Delft University of Technology

Delft Center for Systems and Control

Technical report 18-001

Forecasting spot electricity prices: Deep learning approaches and empirical comparison of traditional algorithms*

J. Lago, F. De Ridder, and B. De Schutter

If you want to cite this report, please use the following reference instead:
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, July 2018. doi:10.1016/j.apenergy.2018.02.069

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/18_001.html

Forecasting spot electricity prices: deep learning approaches and empirical comparison of traditional algorithms

Jesus Lago[a,b,*], Fjo De Ridder[b], Bart De Schutter[a]

[a] *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, Delft, The Netherlands*
[b] *Algorithms, Modeling, and Optimization, VITO, Energyville, ThorPark, Genk, Belgium*

Abstract

In this paper, a novel modeling framework for forecasting electricity prices is proposed. While many predictive models have been already proposed to perform this task, the area of deep learning algorithms remains yet unexplored. To fill this scientific gap, we propose four different deep learning models for predicting electricity prices and we show how they lead to improvements in predictive accuracy. In addition, we also consider that, despite the large number of proposed methods for predicting electricity prices, an extensive benchmark is still missing. To tackle that, we compare and analyze the accuracy of 27 common approaches for electricity price forecasting. Based on the benchmark results, we show how the proposed deep learning models outperform the state-of-the-art methods and obtain results that are statistically significant. Finally, using the same results, we also show that: (i) machine learning methods yield, in general, a better accuracy than statistical models; (ii) moving average terms do not improve the predictive accuracy; (iii) hybrid models do not outperform their simpler counterparts.

Keywords: Electricity Price Forecasting, Deep Learning, Benchmark Study

1. Introduction

Because of the liberalization of the electricity markets in the past decades, the dynamics of electricity prices have become a complex phenomenon with rare characteristics and important consequences. In particular, when compared with other commodities, electricity trade displays a set of attributes that are quite uncommon: constant balance between production and consumption [1]; dependence of the consumption on the time, e.g. hour of the day, day of the week, and time of the year; load and generation that are influenced by external weather conditions [2]; and influence of neighboring markets [3]. Due to these characteristics, the dynamics of electricity prices have become very complex, e.g. highly volatile prices with sudden and unexpected price peaks [2].

In recent years, with the increasing penetration of *renewable energy sources* (RES), the described behavior has aggravated. In particular, while there are no questions regarding the contribution of RES to build a more sustainable world, several concerns have been raised regarding their influence on electricity prices and grid stability. More specifically, as the penetration of RES increases, so does the dependence of electricity production w.r.t. to weather conditions and, in turn, the volatility in electricity prices. This relation has been largely identified in the literature: [4] studied the effect of wind power penetration on the New England electricity market and concluded that price volatility increases with increasing wind penetration. Similarly, [5] carried out a similar study for the Texas market and also concluded that price volatility increased with increasing wind penetration. Looking at the penetration of solar power, [6] indicated that price spikes are expected to occur more frequently as the share of PV increases in the California system. Likewise, looking at the effect of increasing wind penetration in UK for the year 2020, [7] reported that prices are expected to be more volatile than at present.

Due to this effect, as the increasing integration of RES increases the volatility of prices, the behavior of market agents becomes naturally more unpredictable, sudden drops in generation and consumption are more likely to occur, the imbalances between production and consumption increase, and the electrical grid becomes more unstable.

In order to tackle the problems mentioned above, electricity markets together with electricity price forecasting have become a central point of research in the energy sector. In particular, by improving the forecasting accuracy, the negative effects of price uncertainty can be mitigated, the grid can be stabilized, and economic profits can be made.

*Corresponding author
Email address: jlagogarcia@tudelft.nl (Jesus Lago)

1.1. Electricity Price Forecasting

The electricity price forecasting literature is typically divided into five areas: (i) game theory models, (ii) fundamental methods, (iii) reduced-form models, (iv) statistical models, and (v) machine learning methods [2]. Since statistical and machine learning methods have showed to yield the best results [2], they are the focus of this review, and in turn, of the benchmarking experiment that will be performed in this paper.

Common statistical methods are: *autoregressive* (AR) and *autoregressive with exogenous inputs* (ARX) models [8], *double seasonal Holt-Winter* (DSHW) models [9], *threshold ARX* (TARX) models [10], *autoregressive integrated moving average* (ARIMA) models [11, 12], semi/non-parametric models [8, 13], *generalized autoregressive conditional heteroscedasticity* (GARCH) based models [14–16], or *dynamic regression* (DR) and *transfer function* (TF) models [17]. In addition, hybrid versions of the previous models are also common, e.g. wavelet-based models [12, 18, 19].

A pitfall of statistical models is that they are usually linear forecasters, and as such, they might not perform good in data where the frequency is high, i.e. hourly data with rapid variations and high frequency changes. In particular, while they show a good performance if the data frequency is low, e.g. weekly patterns, the nonlinear behavior of hourly prices might become too complicated to predict [20]. To address this issue and predict the nonlinear behaviors of hourly prices, different machine learning methods have been proposed. Among them, *multilayer perceptrons* (MLPs) [21–24], *support vector regressors* (SVRs) [25, 26] and *radial basis function* (RBF) networks [27] are the most commonly used.

While the academic literature comprises a much larger collection of approaches, e.g. see [2, 28], a complete review falls outside of the scope of this paper.

1.2. Deep Learning

In the last decade, the field of neural networks has experienced several innovations that have lead to what is known as *deep learning* (DL). In particular, one of the traditional issues of neural networks had always been the large computational cost of training large models. However, that changed completely when [29] showed that a deep belief network could be trained efficiently using an algorithm called greedy layer-wise pretraining. As related developments followed, researchers started to be able to efficiently train complex neural networks whose depth was not just limited to a single hidden layer (as in the traditional MLP). As these new structures systemically showed better results and generalization capabilities, the field was renamed as deep learning to stress the importance of the depth in the achieved improvements [30, Section 1.2.1].

While this success of DL models initiated in computer science applications, e.g. image recognition [31], speech recognition [32], or machine translation [33], the benefits of DL have also spread in the last years to several energy-related applications [34–39]. Among these areas, wind power forecasting is arguably the field that has benefited the most: [34] shows how, using a deep belief network and quantile regression, probabilistic forecasting of wind speed can be improved. Similar to [34], [39] proposes a deep feature selection algorithm that, in combination with a multi-model framework, improves the wind speed forecasting accuracy by 30%. In the same area of research, [37] proposes an ensemble of *convolutional neural networks* (CNNs) to obtain more accurate probability forecasts of wind power.

In addition to wind power applications, DL has also shown success in other energy-related fields. In the context of load forecasting, [36] proposes a deep autoencoder in combination with an *extreme gradient boosting* (XGB) model and shows how they forecast building cooling load more accurately than alternative techniques; within the same research paper, a *deep neural network* (DNN) to accurately forecast building cooling load is also proposed. For a different application, [38] proposes a DL model to detect islanding and to distinguish this effect from grid disturbances; based on the obtained simulation results, [38] indicates that the DL model can detect islanding with a very high accuracy. In addition, [35] proposes a DL strategy for time series forecasting and shows how it can be used successfully to forecast electricity consumption in households.

1.3. Motivation and Contributions

Despite the success of DL in all these energy-related areas and time series forecasting applications, there has not yet been, to the best of our knowledge, an attempt to bring its ideas and models to the field of electricity price forecasting. In particular, while neural networks have been proposed, they have been traditionally limited to one-hidden-layer networks, e.g. MLPs [21, 22, 40, 41] and RBF networks [27, 42], or to simple versions of *recurrent neural networks* (RNNs), e.g. Elman networks [43, 44]. While these simpler models are sometimes suitable, there are at least three arguments suggesting that using deeper structures could potentially benefit predictive accuracy:

1. Advanced RNN structures, e.g. *long-short term memory* (LSTM) [45] or *gated recurrent unit* (GRU) [46] networks, have shown to be a much better alternative to accurately model complex nonlinear time sequences [47–49], e.g. electricity prices.
2. While a single layer network can in theory model any nonlinear continuous function, a network with more than one hidden layer might be able to model the same function with a reduced number of neurons. Therefore, deep networks might actually be less complex and still generalize better than a simple MLP.
3. Considering the excellent results obtained in forecasting time series in other energy-related applications [34–39], it is possible that forecasting electricity prices might also benefit from using DL architectures.

Based on these arguments, the focus and main contribution of this paper is to propose a collection of different DL models that can be successfully used for forecasting day-ahead electricity prices. In particular, the paper develops a DL modeling framework comprising four models:

1. A DNN as an extension to the traditional MLP.
2. A hybrid LSTM-DNN structure.
3. A hybrid GRU-DNN structure.
4. A CNN model.

Then, considering a large benchmark comparison and a case study, it shows that the proposed DL modeling framework leads to improvements in predictive accuracy that are statistically significant.

In addition, as a second contribution, the paper also tries to establish an extensive benchmark of commonly used forecasters for predicting electricity prices. In particular, since even the largest benchmarks in the literature [8, 9, 50, 51] have been limited to 4-10 different forecasters, the paper considers that a conclusion on the relatively accuracy of the different forecasters cannot be drawn. With that motivation, we aim at providing a large empirical evaluation of 27 common forecasters for day-ahead electricity prices to bring new insights on the capabilities of the various models.

The paper is organized as follows: Section 2 introduces the theoretical concepts and state-of-the-art methods that are used in the research. Next, Section 3 presents the proposed DL framework. Section 4 defines the base forecasters that are collected from the literature and considered in the benchmark. Next, Section 5 evaluates the base and DL models in a case study, compares the obtained predictive accuracy by means of hypothesis testing, and discusses the results. Finally, Section 6 concludes the paper and outlines the main results.

2. Preliminaries

In this section, the theoretical concepts and algorithms that are used in the research are introduced.

2.1. Day-Ahead Forecasting

A type of power exchange that is widely used in many parts of the world is the day-ahead electricity market. In its most general format, bids are submitted for the 24 hours of day d before some deadline on day $d-1$. These bids are usually defined per hour, i.e. every market player has to submit 24 bids. After the deadline has passed, the market operator uses the submitted bids to compute the market clearing price for each of the 24 hours. Then, all the market agents get an energy allocation that depends on the market clearing price and the bids submitted by the market agent.

Considering this market format, a useful forecaster should predict the 24 market clearing prices of day d based on the information available before the deadline on day $d-1$.

2.2. Deep Learning

In this section, we give a brief description of the DL structures considered in the modeling framework. For the sake of conciseness, we provide a large explanation of the DL models in Appendix A.

The basic DL model is the DNN [30], the natural extension of the traditional MLP that uses multiple hidden layers. When compared with a standard MLP, a DNN requires specific model changes to be efficiently trained, e.g. activation functions different from the standard sigmoid.

Slightly more complex than DNNs are RNNs [30], a type of network that builds additional mappings to hold relevant information from past inputs and that are suitable for modeling time series data, e.g. electricity prices. The two state-of-the-art recurrent networks are LSTM [45] and GRU networks [48]; unlike standard RNNs, they are able to model a selective forget-remember behavior. While both structures are very similar, GRUs have a simpler structure and they are faster to train.

A different type of DL structure are CNNs, a type of network that are modeled using three building blocks: a convolution operation, a pooling operation, and a fully connected layer. Given an array of data, the convolution operation slides a filter across the data array and computes local element-wise cross product between the filter and the data. As different filters capture different properties, CNNs typically use various filters to obtain different data arrays known as feature maps. In a subsequent step, the pooling operation reduces the size of these feature maps by reducing large areas into single values. Finally, after several convolutions and pooling operations are done, the values of the last feature maps are used as inputs for a fully connected layer.

