comes to the structure of the dense layers, one layered network with a size of either 64 or 128 tended to yield better results. A fixed-size of 100 epochs was not enough for networks trained with a learning rate of 0.0001, but as the simpler models performed best and their training time is fast, a low

learning rate seemed advantageous. From what we could see, no loss function and batch size had any results significantly different from others.

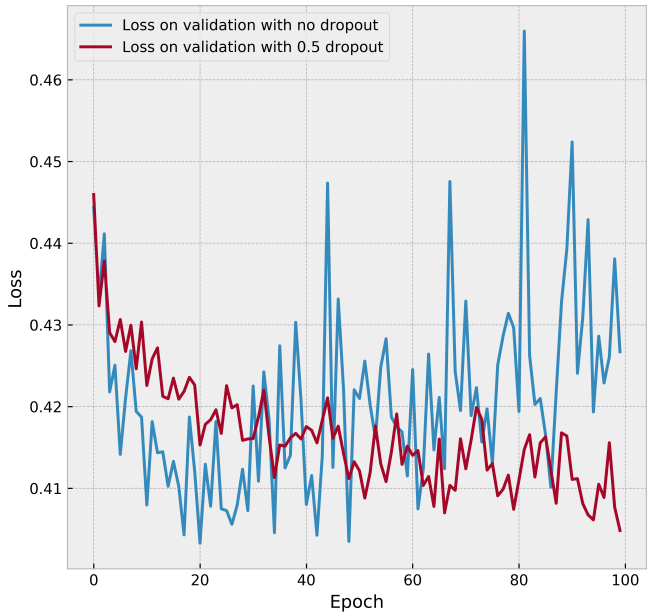

Figure 6.3: Dropout effect in the FFNN. Both models are trained with the same hyperparameters, except for the dropout rate, where blue is trained with no dropout and red with a dropout rate of 0.5. We can clearly see that the dropout yields more robustness.

It was also difficult to come to a conclusion on which parameters that worked best for the GRU and LSTM. These models are more complex than the FFNN, with many more knobs to turn and tweak. This was noticeable as the models, regardless of the hyperparameters, started to overfit on the train set from an early epoch. Figure 6.4 visualizes a typical learning curve for these models. Fortunately, there are hints of learning in the first epochs, and that might be enough to produce good results if we can stop the training at the right time. Due to the overfitting issues of the LSTM and GRU, a low learning rate with a high dropout seemed to be advantageous. When it comes to the other parameters, it

was again difficult to distinguish between their relative performances.

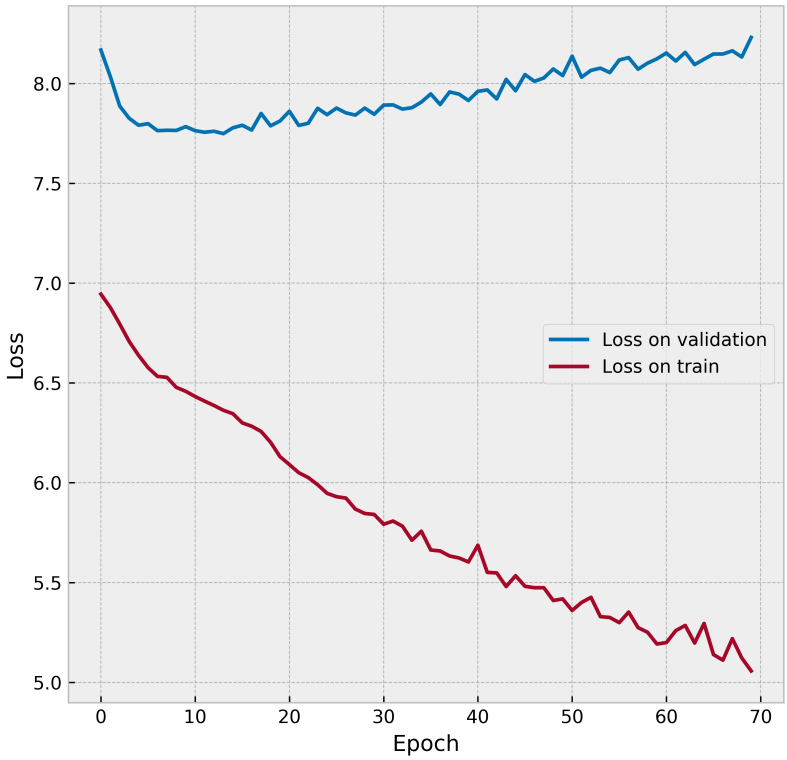

Figure 6.4: Typical GRU/LSTM learning curves in the grid search. The Figure is taken from the GRU grid search for stage 1.

Based on these findings, a smaller hyperparameter space was further explored in the forward chained validation. Since we did not seem to improve performance by using a specific loss function or batch size, we set the batch size to 32 and SmoothL1Loss as the chosen loss function. In PyTorch, SmoothL1Loss is formulated as:

$$SmoothL1Loss(\hat{y}, y) = \frac{1}{n}\sum_{i=i}^{n} z_i$$

where n is the batch size and z_i is given by:

$$z_i = \begin{cases} 0.5 \cdot (\hat{y}_i - y_i)^2, & if|\hat{y}_i - y_i| < 1 \\ |\hat{y}_i - y_i| - 0.5, & otherwise \end{cases}$$

This loss function uses a squared term when the absolute error is below 1 to get more out of the gradients when the prediction is relatively close to the target. This can make sense for a target series that is closely distributed around zero mean. To prevent the gradients from being dominated by outliers, the error is not squared when it is above 1. This behavior can be preferable to avoid overfitting to extreme observations.

6.2.2 Forward chained validation

From Section 4.1, we know that we are dealing with a very noisy target series that can quickly change from short-term dynamics. The risks of overfitting will be huge when fitting a model with many parameters in such domains, as we already have seen. Because of having a relatively short dataset, we should probably not select the final hyperparameters based on the performance on a separate validation period, because that would imply a systematic bias towards the selected period of time. We know that electricity prices are determined from processes with seasonality on the annual level. Furthermore, it is natural to assume that the more data we can train on, the better the results.

The conventional way to solve these types of problems is to make use of cross-validation techniques. It is, however, risky to employ shuffled grouping of train and test data for time series data. A cross-validation strategy implicitly assumes that each observation is independent across the temporal dimension, but for time series there is usually a clear statistical dependence following the natural temporal order. In the EPF domain, future prices might, for instance, be influenced by new installations, changes in market design[3], or new expectations that were constructed among market participants. Training with "the future" in these rapidly evolving markets is a design flaw that potentially could yield a too optimistic estimate of the true test error.

[3]The German intraday market has undergone some major regulatory changes in the last several years. Since July 2017, it is possible to trade until 5 min before delivery inside of the control zones. Second, in October 2018, the Austrian control zone was split from the former German-Austrian market. [Janke and Steinke, 2019].

We therefore employ a forward chained validation technique instead. It works by selecting a number of chains where each chain gradually grows the size of the training set, whilst validating on some subsequent data. Each parameter combination then gets a score, which is made from the basis of the minimal MAE obtained throughout the training. This then proceeds for all of the chains. Figure 6.5 illustrates how we select 5 chains and assigns weights to the validation results proportional to the months involved in the training set. The score of each parameter combination is determined by the weighted sum of MAEs across all of the chains. The combination with the lowest aggregated MAE score will then be chosen as the hyperparameters for the final model. Although individual validation results in a chain are heavily biased towards the subsequent evaluated months, the final model will hopefully not, as its hyperparameters are obtained from weighting validation results across multiple validation months.

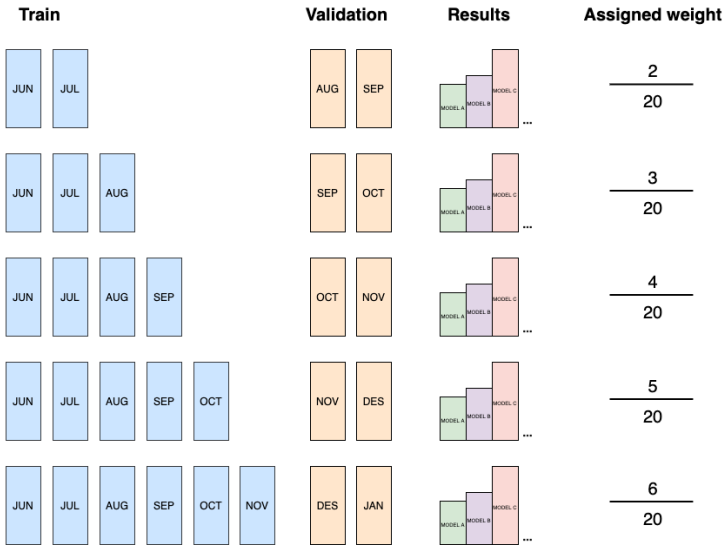

Figure 6.5: Forward Chaining. A new model is built for each hyperparameter combination and for each of the 5 chains. Chain number $n + 1$ is trained on an additional month of data compared to chain number n and then validated on the two subsequent months. The score of a model is after this determined from the weighted average of the minimal MAE obtained across the chains. Weights are assigned to the validation results proportional to the months trained.

Chains that are validated on more recent data is weighted more heavily for two reasons. Firstly, these chains have been trained on more data and are thus likely to have found more robust and generalizable patterns. Secondly, they have been

trained to fit more recent data better. It makes sense to emphasize recent data for time series forecasting. The data generating process might slightly evolve with time, as previously explained. If you fit a model too heavily with outdated data, the model might learn trends that are not relevant anymore.

The tested parameters in the forward chained validation process are presented in Table 6.4 and Table 6.5, trained with 500 and 150 epochs for the FFNN and the recurrent models, respectively.

Learning rate	**d**	**$S_{D1};...;S_{Dn}$**
0.001	0.25	64
0.0005	0.4	128
	0.5	64;64
	0.6	128;128
		128;64

Table 6.4: Tested parameters for FFNN in the forward chained validation.

Learning rate	**d**	**l**	**N_{layers}**	**RNN_{size}**	**RNN_{init}**	**$S_{D1};...;S_{Dn}$**
0.0005	0.25	5	1	64	zero	128
0.0001	0.5	20	2		learn	128;128

Table 6.5: Tested parameters for LSTM and GRU in the forward chained validation.

Final models and hyperparameters

As mentioned initially, some hyperparameters have been left out from the optimization search. In our case, these are the optimizer and activation functions used in dense layers. We should also mention how we chose to scale the input variables. All inputs have their median removed and then scaled to the interquartile range. Intraday price features have also been transformed with the area hyperbolic sine function to reduce variance, as previously explained in Section 4.2.

We decide to use Adam as the optimizer. It is instantiated with default parameters in PyTorch, except for the learning rate, which is chosen for each model in the forward chained validation. Adam is a popular optimizer choice in the field of Deep Learning, where a learning rate is maintained for each parameter in the network, and adapted during learning to find good individual learning rates. Adam was introduced in [Kingma and Ba, 2014]. The Rectified Linear

Unit (Relu) is the selected activation function, due to its good properties when the gradient is backpropagated, especially advantageous in deep networks where vanishing gradients can cause problems. Relu simply outputs the max of 0 and the input, i.e. $relu(x) = max(0, x)$.

Table 6.6 summarizes the hyperparameters that is equal for all models.

Hyperparameter	**Selected value**
Optimizer	Adam
Activation function (a)	Relu
Batch size	32
Loss function	SmoothL1Loss

Table 6.6: Selected hyperparameters that are not model specific.

The rest of the hyperparameters are chosen relative to the model and the stage, and further selected by the parameter combination that got the best score on the forward chaining validation. Table 6.7 to 6.12 present the optimized hyperparameters.

Learning rate	**d**	$\mathbf{S_{D1}}$
0.001	0.4	64

Table 6.7: Optimized hyperparameters for FFNN stage 1.

Learning rate	**d**	$\mathbf{S_{D1}}$
0.0005	0.6	64

Table 6.8: Optimized hyperparameters for FFNN stage 2.

Learning rate	**d**	**l**	$\mathbf{N_{layers}}$	$\mathbf{RNN_{size}}$	$\mathbf{RNN_{init}}$	$\mathbf{S_{D1}}$
0.0001	0.5	20	2	64	learn	128

Table 6.9: Optimized hyperparameters for GRU stage 1.

Learning rate	d	l	N_{layers}	RNN_{size}	RNN_{init}	S_{D1}
0.0005	0.5	20	2	64	learn	128

Table 6.10: Optimized hyperparameters for GRU stage 2.

Learning rate	d	l	N_{layers}	RNN_{size}	RNN_{init}	S_{D1}; S_{D2}
0.0005	0.25	5	1	64	zero	128;128

Table 6.11: Optimized hyperparameters for LSTM stage 1.

Learning rate	d	l	N_{layers}	RNN_{size}	RNN_{init}	S_{D1}; S_{D2}
0.0005	0.25	5	1	64	zero	128;128

Table 6.12: Optimized hyperparameters for LSTM stage 2.

