Electricity Spot Price Forecast: In Depth Analysis of Different Machine Learning Methods

Julian Studt [1], Mattias Hadlak [1], Merten Schuster [1], Henrik Herr [1], Bernd Engel [1]

[1] elenia Institute for High Voltage Technology and Power Systems, Brunswick, Germany, j.studt@tu-bs.de

Abstract

Electricity price forecasts are crucial for portfolio and risk management in electricity markets. Especially in the field of machine learning methodologies, a multitude of techniques to implement electricity price forecasts exist. Comparisons between methods often focus on the prediction error and disregard other important aspects like performance and usability. The aim of this publication is to provide a holistic analysis of data-driven methodologies by looking at additional factors like the search for hyperparameters and model size. The statistical *seasonal autoregressive integrated moving average with exogenous inputs* model and the *autoregressive with exogenous inputs* model are chosen as a benchmark. They are compared to a *feed-forward neural network* and a *recurrent neural network*. The case study shows that neural networks are not always the most suitable choice for price forecasts. In fact, the search for fitting hyperparameters of the neural networks is time intensive and difficult compared to the statistical models. The results also show that linear models yield lower forecasting errors when a small number of features, such as residual load and past prices, are used as input parameters for electricity price forecasting.

1 Introduction

1.1 Motivation

The forecasting of electricity spot market prices is crucial in the energy trading and risk management chain. At the same time a lot of different methods and technologies to perform those forecasts exist, further increasing the complexity. Most of the literature that analyses different methods focus on the forecast error as the main benchmark indicator. Even though it is the most prominent indicator for the quality of a forecasting model it does not show the whole picture of each technique. Additional factors e.g. scalability, maintainability and performance could play a role in the selection process of a forecasting model. For this reason, the aim of this publication is to look into different methods more closely and investigate additional indicators for comparison. As a benchmark, the statistical *autoregressive with exogenous inputs (ARX)* model and *seasonal autoregressive integrated moving average with exogenous inputs (SARIMAX)* model are used. They are compared to two neural network structures: A *feed-forward neural network (FNN)* and a *recurrent neural network (RNN)*. To fully assess these methods the hyperparameter search will be discussed to answer the question on how difficult it is to find the best configuration and on how much time it takes. Furthermore, the parameters of the different methods are analysed and compared by building a connection between the models, which helps clearly identify differences and similarities. The parameter analysis is supported by the analysis of the forecasting errors for the methods for different data sets and different timeframes.

1.2 Literature Review

Electricity spot price forecasts have already been described in various publications. [1] and [2] give a good overview of various methods, that are commonly used. A distinction can be loosely made between statistical, machine learning and hybrid methods. Statistical methods are a common approach and often used as benchmark like in [3, 4], where general *ARX* or *autoregressive integrated moving average (ARIMA)* type models are used. Machine learning methods became more popular in recent years and promise better forecasts due to advanced development of computing power. Often times statistical methods are combined with machine learning methods like in [4], where an autoregressive *LASSO* and *Elastic Net* model are build. In [5] an *ARIMA* model is combined with a *support vector regression* model for short term forecasting. More advanced methods are for example chosen in [3], where a neural network is built that uses a *k nearest neighbours* algorithm to select

features. In [2] other neural networks like *convolutional neural networks (CNN)* and *long short term memory (LSTM)* networks are built. A distinction here is that spot prices for different countries are predicted simultaneously with the neural networks, to consider cross border trades in the forecasts.

This paper begins by introducing the two statistical models ARX and SARIMAX that are used as a benchmark. Afterwards two machine learning models, namely the FNN and RNN are shown and explained in comparison to the statistical models in subchapter 2.2. A short overview of cost functions and optimization procedures is given in subchapter 2.3. In subchapter 2.4 the general method used for finding the hyperparameters is described. Subchapter 3.1 discusses the results of this hyperparameter search in terms of complexity and computation time. In the following subchapter 3.2 the key figures of each model are shown. Afterwards, in subchapter 3.3, the prediction error for the models are discussed from different perspectives. To get more insight into how these models function the subchapter 3.4 provides a look into the actual parameters that map inputs and outputs. The final conclusion is drawn in chapter 4.