2.3. Hyperparameter Optimization

Hyperparameters are model parameters that have to be selected before the estimation process, e.g the number of neurons in a neural network or the lag order in an ARIMA model. In the case of our benchmark study, to objectively analyze and compare the accuracy of each benchmark model, we optimize this selection following the same automated procedure for each individual model. In particular, we employ the tree-structured Parzen estimator [52], a sequential model-based optimization algorithm [53] within the family of *Bayesian optimization* [54] methods.

2.4. Performance Metrics

A performance metric is needed to evaluate and compare the accuracy of the forecasters. In this paper, we consider the *symmetric mean absolute percentage error* (sMAPE) [55] metric:

$$\text{sMAPE} = \frac{100}{N}\sum_{k=1}^{N}\frac{|y_k - \hat{y_k}|}{(|y_k| + |\hat{y}_k|)/2}, \tag{1}$$

where $[y_1, \ldots, y_N]^\top$ are the real outputs to be predicted and $[\hat{y}_1, \ldots, \hat{y}_N]^\top$ the predicted values.

As in [3], sMAPE is selected instead of the more traditional *mean absolute percentage error* (MAPE) metric because of the issues that affect MAPE [55].

2.5. Diebold-Mariano Test

The sMAPE is a metric that can be used to compare which model has a better accuracy. However, the fact that the accuracy of a model is higher, is not enough to guarantee that the model is better. In particular, to have a minimum assurance that a model is better, the difference in accuracy should be statistically significant. To evaluate this, the *Diebold-Mariano* (DM) test [56] is the statistical test that is typically used.

Given a time series vector $[y_1, \ldots, y_N]^\top$ to be forecast, two prediction models M_1 and M_2, and the associated forecasting errors $[\varepsilon_1^{M_1}, \ldots, \varepsilon_N^{M_1}]^\top$ and $[\varepsilon_1^{M_2}, \ldots, \varepsilon_N^{M_2}]^\top$, the DM test builds a covariance stationary loss function $L(\varepsilon_k^{M_i})$ and the associated loss differential:

$$d_k^{M_1,M_2} = L(\varepsilon_k^{M_1}) - L(\varepsilon_k^{M_2}). \tag{2}$$

Then, in its one-sided version, the DM test evaluates the null hypothesis H_0 of M_1 having an accuracy equal to or worse than M_2, i.e. equal or larger expected loss, against the alternative hypothesis H_1 of M_1 having a better accuracy, i.e.:

$$\text{One-sided DM test} \quad \begin{cases} H_0: \ \mathbb{E}[d_k^{M_1,M_2}] \geq 0, \\ H_1: \ \mathbb{E}[d_k^{M_1,M_2}] < 0. \end{cases} \tag{3}$$

If H_0 is rejected, the test concludes that the accuracy of the forecast of M_1 is statistically significantly better.

3. DL Modeling Framework

As indicated in the introduction, the main goal of this paper is to propose a DL modeling framework as a forecasting tool for day-ahead electricity prices. As a first step to achieve that, this section develops the four DL models comprising the framework.

3.1. Market Integration

Before describing each model separately, it is important to note that a common feature to all DL models is market integration. In particular, to improve the predictive accuracy, all the DL models simultaneously predict electricity prices of various day-ahead markets. The idea behind is that, as shown in [3], due to market integration and by multitasking, i.e. predicting prices in different markets, the models can learn more general features and integrate relations across neighboring markets.

In detail, regarding a local market L that is subject to study and a set of c neighboring markets N1, ..., Nc, each DL model predicts the following output:

$$\mathbf{p} = [p_{\mathrm{L}_1}, \ldots, p_{\mathrm{L}_{24}}, p_{\mathrm{N}1_1}, \ldots, p_{\mathrm{N}c_{24}}]^\top, \tag{4}$$

where $\mathbf{p}_\mathrm{L} = [p_{\mathrm{L}_1}, \ldots, p_{\mathrm{L}_{24}}]^\top$ is the vector of day-ahead prices in the local market, and $\mathbf{p}_{\mathrm{N}i} = [p_{\mathrm{N}i_1}, \ldots, p_{\mathrm{N}i_{24}}]^\top$ is the vector of day-ahead prices in the neighboring market i.

3.2. DNN Model

As a simple extension of the traditional MLP, the first DL model for predicting day-ahead prices is a deep neural network with two hidden layers. In particular, defining as $\mathbf{X} = [x_1, \ldots, x_n]^\top$ the input of the model, as n_1 and n_2 the respective number of neurons of the first and the second hidden layer, and by $\mathbf{p} = [p_{\mathrm{L}_1}, \ldots, p_{\mathrm{L}_{24}}, p_{\mathrm{N}1_1}, \ldots, p_{\mathrm{N}c_{24}}]^\top$ the vector of day-ahead prices that we intend to forecast, the corresponding model is represented in Figure 1.

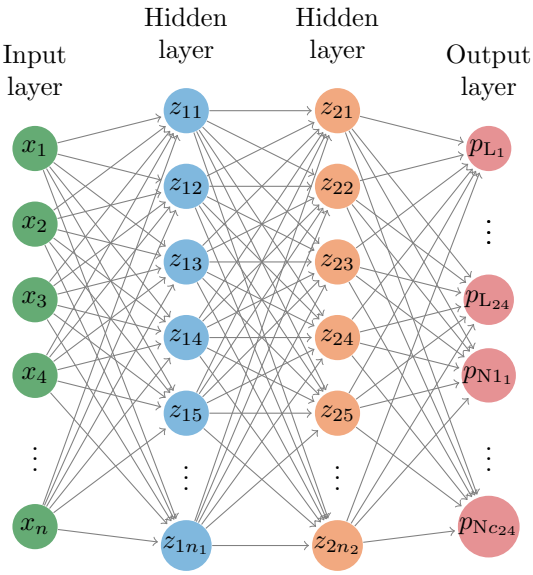

Figure 1: Deep neural network to simultaneously forecast day-ahead prices in several countries.

3.3. LSTM-DNN Model

The second DL model for predicting day-ahead prices is a hybrid forecaster combining an LSTM and a DNN network. The motivation behind this hybrid structure is to include a recurrent layer that can learn and model the sequential relations in the time series data as well as a regular layer that can learn relations that depend on non-sequential data.

In detail, for this new model, the inputs are divided between those that model sequential time data, e.g. past electricity prices, and those that model regular data, e.g. day of the week or day-ahead forecasting of the grid load. This division is necessary because the LSTM network requires a sequence of time series values as an input. However, considering all the possible regressors for electricity price forecasting, it is clear that some of them do not have that property.

In general, for the case of electricity prices, the distinction between these two types of data can be done by considering the time information represented in the data. Specifically, if the data represents a collection of past values, it can normally be modeled as time sequential data and used as an LSTM regressor. By contrast, if the data represents some specific property associated with the day ahead, i.e. it represents direct information of a future event, it cannot be modeled as a time sequence. Examples of the first could be past day-ahead prices or the measured grid load; examples of the second could be the day-ahead forecast of the weather or whether tomorrow (day-ahead) is a holiday. Using this distinction, the inputs of the model are divided between two groups:

- Input vector $\mathbf{X}_{\mathrm{F}} = [x_{\mathrm{F}1}, \ldots, x_{\mathrm{F}n}]^\top \in \mathbb{R}^n$ representing future information.
- A collection $\left\{\mathbf{X}_{\mathrm{S}}^i\right\}_{i=1}^q$ of q input sequences, where $\mathbf{X}_{\mathrm{S}}^i = [x_{\mathrm{S}1}^i, \ldots, x_{\mathrm{S}N}^i]^\top \in \mathbb{R}^N$ is a vector representing past information.

Using this separation, the model uses a DNN to process the inputs $\mathbf{X}_{\mathrm{F}}$ and an LSTM to process the time sequences $\left\{\mathbf{X}_{\mathrm{S}}^i\right\}_{i=1}^q$. Then, the outputs of these two networks are concatenated into one vector and this vector is fed into a regular output layer.

Defining the number of neurons of the DNN and LSTM layers respectively by n_{F} and n_{S}, and by z_{F_i} and $[z_{\mathrm{S}_i}, c_{\mathrm{S}_i}]\top$ the internal state of their neuron i, an example of the proposed model is represented by Figure 2.

3.4. GRU-DNN Model

The third DL model for predicting day-ahead prices is a hybrid model combining a GRU and a DNN network. As with the LSTM-DNN hybrid structure, the motivation behind this model is to include a layer that is tailored to sequential data. However, to reduce the computational burden of the LSTM layer, a GRU layer is used instead to model the time data sequences $\left\{\mathbf{X}_{\mathrm{S}}^i\right\}_{i=1}^q$. Specifically, if in Figure 2 the LSTM cell states $[z_{\mathrm{S}_i}, c_{\mathrm{S}_i}]\top$ are replaced by the corresponding GRU cell state z_{S_i}, the modified figure would represent an example of the new proposed model.

3.5. CNN Model

The fourth DL model for predicting day-ahead prices is a CNN network. As in the previous two cases, the inputs are divided between those that model sequential past data and those that model information regarding the day ahead. For the hybrid models, the division was necessary because the recurrent layers needed sequential data. In this new case, the separation is required in order to group data with the same dimensions as inputs for the same CNN. In particular, the data is separated into two parts:

- The same collection $\left\{\mathbf{X}_{\mathrm{S}}^i\right\}_{i=1}^q$ of q input sequences used for the hybrid models. As before, $\mathbf{X}_{\mathrm{S}}^i = [x_{\mathrm{S}1}^i, \ldots, x_{\mathrm{S}N}^i]^\top \in \mathbb{R}^N$ is a vector representing some sequential past information.

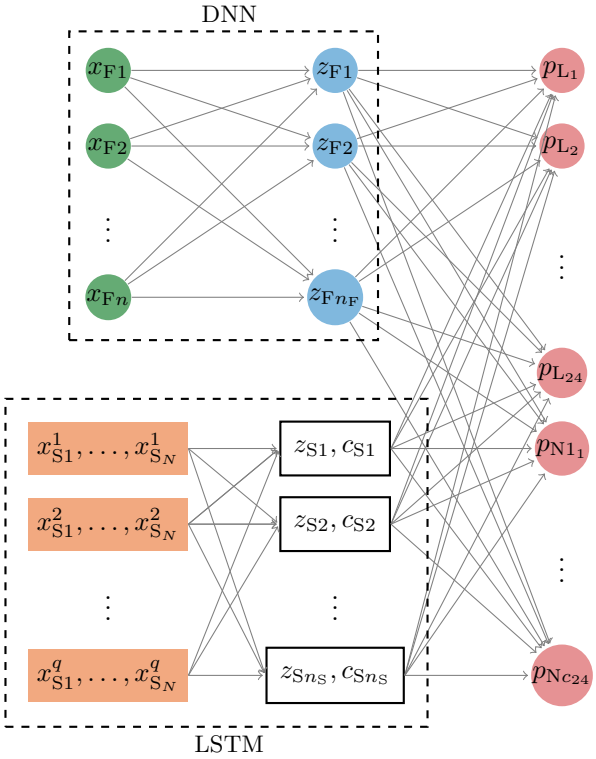

Figure 2: Hybrid DNN-LSTM network to simultaneously forecast day-ahead prices in several countries.

- A new collection $\left\{\mathbf{X}_{\mathrm{F}}^i\right\}_{i=1}^r$ of r input vectors, where each vector $\mathbf{X}_{\mathrm{F}}^i = [x_{\mathrm{F}_1}^i, \ldots, x_{\mathrm{F}_{24}}^i]^\top \in \mathbb{R}^{24}$ represents some future information of the 24 hours of the day ahead. These data are equivalent to the day-ahead inputs $\mathbf{X}_{\mathrm{F}} = [x_{\mathrm{F}1}, \ldots, x_{\mathrm{F}n}]^\top$ of the hybrid models. In particular, the values in $\mathbf{X}_{\mathrm{F}}$ representing hourly day-ahead values, e.g. forecast of the grid load, are directly mapped into the corresponding 24-values sequence. By contrast, the values in $\mathbf{X}_{\mathrm{F}}$ representing some day-ahead property, e.g. holidays, are repeated 24 times to build the equivalent vector.