Model	Stage 1	Stage 2
FFNN	961	1217
GRU	468389	514529
LSTM	147722	159257

Table 6.13: Number of trainable parameters in the proposed models.

The final model architectures are illustrated in Appendix A. In Table 6.13 we include the number of trainable parameters (weights and biases) for each of the proposed models, as a measure of their complexity.

The regularization parameter λ in the LASSO was determined in the same way, i.e., from the weighted average across the 5 chains. It ended up as 0.034 for stage 1 and 0.015 for stage 2 (when validating on the transformed prices). The LASSO was also constructed with an upper bound of 100000 on the number of iterations required to find the coefficients, and the rest were default hyperparameters from Scikit-learn. This library implements ML algorithms made available in a high-level language [Pedregosa et al., 2011]. OLS was also built merely using default hyperparameters from Scikit-learn.

6.2.3 When to stop final training

After the final models have been selected, we merge the previous train and validation sets, to get all the available training data. The challenging part is now to determine when to stop training because we have no distinct validation data to use for early-stopping purposes. Instead, we have to deduce a sensible amount of training based on the experience we have made from training on the chains.

We make this choice because we have little data, to begin with, and cannot afford not to utilize everything for training. Besides, months can be very different from seasonal effects. It is not given that an early-stopping scheme on selected months would yield robust results on unseen testing months.

In Table 6.14, we see the epoch numbers that minimized the mean absolute error on each chain validation for the final models. The number of epochs varies greatly, and individual validation results are likely biased towards the evaluated months. The final number of epochs used to train our models should take this into account somehow. Our decision is to decide on the number of epochs in the same way as we have optimized for any other hyperparameter. That is, we decide to train our final models with the number of epochs equal to the weighted number of epochs that minimized the MAE across all of the chains (where the weight of each chain is proportional by the number of months involved in the training).

	Chain					
	1	**2**	**4**	**4**	**5**	**Weighted Sum**
FFNN stage 1	10	6	5	97	156	74
GRU stage 1	10	8	7	8	12	9
LSTM stage 1	5	7	4	4	17	8
FFNN stage 2	482	410	183	451	479	403
GRU stage 2	11	7	4	6	6	6
LSTM stage 2	21	10	8	12	11	12

Table 6.14: Number of epochs that minimized the MAE on each chain for the final models. Weighted sum is the weighted number of epochs across all of the chains, where the weight of each chain is proportional by the number of months involved in the training.

6.2.4 Rolling window evaluation

The out-of-sample testing set is from 01/02/2020 to 27/03/2020. To exhibit having conditions as real as possible, we utilize a rolling window for the testing

phase. More precisely, we choose to retrain the models once a day at midnight[4] with all available data. Because we make the forecast two hours before delivery, all available data at midnight in this setting is data up until 10 PM on the night before. This means that, for example, when predicting the prices for 27/03/2020, the models are trained with data from 01/06/2019 00:00 to 26/03/2020 22:00[5].

6.2.5 Ensemble learning

Furthermore, we do not fit just a single neural network every new day, but rather 10 and select the median $\hat{y}_i^{d,h}, \{i = 1, ..., 10\}$ outputted from all the similar models. The reason for this is that neural networks are sensitive models, especially for smaller datasets. They can, for example, be unfortunate with the randomized weight initialization or get stuck in local minima during gradient descent. The median is a well-suited estimate of the central tendency for outlier-prone distributions. The baselines are deterministic in their training and do not require similar care-taking. The eager reader who is curious about the variance observed when training the proposed neural networks can visit Appendix B.

[4]Theoretically, we could have retrained the models every hour, but with a substantial increase in computational burden, for practical reasons, we did not do that.

[5]As a reminder, note that almost 3 weeks from the total dataset had to be removed due to missing prices in the final hour prior to delivery, see Section 4.1.

Chapter 7

Results and Discussion

In this chapter, we first evaluate the forecasts for stage 1 and stage 2 on the unseen test data, and plot the forecasts that have obtained the lowest linear errors for each stage. Further in Section 7.2, we describe in more detail properties of the the forecasted series. Section 7.3 assess the significance of different forecast accuracies. We finally discuss the results that deserve specific attention in Section 7.4.

Some results have been placed in the Appendix. This accounts for metrics reported on specific subsamples of the test-data, see Appendix C. Metrics reported on the full back-transformed prices, i.e. the sum of the spot price and predicted deltas since spot, can be found in Appendix D. Visualizations of the forecasted time series for all models are left in Appendix E.

7.1 Forecast Evaluation

Model	**MAE**	**R^2**	**Correct sign**
MOR	9.198	0.069	0.593
OLS	9.180	0.078	0.586
LASSO	9.094	0.067	0.609
FFNN	9.122	0.083	0.588
LSTM	**8.938**	**0.105**	0.611
GRU	8.965	0.086	**0.622**

Table 7.1: Results from stage 1 for selected evaluation metrics. The best score for each metric is emphasized in bold.

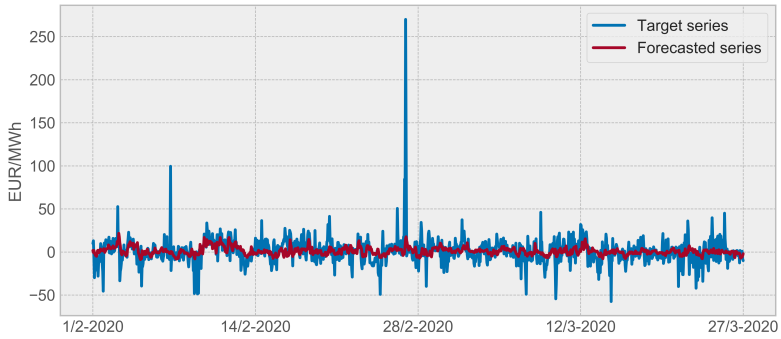

Figure 7.1: LSTM stage 1 forecast.

Model	**MAE**	**R^2**	**Correct sign**
Naïve	7.545	**0.364**	0.726
OLS	7.651	0.300	0.722
LASSO	7.648	0.294	**0.727**
FFNN	7.659	0.293	0.722
LSTM	7.924	0.265	0.717
GRU	7.926	0.234	0.712
ens(MOR)	7.971	0.276	0.711
ens(OLS)	7.551	0.352	0.723
ens(LASSO)	7.568	0.338	**0.727**
ens(FFNN)	**7.534**	0.351	0.726
ens(LSTM)	7.632	0.329	0.718
ens(GRU)	7.640	0.316	0.723

Table 7.2: Results from stage 2 for selected evaluation metrics. The best score for each metric is emphasized in bold.

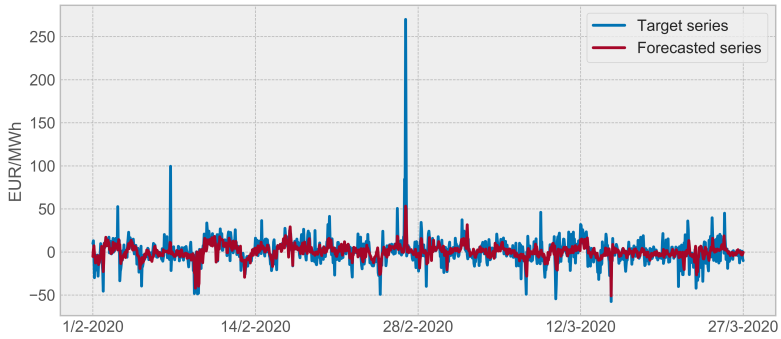

Figure 7.2: ens(FFNN) stage 2 forecast.

7.2 Forecast Characteristics

Model	mean	std	min	25	50	75	max
MOR	-0.069	5.012	-32.552	-2.142	0.000	2.292	22.743
OLS	1.546	3.940	-16.494	-0.912	1.406	3.793	20.446
LASSO	0.569	2.974	-15.795	-1.112	0.473	2.310	10.063
FFNN	2.403	5.054	-12.647	-0.230	1.890	4.322	35.722
LSTM	1.004	4.209	-8.779	-1.629	0.589	3.253	21.727
GRU	0.787	3.705	-9.423	-1.589	0.346	2.959	13.989

Table 7.3: Forecast characteristics stage 1

Model	mean	std	min	25	50	75	max
Naïve	0.301	7.965	-56.960	-3.610	0.290	4.130	60.920
OLS	0.660	6.653	-24.330	-3.378	0.661	4.427	34.050
LASSO	0.450	6.360	-26.903	-3.561	0.282	4.380	31.348
FFNN	0.765	6.892	-37.178	-2.554	0.586	3.902	33.412
LSTM	0.452	6.590	-30.823	-2.989	0.186	3.520	41.156
GRU	0.372	6.356	-23.717	-2.830	-0.034	3.494	24.624
ens(MOR)	0.116	5.469	-28.308	-2.558	0.093	2.762	35.858
ens(OLS)	0.390	7.600	-48.773	-3.575	0.362	4.222	53.867
ens(LASSO)	0.368	7.199	-43.434	-3.580	0.322	4.294	47.613
ens(FFNN)	0.417	7.637	-50.918	-3.423	0.374	4.149	53.295
ens(LSTM)	0.376	7.060	-37.378	-3.256	0.281	3.987	46.770
ens(GRU)	0.336	6.920	-32.352	-3.335	0.160	3.971	40.711

Table 7.4: Forecast characteristics stage 2

7.3 Significance of Different Forecasts

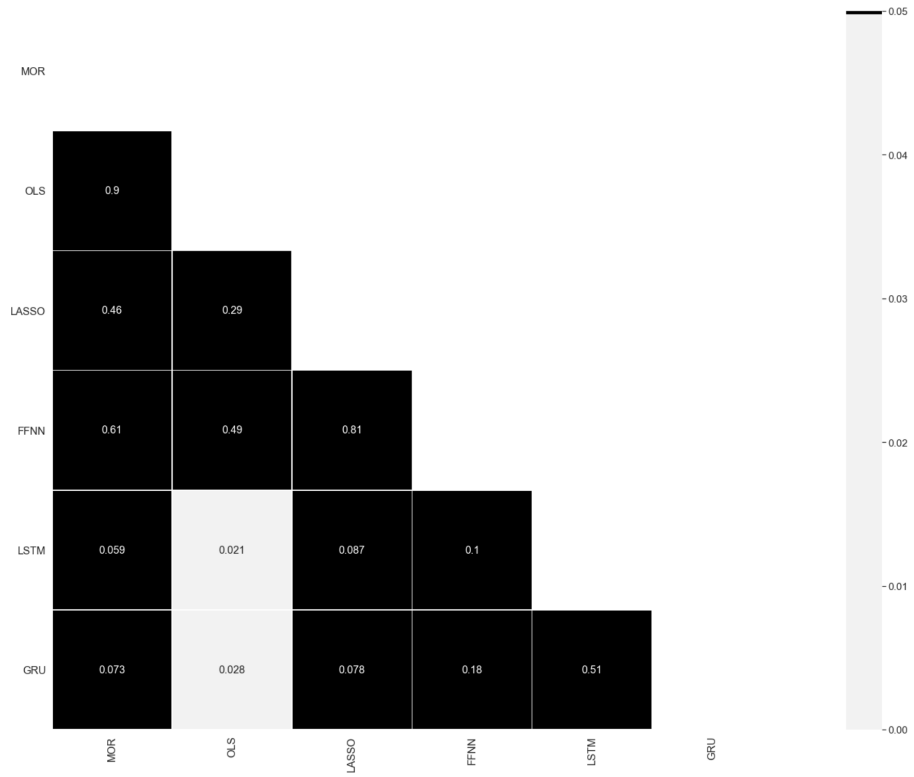

Figure 7.3: Heatmap to indicate range of p-values obtained from the Diebold-Mariano test on stage 1. P-values less than 0.05 are colored white.

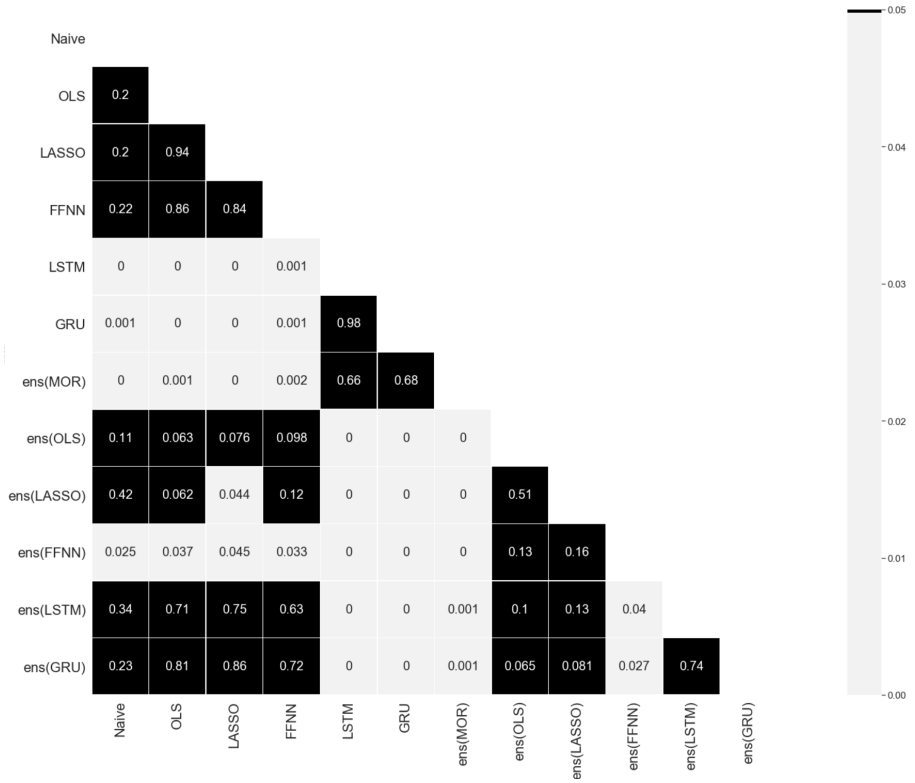

Figure 7.4: Heatmap to indicate range of p-values obtained from the Diebold-Mariano test on stage 2. P-values less than 0.05 are colored white.