2 Problem, Methods and Hyperparameter Optimization

2.1 DayAhead Price Forecast

The DayAhead Spot auction takes place every day and determines hourly prices for the next full day. Participants of the auction submit their bids to buy or sell a certain amount of electricity for a certain price to the market. The spot market orders the sell and buy bids and uses a special algorithm to determine a unified price for which all bids within the price range are executed. The task of the DayAhead spot price forecast is to predict this unified price for each hour of the next day considering only information that is available at gate closure. In this case the prices of the previous days, the load prediction for the whole German market area and the *renewable energy sources (RES)* prediction coming from photovoltaic and wind production are used.

2.2 Methods

To analyse the machine learning methods, they are compared to statistical methods. In this context often used models are the ARX models as well as the SARIMAX. These models are going to be compared with the two neural network structures FNN and RNN.

The **ARX** model uses past values of the target variable, so called endogenous values (the *AR* part) and exogenous values (the *X* part) that influence the target. Together they form a linear regression model that can be simply extended to predict multiple values, which leads to equation (1).

$$\boldsymbol{y}_t = \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{\varepsilon}_t \qquad (1)$$

The vector $\boldsymbol{x}_t$ is the input at time *t* consisting of past values of $\boldsymbol{y}_t$ as well as the exogenous variables. The parameter matrix $\boldsymbol{W}$ maps the inputs to the outputs and $\boldsymbol{\varepsilon}_t$ describes the forecasting error at time *t*. The model is shown as a graph in **Figure 1**. On the left side are the input values in red that are mapped to the target values in blue via the black box, that describes the linear relation.

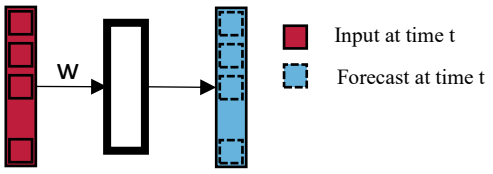

Figure 1 Graphical representation of an ARX model

The **SARIMAX** model is a more general version of the ARX model with additional options. Besides the already described AR and X part it consists of a seasonal (*S*), an integrated (*I*) and moving average (*MA*) part. Seasonal means that two frequencies of values are used. A time series in hourly resolution can thus be described by hourly time steps and daily time steps to consider effects between consecutive hours and consecutive days. The I part is a form of pre-processing, that is used for non-stationary time series by calculating the difference between consecutive values. To solve the resulting equation, it has to be integrated, thus the I. The MA part considers

past forecasting errors and therefore introduces a recursive loop in the model as seen in **Figure 2**.

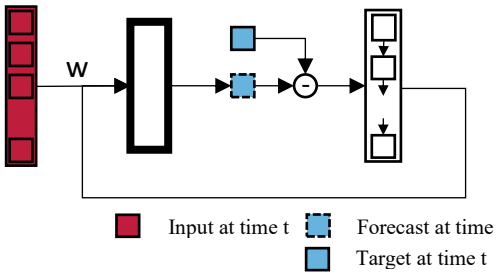

Figure 2 Simplified graphical representation of a SARIMAX model

As shown in Figure 2, the prediction is compared to the real value and then saved for several steps, marked by the structure to the right, depending on the MA hyperparameter. In order to use a similar representation throughout the paper, the model is written in state space form, shown in (2).

$$\begin{aligned} \boldsymbol{h}_t &= \boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}\boldsymbol{h}_{t-1} \\ y_t &= \boldsymbol{v}^T\boldsymbol{h}_t + \varepsilon_t \end{aligned} \qquad (2)$$

The vector $\boldsymbol{h}_t$ is called state vector that gets updated in each step by a combination of new inputs $\boldsymbol{x}_t$ and its past state. New predictions y_t are made by a linear combination of the state vector $\boldsymbol{h}_t$ with the vector $\boldsymbol{v}$. The state space representation is not unique, one possibility will be shown in subchapter 3.4.