Given this separation, the model uses 2 parallel CNNs to model the electricity price dynamics. In particular, a first CNN considers the r input sequences $\left\{\mathbf{X}_{\mathrm{F}}^i\right\}_{i=1}^r$ as r input channels. Then, a parallel CNN regards the remaining q input sequences $\left\{\mathbf{X}_{\mathrm{S}}^i\right\}_{i=1}^q$ as q input channels. Next, both networks perform a series of convolution and pooling operations. Finally, the feature maps at the end of both CNNs are connected into a fully connected layer that models the day-ahead prices $\mathbf{p} = [p_{\mathrm{L}_1}, \ldots, p_{\mathrm{L}_{24}}, p_{\mathrm{N}1_1}, \ldots, p_{\mathrm{N}c_{24}}]\top$. As with the hybrid networks, the motivation behind using this structure is to have a network with layers tailored to sequential past data as well as with layers tailored to non-sequential data.

Defining the internal states of both networks by $z_{\mathrm{F}_k}^{i,j}$ and $z_{\mathrm{S}_k}^{i,j}$, with i representing the layer of the network, j the specific feature map in layer i, and k the state within the

feature map j of layer i, Figure 3 depicts an example of this type of structure. For the sake of simplicity, the example illustrates both CNNs performing just a single convolution and pooling operation and using only two filters.

3.6. Selection of the Network Structure

To complete the modeling framework, the details of the models have to be selected; in particular, for each of the proposed forecasters, there are many hyperparameters to be selected, e.g. the number of neurons, the type of activation function, etc. However, while the structure of the proposed models is general for any electricity market, the specific architecture and implementation details might be not. Specifically, hyperparameters such as the number of neurons might depend on the market under study, and thus, they should be optimized accordingly. As a result, in this section, we limit the explanation to which hyperparameters are optimized. Next, in later sections, we indicate the specific optimal selection for the case study.

3.6.1. Common Hyperparameters

While some hyperparameters are model-specific, three of them are common to the four models:

1. **Activation function**: Except for the output layer that does not use any, all the layers within a network use, for the sake of simplicity, the same activation function. This function is chosen with a single hyperparameter, and in the case of the hybrid models, i.e. GRU-DNN and LSTM-DNN, two hyperparameters are used so that each network type can employ a different activation function.
2. **Dropout**: Dropout [57] is included as a possible regularization technique to reduce overfitting and to improve the training performance. To do so, at each iteration, dropout selects a fraction of the neurons and prevents them from training. This fraction of neurons is defined as a real hyperparameter between 0 and 1.
3. **L1-norm penalization**: In addition to dropout, the models can add an L1-norm penalization to the network parameters as a different way of regularizing. Defining the network weights by W and using another binary hyperparameter, the models can choose whether to add to the cost function the following term:
$$\lambda \|W\|_1^2. \tag{5}$$
If regularization is selected, λ becomes a real hyperparameter.

3.6.2. DNN Hyperparameters

The DNN model uses two additional model-specific hyperparameters:

- $\boldsymbol{n_1}/\boldsymbol{n_2}$: number of neurons in the first/second hidden layer.

3.6.3. LSTM-DNN / GRU-DNN Hyperparameters

For the two hybrid models, there are three additional model-specific hyperparameters:

1. $\boldsymbol{n_{\mathbf{LSTM}}}/\boldsymbol{n_{\mathbf{GRU}}}$: number of neurons in the recursive layer.
2. $\boldsymbol{n_{\mathbf{DNN}}}$: number of neurons in the DNN layer.
3. **Sequence length**: For the LSTM structure, each input is modeled as a sequence of past values. Considering that values too far in the past do not cause any effect in the day-ahead prices, selecting the right length for the input sequences might remove unnecessary complexities. Therefore, a third hyperparameter is used to select the length of the input sequences.

3.6.4. CNN Hyperparameters

Depending on which of the two CNN structures they affect, the specific hyperparameters of the CNN model can be divided into three groups:

1. The hyperparameters that are common and equal to the two CNN structures:
   (a) **Pooling frequency**: The pooling operation does not have to be always performed right after every convolution. Therefore, an integer hyperparameter is used to select how frequently, i.e. after how many convolutional layers, pooling is performed.
   (b) **Pooling type**: To enlarge the number of possible architectures, a binary hyperparameter selects whether the model uses the average pooling or the maximum pooling operation.
2. The hyperparameters that only apply to one of the two CNN structures:
   (c) **Channel length**: For the CNN with past sequences, the length of the input channels is selected as an integer hyperparameter. In the case of the other CNN, the input channels have a length of 24 that correspond with the 24 hours of the day ahead.
3. The integer hyperparameters that, while employed in both networks, their value can be different.
   (d) **Filter size**: the size of the filter of the convolution operation.
   (e) **Number of convolutions**: the number of convolutional layers in each CNN.
   (f) **Feature maps in first layer**: The number of feature maps in every layer is determined by selecting the number of feature maps in the first layer. In particular, the number of feature maps in successive layers is simply doubled every two convolutional layers. This choice is used to reduce the total number of hyperparameters. In particular, a more general approach could be to select the number of convolution layers, and then, to model the number of features maps in each of these layers with a different hyperparameter. However, this approach is avoided as it requires a much larger computational cost.

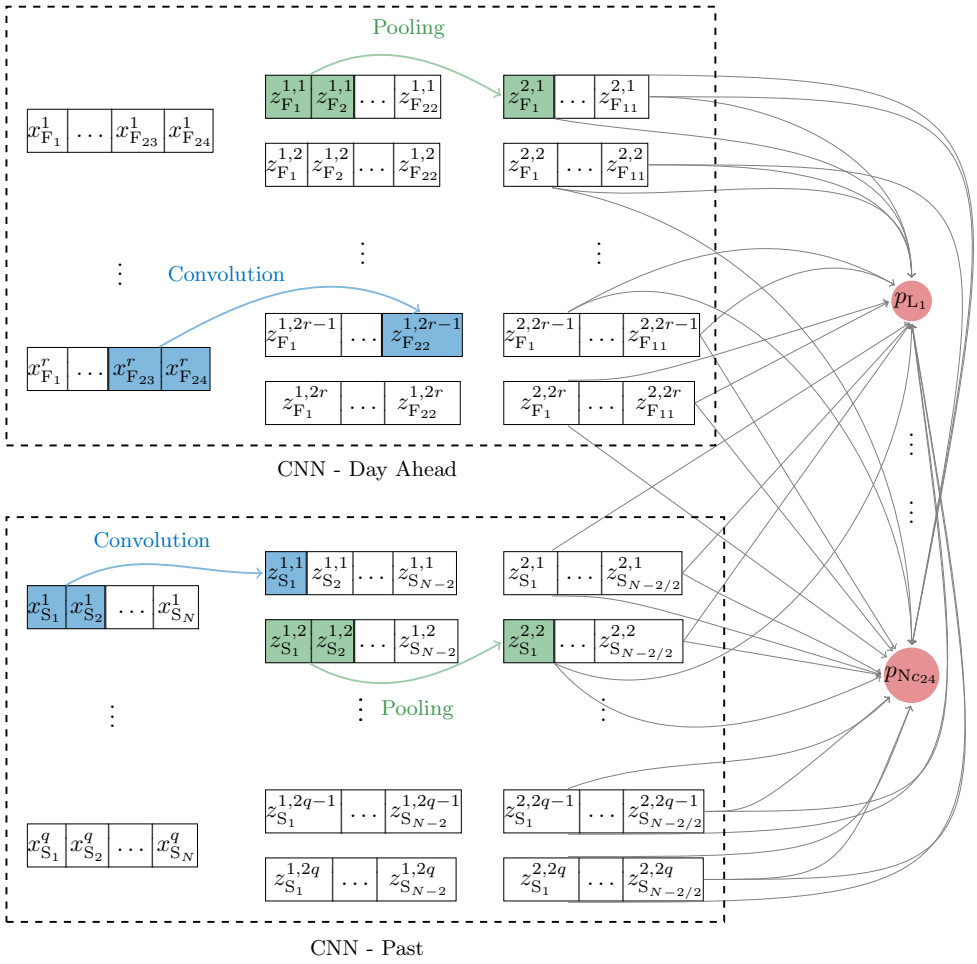

Figure 3: Hybrid DNN-LSTM network to simultaneously forecast day-ahead prices in several countries.

3.7. Model Estimation

In the proposed framework, all the neural networks are trained by minimizing the mean absolute error. In particular, given the training set $\mathcal{S}_{\mathcal{T}} = \left\{(\mathbf{X}_k, \mathbf{p}_k)\right\}_{k=1}^{N}$ with N data points, the networks are trained via the following optimization problem:

$$\underset{\mathbf{w}}{\text{minimize}} \quad \sum_{k=1}^{N} \|\mathbf{p}_k - F(\mathbf{X}_k, \mathbf{w})\|_1, \tag{6}$$

where $\mathbf{w}$ represents the vector of all network weights and $F : \mathbb{R}^n \to \mathbb{R}^{24\,(c+1)}$ the neural network map. The selection of the mean absolute error instead of the more traditional root mean square error is done for a simple reason: as the electricity prices have large spikes, the Euclidean norm would put too much importance on the spiky prices. The optimization problem is solved using Adam [58], a stochastic gradient descent method [59] that uses adaptive learning rates. The advantage of using this optimization method is that the learning rate does not need to be tuned online. Together with Adam, the proposed models also considers early stopping [60] to avoid overfitting.

4. Benchmark Models for Electricity Price Forecasting

In order to have a large benchmark study, we consider, in addition to the 4 proposed DL forecasters, a set of 23 different models that have been proposed in the literature of electricity prices forecasting. In addition, to further enlarge the benchmark, we consider different versions of each of the 27 individual models in order to have a benchmark of 98 models.

As the 23 models from the literature will be used to evaluate the proposed DL models, they are referred to as base forecasters. Moreover, as the aim of this study is not only the evaluation of the DL models but also to establish a large benchmark within the community of electricity price forecasting, we try to consider a fair selection of base models by including the most common and known forecasters from the literature. In particular, we use the excellent literature review of [2] and the newest advances in the field to make the selection as complete as possible. It is important to note that, while the main principles of each base model are defined below, the model equations are not provided. Instead, we refer to the original papers

for full documentation.

Based on the model separation of [2], the 23 base forecasters are divided into three different classes: statistical methods without exogenous inputs, statistical methods with exogenous inputs, and machine learning methods.

4.1. Statistical Methods Without Exogenous Inputs

The first class of models comprises statistical methods that only use past prices as input features. Among them, we make the distinction between AR models, GARCH models, and exponential smoothing methods.

4.1.1. AR-Type Models

The first subclass of forecasters assumes homoskedasticity, i.e. constant variance and covariance functions, and models time correlation in the time series using a linear model. Within this subclass, we have selected four models:

1. The well-known Wavelet-ARIMA model [18], a method that has been regularly used in other empirical evaluations [42, 61–63]. This model will be denoted as *wavelet-ARIMA* (WARIMA).
2. The *double seasonal ARIMA* (DSARIMA) model [9], an ARIMA model that considers the double seasonality, i.e. weekly and daily, of electricity prices.
3. The AR model of [64], an autoregressive model with lags of 24, 48, and 168 hours, that also models differences among days of the week.
4. The Wavelet-ARIMA-RBF model [42], a forecaster that considers the traditional Wavelet-ARIMA structure but adds an RBF network to model the residuals. This model will be denoted as WARIMA-RBF.