7.4 Discussion

7.4.1 General results for stage 1

From Table 7.1, the GRU and LSTM obtain the best scores in stage 1. The FFNN and LASSO then follow, when linear errors are concerned. Finally, the OLS and MOR perform the worst.

With that being said, errors are generally quite comparable with a 2.7 percent difference in MAE from the best to the worst-performing model. This does not imply that the models are equally as good, but more than anything is a testament to the difficulty of predicting prices in the final trading hour. The market tightens up very much shortly before delivery. Many operators try to

balance their remaining forecast errors. At the same time, resources designated for balancing purposed become scarce due to finite availability and long ramp-up times [Garnier and Madlener, 2015]. The direction of the upcoming system imbalance further becomes clearer. The entire situation looks very difficult to estimate two hours prior to delivery, which is when we make the forecast. It seems very plausible that even a significant advancement in modeling can result in just a minor error reduction on average.

As a result, the significance of outperformance of different forecasts obtained with the DM test is not very evident, as can be seen from Figure 7.3. The only significant difference obtained at the 0.05 critical level is that the forecasts produced by the recurrent networks do not equal the forecast accuracy obtained from the OLS. We also have a limited test set and a heavy-tailed target distribution. The null-hypothesis of equal forecast accuracy could avoid getting rejected due to having a pretty small test set of only 1257 hours. The complexity of the objective, combined with scarce data, is a threat to validity and makes drawing general conclusions more difficult.

However, note that the DM test has no concept of knowing how robustly the forecasts are constructed. Rerunning the experiments would, in all likelihood, yield very similar results on the reported metrics. We argue that this is the case because we re-calibrated each network every day with the latest available data. Moreover, to enhance robustness, we made the final prediction from the median across ten independently trained models. Hundreds of models were thus collectively built during the approximately two months of testing data, undoubtedly lowering the probability of good results to be caused merely by random effects.

Differences in model performance also become greater when price spikes are removed. If we consider scores obtained when predicting only the prices between the 5 and 95 percent quantile levels, the difference in MAE from the best model (GRU) to the worst model (MOR) grows to well above 5 percent, highlighted in Table C.1.

Interestingly, as can be seen in Table C.2, the MOR performs best at the outer percentile levels. This could indicate that extreme prices usually happen in the steep merit-order regime and that the other models underestimate the importance of fundamental merit-order characteristics relative to other exogenous input variables. A more likely explanation is perhaps that the models have not been trained with enough historical occurrences of data with high price spreads. On top of that, they appear to be regularized very strictly with the purpose of minimizing mean errors and hence perform poorly for prices at outer quantile levels.

Generally speaking, the models seem to perform the worst in cases when the price spread is assumed to be the highest. Detailed scores on outer percentile

levels of input variables is found in Appendix C. High or low spot prices, extreme system imbalances, major power outages, and rapid movement in the merit-order (i.e., ramping effects) are all correlated with higher than normal price spreads. Generalized models constructed to minimize mean absolute errors do poor in these situations.

7.4.2 General results for stage 2

For stage 2, the highest obtained R^2 has increased to over 0.36 from what previously was ca 0.10 in stage 1, see Table 7.2. The relative results between models are, in turn, more similar. The truth is likely that the naïve input, which is the most recent available price for the target contract, is already very correlated with the target price. Models that are built from minimizing errors tend to assign much importance to this input variable, and pattern finding in other exogenous variables is down-prioritized. Simplicity is further rewarded because the naïve input is coupled with fewer interactions from additional model complexity during inference.

It seems as fundamental information is already partly or fully incorporated into the most recent transaction price. Complex networks with more historical inputs, i.e., the LSTM and GRU, struggle to give enough weight to the naïve input. The passes of historical fundamentals through the recurrent layers introduce noise to the final output.

The difference in MAE from best to worst model drops a whole percentage when concerning only the prices between the 5 and 95 percent quantile levels. In this case, the FFNN is getting a slightly lower MAE than the naïve baseline, so the network seems capable of adding some useful information for typical price spreads. Linear concepts also outperform the naïve price when spikes are removed. The differences are, however, not very significant, as can be seen from Figure 7.4, and which model is preferable in this scenario looks inconclusive based on these empirical results.

Averaging forecasts does better on a macro-scale. The autocorrelation coming from prior transaction prices is very rewarding for errors when there is a price spike. In contrast, more sophisticated models that incorporate the latest fundamental information looks to do better in more normal price ranges. Averaging them hence reduces mean errors.

Although it is hard to significantly outperform the naïve two hours prior to delivery, other forecast horizons would likely give different results. Our models are built profoundly with fundamental inputs. Fundamental weather drivers are often known in advance, but not captured in the observable market prices until

very close to delivery. Other studies have, for instance, shown that it is possible to beat the naïve 4 hours before delivery significantly with the LASSO estimator [Marcjasz et al., 2020].

7.4.3 Comparing the networks

The proposed neural networks are very different. The recurrent models have more than a hundred times as many model parameters that must be tuned, compared to the FFNN (see Table 6.13). One might think that this should result in the recurrent models to require more training, but what we see is the opposite. The recurrent networks overfit quickly; the lowest errors on the validation sets were obtained fast, and we in-turn, had to stop training from an early epoch.

The recurrent models are trained with more historical data given as input features. The advantage of this is that they are less sensitive to single noisy measurements. The disadvantage is formerly knowing if all of the extra input data is relevant. Furthermore, if one of the input variables happens to be of particularly high importance, parts of its information could more easily be lost from the operations jointly performed on all of the input data. The latter seems to be the case for stage 2 with the influential naïve transaction price added.

Recall that we fitted each network several times, where the final output was calculated as the median across ten independent predictions. In Appendix B, we have assessed the sensitivity of the networks by exploring the maximum and minimum outputs for each of the predictions made from the ten networks. In stage 1, it turns out that the LSTM shows the highest variance in training. On the other hand, the GRU is a more regularized network and has the lowest empirical standard deviations of predictions across the 10 networks. This situation is not repeated in stage 2, where the GRU and the LSTM are a lot more sensitive compared to the FFNN.

In stage 1, the GRU and LSTM get the best results. These results are interesting as they are the most complex models and trained with the smallest number of epochs. By comparing it with the much simpler FFNN, we see that the recurrent networks are more robust, indicating that useful non-linearities perhaps are picked up in the 12 hours of historical data through the recurrent layers. The GRU stands out in this regard, as it utilizes two stacked layers. LSTM has just one recurrent layer but a deeper dense section.

We should mention that the FFNN showed more promising learning behavior during the grid search and forward chained validation. Admittedly, we were somewhat surprised to see that the more advanced recurrent models obtained the lowest errors on the unseen test data. New questions arise to validate this

more thoroughly beyond two months of testing.

7.4.4 Comparing the networks to the statistical baselines

Linear concepts tend to show convincing results for EPF, and our results show that they indeed perform comparable, but just slightly worse than the proposed non-linear neural networks. The difference between model classes is greater for stage 1. However, whether this difference is statistically significant is still an open question from the studies conducted on a rather small test set of 1257 hours.

The LASSO has the lowest variance in predictions (see Table 7.3 and 7.4), as it is regularized the hardest of all models. Consequently, it performs very well on the metric that counts the percentage of times the predicted value has the same sign as the true observation. Magnitudes do not matter for the calculation of this score.

Having simple models that are easily interpretable is arguably more important as a decision support tool than a complex price model, even if the latter has just a marginally lower error. Yet we have seen from Section 2.5 that many non-linearities determine the intraday prices, so coupling non-linear methods with explainable ML is likely the way to go.

7.4.5 The importance of accounting for the merit-order effect

Although the MOR is performing the worst, it is still doing surprisingly well when considering the very simplistic assumptions that we made. Recall that we assumed that only half of the power were traded on exchanges and the rest through bilateral contracts.

Also, note the MOR is the only model that does not incorporate information about the system imbalance; it only uses the forecast errors since spot time to traverse the bid-offer curve. It is known that the system imbalance 2 hours and 15 minutes prior to delivery correlates a lot with the constructed series (we measured a coefficient of 0.25 in our dataset). Performance would likely be higher if the MOR somehow included data on system imbalance to make its future estimates.

Nevertheless, we argue that these results highlight that sensible price models should take fundamental characteristics from the merit-order into account.

7.4.6 Training with enough data and accounting for seasonal effects

The FFNN in stage 1 shows some unhealthy signs concerning generalization capabilities. We see in Table 7.3 that the FFNN is overestimating in general, with a mean prediction of 2.403, which is a lot larger than the actual mean of 0.923. The same can be said about OLS. Interestingly, the FFNN and OLS are fed with exactly the same features, indicating that these models perhaps were not trained with enough data to find generalizable patterns in the input variables. This argument is further strengthened from the observation that the average price in the training data is skewed slightly positive, which causes the models to be biased towards predicting positive deltas. This weakness stems from having too few cases in the historical dataset; in the long run, price deltas will likely revert to zero mean. The models trained with richer feature sets do not show any tendency to overestimate towards positive deltas.

A further problem in EPF studies with a limited sized dataset is to account for seasonal effects. The models' final hyperparameters were chosen from the weighted sum of MAEs across all of the monthly chains. This was done to not get parameters heavily biased towards specific months, which can be very different. Accounting for seasonal effects is extremely important in order to obtain robust results. A pitfall would, for instance, be to use January explicitly for hyperparameter tuning. With this scheme, solar generation forecasts would get close to zero importance because it is not sunny in the winter. That clearly would not be desirable when testing on months with more sun. Forecasts built from at least a year of data bypasses this problem to some extent.

For seasonality in smaller frequencies, some readers might be surprised that we avoid using hourly and weekly dummy input variables. We make this choice because calendar effects are already implicitly captured in the consumption forecasts. Besides, throwing in another 24 or 31 input variables would make learning a lot more difficult from the curse of dimensionality.

7.4.7 Some words on the limitations of the conducted DM-test

A hurdle in the DM test is, as just briefly mentioned in Section 2.4, that the loss differential series can be autocorrelated. Generally speaking, prices close in temporal order are predicted using similar fundamental input drivers, and residuals might thus also become serially correlated. The modified DM test proposed by [Harvey et al., 1997] that was conducted, is very sensitive to the chosen lag-order h. In our study, we chose to employ a two-step ahead h because from inspecting

the autocorrelations from the residuals, lags greater than 2 were usually close to zero and thus had a negligible covariance. Results would seem more significant if we instead were to use $h = 1$; especially noteworthy is that all forecasts obtained from neural networks on stage 1 would now statistically outperform the forecasts obtained from the baselines at the 5 percent significance level.

The optimal lag order to use in the DM test is likely model-specific as they take different inputs. Ideally, a more sophisticated method should be performed in order to estimate the individual orders of covariance-stationary. [Ziel et al., 2015] used an autoregressive process to tackle correlation in the DM test. Taking these types of measures was outside the scope of this study.

Chapter 8

Conclusion

This chapter is the final chapter. We first conclude and give answers to the research questions. In Section 8.2, we argue if any contributions have been made to the field. Section 8.3 says some words about suggested future work.

8.1 Conclusion

This work has been centered around the creation of artificial neural networks to tackle the task of predicting final electrical intraday prices on the German intraday market. The constructed target series was an approximation of the average price delta from the spot price for hourly products traded in the final 60 minutes before delivery.

We have seen that the market tightens up considerably in the final trading hour because market participants need to be able to react to the latest information. Predicting prices in the final trading hour while making the forecast two hours in advance remains a very challenging task. A major improvement in modeling seems to translate into just a small MAE reduction on average.

The very noisy target series resulted in strict regularization and careful training of the proposed FFNN, LSTM, and GRU. A comprehensive forward-chained validation search was applied to combat the overfitting-problem. With this technique, the optimal training time and hyperparameters were determined from a weighted average of the minimal MAEs acquired across numerous validation months.

Linear models tend to show convincing results in EPF studies, and our results show that these baselines indeed perform comparable, but slightly worse than the

proposed non-linear networks. The significance of the outperformance of different forecasts obtained with the Diebold-Mariano test is further debatable from the studies that we have conducted on a limited test set of 1257 hours.