The neural networks are similar to the statistical methods. The most important difference is, that they do not make any assumptions about the relationship between the inputs and the outputs. On the one hand this is an advantage cause theoretically every relationship can be modelled. On the other hand, this high flexibility makes it difficult to find a suitable configuration for a neural network. Especially the optimization process can be long and complex.

The similarity between the **FNN** and the ARX model are obvious by comparing Figure 1 and **Figure 3**. In Figure 3 it is shown, that the neural network uses the same inputs as the ARX model.

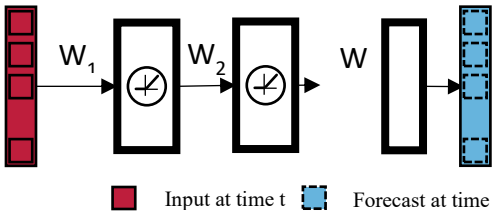

Figure 3 Graphical representation of the FNN model

The nonlinearity is introduced by the *rectified linear unit (ReLu)* function in each layer. By chaining several layers an arbitrary complexity can be obtained. Noting the ReLu function by g, a network with two hidden layers and one output layer is shown in (3).

$$\boldsymbol{y}_t = \boldsymbol{W}\big(g(\boldsymbol{W_2}g(\boldsymbol{W_1}\boldsymbol{x}_t))\big) + \boldsymbol{\varepsilon}_t \qquad (3)$$

A particular concept of a neural network is the **RNN** which is used for reading in sequences. Using the new input and an internal state it predicts the next state as seen in **Figure 4**.

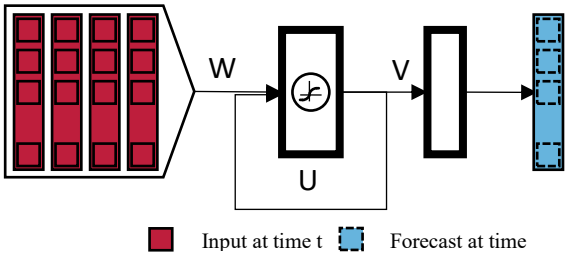

Figure 4 Graphical representation of the RNN model

Due to the recursion there is a resemblance with the SARIMAX model in Figure 2. This is further highlighted by looking at the *RNN* equation (4).

$$\begin{aligned} \boldsymbol{h_t} &= g(\boldsymbol{W}\boldsymbol{x_t} + \boldsymbol{U}\boldsymbol{h_{t-1}}) \\ \boldsymbol{y_t} &= \boldsymbol{V}\boldsymbol{h_t} + \boldsymbol{\varepsilon_t} \end{aligned} \qquad (4)$$

The nonlinear activation function g of an RNN is the *tanh*.

The normal RNN has the vanishing or exploding gradient problem during optimization, which means that due to the optimization procedure it poses problems for long sequences [6]. For this reason, the *long short-term memory (LSTM)* network is used. In this paper during the RNN optimization process, also LSTM networks will be tested.