4.1.2. GARCH-Based Models

Unlike the AR-type models, GARCH-based models do not require homoskedasticity in the time series. However, unlike the former, GARCH models are not accurate in forecasting spot electricity prices in standalone applications; particularly, they need to be coupled with AR-type models to boost their predictive accuracy [2, Section 3.8.6]. As a result, within this subclass, we regard the following hybrid model:

5. The ARIMA-GARCH model [15], a forecaster that considers a standard ARIMA model with GARCH residuals.

4.1.3. Exponential Smoothing Methods

The last subclass is exponential smoothing, a family of algorithms that make a prediction using an exponentially weighted average of past observations. Among these methods, we have selected two different forecasters:

6. The DSHW [65] model, an algorithm that was successfully used by [9] for forecasting spot electricity prices.
7. The *exponential smoothing state space model with Box-Cox transformation, ARMA errors, trend and seasonal components* (TBATS) [66], a forecaster that is able to model multiple seasonality. While this method has never been used before for electricity price forecasting, it is a generalization of the DSHW model [66]. Therefore, it is an interesting method to consider.

4.2. Statistical Methods with Exogenous Inputs

The second class of models are statistical methods that consider regressors to enhance the predictive accuracy. Typical regressors for forecasting electricity prices are the grid load, the available capacity, or the ambient temperature. Among these models, we can distinguish four subclasses: ARX-type models, regime-switching models, semi-parametric models, and models with automated input selection.

4.2.1. ARX-Type Models

The first subclass is the natural generalization of adding exogenous inputs to the AR-based models of Section 4.1.1. Like the AR models, they also assume homoskedasticity of the data. For the benchmark, we consider four ARX models:

8. The DR model [17], an ARX model that uses the grid load as a regressor and that has been used in other empirical evaluations [51].
9. The TF model [17], an ARX model with moving average terms that, like the DR model, it uses the grid load as a regressor and it has also been used in other comparisons [51].
10. The ARX model proposed in [64], an extension of the AR method defined in Section 4.1.1 that uses the grid load as a regressor. We will refer to this model as ARX.
11. The *full-ARX* (fARX) model [67], an ARX model that is an extension of the previous ARX.

4.2.2. Regime-Switching Models

The second subclass, i.e. regime-switching models, considers that the time series can be modeled by different regimes, that each regime has an independent model, and that switches between regimes can be modeled by the value of some variable. We consider a single regime switching model:

12. The TARX model defined in [10], a model with two regimes that separate normal prices from spiky dynamics. As decision variable, the model uses the difference between the mean price of one day and of eight days before. Then, each of the regimes is modeled with an ARX model that uses the grid load as an exogenous input.

4.2.3. Semiparametric Models

Semiparametric models are based on the premise that, given some empirical data, a nonparametric kernel density estimator might lead to a better fit than any parametric distribution. To benefit from this hypothesis, they relax the assumption about the probability distribution that is typically needed when estimating their parametric counterparts. An example of semiparametric models are the semiparametric ARX models, which have the same functional form as the equivalent ARX models, but they relax the normality assumption needed for the maximum likelihood estimation [8, 68]. For the benchmark, we regard two different semiparametric models:

13. The *Hsieh-Manski ARX* (IHMARX) estimator, an algorithm originally analyzed in [68] and studied in the context of electricity price forecasting in [8].
14. The *smoothed nonparametric ARX* (SNARX) estimator, a semiparametric model that was also originally analyzed in [68] and applied to electricity price forecasting in [8].

4.2.4. Models with Automated Input Selection

In the last subclass, we consider a set of models that automatically select the important exogenous inputs. While this type of models are instantiations of the previous three subclasses, we separate them in a fourth subclass due to their special structure. For the benchmark, we consider two of them:

15. The *fARX regularized with Lasso* (fARX-Lasso) [67] model, the fARX model defined in the subclass of ARX models that uses Lasso [69] as a regularization tool to automatically reduce the contribution of unimportant inputs.
16. The fARX-EN [67] model, the same model but using elastic nets [70] as a regularization tool.

4.3. Artificial Intelligence Models

The last class of models comprises the machine learning models, a family of algorithms that, while also including exogenous inputs, are able to model more complex nonlinear relations than the previously defined models. Within this class, we can distinguish three subclasses: models based on neural networks, SVR-based models, and ensemble methods.

4.3.1. Neural network based models

This subclass can be seen as a family of simpler DL algorithms. For the benchmark, we regard two different models:

17. The traditional MLP model, a standard neural network with a single hidden layer widely used by many authors [9, 21, 22].
18. The RBF network, a model introduced in Section 4.1.1 as part of a hybrid forecaster that has also had standalone applications [27].

4.3.2. SVR Based Models

Support vector regressors perform a nonlinear mapping of the data to a higher-dimensional space where linear functions are used to perform regression. For the benchmark, we include the following three models:

19. The plain SVR model as used in [71].
20. The SOM-SVR [25, 72] model, a forecaster that first clusters data via *self-organizing maps (SOM)* and then predicts prices using a different SVR model per cluster.
21. The SVR-ARIMA [26] model, a hybrid forecaster that uses a SVR model to capture the nonlinearity of prices and an ARIMA model for the linearities.

4.3.3. Ensemble Models

Within this final subclass, we include algorithms based on ensemble methods. Particularly, we consider the two well-known algorithms based on regression trees [73]:

22. The *random forest* (RF) [74] model, a forecaster that predicts data by combining several regression trees. It is based on the principle of bagging [73, Chapter 8], i.e. combining models with low bias and high variance error in order to reduce the variance while keeping a low bias.
23. The XGB [75] model, which also forecasts data by combining regression trees, but it is based on the principle of boosting [73, Chapter 10], i.e. combining models with high bias and low variance in order to reduce the bias while keeping a low variance.

It is important to note that, while to the best of our knowledge, these models have never been used for electricity price forecasting, we include them in the benchmark as they display reasonable results.

4.4. Modeling Options

To have a more fair comparison, the mentioned models are not only considered in their traditional form; particularly, for each model, three modeling options with two alternatives per modeling option are considered, i.e. a model that could use the 3 modeling options would have $2^3 = 8$ model alternatives.

4.4.1. Modeling Option 1: Spikes Preprocessing

Due to the fact that the dynamics of electricity prices are characterized by large, but infrequent, spikes [2], better models might be obtained if spikes are disregarded during the estimation process. As a result, when estimating the model parameters, we consider two model alternatives:

1. **MO1A1**: A first alternative that limits the spike amplitude to the mean plus/minus three times the standard deviation.
2. **MO1A2**: A second one that uses raw prices.

4.4.2. Modeling Option 2: Feature Selection

For all the models that include exogenous inputs, there are two additional model alternatives:

1. **MO2A1**: A first alternative that uses the features from the original paper. For all the base models, the original input is the day-ahead grid load forecast given by the transmission system operator.
2. **MO2A2**: A second alternative where the features are optimally selected considering all the available data in the market under study. This step is done following the feature selection method described in [3], where the features are optimally selected by minimizing the sMAPE of the model in a validation set.

4.4.3. Modeling Option 3: Market Integration

As explained in Section 3, all the DL models simultaneously predict electricity prices in various spot markets. This was done because, as shown in [3], the accuracy of forecasting electricity prices can be enhanced by including market integration. Therefore, for all the forecasters that model the day-ahead prices in a single model, i.e. that do not need 24 independent models, two additional model alternatives are considered:

1. **MO3A1**: A first alternative where the models only predict the prices in the local market.
2. **MO3A2**: A second alternative where the models consider market integration and simultaneously predict the prices in various markets.

It is important to note that, while this modeling option is only possible for some models, considering market integration is available for many more. In particular, for any of the models with exogenous inputs, market integration could be modeled using features from connected markets as model inputs. Therefore, when evaluating the second alternative of *modeling option 2*, i.e. MO2A2, market integration is implicitly considered if features from connected markets are part of the optimal set of inputs.

4.5. Hyperparameter Optimization

In order to have a fair comparison, not only different modeling options should be considered, but also the hyperparameters of the models should be optimized. In particular, considering that the hyperparameters of the DL models are tuned, the configuration of the base models should also be tuned. As motivated in Section 2.3, this optimization step is performed using Bayesian optimization. Examples of hyperparameters in the base models are: the size of the lags in all the AR-based models, the penalty coefficient in the SVR model, or the number of trees in the random forest.

4.6. Summary

We summarized in Table 1 all the considered benchmark methods with their properties and modeling options. In particular, the first column denotes whether a model is nonlinear, the second one whether it considers exogenous inputs, and the last three whether the model can make use respectively of modeling options 1, 2, and 3. It necessary to remark that these three columns do not indicate which alternative is the best; more specifically, they simply show whether a model can consider the alternatives of each modeling option.

	Properties		**Options**		
Model	Non-linear	Exog. Inputs	MO1	MO2	MO3
AR			X		
DSARIMA			X		
WARIMA			X		
WARIMA-RBF	X		X		
ARIMA-GARCH			X		
DSHW			X		
TBATS			X		
DR		X	X	X	
TF		X	X	X	
ARX		X	X	X	
TARX		X	X	X	
IHMARX		X	X	X	
SNARX		X	X	X	
fARX		X	X	X	
fARX-Lasso		X	X	X	
fARX-EN		X	X	X	
MLP	X	X	X	X	X
RBF	X	X	X	X	X
SVR	X	X	X	X	
SOM-SVR	X	X	X	X	
SVR-ARIMA	X	X	X	X	
RF	X	X	X		X
XGB	X	X	X		X
DNN	X	X	X		X
LSTM	X	X	X		X
GRU	X	X	X		X
CNN	X	X	X		X

Table 1: Compilation of methods considered in the benchmark. The first two columns denote possible properties of the model. The last three columns respectively denote whether a model can make use of the 2 alternatives of modeling option 1, the 2 alternatives of modeling option 2, and the 2 alternatives of modeling option 3.

It is important to note that, while 27 individual benchmark models have been defined, a total of 98 models are in fact included in the benchmark. In particular, considering the three modeling options, a total of $27 \cdot 2\,(\text{MO1}) + 14 \cdot 2\,(\text{MO2}) + 8 \cdot 2\,(\text{MO3}) = 98$ forecasters are included. However, as a comparison of 98 models would be too vast, the results in the case study are directly given in base of

the best alternative for each of the 27 individual models. A description of which alternative performs the best for each model is listed in 5.3.

5. Case Study

In this section, we perform the empirical study to evaluate the proposed DL models and to analyze the predictive accuracy of the various base models. To do so, we consider the day-ahead market in Belgium, i.e. *European power exchange* (EPEX)-Belgium, in the period from 01/01/2010 to 31/11/2016. In particular, as a first step to analyze the models, we motivate the data that is considered. Then, we perform the required hyperparameter optimization so that all the forecasters employ an optimized structure. Next, after the hyperparameters are optimized, we compare the predictive accuracy of the various forecasters using a year of out-of-sample data. From this comparison, we are able to establish a first evaluation of the DL models as well as to rank the benchmark models according to their performance. Finally, the differences in performance are analyzed via statistical testing.

5.1. Data

In general, when looking at the day-ahead forecasting literature, several inputs have been proposed as meaningful explanatory variables, e.g. temperature, gas and coal prices, grid load, available generation, or weather [2].

5.1.1. Data Selection

For this research, in addition to the past prices p_{B} in the EPEX-Belgium, we consider several exogenous inputs. As defined by the second modeling alternative MO2 in Section 4.4.2, the specific subset of inputs is given as either one of the following alternatives:

1. A first subset that considers as exogenous input the day-ahead grid load forecast given by the transmission system operator. This selection is done as this variable has been widely used in the literature [8, 10, 67], and for all the base models, it is the exogenous input used in the original papers.
2. A second subset that is obtained by regarding all the available information for the market under study and performing feature selection. This step is done following the feature selection method described in [3]. The available input features are:
   (a) The day-ahead forecast l_{B} of the grid load in the EPEX-Belgium.
   (b) The day-ahead forecast g_{B} of the available generation in the EPEX-Belgium.
   (c) Past prices p_{F} in the neighboring EPEX-France market.
   (d) The day-ahead forecast l_{F} of the grid load in the EPEX-France.
   (e) The day-ahead forecast g_{F} of the available generation in the EPEX-France.