A rolling window-technique was utilized during the testing phase such that models were re-trained every new day with the latest available data. To enhance robustness, we fitted ten similar networks and selected the median outputted from all the alike models. Thus, hundreds of neural networks were built during the entire testing phase. This suggests that the neural network's reported performance is unlikely to be caused by chance, even if the null-hypothesis of equal forecast accuracy could not be rejected.

Research question 1 *To what extent does exogenous factors determine the final intraday prices that can be predicted?*

The results show that up to 10 percent of the variance of the price spread can be predicted merely from exogenous effects. When allowing for the inclusion of variables constructed from preceding intraday prices, i.e., in stage 2, the most recently available transaction price sampled two hours before delivery accounts alone for over a third of the variance. The best estimates are obtained when the preceding price is combined with models enhanced with timely constructed fundamental variables.

Research question 2 *How do recurrent networks compare to feed-forward networks when predicting final intraday prices?*

There are some indications that recurrent models can better capture the serial dependence that evolves in the time series of fundamental variables. Although the LSTM obtains the lowest MAE for stage 1, error measurements are generally susceptible to price spikes. If spikes are removed, a GRU with two stacked recurrent layers gets the lowest errors. Besides, the GRU has a lower variance in training. For these reasons, GRU seems like a more robust alternative.

We experience that complex models do worse when prior intraday prices are added to the feature set. With greater model expressivity comes increased sensitivity and more unstable behavior in the training process. In this case, a shallow feed-forward network with extensive use of dropout seems preferable based on our empirical results.

8.2 Contributions

We have shown that parameter-rich artificial neural networks can outperform state-of-the-art statistical baselines, even for smaller datasets with a noisy target series constructed from final intraday prices. Complex models perform better

relative to simpler models in typical price ranges when only exogenous effects are used as input variables. In this way, we have contributed with positive indications of the further potential of Deep Learning within the EPF domain.

A necessary prerequisite for robust performance with neural networks is very careful training. We hope that our forward chained methodology perhaps can help aspiring practitioners to not walk into the pitfalls of excluding the potential of neural networks in EPF applications even if deep structures seem to overfit at first sight.

A more practical contribution is the suggestion on how to engineer simple features from the submitted bid-offers from the day-ahead to boost predictive performance. We have empirically validated the known importance of accounting for fundamental characteristics in the merit-order.

8.3 Future Work

In order to assess the significance of results more robustly, this study should ideally be done again when there is enough test-data for at least a year. Furthermore, it would be wise to account for the power volumes traded in the final hour. We suggest utilizing the ID1 index as the target series for reproducibility purposes.

In future work, it is possible to do a more systematic study on the feature selection problem. A natural extension is to use more sophisticated algorithms and automated methods. Additional work can especially be done into engineering fundamental characteristics from the merit-order into the shape of appropriate input variables. It is then advisable to dynamically find the more precise amounts of power traded on exchanges.

An interesting study to conduct on artificial neural networks is to explore the potential of transfer learning. The idea is that a network pre-trained on neighboring or prior transaction prices can adjust the weights to something better than random initialization. We have briefly experimented with pre-training networks, but it deserves a more systematic study.

Furthermore, Convolutional Neural Networks (CNNs) is a model-type known to work well in applications where locally grouped data is correlated. It might be a good fit for EPF studies aiming to predict prices in the very short term.

Finally, although profits are thought to be closely related to small price prediction errors, it is plausible that certain positions can be more optimal than others, albeit generating higher predictive errors on average. Complementary analysis of economic benefits is something to consider in the future.

Bibliography

Amjady, N. and Hemmati, M. (2006). Energy price forecasting-problems and proposals for such predictions. *IEEE Power and Energy Magazine*, 4(2):20–29.

Andrade, J. R., Filipe, J., Reis, M., and Bessa, R. J. (2017). Probabilistic price forecasting for day-ahead and intraday markets: Beyond the statistical model. *Sustainability*, 9(11):1990.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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

Diebold, F. M. and Mariano, R. (1995). R.(1995). comparing predictive accuracy. *Journal of Business & economic statistics*, 20(1).

Diebold, F. X. (2015). Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of diebold–mariano tests. *Journal of Business & Economic Statistics*, 33(1):1–1.

Epias (2018). Epias annual electricity market report 2018.

Fernández-Redondo, M. and Hernández-Espinosa, C. (2001). Weight initialization methods for multilayer feedforward. In *ESANN*, pages 119–124.

Garnier, E. and Madlener, R. (2015). Balancing forecast errors in continuous-trade intraday markets. *Energy Systems*, 6(3):361–388.

Germany Trade And Invest (2018). The german electricity market - a brief overview. `https://www.`

gtai.de/gtai-en/invest/industries/life-sciences/the-german-electricity-market-a-brief-overview-105274.pdf. Online; accessed 9-May-2020.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

Glas, S., Kiesel, R., Kolkmann, S., Kremer, M., von Luckner, N. G., Ostmeier, L., Urban, K., and Weber, C. (2020). Intraday renewable electricity trading: advanced modeling and numerical optimal control. *Journal of Mathematics in Industry*, 10(1):3.

Goodarzi, S., Perera, H. N., and Bunn, D. (2019). The impact of renewable energy forecast errors on imbalance volumes and electricity spot prices. *Energy Policy*, 134:110827.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE.

Gürtler, M. and Paulsen, T. (2018). The effect of wind and solar power forecasts on day-ahead and intraday electricity prices in germany. *Energy Economics*, 75:150–162.

Hagemann, S. (2015). Price determinants in the german intraday market for electricity: an empirical analysis. *Journal of Energy Markets*.

Hagemann, S. and Weber, C. (2013). An empirical analysis of liquidity and its determinants in the german intraday market for electricity.

Harvey, D., Leybourne, S., and Newbold, P. (1997). Testing the equality of prediction mean squared errors. *International Journal of forecasting*, 13(2):281–291.

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

Irsoy, O. and Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 720–728.

Janke, T. and Steinke, F. (2019). Forecasting the price distribution of continuous intraday electricity trading. *Energies*, 12(22):4262.

Kaminski, V. (2012). *Energy markets*. Risk Books.

Kath, C. (2019). Modeling intraday markets under the new advances of the cross-border intraday project (xbid): Evidence from the german intraday market. *Energies*, 12(22):4339.

Kath, C. and Ziel, F. (2018). The value of forecasts: Quantifying the economic gains of accurate quarter-hourly electricity price forecasts. *Energy Economics*, 76:411–423.

Kiesel, R. and Paraschiv, F. (2017). Econometric analysis of 15-minute intraday electricity prices. *Energy Economics*, 64:77–90.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koch, C. (2019). Intraday imbalance optimization: Incentives and impact of strategic intraday bidding in germany.

Koch, C. and Maskosa, P. (2019). Passive balancing through intraday trading.

Kolberg, J. K. and Waage, K. (2018). Artificial intelligence and nord pool's intraday electricity market elbas: a demonstration and pragmatic evaluation of employing deep learning for price prediction: using extensive market data and spatio - temporal weather forecasts. Master's thesis, NHH Norwegian School Of Economics.

Kremer, M., Kiesel, R., and Paraschiv, F. (2019). A fundamental model for intraday electricity trading. *Available at SSRN*.

Kuo, P.-H. and Huang, C.-J. (2018). A high precision artificial neural networks model for short-term energy load forecasting. *Energies*, 11(1):213.

Lago, J., De Ridder, F., and De Schutter, B. (2018). Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms. *Applied Energy*, 221:386–405.

Le, Q. V. et al. (2015). A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain*, pages 1–20.

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

Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

Maciejowska, K. and Weron, R. (2019). Hsc research report. *Energy Economics*.

Marcjasz, G., Uniejewski, B., and Weron, R. (2020). Beating the naive - combining lasso with naive intraday electricity price forecasts. *Energies*, 13(7):1667.

Maskos, P. (2017). *Der Einfluss des Netzregelverbundsaldos auf den Handel am kontinuierlichen Intraday-Markt*. PhD thesis, Master Thesis. Berlin: Technical University of Berlin.

Monteiro, C., Ramirez-Rosado, I. J., Fernandez-Jimenez, L. A., and Conde, P. (2016). Short-term price forecasting models based on artificial neural networks for intraday sessions in the iberian electricity market. *Energies*, 9(9):721.

Murdock, H. E., Gibb, D., André, T., Appavou, F., Brown, A., Epp, B., Kondev, B., McCrone, A., Musolino, E., Ranalder, L., et al. (2019). Renewables 2019 global status report.

Narajewski, M. and Ziel, F. (2019). Econometric modelling and forecasting of intraday electricity prices. *Journal of Commodity Markets*, page 100107.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems*. Pearson education.

Nowotarski, J. and Weron, R. (2015). Computing electricity spot price prediction intervals using quantile regression and forecast averaging. *Computational Statistics*, 30(3):791–803.

Oksuz, I. and Ugurlu, U. (2019). Neural network based model comparison for intraday electricity price forecasting. *Energies*, 12(23):4557.

Pape, C., Hagemann, S., and Weber, C. (2016). Are fundamentals enough? explaining price variations in the german day-ahead and intraday power market. *Energy Economics*, 54:376–387.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pelagatti, M. (2018). Trend and long-run relations in electricity prices - why pre-filtering is inevitable. `http://www.aieeconference2018milan.eu/presentations/PELAGATTI.pdf`. Third AIEE Energy Symposium, Online; accessed 8-May-2020.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Russell, S. J. and Norvig, P. (2010). Artificial intelligence-a modern approach (3rd internat. edn.).

Skajaa, A., Edlund, K., and Morales, J. M. (2015). Intraday trading of wind energy. *IEEE Transactions on Power Systems*, 30(6):3181–3189.

Specht, D. F. and Shapiro, P. D. (1991). Generalization accuracy of probabilistic neural networks compared with backpropagation networks. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume 1, pages 887–892. IEEE.

Stork, D. G. (1989). Is backpropagation biologically plausible. In *International Joint Conference on Neural Networks*, volume 2, pages 241–246. IEEE Washington, DC.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

Touretzky, D. S. and Pomerleau, D. A. (1989). What's hidden in the hidden layers. *Byte*, 14(8):227–233.

Ugurlu, U., Oksuz, I., and Tas, O. (2018). Electricity price forecasting using recurrent neural networks. *Energies*, 11(5):1255.

Ullern, S. and Fjeldberg, V. P. (2019). Using deep learning to predict hourly net imbalance volumes on power markets in germany and the nordics. Project report in TDT4900, Department of Computer and Information Science, NTNU – Norwegian University of Science and Technology.

Uniejewski, B., Marcjasz, G., and Weron, R. (2019). Understanding intraday electricity markets: Variable selection and very short-term price forecasting using lasso. *International Journal of Forecasting*, 35(4):1533–1547.

Uniejewski, B., Weron, R., and Ziel, F. (2017). Variance stabilizing transformations for electricity spot price forecasting. *IEEE Transactions on Power Systems*, 33(2):2219–2229.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Weron, R. (2007). *Modeling and forecasting electricity loads and prices: A statistical approach*, volume 403. John Wiley and Sons.

Weron, R. (2014). Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International journal of forecasting*, 30(4):1030–1081.

Würzburg, K., Labandeira, X., and Linares, P. (2013). Renewable generation and electricity prices: Taking stock and new evidence for germany and austria. *Energy Economics*, 40:S159–S171.

Zhang, F. and Fleyeh, H. (2019). A review of single artificial neural network models for electricity spot price forecasting. In *2019 16th International Conference on the European Energy Market (EEM)*, pages 1–6. IEEE.

Ziel, F. (2017). Modeling the impact of wind and solar power forecasting errors on intraday electricity prices. In *2017 14th International Conference on the European Energy Market (EEM)*, pages 1–5. IEEE.

Ziel, F., Steinert, R., and Husmann, S. (2015). Forecasting day ahead electricity spot prices: The impact of the exaa to other european electricity markets. *Energy Economics*, 51:430–444.