2.3 Cost Function and optimization

The function that maps the chosen inputs $\boldsymbol{x}$ to the outputs $\boldsymbol{y}$ is called f as shown in (5).

$$\boldsymbol{y}_\mathrm{t} = f(\boldsymbol{x}_\mathrm{t}; \boldsymbol{W}) + \boldsymbol{\varepsilon}_\mathrm{t} \tag{5}$$

To find the optimal parameters $\boldsymbol{W}$ of each function f that maps the chosen inputs $\boldsymbol{x}_\mathrm{t}$ to the outputs $\boldsymbol{y}_\mathrm{t}$, a cost function C is defined, that has to be minimized. It uses all N training sample inputs $\boldsymbol{X} = [\boldsymbol{x_1}\ \boldsymbol{x_2}\ \ldots \boldsymbol{x_N}]$ and the matching outputs $\boldsymbol{Y} = [\boldsymbol{y_1}\ \boldsymbol{y_2}\ \ldots \boldsymbol{y_N}]$ **:**

$$\underset{\boldsymbol{W}}{\operatorname{argmin}}\, C(f(\boldsymbol{X}; \boldsymbol{W}), \boldsymbol{Y}) \tag{6}$$

A typical cost function is the mean squared error (MSE) respectively its root (*RMSE*). The MSE for all training samples N is defined as in (7):

$$C_{MSE}(f(\boldsymbol{X}; \boldsymbol{W}), \boldsymbol{Y}) = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{y_i} - f(\boldsymbol{x_i}; \boldsymbol{W}))^2 \tag{7}$$

To find the minimum parameters in the ARX model a linear equation system is solved, which is known as the ordinary least squares problem. For SARIMAX a slightly more complex recursive algorithm is employed, since the prediction errors of the MA part are not known while building the model.

The neural networks in this paper use the same cost function with an additional regularization term:

$$C_{NN}(f(\boldsymbol{X}; \boldsymbol{W}), \boldsymbol{Y}) = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{y_i} - f(\boldsymbol{x_i}; \boldsymbol{W})\right)^2 + \lambda\|\boldsymbol{W}\|_1 \tag{8}$$

The term $\lambda\|\boldsymbol{W}\|_1$ sets low weights w to zero depending on the size of the hyperparameter λ. It is commonly referred to as *L1* regularization and is supposed to avoid overfitting. If the term is used in an ARX model it is called the *ARX-LASSO* model and it has the effect of an input selection since less important input weights are set to zero.

Due to their size and nonlinearity neural networks are usually optimized with a stochastic gradient descent algorithm, where the gradient of the cost function is calculated in each iteration and propagated backwards through the network. This algorithm is either stopped after a fixed amount of iterations or when there are no more improvements in the cost function. In this paper early stopping is used, which measures the cost function on the training set and on a validation set simultaneously. If the two cost functions differ too much the algorithm stops to avoid overfitting.

2.4 Hyperparameter optimization

This subchapter further explains the model hyperparameters and the methodology used to find them. An overview of all hyperparameters under consideration is given in **Table 1**.

Table 1 Hyperparameter options

Model	Hyperparameters
ARX	endogenous history (AR), exogenous history (X), intraday effects
SARIMAX	seasonality (S) endogenous history (AR), differencing (I), prediction history (MA), exogenous values (X)
FNN	endogenous history, exogenous history, intraday effects, layers, neurons per layer, L1 regularization
RNN	endogenous history, exogenous history, intraday effects, neurons per layer, LSTM, L1 regularization

For the ARX model the number of past endogenous as well as exogenous values can be chosen independently from each other. The model can either predict a whole day at once, which in practice means that effects between the hours of the day are considered or it can predict just one hour of a day, which would not consider effects between hours.

For the SARIMAX model the AR, MA and I parameters are found via the *Box-Jenkins* method [8]. Effects between different hours are considered via the seasonality: In contrast to the ARX model, the final model takes the value from the previous hour and from the previous day. ARX only uses values from previous days.

The FNN model can use the same input options as the ARX model. On top the number of layers and the number of neurons per layer can be chosen. To simplify the available options, the number of parameters in each layer is equal. In the search the option of zero layers will also be considered which would make the model a linear ARX model optimized by a stochastic gradient descent algorithm.

The RNN model only considers one layer with a varying number of neurons. The exogenous history has to be the same length as the endogenous history to properly read them into the network together. To consider long sequences the option of a LSTM network is also considered.