We make the distinction between these two alternatives because, while it is necessary to optimize each model for our case study, it is also important to evaluate them in their original format, i.e. as they were originally proposed in the literature.

It is important to note that, while we optimize the input features for every model, discussing the results of the feature selection would be too large to include within the manuscript (we evaluate 27 models, each model predicts 24 hours, and there are available more than 750 individual input features that can be selected per hour and per model). As a consequence, the main results of the feature selection, i.e. which features are in general relevant to predict the different hours of the day, are provided as supplementary material in Appendix B.

5.1.2. Data Division

To perform the different experiments, we divide the data into three sets:

1. Training set (01/01/2010 to 30/11/2014): these data are used for training and estimating the different models.
2. Validation set (01/12/2014 to 30/11/2015): a year of data is used to select the optimal hyperparameters.
3. Test set (01/12/2015 to 30/11/2016): a year of data that is not used at any step during the model estimation process, is employed as the out-of-sample data to compare the models.

Considering that there are 24 electricity prices per day, the training dataset comprises 43536 data points. Likewise, both validation and test datasets comprise 8760 data points each.

5.1.3. Data Processing

In order to obtain time series that are easier to forecast, the data used for the statistical models are processed using a Box-Cox transformation [76]. This preprocessing step, which includes the log-transformation as a special case, is a standard one in the literature of electricity price forecasting [9–11, 19, 51]. For the machine learning and DL models, the data is respectively normalized to the intervals $[0, 1]$ and $[-1, 1]$. This transformation is done because, based on experimental results using the validation set, these two preprocessing steps help to obtain more accurate models.

It is important to note that these transformations are only applied when estimating the parameters, not for computing metrics or statistical significance.

5.1.4. Data Access

For the sake of reproducibility, we have only considered data that are publicly available. Particularly, the electricity prices can be obtained from the ENTSO-E transparency platform [77]. Similarly, the load and generation day-ahead forecasts are available on the webpages of RTE

[78] and Elia [79], the respective TSOs in France and Belgium.

5.2. Modeling Implementation: Frameworks and Libraries

In order to implement the proposed DL framework, we use the Keras [80] DL library in combination with the mathematical language Theano [81]. The full framework is developed in `python`.

For the base models, the libraries employed differ more. In general, most of the forecasters are also modeled in `python`. The only exception are the DSHW and the TBATS forecasters, both of which are modeled using the `R` language and its forecast library [82]. For the remaining 17 models, we can distinguish several groups according to the library/framework used:

1. For the RF, the AR, the DR, the ARX, the TARX, the RBF, the three fARX-based models, and the three SVR-based models, the scikit-learn library [83] is used.
2. The XGB model is built using the xGBoost library [75] which is developed by the same authors that proposed the algorithm.
3. The MLP is modeled using the same frameworks as the other DL models.
4. The remaining models, i.e. the IHMARX, the SNARX, the TF, and the 4 ARIMA-based models, are estimated by solving the corresponding maximum likelihood estimation problem. In particular, to solve the various nonlinear optimization problems that arise from the maximum likelihood technique, we employ `CasADi` [84], a symbolic framework for automatic differentiation and numerical optimization. Within this group, we also model the ARIMA part of the SVR-ARIMA model.

In addition, to solve the optimization problems that estimate the models' parameters, we distinguish between two different stopping criteria:

1. Except for the neural network models, the stopping criterion is given by the moment that a (local) minimum is reached. We assume that a local minimum is reached when the gradient of the objective function is lower than some tolerance; in our study, that was 10^{-6}.
2. For the neural network models, we monitor the performance of a validation set and we stop the training when the improvements on this validation set cease (we assume that the improvement ceases if the accuracy in the validation set worsens for ten consecutive epochs). This criterion is called early stopping [60], and it is done because neural networks would overfit to the training data and would not generalize well if a (local) minimum is reached.

It is important to note that, for all non-convex models, the described stopping criteria cannot ensure that the best model is found, i.e. the optimal solutions are in local minima or in their vicinity. To improve this situation, we have added multi-start optimization to the hyperparameter selection; by doing so, when optimizing the hyperparameters, larger regions of the parameter space are explored and the quality of the obtained local solution can be improved.

5.3. Best Alternative per Modeling Option

In Section 4.4, we have described the three modeling options that are available for each benchmark model. In this section, we present and explain the best alternative for each of the options when considering the case study. It is important to note that all the results listed here are based on the validation dataset.

The obtained results are listed in Table 2 where, for each benchmark model and each modeling option, i.e. MO1, MO2, and MO3, the best model alternative is shown. In particular, the optimal alternative is given by one of the following labels:

- A1 (A2) to respectively denote that alternative 1 (2) performs the best.
- NI (non-important) to denote that the modeling option has no effect, i.e. both alternatives perform similarly.
- No label if the model cannot use the modeling option.

Based on the results of Table 2 we can draw the following conclusions:

1. Considering the results of modeling option MO1, preprocessing price spikes (Alternative A1) seems to be helpful for all statistical models. In contrast, preprocessing seems to be irrelevant or decrease the performance in the case of machine learning models. A possible explanation for this effect is the fact that price spikes are nonlinear effects, and as such, they can compromise the prediction quality of statistical models since they are largely linear [20]. In contrast, as machine learning models are able to model more complex nonlinear relations, it is possible that they can predict up to certain degree some of the nonlinear price spikes.
2. Observing the results of modeling option MO2, it is clear that, except for the non-parametric models, when the input features are optimally selected (Alternative A2) the accuracy of the models improves. In particular, the models obtain better performance when, instead of simply considering the load in the local market (Alternative A1), the model also includes input features like the load or generation in a neighboring market.
3. Analyzing the results of modeling option MO3, we can observe how the accuracy improvements by predicting multiple markets at the same time (Alternative A2)

	MO1	MO2	MO3
AR	A1		
DSARIMA	A1		
WARIMA	A1		
WARIMA-RBF	A1		
ARIMA-GARCH	A1		
DSHW	A1		
TBATS	A1		
DR	A1	NI	
TF	A1	NI	
ARX	A1	NI	
TARX	A1	NI	
IHMARX	A1	A1	
SNARX	A1	A1	
fARX	A1	A2	
fARX-Lasso	A1	A2	
fARX-EN	A1	A2	
MLP	NI	A2	NI
RBF	A1	A2	A1
SVR	NI	A2	
SOM-SVR	NI	A2	
SVR-ARIMA	NI	A2	
RF	A2		A1
XGB	A2		A1
DNN	A2		A2
LSTM	A2		A2
GRU	A2		A2
CNN	A2		A2

Table 2: Summary of which alternatives of the three modeling options perform the best for each of the 27 individual models. The labels A1|A2 respectively denote the case where alternative 1|2 performs the best. NI denotes the case where the modeling option has no effect. An empty cell means that the model cannot use the modeling option.

are restricted to the deep learning models. As originally argued in [3], this result is due to multi-tasking, a technique that can be successfully used to improve the predictive accuracy of deep neural networks but that might not be helpful for other models. In particular, when multi-tasking, deep neural networks solve auxiliary and related tasks, e.g. predicting neighboring markets, in order to generalize better and avoid overfitting.

5.4. Hyperparameter Optimization

In Section 3.6, we have described the hyperparameters that should be optimized for each DL model. In this section, we present the obtained optimal configurations for the case study. For the base models, while their hyperparameters are also optimized, including here the optimization results and hyperparameter definitions would require a huge amount of space. As a result, for the sake of conciseness, the results and definitions are listed in Appendix C.

When analyzing the results, it is important to keep in mind that all the hyperparameter solutions (and in turn the model sizes) depend on the current amount of data. In particular, as deep learning models employ a large number of parameters, they also require large amounts of data to accurately estimate their parameters. As a result, if the amount of data is not enough to obtain the best model in terms of prediction performance, the hyperparameter optimization could select a smaller model that performs better with the current amount of data but that is not the best model overall. As we argued in Section 5.7, this effect might explain the lower empirical performance observed for the most complex model, i.e. the CNN.

5.4.1. DNN Model

For the DNN, the optimal structure consists of a first and second hidden layers with respectively 239 and 162 neurons, the *rectifier linear unit* (ReLU) as the activation function, and no regularization nor dropout. The obtained optimal hyperparameters are summarized in Table 3.

Hyperparameter	**Value**
Activation Function	ReLU
Dropout	No
Regularization	No
n_1	239
n_2	162

Table 3: Optimal Hyperparameters for the DNN model.

5.4.2. LSTM Model

For the second proposed model, the optimal structure is an LSTM layer with 83 neurons and a regular layer with 184 neurons. Moreover, for the LSTM layer, the activation function is a *hyperbolic tangent (tanh)* function and the sequence length of input values is 2 weeks of past data. For the regular layer, the optimal activation is a ReLU function. In addition, none of the two layers require regularization nor dropout. The obtained optimal hyperparameters are represented in Table 4.

5.4.3. GRU Model

Similar to the LSTM-DNN model, the optimal hyperparameters for the GRU-DNN model are summarized in Table 5.

Hyperparameter	Value
Activation Function - DNN	ReLU
Activation Function - LSTM	Tanh
Dropout	No
Regularization	No
n_{DNN}	184
n_{LSTM}	83
Sequence Length	2 weeks

Table 4: Optimal Hyperparameters for the LSTM model.

Hyperparameter	Value
Activation Function - DNN	ReLU
Activation Function - LSTM	Tanh
Dropout	0.32
Regularization	No
n_{DNN}	166
n_{GRU}	132
Sequence Length	3 weeks

Table 5: Optimal Hyperparameters for the GRU model.

5.4.4. CNN Model

Finally, for the CNN model, the network that processes past data consists of three convolutional layers with respectively 64, 128, and 256 feature maps, each of them with a filter of size 3. After each of these layers, a max pooling operation and a batch normalization are performed. For the network that processes day-ahead data, the optimal structure is exactly the same. Both networks use the ReLU as activation function, a dropout factor of 0.31, and no regularization. The obtained optimal hyperparameters are summarized in Table 6.

5.4.5. General Observations

When analyzing the optimal hyperparameter results for the DL models, we can observe two interesting results that are common to the four models:

1. Except for the recurrent layers that require a `tanh` activation function, the optimal activation function for all the other deep learning layers is the `ReLU` function. This result agrees with the general observations in the field of DL, see e.g. [30], where `ReLU` is the default recommended activation function for any modern neural network with the exception of the LSTM and GRU cells, which by default require a `tanh` activation function.
2. Traditional regularization, i.e. performing dropout or penalizing with a L_1 norm the parameters of the neural network to impose sparsity on the network parameters, is in general not helpful (the only exception is the CNN model that does requires dropout). While this result might seem surprising (considering the small size of the datasets and the large number of parameters of the DL networks), it can be explained due to the combination of two effects:
   (a) While the proposed models are deep structures, they are less deep than DL networks used for more traditional applications, e.g. image or speech recognition. As a result, the number of parameters is smaller, and thus, the regularization step is less critical.
   (b) The models are trained using early stopping. While this is not a regularization technique by itself, it prevents overfitting. As a result, the regularization step becomes less critical.

Hyperparameter	Value
Activation Function	ReLU
Dropout	0.31
Regularization	No
Pooling frequency	1
Pooling type	Max pooling
Filter size - Past	3
Filter size - D.A.	3
Number of convolutions - Past	3
Number of convolutions - D.A.	3
Initial feature maps - Past	64
Initial feature maps - D.A.	64
Channel length	1 week

Table 6: Optimal hyperparameters for the CNN model. The label *D.A.* refers to the network that processes day-ahead data. The label *Past* refers to the network for past data.

5.5. Comparing Predictive Accuracy

After describing the experimental setup and obtaining the optimal model structures, we can compute and compare the predictive accuracy of the various models. However, to have a meaningful and complete assessment, not only the accuracy of the models should be computed, but also the statistical significance of the results should be established. In this section, we perform the first step of this analysis, i.e. we compute the accuracy of the models. Next, in the following section, the statistical tests are performed.