Appendices

A Figures of the final proposed models

A.1 Stage 1

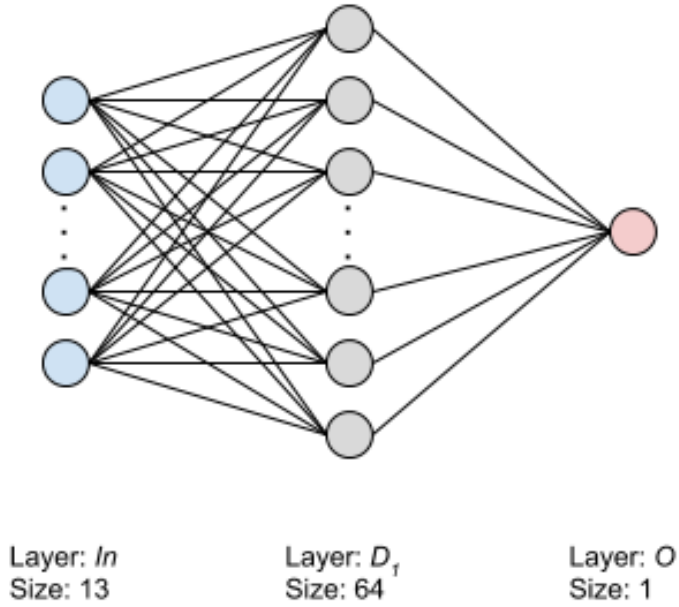

Figure A.1: FFNN architecture stage 1

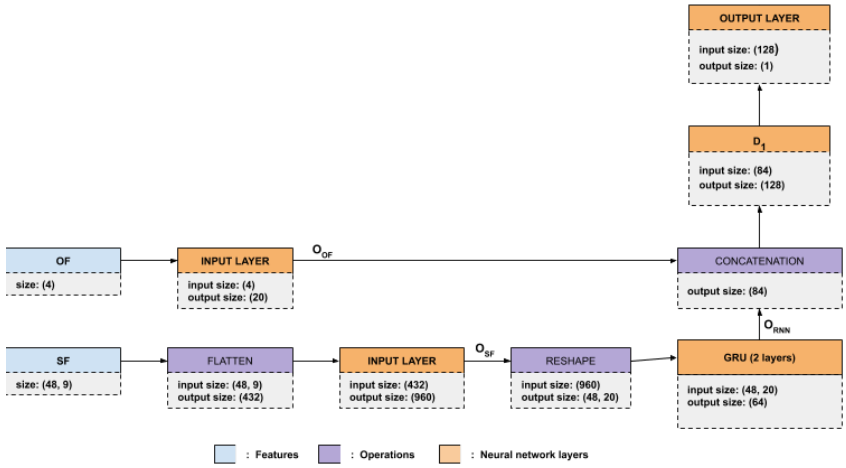

Figure A.2: GRU architecture stage 1

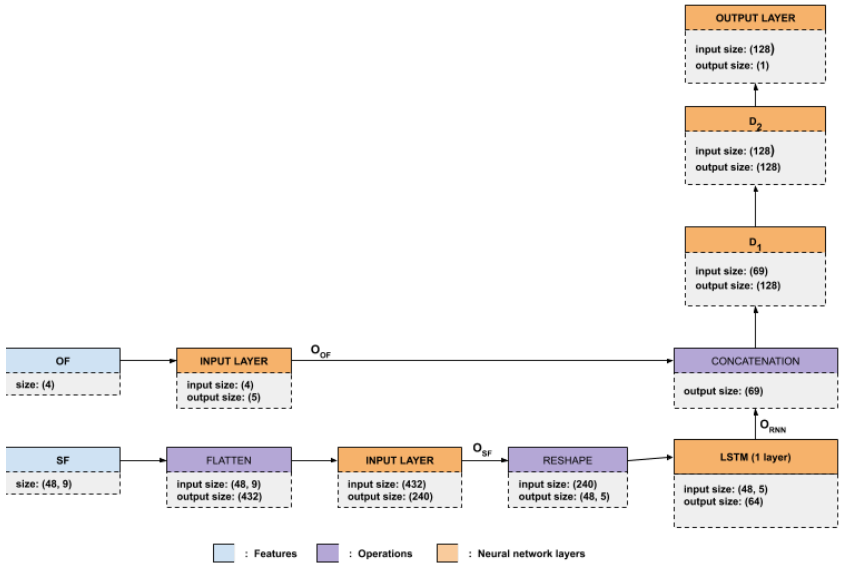

Figure A.3: LSTM architecture stage 1

,

A.2 Stage 2

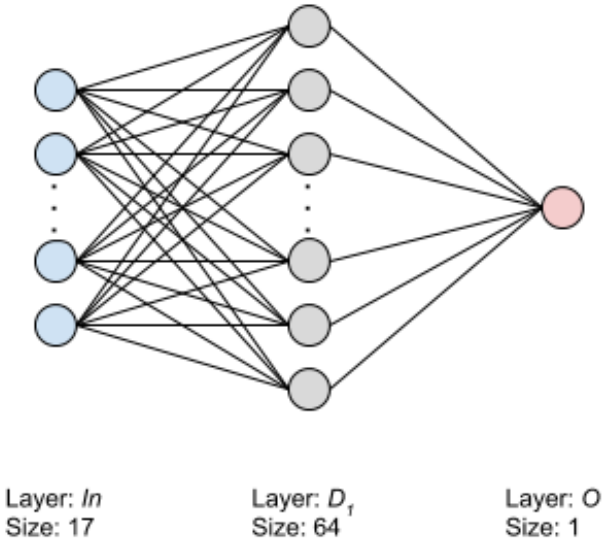

Figure A.4: FFNN architecture stage 2

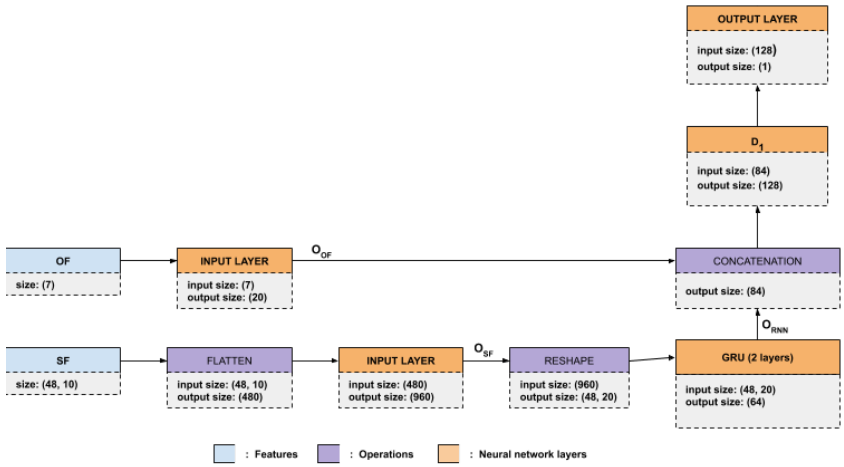

Figure A.5: GRU architecture stage 2

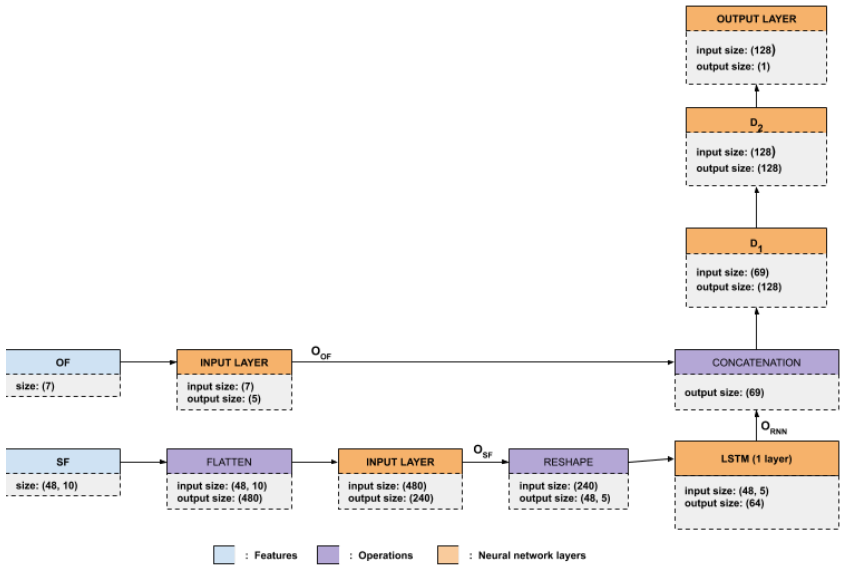

Figure A.6: LSTM architecture stage 2

B Variance in training the networks

When testing our models, the final prediction, $y^{d,h}$, was selected by calculating the mean of the output of 10 trained networks. We can assess the sensitivity of the models by exploring the output of similar networks. For each model, we have constructed two time series, $pred^{max}$ and $pred^{min}$, where $pred^{max}$ contains the maximum outputs of the 10 trained networks for each hour in the test set, while $pred^{min}$ contains the minimum outputs. The times series is mathematically constructed as follow:

$$pred^{max} = \{max(net_i^1, ..., net_i^{10}), ..., max(net_n^1, ..., net_n^{10})\}$$

$$pred^{min} = \{min(net_i^1, ..., net_i^{10}), ..., min(net_n^1, ..., net_n^{10})\}$$

where n is the number of hours in the test set, and net_i^1 is the first network's prediction of the i'th hour. Then, we can calculate the difference of the time series, where:

$$diff_i = pred_i^{max} - pred_i^{min}$$

Table B.1 describes the $pred^{max}$ and $pred^{min}$ for our proposed models, while Table B.2 describes the $diff$. What we can see is that $pred^{max}$ and $pred^{min}$ yield quite different characteristics for all models saying something about the maximum and minimum boundaries for the models. By inspecting the different $diff$'s we see that the models behave differently. In stage 1, LSTM is the model with the highest variability and most sensitive during training. Interestingly, given the number of model parameters, GRU has the most. In this case, it leads to more stable training compared to the other networks, as its mean error, maximum error and the standard deviation of the errors are smaller. The GRU is therefore the most regularized network, and it seems to be advantageous with complex models when only exogenous factors are considered in the feature set.

In stage 2, less complex architectures seem preferable, as some features have such great explanatory power relative to other features, so other features mainly contribute to noise.

		mean	**std**	**min**	**25**	**50**	**75**	**max**
FFNN	$pred^{max}$	3.71	5.82	-10.10	0.55	2.67	5.57	44.30
Stage 1	$pred^{min}$	1.19	4.70	-15.92	-1.29	1.01	3.24	26.19
GRU	$pred^{max}$	1.95	3.97	-7.93	-0.77	1.25	4.30	15.83
Stage 1	$pred^{min}$	-0.27	3.59	10.40	2.52	0.68	1.83	10.98
LSTM	$pred^{max}$	3.67	5.38	-6.75	0.30	2.76	5.82	38.60
Stage 1	$pred^{min}$	-1.29	3.97	-13.35	-3.70	-1.36	1.19	14.74
FFNN	$pred^{max}$	1.60	7.07	-34.89	-1.98	1.01	4.62	37.51
Stage 2	$pred^{min}$	0.02	6.92	-43.05	-3.10	0.20	3.35	29.39
GRU	$pred^{max}$	2.95	7.28	-18.67	-1.32	1.06	6.23	43.71
Stage 2	$pred^{min}$	-1.92	6.560	-31.77	-4.83	-1.08	1.55	19.43
LSTM	$pred^{max}$	3.49	8.46	-19.25	-1.44	1.69	5.86	58.16
Stage 2	$pred^{min}$	-2.22	7.44	-47.29	-4.57	-1.47	1.82	24.64

Table B.1: $pred^{max}$ and $pred^{min}$ for the proposed models

diff for model:	**mean**	**std**	**min**	**25**	**50**	**75**	**max**
FFNN stage 1	2.53	1.98	0.25	1.38	2.07	2.93	20.51
GRU stage 1	2.21	0.98	0.38	1.53	2.04	2.73	7.51
LSTM stage 1	4.96	2.87	0.72	3.02	4.38	6.20	25.84
FFNN stage 2	1.58	1.25	0.25	0.82	1.20	1.90	10.61
GRU stage 2	4.87	3.29	0.64	2.43	4.05	6.29	24.28
LSTM stage 2	5.71	5.18	1.03	2.68	3.95	6.45	46.95

Table B.2: $diff$ for the proposed models

C Results on selected subsamples of test data

C.1 Stage 1

Model	**MAE**	**R^2**	**Correct sign**
MOR	6.938	-0.054	0.580
OLS	6.809	0.046	0.575
LASSO	6.655	0.067	0.600
FFNN	6.846	0.019	0.578
LSTM	6.624	0.065	0.603
GRU	6.596	0.075	0.613

Table C.1: Results from stage 1 for selected evaluation metrics when spikes are removed, i.e. only consider prices $y^{d,h}$ between the 5 and 95 percentile levels in the test set.

Model	**MAE**	**R^2**	**Correct sign**
MOR	29.486	0.115	0.714
OLS	30.463	0.089	0.690
LASSO	30.987	0.066	0.690
FFNN	29.550	0.106	0.675
LSTM	29.706	0.119	0.683
GRU	30.233	0.089	0.706

Table C.2: Results from stage 1 for selected evaluation metrics on spikes only, i.e. only consider prices $y^{d,h}$ outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	**R^2**	**Correct sign**
MOR	9.731	0.072	0.602
OLS	9.581	0.104	0.630
LASSO	9.716	0.070	0.628
FFNN	9.523	0.102	0.619
LSTM	9.436	0.120	0.628
GRU	9.537	0.091	0.640

Table C.3: Results from stage 1 for selected evaluation metrics on February data only.

Model	**MAE**	**R^2**	**Correct sign**
MOR	8.610	0.030	0.584
OLS	8.739	-0.018	0.538
LASSO	8.409	0.028	0.589
FFNN	8.681	0.004	0.554
LSTM	8.388	0.037	0.592
GRU	8.336	0.045	0.602

Table C.4: Results from stage 1 for selected evaluation metrics on March data only.