In order to find the optimal hyperparameters out of this selection for each model, the following process is used:

1. Split data in:
   - Training set (01/01/2015 – 31/09/2017)
   - Validation set (01/10/2017 – 31/09/2018)
   - Test set (01/10/2018 – 31/09/2019)
2. Choose possible hyperparameters for each model: To narrow down possible parameters literature results are considered and data analysis are performed
3. Train all models on training set
4. Test the model RMSE on validation set
5. Choose the models with lowest validation RMSE
6. Retrain models with the best configuration on training and validation set together
7. Check the model RMSE on test set

The final error on the test sets indicates how good the model predictions are for unseen data.

3 Results

3.1 Hyperparameter search

The main indicators for the search of hyperparameters are the tested combinations and the computation time, which is highly dependent on hardware, software library and chosen optimizers. This does not make it a fully objective criterion but nevertheless it is important to note since it shows a clear indication.

- ARX – 220 combinations – 194s: The number of tested combinations is relatively low since there are not too many options available. The model was built with the linear regression model of the *Scikit-learn* library [7]. The model training is quite fast with each combination tested in less than one second.
- SARIMAX – 2 combinations – 40s: The Box-Jenkins method gives a guideline on how to identify possible models by data analysis, which left only two options to be tested. The model was built with the *statsmodels* library [9]. The training for one model is relatively long compared to the ARX model and depends a lot on the length of the history and the chosen optimization method.
- FNN – 400 combinations – 197min: A high number of hyperparameter combinations are possible. The search got stopped limited to 400 randomly chosen combinations. The model was built with the *TensorFlow* library [10]. Optimization time is mainly dependant on the number of layers of the network. No GPU was used for the optimization.
- RNN – 400 combinations – 18h: A high number of hyperparameter combinations are possible. The search got limited to 400 randomly chosen combinations. The additional optimization time arises from the temporal dimension that is introduced by the recurrent layer. Instead of a simple backpropagation algorithm the backpropagation through time algorithm is used by the *TensorFlow* library.

3.2 Key facts of models

The models resulting from the procedure described in subchapter 2.4 are shown in **Table 2**.

Table 2 Optimal hyperparameters for each model.

Model	Hyperparameter	Value
ARX	endogenous history	18
	exogenous history	17 +1
	intraday effects	no
SARIMAX	seasonality	24
	endogenous history	1
	differencing	1 seas.
	prediction history	2 seas.

FNN	endogenous history	3
	exogenous history	2+1
	intraday effects	yes
	layers	0
	neurons per layer	-
	L1 regularization	0.001
RNN	endogenous history	10
	exogenous history	9+1
	intraday effects	yes
	neurons per layer	170
	LSTM	no
	L1 regularization	0.001

The "+1" in exogenous history means that the exogenous value at prediction time is used.

The ARX model uses a long history of both endogenous and exogenous values but does not model intraday effects. SARIMAX models the intraday effects via the seasonality. For each prediction it uses the prediction of the previous hour and the true value from the same hour the day before. In contrast to the ARX model the used history is low. Only values of the past two days are considered. The FNN model does not have any hidden layer which makes it a linear vector ARX model with *L1* regularization, which is a LASSO model similar to [11]. To avoid confusion the term FNN is continued to be used. For the RNN, a moderate length history is used whereas the number of neurons in the hidden layer is relatively large. Both neural networks model intraday effects by full day predictions.

3.3 Error Analysis

In **Figure 5** the error for all models on the different data sets is shown.