5.5.1. Main Results

To compare and analyze the predictive accuracy of the various forecasters, we compute their sMAPE on the test set. In addition, to guarantee that the assessment is similar to real conditions, i.e. that the forecaster is re-estimated when new data is available, the models are re-estimated on daily basis. The obtained results are listed in Table 7.

Model	sMAPE [%]	Class
DNN	12.34	ML
GRU	13.04	
LSTM	13.06	
MLP	13.27	
SVR	13.29	
SOM-SVR	13.36	
SVR-ARIMA	13.39	
XGB	13.74	
fARX-EN	13.76	SM
CNN	13.91	ML
fARX-Lasso	13.92	SM
RBF	14.77	ML
fARX	14.79	ST
RF	15.39	ML
IHMARX	16.72	ST
DR	16.99	
TARX	17.08	
ARX	17.34	
SNARX	17.58	
TBATS	17.9	
ARIMA-GARCH	19.3	
AR	19.31	
DSHW	19.4	
WARIMA-RBF	22.82	
WARIMA	22.84	
DSARIMA	23.40	
TF	23.57	

Table 7: Comparison of the predictive accuracy of the various forecasters by means of sMAPE. The labels ML and SM respectively refer to machine learning and statistical methods.

5.5.2. Observations

From the results displayed in Table 7, we can make various observations:

i. The DNN, GRU, and LSTM models, i.e. 3 of the 4 proposed DL forecasters, seem to outperform all the considered literature models.
ii. A line can be drawn between statistical models and machine learning methods. In particular, except for the fARX-based models, the other statistical methods perform worse than any artificially intelligence model.
iii. According to their performance, the models seem to be divided in eight clusters:
   (1) The DNN model with a 12.3% sMAPE.
   (2) The DL models with a recurrent layer, i.e. LSTM and GRU, with a 13% sMAPE.
   (3) The three SVR-based models and the MLP with a 13.3-13.4% sMAPE.
   (4) The CNN, the XGB, and the statistical models with automatic feature selection with a sMAPE between 13.7-13.9%.
   (5) The RF, the fARX, and the RBF models with a 14.7-15-3% sMAPE.
   (6) With a 16.7-17.9% sMAPE, the TBATS and the statistical methods with exogenous inputs but without moving average (except for the fARX).
   (7) With a 19.3-19.4% sMAPE, the ARIMA-GARCH and 2 of the 3 models without exogenous inputs nor moving average.
   (8) With a 22-23 % sMAPE, the statistical methods with a moving average term (except for the ARIMA-GARCH) .
iv. Surprisingly, the models with moving average seem to perform worse that their simpler AR counterparts.
v. The TBATS model appears to be the best alternative when no exogenous inputs are available. In particular, it even matches the performance of some statistical methods with exogenous inputs.
vi. From the considered models from the literature, SVRs and MLPs perform the best.
vii. The SVR hybrid methods, i.e. SVR-ARIMA and SOM-SVR, perform no different that the simple SVR model.

5.6. Statistical Testing

In this section, we study the statistical significance of the differences in predictive accuracy among the various forecasters.

5.6.1. Diebold-Mariano Test

To assess this statistical significance, we use the DM test as defined by (2)-(3), where the loss differential at time k is built using the absolute error:

$$d_k^{M_1,M_2} = |\varepsilon_k^{M_1}| - |\varepsilon_k^{M_2}|. \tag{7}$$

Moreover, we follow the procedure of [3, 85, 86] and we perform an independent DM test for each of the 24 time series representing the hours of a day. The reason for that is twofold: first, as we use the same information to forecast the set of 24 day-ahead prices, the forecast errors within the same day would exhibit a high correlation, and thus, the full error sequence would be correlated. Second, as we study each hour separately, the DM tests allow us to distinguish between three situations:

1. The accuracy of forecaster M_1 is significantly better than the one of forecaster M_2.
2. The accuracy of M_1 is significantly better than the accuracy of M_2, but at some hours, M_2's accuracy is significantly better.
3. M_1's accuracy is never significantly better than M_2's.

In detail, for each hour $h = 1, \ldots, 24$ and for each model pair M_1 and M_2, we perform a one-sided DM test, at a 95% confidence level, with the null hypothesis of the predictive accuracy of M_1 being equal or worse than M_2's:

$$\mathrm{DM}_h \begin{cases} H_0: \ \mathbb{E}[d_{h,k}^{M_1,M_2}] \geq 0, \\ H_1: \ \mathbb{E}[d_{h,k}^{M_1,M_2}] < 0, \end{cases} \quad \text{for } h = 1, \ldots 24, \tag{8}$$

where $[d_{h,1}^{M_1,M_2}, \ldots, d_{h,N/24}^{M_1,M_2}]$ represents the vector of loss differentials at hour h.[1]

Next, we perform the complementary one-side DM test with the null hypothesis of M_2 having the same or worse accuracy than M_1:

$$\hat{\mathrm{DM}}_h \begin{cases} H_0: \ \mathbb{E}[-d_{h,k}^{M_1,M_2}] \geq 0, \\ H_1: \ \mathbb{E}[-d_{h,k}^{M_1,M_2}] < 0, \end{cases} \quad \text{for } h = 1, \ldots 24. \tag{9}$$

Finally, we establish that the predictive accuracy of M_1 is significantly better than M_2's if two conditions are met:

1. In at least one of the regular DM_h tests the null hypothesis is rejected, i.e. the predictive accuracy of M_1 is at least significantly better in 1 of the 24 prediction windows.
2. None of the complementary $\hat{\mathrm{DM}}_h$ tests rejects the null hypothesis, i.e. the predictive accuracy of M_2 is not significantly better in any of the 24 prediction horizons.

If both M_1 and M_2 are at least significantly better in one of the 24 prediction windows, we perform a further DM test considering the full vector of loss differential $[d_1, \ldots, d_N]^\top$.[2] Specifically, recalling that optimal k-step-ahead forecast errors are at most $(k-1)$-dependent [56], we perform a DM test on the full loss differential considering serial correlation of order 23:

$$\mathrm{DM}_{\mathrm{sc}} \begin{cases} H_0: \ \mathbb{E}[d^{M_1,M_2}] \geq 0, \\ H_1: \ \mathbb{E}[d^{M_1,M_2}] < 0. \end{cases} \tag{10}$$

If the null hypothesis of $\mathrm{DM}_{\mathrm{sc}}$ is rejected, we consider that, while at some hours M_2's accuracy is significantly better than M_1's, M_1's accuracy is significantly better when considering the full error sequence.

5.6.2. Results

The obtained results are summarized in Table 8. There are three possible scenarios:

1. Cells that display a ✓ represent the cases where the alternative hypothesis is accepted with a 95% confidence, i.e. the predictive accuracy of M_1 is statistically significantly better than the one of M_2.
2. Cells that display a ✓$_\mathbf{s}$ represent the cases where, while the predictive accuracy of M_2 is at least significantly better in one of the 24 predictive horizons, the overall predictive accuracy of M_1 when considering the full loss differential is still statistically significantly better.
3. Empty cells represent the cases where M_1 is not significantly better than M_2.

Considering the results listed in Table 8, we confirm the various observations made in Section 5.5.2:

1. The DNN, LSTM, and GRU models, i.e. 3 of the 4 proposed forecasters, are indeed statistically significantly better than the rest. In particular, the DNN shows a predictive accuracy that is statistically significantly better than the accuracy of all others. In addition, the LSTM and GRU models have an accuracy that is statistically significantly better than all others except the MLP.
2. Except for the fARX-based models, the accuracy of the machine learning methods is statistically significantly better than the accuracy of statistical methods.
3. Based on accuracy differences that are statistically significant, we can observe a very similar group separation pattern as the one described in Section 5.5.2.
4. The models with moving average terms have an accuracy that is statistically significantly worse than their AR counterparts.
5. The TBATS model has an accuracy that is statistically significantly better than any other model without exogenous inputs.
6. The accuracy of the SVR and hybrid-SVR models is not statistically significantly different.

To illustrate the first observation, i.e. that the proposed DNN, GRU and LSTM models are significantly better than the rest, we depict in Figures 4 and 5 the test statistics obtained when applied to the DNN and GRU models. In these figures, at each hour h, the points above the upper horizontal line accept, at a 95 % confidence, the alternative hypothesis in DM_h, i.e. that the specific DL model has an accuracy that is statistically significantly better. Similarly, any point below the lower horizontal line accepts, at a 95 % confidence, the alternative hypothesis in $\hat{\mathrm{DM}}_h$, i.e. that the specific DL model has an accuracy that is statistically significantly worse.

From Figure 4 representing the DNN results we can observe how, except for the LSTM and GRU models, for any other forecaster the DNN is at least significantly better at one hour and never significantly worse. In other words, the DNN is statistically significantly better than all other models except the LSTM and GRU forecasters. When compared with these two, while the DNN shows an overall accuracy that is statistically significantly better, the LSTM's accuracy is better at hours 01:00 and 22:00, and the GRU's accuracy at hours 01:00, 02:00, 03:00, and 06:00.

[1] $N/24$ losses $d_{h,k}$ per hour h as there are N time points.

[2] Notice that $d_{h,k}^{M_1,M_2} = d_{24(k-1)+h}^{M_1,M_2}$

Model Class	Statistical Methods													ML	ST	ML	ST	ML	ST	Machine Learning							
M_1 \ M_2	TF	DSARIMA	WARIMA	WARIMA-RBF	DSHW	AR	ARIMA-GARCH	TBATS	SNARX	ARX	TARX	DR	IHMARX	RF	fARX	RBF	fARX-Lasso	CNN	fARX-EN	XGB	SVR-ARIMA	SOM-SVR	SVR	MLP	LSTM	GRU	DNN
TF																											
DSARIMA																											
WARIMA																											
WARIMA-RBF																											
DSHW	✓s	✓s	✓s	✓s																							
AR	✓	✓s	✓s	✓s																							
ARIMA-GARCH	✓	✓	✓	✓																							
TBATS	✓s	✓s	✓s	✓s	✓s	✓s	✓s																				
SNARX	✓	✓	✓	✓	✓s	✓	✓s																				
ARX	✓s	✓s	✓s	✓s	✓s	✓s	✓s																				
TARX	✓s	✓s	✓s	✓s	✓s	✓s	✓s			✓																	
DR	✓	✓	✓	✓	✓	✓s	✓		✓s																		
IHMARX	✓	✓	✓	✓	✓s	✓s	✓s	✓s	✓	✓																	
RF	✓	✓	✓	✓	✓	✓s	✓	✓s	✓	✓	✓s	✓	✓s														
fARX	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓														
RBF	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s													
fARX-Lasso	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓s												
CNN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓												
fARX-EN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓		✓s										
XGB	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓								
SVR-ARIMA	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓	✓	✓s		✓s								
SOM-SVR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓	✓	✓	✓s	✓s								
SVR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s								
MLP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓s	✓s	✓s	✓s						
LSTM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓s	✓s	✓s	✓s				
GRU	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓s	✓	✓	✓				
DNN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓s	✓s	

Table 8: DM results for the base and DL models. ✓ represents the cases where M_1's accuracy is statistically significantly better than M_2's. ✓s represent the cases where, while M_2's accuracy is at least significantly better in one of the 24 hours, the accuracy of M_1 is still statistically significantly better if the whole loss differential sequence is considered. The labels ST and ML respectively refer to statistical and machine learning methods.

From Figure 5 representing the GRU results we can draw similar conclusions. In particular, the GRU model is statistically significantly better than all models except the DNN, LSTM, GRU, MLP, XGB and fARX-EN. However, for the XGB and fARX-EN models, while their accuracy is statistically significantly better at one hour, the GRU has an overall accuracy that is significantly better. From Figure 6 representing the LSTM results, we can draw similar conclusions as the ones obtained from Figure 5.