Model	**MAE**	**R^2**	**Correct sign**
MOR	9.526	-0.375	0.617
OLS	8.462	-0.237	0.732
LASSO	9.014	-0.343	0.681
FFNN	8.027	-0.183	0.786
LSTM	8.657	-0.247	0.657
GRU	8.836	-0.301	0.647

Table C.5: Results from stage 1 for selected evaluation metrics on positive target observations only, i.e. $y^{d,h} > 0$.

Model	**MAE**	**R^2**	**Correct sign**
MOR	8.826	-0.854	0.567
OLS	9.995	-1.182	0.421
LASSO	9.185	-0.954	0.528
FFNN	10.364	-1.304	0.363
LSTM	9.257	-0.990	0.559
GRU	9.113	-0.954	0.594

Table C.6: Results from stage 1 for selected evaluation metrics on negative target observations only, i.e. $y^{d,h} < 0$.

Model	**MAE**	**R^2**	**Correct sign**
MOR	16.907	0.047	0.571
OLS	15.791	0.067	0.651
LASSO	16.684	0.030	0.659
FFNN	16.126	0.067	0.611
LSTM	16.493	0.077	0.571
GRU	16.751	0.038	0.595

Table C.7: Results from stage 1 for selected evaluation metrics at extreme spot prices, i.e. only consider predictions when $DA^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	**R^2**	**Correct sign**
MOR	9.215	-0.077	0.690
OLS	8.724	0.116	0.667
LASSO	8.716	0.144	0.730
FFNN	8.840	0.053	0.675
LSTM	8.388	0.164	0.722
GRU	8.444	0.156	0.738

Table C.8: Results from stage 1 for selected evaluation metrics at extreme wind forecast errors, i.e. only consider predictions when $\Delta WI_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	**R^2**	**Correct sign**
MOR	8.186	0.088	0.603
OLS	8.359	0.083	0.556
LASSO	8.129	0.144	0.627
FFNN	8.056	0.125	0.556
LSTM	7.799	0.191	0.675
GRU	7.694	0.214	0.667

Table C.9: Results from stage 1 for selected evaluation metrics at extreme solar forecast errors, i.e. only consider predictions when $\Delta SO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R^2}$	**Correct sign**
MOR	6.681	0.115	0.603
OLS	7.209	0.051	0.516
LASSO	6.775	0.109	0.611
FFNN	7.067	0.088	0.540
LSTM	6.527	0.159	0.683
GRU	6.499	0.161	0.675

Table C.10: Results from stage 1 for selected evaluation metrics at extreme consumption forecast errors, i.e. only consider predictions when $\Delta CO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R^2}$	**Correct sign**
MOR	11.812	0.037	0.621
OLS	11.525	0.069	0.583
LASSO	11.357	0.014	0.544
FFNN	11.276	0.094	0.612
LSTM	11.226	0.080	0.563
GRU	11.416	0.027	0.621

Table C.11: Results from stage 1 for selected evaluation metrics at extreme availability forecast errors, i.e. only consider predictions when $\Delta AV_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R^2}$	**Correct sign**
MOR	8.881	0.056	0.738
OLS	8.676	0.215	0.738
LASSO	8.892	0.215	0.778
FFNN	8.831	0.138	0.778
LSTM	8.384	0.264	0.754
GRU	8.487	0.254	0.770

Table C.12: Results from stage 1 for selected evaluation metrics at extreme sum of weather forecast errors, i.e. only consider predictions when $\Delta WE_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	10.174	0.056	0.706
OLS	9.482	0.241	0.698
LASSO	9.805	0.166	0.706
FFNN	9.372	0.228	0.746
LSTM	9.292	0.236	0.706
GRU	9.375	0.194	0.722

Table C.13: Results from stage 1 for selected evaluation metrics at extreme sum of all forecast errors, i.e. only consider predictions when $\Delta AE_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	8.983	0.205	0.698
OLS	9.773	0.136	0.643
LASSO	9.940	0.064	0.706
FFNN	8.800	0.189	0.675
LSTM	8.584	0.214	0.714
GRU	8.930	0.183	0.714

Table C.14: Results from stage 1 for selected evaluation metrics at extreme wind power forecasts made at spot time, i.e. only consider predictions when $\hat{WI}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	9.573	0.073	0.635
OLS	10.135	-0.104	0.508
LASSO	9.345	0.090	0.587
FFNN	10.008	-0.051	0.524
LSTM	9.518	0.033	0.619
GRU	9.263	0.074	0.635

Table C.15: Results from stage 1 for selected evaluation metrics at extreme solar power forecasts made at spot time, i.e. only consider predictions when $\hat{SO}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	11.371	0.067	0.667
OLS	11.157	0.074	0.571
LASSO	10.983	0.040	0.683
FFNN	11.395	0.080	0.611
LSTM	10.672	0.112	0.667
GRU	10.792	0.062	0.698

Table C.16: Results from stage 1 for selected evaluation metrics at extreme consumption power forecasts made at spot time, i.e. only consider predictions when $\hat{CO}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	14.527	0.068	0.611
OLS	13.889	0.097	0.643
LASSO	13.982	0.054	0.667
FFNN	13.660	0.117	0.635
LSTM	13.676	0.114	0.651
GRU	13.834	0.067	0.667

Table C.17: Results from stage 1 for selected evaluation metrics at extreme system imbalances, i.e. only consider predictions when $SI_9^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	14.299	0.058	0.690
OLS	13.910	0.106	0.722
LASSO	14.588	0.060	0.730
FFNN	13.556	0.118	0.706
LSTM	13.203	0.152	0.714
GRU	13.538	0.097	0.714

Table C.18: Results from stage 1 for selected evaluation metrics at extreme traversals of the bid-offers, i.e. only consider predictions when $MO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
MOR	14.973	0.063	0.627
OLS	14.105	0.088	0.627
LASSO	14.449	0.071	0.659
FFNN	14.800	0.088	0.611
LSTM	14.165	0.132	0.651
GRU	14.229	0.086	0.690

Table C.19: Results from stage 1 for selected evaluation metrics at extreme changes in traversals of the bid-offers, i.e. only consider predictions when $\Delta MO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

C.2 Stage 2

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	5.875	0.153	0.711
OLS	5.756	0.216	0.707
LASSO	5.723	0.225	0.710
FFNN	5.830	0.191	0.707
LSTM	5.994	0.168	0.700
GRU	5.960	0.169	0.694
ens(MOR)	5.982	0.183	0.693
ens(OLS)	5.835	0.173	0.708
ens(LASSO)	5.793	0.191	0.711
ens(FFNN)	5.827	0.172	0.711
ens(LSTM)	5.833	0.192	0.700
ens(GRU)	5.821	0.191	0.704

Table C.20: Results from stage 2 for selected evaluation metrics when spikes are removed, i.e. only consider prices $y^{d,h}$ between the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	22.529	0.444	0.865
OLS	24.654	0.331	0.849
LASSO	24.932	0.320	0.881
FFNN	24.076	0.331	0.857
LSTM	25.256	0.301	0.865
GRU	25.572	0.258	0.873
ens(MOR)	25.818	0.311	0.873
ens(OLS)	22.949	0.419	0.857
ens(LASSO)	23.498	0.393	0.873
ens(FFNN)	22.857	0.419	0.865
ens(LSTM)	23.776	0.381	0.881
ens(GRU)	23.966	0.363	0.897

Table C.21: Results from stage 2 for selected evaluation metrics on spikes only, i.e. only consider prices $y^{d,h}$ outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	7.628	0.411	0.769
OLS	7.765	0.325	0.751
LASSO	7.798	0.313	0.762
FFNN	7.791	0.310	0.750
LSTM	8.172	0.284	0.748
GRU	8.150	0.245	0.744
ens(MOR)	8.232	0.296	0.745
ens(OLS)	7.639	0.392	0.765
ens(LASSO)	7.671	0.372	0.763
ens(FFNN)	7.625	0.390	0.763
ens(LSTM)	7.774	0.359	0.750
ens(GRU)	7.782	0.342	0.754

Table C.22: Results from stage 2 for selected evaluation metrics on February data only.

Model	**MAE**	**R^2**	**Correct sign**
Naive	7.453	0.224	0.679
OLS	7.524	0.214	0.689
LASSO	7.483	0.225	0.689
FFNN	7.514	0.226	0.692
LSTM	7.652	0.194	0.682
GRU	7.678	0.181	0.677
ens(MOR)	7.682	0.203	0.674
ens(OLS)	7.453	0.228	0.677
ens(LASSO)	7.453	0.231	0.687
ens(FFNN)	7.434	0.231	0.686
ens(LSTM)	7.475	0.233	0.684
ens(GRU)	7.484	0.228	0.689

Table C.23: Results from stage 2 for selected evaluation metrics on March data only.

Model	**MAE**	**R^2**	**Correct sign**
Naive	7.673	0.070	0.732
OLS	7.618	-0.031	0.749
LASSO	7.730	-0.056	0.737
FFNN	7.591	-0.049	0.753
LSTM	8.022	-0.082	0.719
GRU	8.089	-0.149	0.698
ens(MOR)	8.244	-0.092	0.707
ens(OLS)	7.650	0.048	0.737
ens(LASSO)	7.684	0.018	0.734
ens(FFNN)	7.625	0.045	0.738
ens(LSTM)	7.739	0.010	0.725
ens(GRU)	7.796	-0.023	0.723

Table C.24: Results from stage 2 for selected evaluation metrics on positive target observations only, i.e. $y^{d,h} > 0$.

Model	**MAE**	**R^2**	**Correct sign**
Naive	7.399	-0.292	0.720
OLS	7.687	-0.404	0.691
LASSO	7.555	-0.365	0.716
FFNN	7.736	-0.393	0.688
LSTM	7.814	-0.472	0.715
GRU	7.740	-0.476	0.728
ens(MOR)	7.661	-0.377	0.716
ens(OLS)	7.437	-0.306	0.708
ens(LASSO)	7.435	-0.306	0.720
ens(FFNN)	7.431	-0.303	0.713
ens(LSTM)	7.510	-0.336	0.711
ens(GRU)	7.463	-0.328	0.723

Table C.25: Results from stage 2 for selected evaluation metrics on negative target observations only, i.e. $y^{d,h} < 0$.

Model	**MAE**	**R^2**	**Correct sign**
Naive	11.349	0.423	0.833
OLS	12.318	0.288	0.833
LASSO	12.284	0.278	0.833
FFNN	12.453	0.268	0.833
LSTM	13.364	0.248	0.794
GRU	13.561	0.180	0.817
ens(MOR)	13.276	0.279	0.825
ens(OLS)	11.539	0.394	0.833
ens(LASSO)	11.704	0.364	0.833
ens(FFNN)	11.587	0.388	0.825
ens(LSTM)	12.164	0.348	0.825
ens(GRU)	12.331	0.317	0.841

Table C.26: Results from stage 2 for selected evaluation metrics at extreme spot prices, i.e. only consider predictions when $DA^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	7.547	0.334	0.698
OLS	7.587	0.354	0.683
LASSO	7.615	0.347	0.706
FFNN	7.537	0.352	0.683
LSTM	7.803	0.317	0.714
GRU	7.735	0.300	0.722
ens(MOR)	8.016	0.248	0.714
ens(OLS)	7.547	0.345	0.690
ens(LASSO)	7.565	0.348	0.698
ens(FFNN)	7.503	0.345	0.683
ens(LSTM)	7.636	0.344	0.690
ens(GRU)	7.581	0.341	0.698

Table C.27: Results from stage 2 for selected evaluation metrics at extreme wind forecast errors, i.e. only consider predictions when $\Delta WI_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	6.638	0.472	0.698
OLS	7.014	0.384	0.738
LASSO	6.919	0.409	0.706
FFNN	6.711	0.442	0.738
LSTM	7.074	0.349	0.722
GRU	7.086	0.297	0.714
ens(MOR)	7.039	0.387	0.706
ens(OLS)	6.720	0.466	0.706
ens(LASSO)	6.755	0.459	0.698
ens(FFNN)	6.617	0.478	0.714
ens(LSTM)	6.749	0.452	0.714
ens(GRU)	6.780	0.436	0.706

Table C.28: Results from stage 2 for selected evaluation metrics at extreme solar forecast errors, i.e. only consider predictions when $\Delta SO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	6.151	0.194	0.730
OLS	5.980	0.224	0.730
LASSO	5.954	0.231	0.738
FFNN	5.941	0.227	0.738
LSTM	5.782	0.269	0.738
GRU	5.975	0.225	0.706
ens(MOR)	6.149	0.213	0.730
ens(OLS)	6.102	0.203	0.730
ens(LASSO)	6.054	0.213	0.730
ens(FFNN)	6.068	0.207	0.738
ens(LSTM)	5.876	0.248	0.738
ens(GRU)	5.947	0.226	0.730