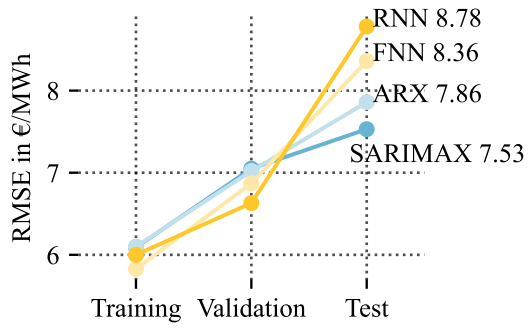

Figure 5 RMSE on the different sets for all models

As expected all the models show the lowest error on the training set followed by the validation set. Especially the neural networks show low errors on these sets. But in contrast to the statistical models they perform significantly worse on the test set. The poor performance is an indication that the neural networks are overfitting despite the regularization measures taken. It also shows that a good validation technique is necessary for training neural networks to avoid misleading results. Apparently too much information of the validation set was introduced into the neural networks during the hyperparameter optimization.

To further analyse the errors and compare the methods, **Figure 6** shows the RMSE for each hour of the predicted day on the test set.

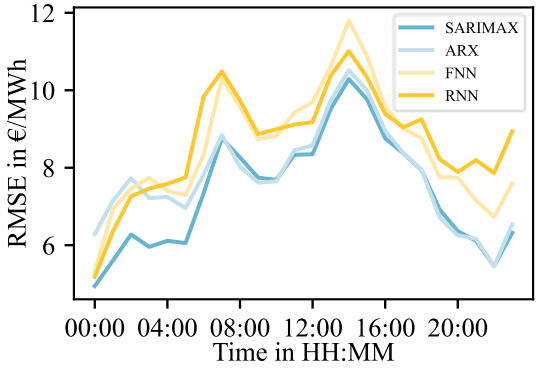

Figure 6 RMSE for each hour of the day ahead prediction

All models show a similar error curve. During the day, especially the morning and the afternoon, errors are highest, which is due to high price volatility in these hours. The neural networks show bigger errors in almost every hour compared to the statistical models. Only exception is the morning, where they are performing better than the ARX model. The reason is the considered intraday effect. This becomes especially apparent if ARX is compared to SARIMAX, since they show almost the same performance after the first hours in the morning. It leads to the conclusion that the intraday effects are only important for the morning hours, which is also found in [11]. The behaviour can be explained by the SARIMAX model. It models intraday effects by the autoregression with values of the previous hour. Since the model predicts 24 hours, only the first hour has the exact value of the previous hour. The next predictions have to use the predicted value and so forth. As an example, in

the following equation (9) only the AR part is considered.

$$y_t = a\, y_{t-1} + \boldsymbol{w}^T \boldsymbol{x_t} + \varepsilon_t \qquad (9)$$

To forecast the value at t+1 the prediction $\hat{y}_t = y_t - \varepsilon_t$ is used:

$$\begin{aligned} y_{t+1} &= a\, \hat{y}_t + \boldsymbol{w}^T \boldsymbol{x}_t + \varepsilon_{t+1} \\ y_{t+1} &= a\,(a y_{t-1} + \boldsymbol{w}^T \boldsymbol{x}_t) + \boldsymbol{w}^T \boldsymbol{x}_{t+1} \\ &\quad + \varepsilon_{t+1} \end{aligned} \qquad (10)$$

The parameter a < 1 for stationary process which makes the influence of the price y_{t-1} exponentially decreasing in each step. For this reason, the autoregression during the day is only relevant in the morning hours.

3.4 Parameter Analysis

For further analysis of the models the parameters found during the training process are analysed. In **Figure 7** the parameter matrix of the FNN is visualized in a heatmap split according to the input it maps to the output.