For the sake of simplicity, Table 8 only represents a summary of all the performed DM tests; particularly, as a total of 17550 DM tests were performed ($\frac{27!}{25!2!}$ model pairs $\times$ 50 DM test per model pair), it is impossible to list them all neither here nor even in an appendix. To address that, we have created a website (`goo.gl/FzA4Cb`) where all the DM test results can be obtained. In particular, following the same structure as Figures 4 and 5, we have upload 27 figures representing the DM results for the 27 models. In addition, we have also uploaded an excel sheet with all the p-values of the 17550 DM tests.

5.7. Discussion

To discuss the obtained results, we distinguish between three different topics: an analysis specific to the proposed DL models, an evaluation of the general results of the benchmark study, and a discussion on why neural networks have usually failed to predict electricity prices but in this paper they represent the best model.

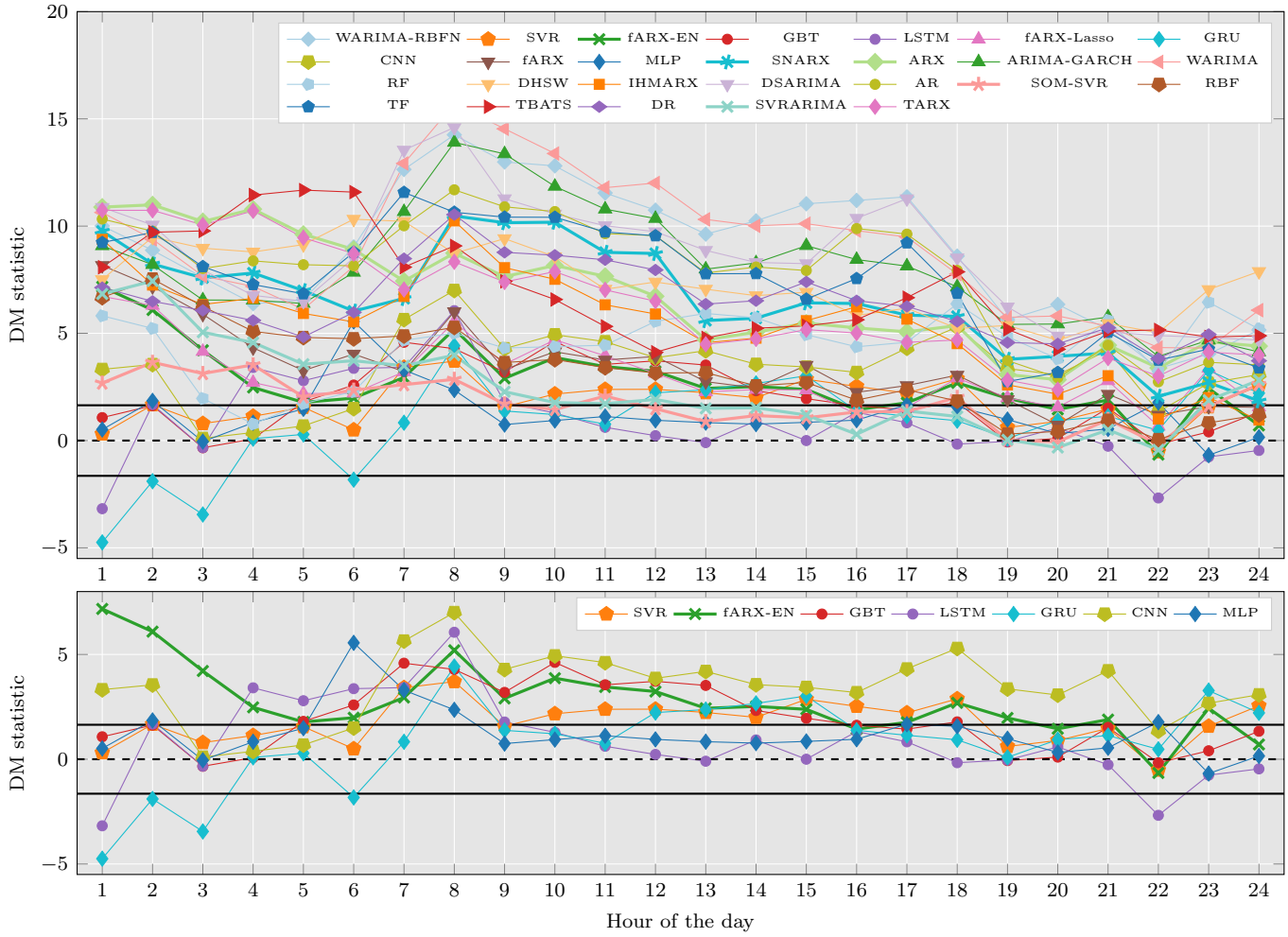

Figure 4: DM results for the DNN model. **Top**: test results for all 26 models. **Bottom**: test results for the top performing models. Values above the top dashed line represent cases where, with a 95 % confidence level, the DNN is significantly better. Similarly, values below the lower dashed line accept at a 95 % confidence level that the DNN is significantly worse.

5.7.1. DL Models

From the results and observations that are drawn in the previous section, we can conclude that the proposed DNN, GRU and LSTM models are the best alternative for forecasting day-ahead prices in the Belgian market. In particular, the benchmark is quite large and these 3 models outperform all the rest in a statistically significant manner.

Moreover, while the DNN is significantly better than the GRU and LSTM forecasters, these two are better at some specific hours. Therefore, if a highly accurate system is targeted, e.g. by combining several forecasts, the three DL models are still necessary. However, if a single model is to be used, e.g. due to limitations in computation, the DNN is clearly the forecaster of choice.

Something that is interesting worthy to discuss is the reason why the GRU, LSTM, and CNN models perform worse than the DNN. In particular, the four of them are deep structures with the potential to model complex non-linear patterns and, in the case of the GRU and LSTM models, they are especially appropriate for modeling time series data. So, how can it be that the DNN has an accuracy that is statistically significantly better? There are two possible hypotheses:

1. The amount of data: DL models require large amounts of data to be properly trained. When comparing the four DL models, the DNN has fewer parameters than the other three; as a result, it might be easier to train. This hypothesis also agrees with the fact that the CNN performance is the worse of the four as it is the model with the largest number of parameters.
2. A second possible reason is the structure of the networks. In particular, the GRU, LSTM, and CNN models separate the data corresponding to the day-ahead and the past data in two different networks. As a result, if some past data and day-ahead data are heavily related, none of the three structures is able to build these relations properly. By contrast, the DNN

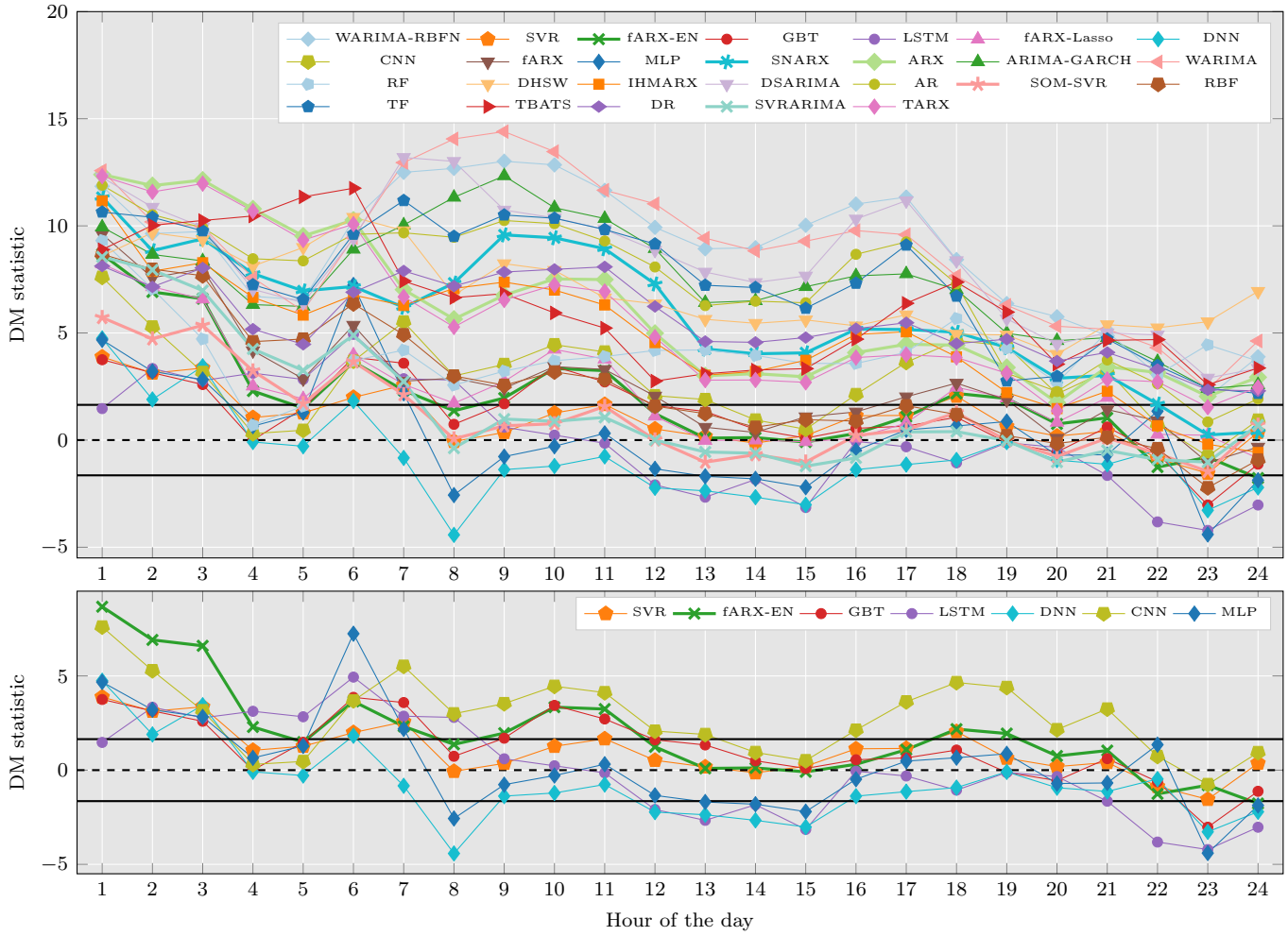

Figure 5: DM results for the GRU model. **Top**: results for all 26 models. **Bottom**: results for the top performing models. Values above the top dashed line represent cases where, with a 95 % confidence level, the GRU is significantly better. Similarly, values below the lower dashed line accept at a 95 % confidence level that the GRU is significantly worse.

model makes no assumption about the input data and allows any possible relation to be built.

It is important to note that these are just hypothesis. In particular, further research is necessary to properly explain this effect.

The last finding worthy to discuss is the performance of the CNN. In particular, the fourth proposed DL model performs no better than simpler machine learning methods like XGB or SVR. An extra hypothesis (in addition to the provided two) to explain this effect is the fact that the CNN uses local operations. In particular, given some layer, the CNN does not interrelate all its values when making the connections to the next layer, but performs local convolution operations that interrelate local groups of data. As a result, while this structure is very convenient to process some specific type of data, e.g. pictures, it might not be appropriate if all the input data is highly correlated, e.g. seasonal time series data like electricity prices.

5.7.2. Benchmark

Regarding the benchmark results, besides being the proposed DNN, GRU, and LSTM models the best forecasters, several other effects need to be discussed.

Moving Average Models

One of the most important effects to be examined is the fact that statistical models with moving average terms perform worse than their AR-counterparts. In particular, as the moving average terms provide an additional resource to model error correlation, they should have the potential to be more accurate. However, if we consider the structure of the model estimation, we can observe that the former is not necessarily true: as the moving average term leads to models that are estimated using non-convex optimization, the global minima is not guaranteed and the resulting models might have a lower performance.