Table C.29: Results from stage 2 for selected evaluation metrics at extreme consumption forecast errors, i.e. only consider predictions when $\Delta CO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	10.027	0.385	0.660
OLS	10.276	0.234	0.689
LASSO	10.405	0.215	0.650
FFNN	10.692	0.195	0.660
LSTM	10.428	0.217	0.689
GRU	10.670	0.136	0.660
ens(MOR)	10.378	0.235	0.738
ens(OLS)	10.082	0.350	0.670
ens(LASSO)	10.195	0.313	0.650
ens(FFNN)	10.169	0.342	0.680
ens(LSTM)	10.155	0.308	0.670
ens(GRU)	10.290	0.271	0.660

Table C.30: Results from stage 2 for selected evaluation metrics at extreme availability forecast errors, i.e. only consider predictions when $\Delta AV_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	7.723	0.407	0.770
OLS	7.873	0.414	0.762
LASSO	7.729	0.425	0.770
FFNN	7.768	0.417	0.754
LSTM	7.962	0.388	0.770
GRU	8.004	0.360	0.770
ens(MOR)	7.835	0.350	0.762
ens(OLS)	7.740	0.415	0.770
ens(LASSO)	7.710	0.423	0.770
ens(FFNN)	7.686	0.416	0.762
ens(LSTM)	7.781	0.419	0.762
ens(GRU)	7.802	0.409	0.770

Table C.31: Results from stage 2 for selected evaluation metrics at extreme sum of weather forecast errors, i.e. only consider predictions when $\Delta WE_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	8.806	0.442	0.667
OLS	9.030	0.368	0.659
LASSO	8.950	0.354	0.667
FFNN	9.029	0.337	0.659
LSTM	9.058	0.369	0.683
GRU	9.090	0.300	0.675
ens(MOR)	9.072	0.316	0.706
ens(OLS)	8.842	0.430	0.667
ens(LASSO)	8.855	0.410	0.667
ens(FFNN)	8.824	0.425	0.667
ens(LSTM)	8.894	0.425	0.675
ens(GRU)	8.920	0.393	0.667

Table C.32: Results from stage 2 for selected evaluation metrics at extreme sum of all forecast errors, i.e. only consider predictions when $\Delta AE_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	7.698	0.362	0.770
OLS	7.783	0.348	0.778
LASSO	7.739	0.350	0.770
FFNN	7.859	0.340	0.770
LSTM	8.143	0.277	0.794
GRU	7.885	0.293	0.802
ens(MOR)	8.045	0.334	0.770
ens(OLS)	7.709	0.360	0.762
ens(LASSO)	7.690	0.358	0.770
ens(FFNN)	7.681	0.360	0.754
ens(LSTM)	7.759	0.336	0.794
ens(GRU)	7.679	0.339	0.794

Table C.33: Results from stage 2 for selected evaluation metrics at extreme wind power forecasts made at spot time, i.e. only consider predictions when $W\hat{I}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	7.199	0.518	0.714
OLS	8.297	0.346	0.762
LASSO	7.938	0.417	0.730
FFNN	7.851	0.426	0.730
LSTM	8.428	0.299	0.714
GRU	8.939	0.191	0.698
ens(MOR)	7.853	0.441	0.698
ens(OLS)	7.392	0.498	0.698
ens(LASSO)	7.474	0.495	0.714
ens(FFNN)	7.313	0.508	0.730
ens(LSTM)	7.690	0.460	0.714
ens(GRU)	7.915	0.422	0.698

Table C.34: Results from stage 2 for selected evaluation metrics at extreme solar power forecasts made at spot time, i.e. only consider predictions when $S\hat{O}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	8.290	0.428	0.802
OLS	8.846	0.274	0.786
LASSO	8.833	0.262	0.786
FFNN	8.820	0.251	0.770
LSTM	9.067	0.250	0.825
GRU	9.219	0.187	0.841
ens(MOR)	9.337	0.271	0.786
ens(OLS)	8.412	0.392	0.794
ens(LASSO)	8.524	0.357	0.794
ens(FFNN)	8.374	0.387	0.778
ens(LSTM)	8.563	0.345	0.786
ens(GRU)	8.652	0.316	0.810

Table C.35: Results from stage 2 for selected evaluation metrics at extreme consumption power forecasts made at spot time, i.e. only consider predictions when $\hat{CO}^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	11.388	0.413	0.778
OLS	11.934	0.277	0.746
LASSO	11.841	0.265	0.778
FFNN	11.737	0.261	0.778
LSTM	12.273	0.259	0.738
GRU	12.492	0.192	0.746
ens(MOR)	12.393	0.270	0.730
ens(OLS)	11.500	0.382	0.770
ens(LASSO)	11.536	0.351	0.770
ens(FFNN)	11.437	0.379	0.770
ens(LSTM)	11.774	0.343	0.754
ens(GRU)	11.846	0.313	0.754

Table C.36: Results from stage 2 for selected evaluation metrics at extreme system imbalances, i.e. only consider predictions when $SI_9^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	12.386	0.400	0.738
OLS	12.508	0.274	0.746
LASSO	12.675	0.259	0.746
FFNN	12.682	0.253	0.754
LSTM	12.383	0.277	0.746
GRU	12.603	0.196	0.754
ens(MOR)	12.728	0.274	0.714
ens(OLS)	12.402	0.372	0.738
ens(LASSO)	12.473	0.342	0.746
ens(FFNN)	12.410	0.368	0.738
ens(LSTM)	12.250	0.348	0.738
ens(GRU)	12.404	0.311	0.738

Table C.37: Results from stage 2 for selected evaluation metrics at extreme traversals of the bid-offers, i.e. only consider predictions when $MO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

Model	**MAE**	$\mathbf{R}^2$	**Correct sign**
Naive	11.088	0.444	0.762
OLS	11.837	0.306	0.770
LASSO	11.843	0.296	0.794
FFNN	11.995	0.296	0.754
LSTM	12.374	0.289	0.754
GRU	12.772	0.209	0.754
ens(MOR)	12.482	0.299	0.738
ens(OLS)	11.228	0.415	0.754
ens(LASSO)	11.390	0.384	0.778
ens(FFNN)	11.236	0.412	0.762
ens(LSTM)	11.535	0.381	0.746
ens(GRU)	11.743	0.345	0.746

Table C.38: Results from stage 2 for selected evaluation metrics at extreme changes in traversals of the bid-offers, i.e. only consider predictions when $\Delta MO_8^{d,h}$ is outside the 5 and 95 percentile levels in the test set.

D Metrics evaluated on full intraday prices

This section evaluates on the full back-transformed prices, i.e. the sum of the spot price and predicted deltas since spot. The best MAE from each stage will also be plotted.

D.1 Stage 1

Model	MAE	R^2	Correct sign
MOR	9.198	0.547	0.902
OL	9.180	0.552	0.909
LASSO	9.122	0.554	0.912
FFNN	9.094	0.546	0.909
LSTM	8.938	0.565	0.908
GRU	8.965	0.556	0.909

Table D.1: Results from stage 1 for selected evaluation metrics on full intraday prices.

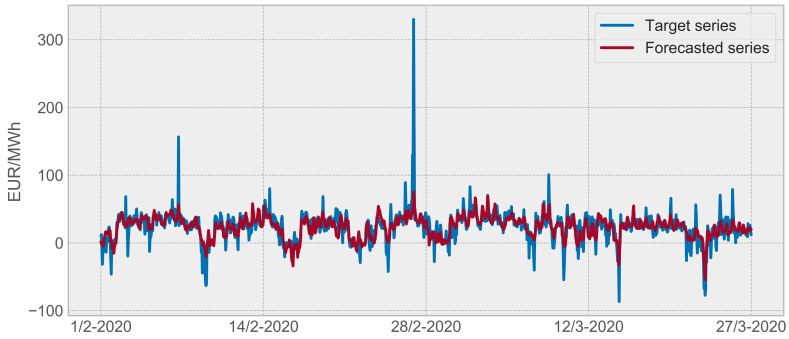

Figure D.1: LSTM stage 1 forecast full intraday prices.

D.2 Stage 2

Model	**MAE**	**R^2**	**Correct sign**
Naive	7.545	0.691	0.921
OLS	7.651	0.659	0.915
LASSO	7.648	0.657	0.919
FFNN	7.659	0.656	0.916
LSTM	7.924	0.643	0.913
GRU	7.926	0.627	0.914
ens(MOR)	7.971	0.648	0.924
ens(OLS)	7.551	0.685	0.921
ens(LASSO)	7.568	0.678	0.922
ens(FFNN)	7.534	0.684	0.918
ens(LSTM)	7.632	0.674	0.914
ens(GRU)	7.640	0.667	0.913

Table D.2: Results from stage 2 for selected evaluation metrics on full intraday prices.

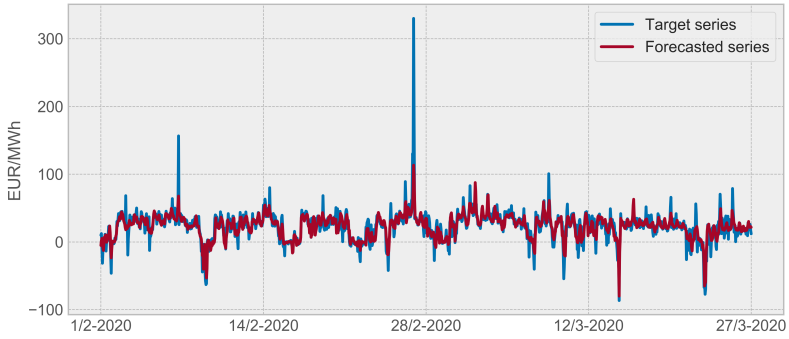

Figure D.2: ens(FFNN) stage 2 forecast on full intraday prices.

E Plots of all forecasts

E.1 Stage 1

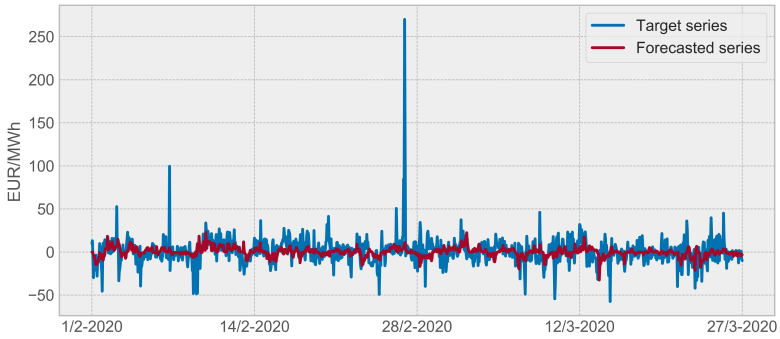

Figure E.1: MOR stage 1 forecast.

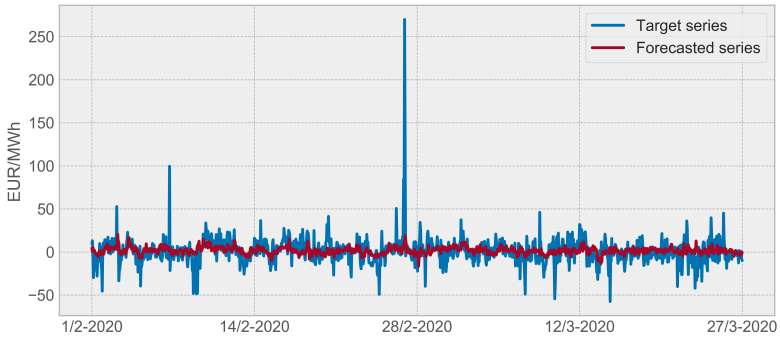

Figure E.2: OLS stage 1 forecast.

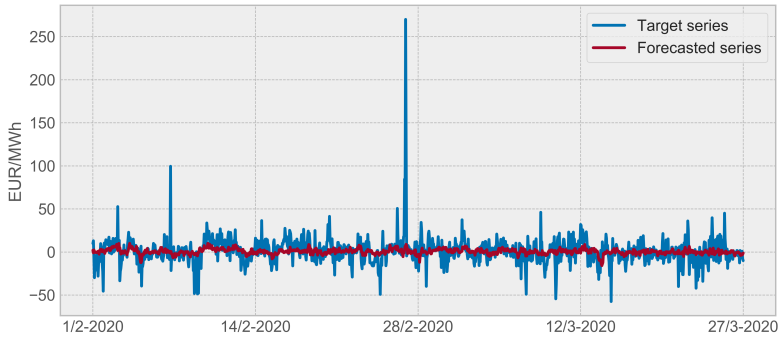

Figure E.3: LASSO stage 1 forecast.

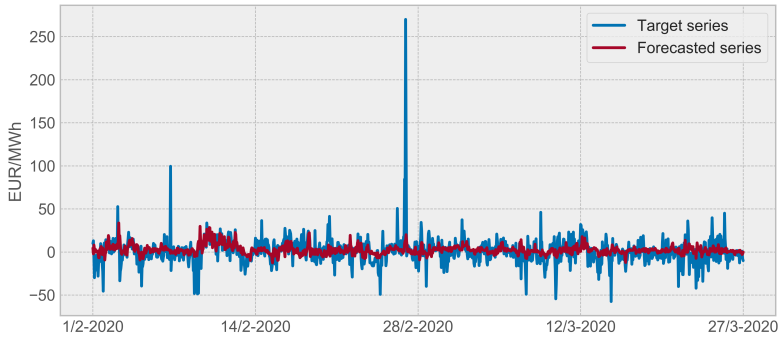

Figure E.4: FFNN stage 1 forecast.