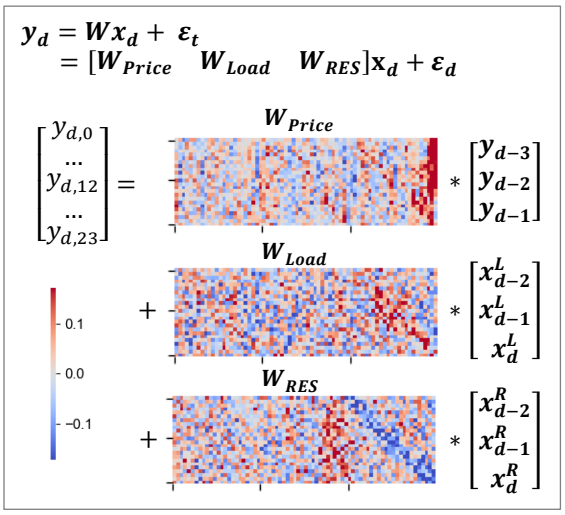

Figure 7 The parameter matrix of the FNN model visualized in colours.

The first matrix maps the historic prices to the output, the second maps the load prediction and its history and the last the RES infeed. As expected the RES prediction clearly has an effect on the prices at the same hour. This is indicated by the blue diagonal in the last heatmap. A negative value (blue) means that a high predicted renewable energy infeed leads to lower prices, which is plausible. For the load forecast in the middle, the pattern is less obvious but visible in red. For prices a different pattern arises. The biggest influence is apparently indicated by the prices of the last hour of the previous day. This matches with the result from the previous chapter, where neural networks performed better in the morning as the ARX model that does not use this price. Even though using more previous days has been discovered beneficial by the hyperparameter search, the parameters here can't be interpreted and it is not clear if they give any benefit to the prediction.

The RNN has multiple parameter matrices, as seen in subchapter 2.2. **Figure 8** shows the parameter matrix $\boldsymbol{U}$ of the recurrent layer that maps the old state to the new state.

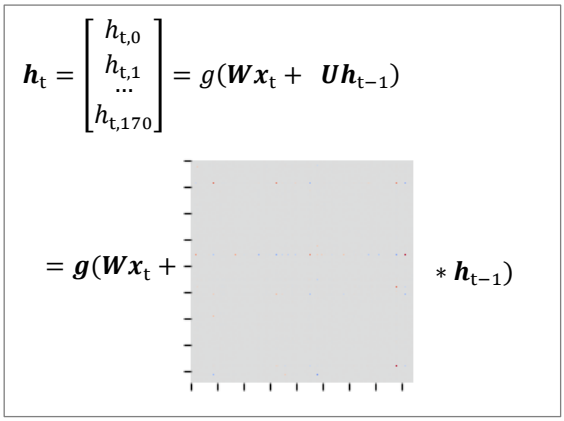

Figure 8 The parameter matrix of the recurrent layer visualised in colours.

Grey means that parameters are zero, so almost all of the parameters are zero. Except a few coloured dots there is no pattern visible. Most of the parameters are close to zero, which means that the transition matrix of this layer is sparse. To interpret this an understanding for the hidden state is needed. In chapter 2.1 the similarity between the RNN and SARIMAX model in state space form was stated. In consequence the matrix $\boldsymbol{U}$ of the SARIMAX and especially the state vector $\boldsymbol{h_t}$ has to be sketched. As already stated this representation is not unique. To simplify the problem, the state equation of an of an ARMA(p,q) model according to [12] is introduced, which gives enough information for a comparison. With $r = \max\{p, q+1\}$:

$$\boldsymbol{h_t} = \begin{pmatrix} \varphi_1 & \varphi_2 & \dots & \varphi_{r-1} & \varphi_r \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix} \boldsymbol{h_{t-1}} + \begin{pmatrix} \varepsilon_t \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad (11)$$

And the observation equation:

$$y_t = (1 \quad \theta_1 \quad \dots \quad \theta_{r-1})\boldsymbol{h_{t-1}} \qquad (12)$$

As seen in the state equation a sparse matrix is obtained to transition between two states, similar to what can be seen in Figure 8. The inner state $\mathbf{h_t}$ considers the history and would be 48 for the SARIMAX model found in this paper. It leads to the conclusion that the RNN learns a SARIMAX like structure. The main difference between the RNN and SARIMAX is that the structure is not explicitly given from the beginning. The RNN therefore has more flexibility but is also more sensitive to errors since this structure has to be learned first which leads to noisy solutions.