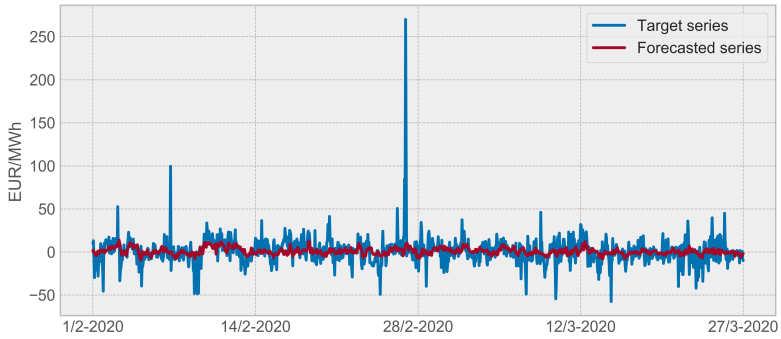

Figure E.5: GRU stage 1 forecast.

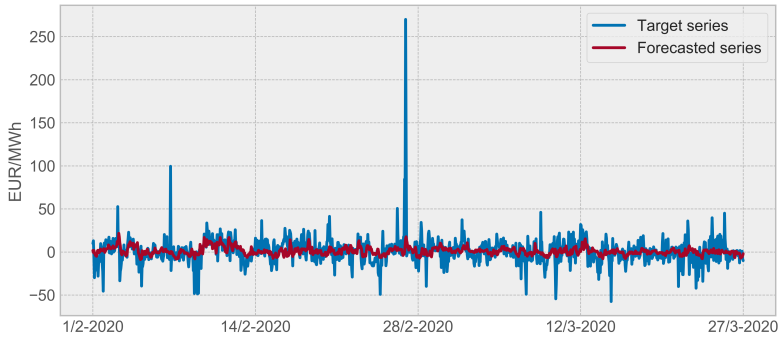

Figure E.6: LSTM stage 1 forecast.

E.2 Stage 2

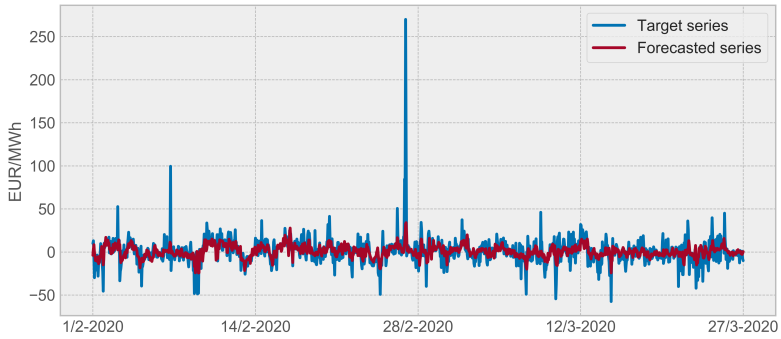

Figure E.7: OLS stage 2 forecast.

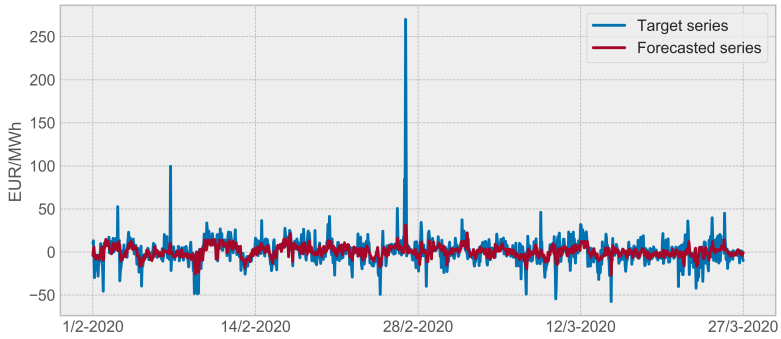

Figure E.8: LASSO stage 2 forecast.

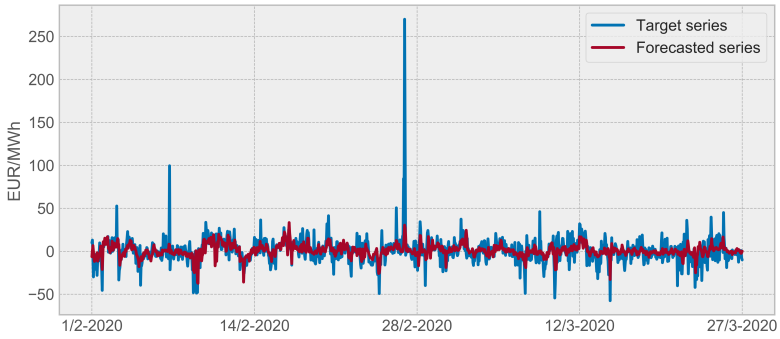

Figure E.9: FFNN stage 2 forecast.

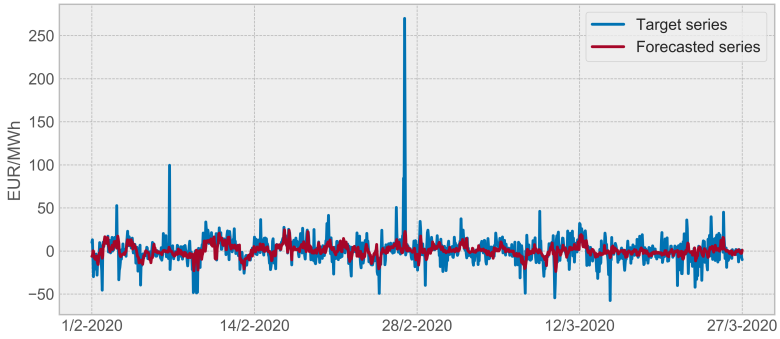

Figure E.10: GRU stage 2 forecast.

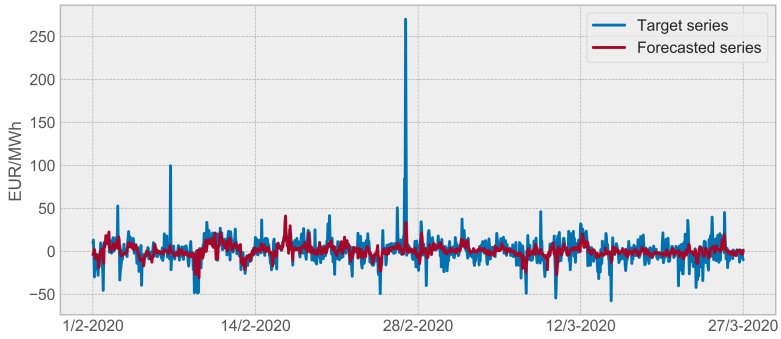

Figure E.11: LSTM stage 2 forecast.

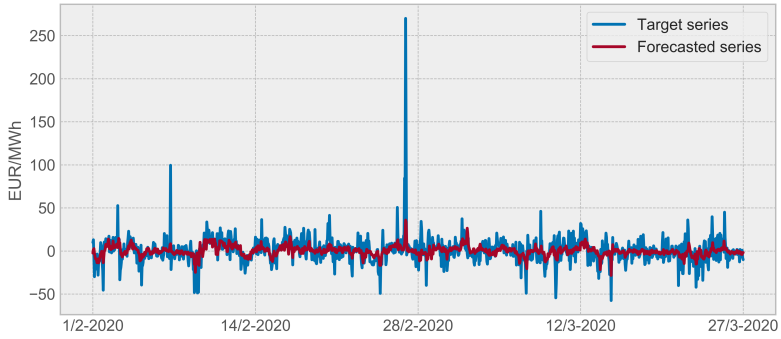

Figure E.12: ens(MOR) stage 2 forecast.

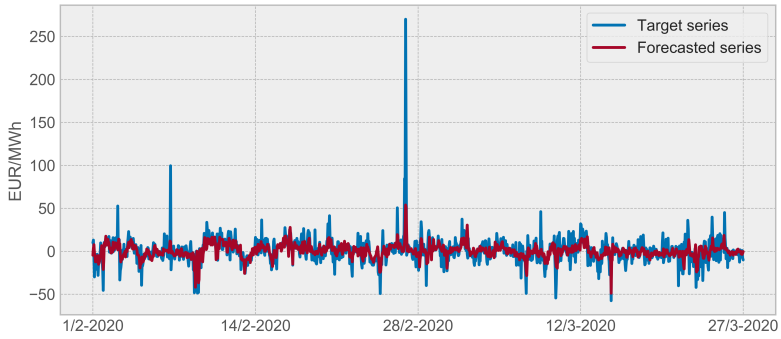

Figure E.13: ens(OLS) stage 2 forecast.

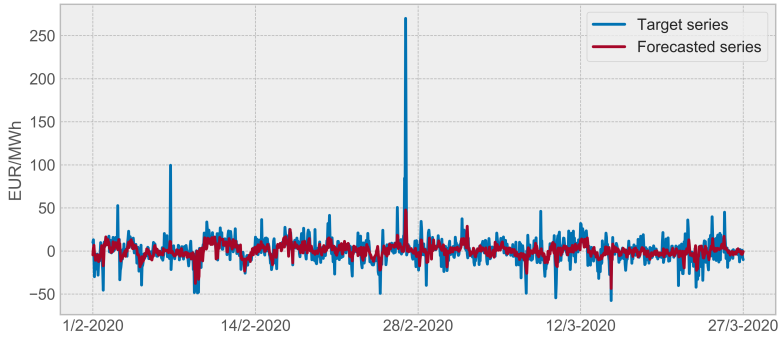

Figure E.14: ens(LASSO) stage 2 forecast.

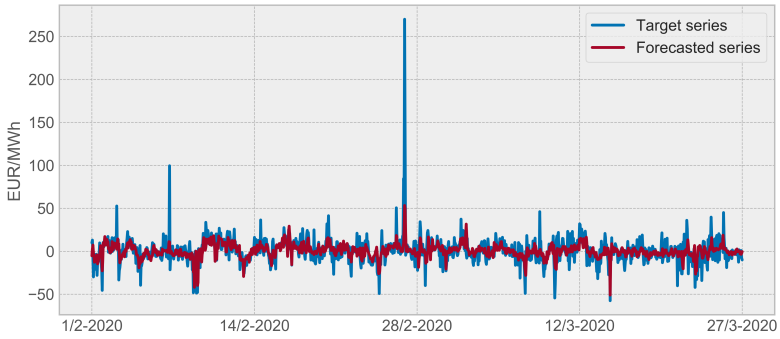

Figure E.15: ens(FFNN) stage 2 forecast.

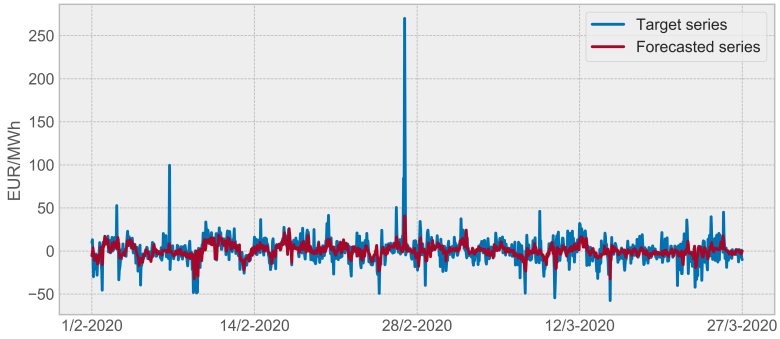

Figure E.16: ens(GRU) stage 2 forecast.

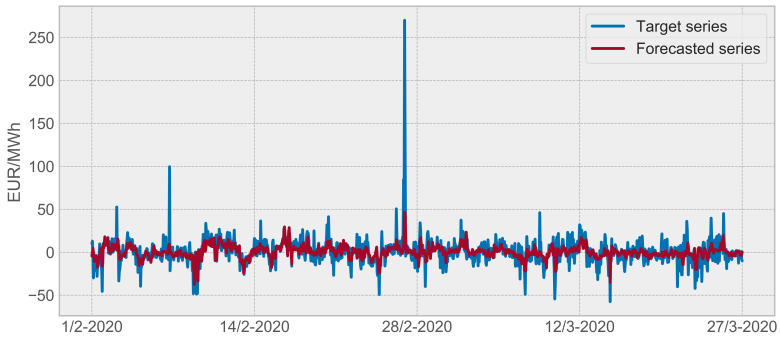

Figure E.17: ens(LSTM) stage 2 forecast.

Simen Ullern & Vebjørn Przytula Fjeldberg

Predicting Final Intraday Electricity Prices

NTNU
Norwegian University of
Science and Technology