4 Conclusion

The study has shown that *feed-forward neural networks* and *recurrent neural networks* have higher forecasting errors than simpler statistical models when past prices, load forecast and renewable energy forecast are used as input parameters. Besides that, they have additional downsides.

The search for optimal neural network hyperparameters is complex, since there are no clear guidelines on how to set up such a model, like the Box-Jenkins method for a SARIMAX model. This leads to a variety of options that have to be tested, each of them being a time-consuming optimization itself.

For machine learning methods a suitable validation strategy must be chosen since they normally have a high number of parameters, which favours overfitting. A more exact modelling and feature engineering similar to SARIMAX can help to reduce parameters. E.g., instead of using a full day prediction it is sufficient to use the last price of the previous day for intraday effects, as it was shown.

The statistical models only outperform the neural networks if the assumptions of linearity are sufficiently met. For renewable energy forecast, load forecast and historic prices used as inputs it is the case. This was also apparent because the FNN hyperparameter search in the end found a linear model. Theoretically the very flexible neural network can model these dependencies as well, which was shown in the parameter analysis, but due to the high number of parameters and the optimization technique noise is introduced into the solution. It means that neural networks have to learn the dependencies of statistical models first. Combined with the validation problem they are not suitable for models that can be described explicitly. Their strength is detecting patterns that cannot be described easily with an explicit model.

This leads to the conclusion that machine learning models for spot price predictions are not always the best option. They are very difficult to handle and depending on the available data, simpler models deliver better forecasting results with much less effort put in hyperparameter searches. In general it is always beneficial to use a simple model as benchmark and try to employ neural networks for cases where a pattern cannot be modelled explicitly.

5 References

[1] R. Weron, “Electricity price forecasting: A review of the state-of-the-art with a look into the future,” *International Journal of Forecasting*, vol. 30, no. 4, pp. 1030–1081, 2014, doi: 10.1016/j.ijforecast.2014.08.008.

[2] J. Lago, F. de Ridder, and B. de Schutter, “Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms,” *Applied Energy*, vol. 221, pp. 386–405, 2018, doi: 10.1016/j.apenergy.2018.02.069.

[3] D. Keles, J. Scelle, F. Paraschiv, and W. Fichtner, “Extended forecast methods for day-ahead electricity spot prices applying artificial neural networks,” *Applied Energy*, vol. 162, pp. 218–230, 2016, doi: 10.1016/j.apenergy.2015.09.087.

[4] B. Uniejewski, J. Nowotarski, and R. Weron, “Automated Variable Selection and Shrinkage for Day-Ahead Electricity Price Forecasting,” *Energies*, vol. 9, no. 8, p. 621, 2016, doi: 10.3390/en9080621.

[5] J. Che and J. Wang, “Short-term electricity prices forecasting based on support vector regression and Auto-regressive integrated moving average modeling,” *Energy Conversion and Management*, vol. 51, no. 10, pp. 1911–1917, 2010, doi: 10.1016/j.enconman.2010.02.023.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*: MIT Press, 2016.

[7] F. Pedregosa *et al.,* “Scikit-learn: Machine Learning in Python,” *Journal of Machine*

Learning Research, vol. 12, pp. 2825–2830, 2011.

[8] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control,* 5th ed. Hoboken: Wiley, 2015.

[9] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in *9th Python in Science Conference*, 2010.

[10] Martíın Abadi *et al., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* [Online]. Available: https://www.tensorflow.org/

[11] F. Ziel, "Forecasting Electricity Spot Prices using Lasso: On Capturing the Autoregressive Intraday Structure," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4977–4987, 2016, doi: 10.1109/TPWRS.2016.2521545.

[12] J. D. Hamilton, *Time Series Analysis*: Princeton University Press, 1994.