Despite this explanation, the truth is that, when looking at the literature of electricity price forecasting, mov-

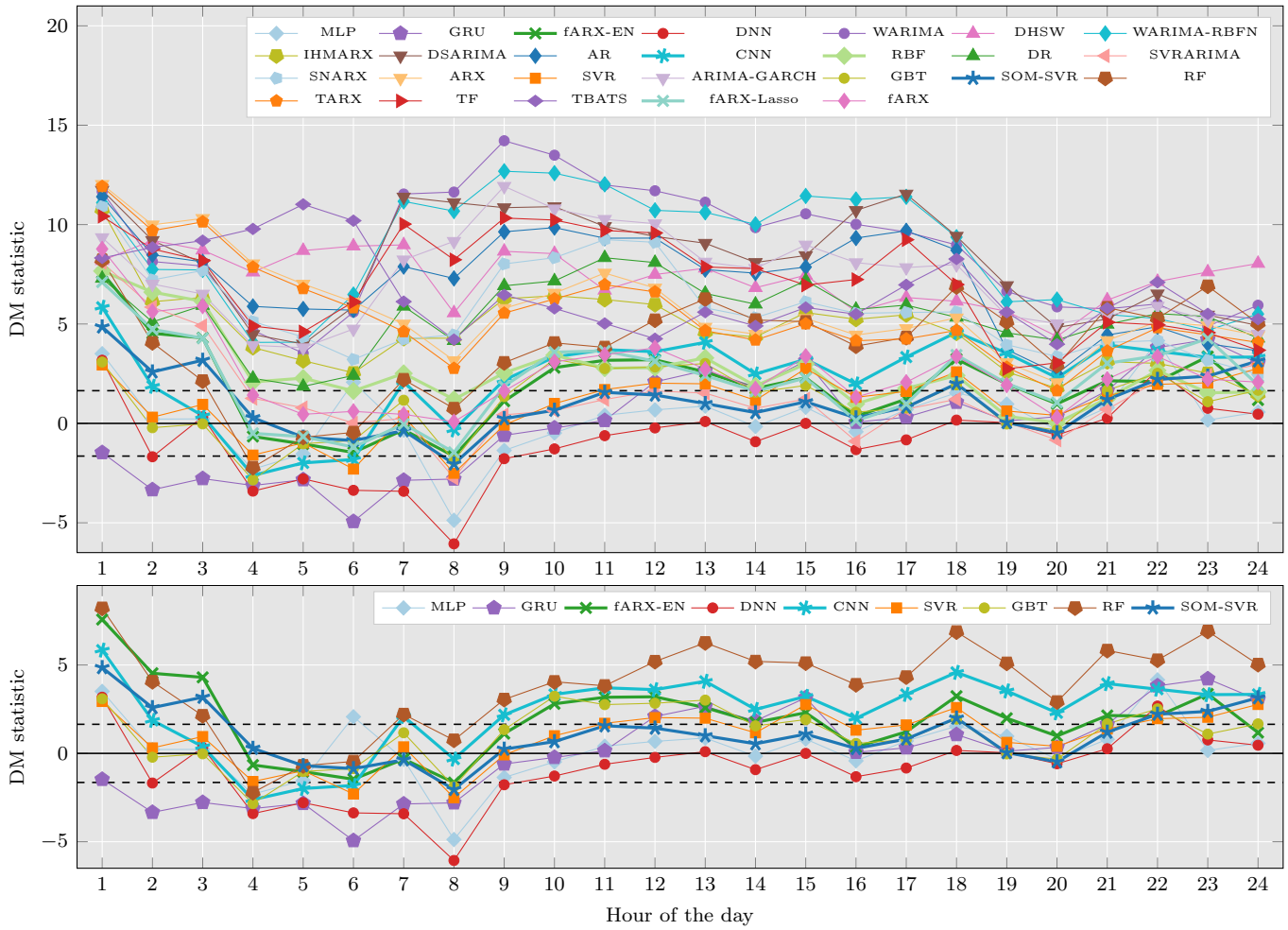

Figure 6: DM results for the LSTM model. **Top**: results for all 26 models. **Bottom**: results for the top performing models. Values above the top dashed line represent cases where, with a 95 % confidence level, the LSTM is significantly better. Similarly, values below the lower dashed line accept at a 95 % confidence level that the LSTM is significantly worse.

ing average models have traditionally outperformed their AR counterparts. A possible explanation for this performance mismatch between past studies and this paper is the change in the dynamics of day-ahead electricity prices during the last years. In particular, due to the increasing penetration of renewable sources, day-ahead prices are becoming more volatile and price spikes are occurring more often. Due to these effects, the resulting optimization problems might be more nonlinear, and in turn, the global minima might become harder to reach.

Machine Learning vs. Statistical Methods

Another effect worth discussing is the fact that machine learning methods clearly outperform statistical methods. In particular, while several past studies led to empirical results that showed that the accuracy of machine learning methods was not better than the one of statistical methods, we can clearly observe that this is not the case in the EPEX-Belgium market. Possible explanations for this effect can be the following: as before, the market under study has large nonlinearities and spikes, and thus, it requires complex nonlinear models to accurately forecast the prices. In addition, the computational power of recent years has dramatically increased, and thus, more data can be used for parameter estimation and the structure of the considered machine learning methods can be more complex. The latter argument also agrees with the fact that DL models have the best performance.

fARX-Based Models

An exception to the previous statement are the fARX-based models. In particular, despite being statistical methods, they clearly perform better than any other statistical method and even better that some machine learning algorithms. These results confirm the findings of [67] and show that this model is one of the best statistical methods for predicting electricity prices. A possible explanation for this performance is the combination of two characteristics:

1. The structure of these models is very general and includes many possible exogenous inputs, which makes them very flexible.
2. At the same time, they use automatic feature selection to reduce the model complexity and make the models tailored to the market under study.

Hybrid Models

A fourth important consideration is the fact that, in general, hybrid models do not outperform their regular versions. In particular, nor SVR-ARIMA nor SOM-SVR outperform the simpler SVR model. Likewise, WARIMA-RBF does not outperform the simpler WARIMA. An exception might be the ARIMA-GARCH, which outperforms the WARIMA and DSARIMA models.

TBATS

A final remark to be made is the fact that the TBATS model is clearly the best choice to predict prices when no regressors are available, and it even is a good choice when exogenous inputs exist. This observation is very important as, to the best of our knowledge, nobody has ever tested the accuracy of the TBATS model for predicting day-ahead electricity prices.

5.7.3. Why do the Proposed Models Improve the Performance?

When we consider the literature of electricity price forecasting, there are many examples where neural networks have been outperformed by other forecasters [12, 27, 42, 51, 62, 63, 87–90]. The results obtained in this paper lead to the opposite conclusion: in this case study, neural networks outperform all other models. In this section, to clarify this discrepancy, we provide the rationale behind the superior performance of the proposed DL models. In particular, we examine four features that past studies have typically not considered and we argue that by not considering them the accuracy worsens.

Depth

As briefly motivated in the introduction, deep neural networks can generalize and obtain better results than their shallow counterparts. This effect is related to the universal approximation theorem [91], which states that a neural network with a linear output layer can approximate any continuous function on compact subsets of $\mathbb{R}^n$ provided that it has enough neurons, but does not indicate whether this number is tractable [30]. In particular, to approximate some families of functions, the number of neurons required by a shallow network can grow exponentially with the input size and in turn become intractable [30]. In the same context, the family of functions could be approximated by a tractable number of neurons if the depth is larger than some threshold number d [30].

As a result, when approximating a target function, a deep network might need a much smaller number of neurons than its shallow counterpart, and thus, it might be able to approximate the function more easily and better. In our case study, this effect is observed in Table 7, where a shallow neural network, i.e. the MLP model, has a lower accuracy than the DNN, GRU, and LSTM models.

If we look now at the literature of electricity price forecasting, the neural networks that have usually been proposed have been shallow networks [12, 22, 42, 51, 62, 63, 87–90]. Therefore, considering the above argument, it might be normal for them to perform worse that the deeper models than we propose.

Number of Neurons

Intrinsically related to the previous argument and with the universal approximation theorem [91] is the fact that, in order for a network to correctly approximate a function, the number of neurons needs to be large enough. However, when we consider the literature of electricity price forecasting, most of the studies have employed small MLPs with less than 50 neurons [22, 40–42, 51, 62, 63, 87, 89, 90] and have not performed any hyperparameter optimization to select the required number of neurons.

If this case study, the empirical results show that the optimal number of neurons for the MLP model is 117 (see Appendix C). While the optimal number will change from case to case, we can use it as a reference to argue that the small-sized neural networks previously proposed in the literature might not be large enough to model the complex dynamics of electricity prices.

To strengthen our argument, we analyze this effect in our case study: we consider a MLP with 50 neurons and we compare its performance against the optimized MLP using 117 neurons. As it would be expected, the MLP with 50 neurons fails to perform as good as the optimized one: its accuracy on the test set drops from 13.27% to 14.30% sMAPE and this difference in accuracy is statistically significantly for all 24 hours. In addition, another finding that reinforces our argument is the fact that, if we were to use this smaller MLP in the benchmark it would not be better than half of the models, which would agree with the literature results.

Size of Training Dataset

Even if the network is large enough to approximate the targeted function, the optimizer might fail to estimate the right parameters [30]. In particular, a possible problem that the optimizer might face is not having enough training data to estimate the large number of parameters in a neural network, e.g. in our MLP model with 117 neurons there are approximately 28200 parameters.

When we examine the literature of electricity price forecasting, studies have usually considered networks that were trained using 1 year of data or less [12, 22, 27, 42, 51,

62, 63, 87–90]. If we consider our case study, that might not be enough: if trained with 1 year of data, the accuracy of the DNN drops from 12.34 % to 13.27 % sMAPE, an accuracy that is worse than the performance of many benchmark models, and which might explain again some of the literature results.

Stochastic Gradient Descent

A second problem that might also affect the parameter estimation regards the properties of the optimization algorithm itself. In particular, in the literature of electricity price forecasting, network parameters have traditionally been estimated using standard gradient descent methods, e.g. batch gradient descent (also known as backpropagation) or the Levenberg–Marquardt optimization [2, 22, 40–42, 51, 88–90]. These methods, while they might work well for small sized-networks, they display computational and scalability issues and they often obtain worse results [92].

A better alternative is the family of *stochastic gradient descent* methods [92, 93], which, instead of computing the gradient w.r.t. to the whole training dataset, they do it w.r.t. to subsets of it. In our case study, if batch gradient descent is used instead of adam, i.e. a type of stochastic gradient descent method, the accuracy of the DNN drops from 12.34 % to 14.15 %. Based on this empirical result and the argument above, it is clear that this effect might also account for some of the discrepancies between our work and the literature.

6. Conclusions

In this paper, four different *deep learning* (DL) models to predict day-ahead electricity prices are proposed. Moreover, a large benchmark study is set up in order to compare the predictive accuracy of the proposed models w.r.t. to other models in the literature. This benchmark is selected to comprise as many models as possible and to serve as a reference within the field of day-ahead electricity price forecasting.

Three of the four proposed DL forecasters, i.e. the *deep neural network* (DNN) model, the *long-short term memory* (LSTM) model, and the *gated recurrent unit* (GRU) model, are shown to obtain a predictive accuracy that is statistically significantly better than all other models. In addition, among these three models, the DNN is able to outperform the other two with a difference in accuracy that is statistically significant. Despite this difference, the three models are necessary to obtain a high-performing forecaster, as the accuracy of the GRU and LSTM models still better at some specific hours.

Among the rest of the forecasters, it is observed a clear division is observed between machine learning and statistical methods, where the former display an accuracy that is statistically significantly better. In addition, models with moving average terms are shown to suffer the worst performance and hybrid methods are shown not to outperform their simpler counterparts.

In future work, this research will be expanded with four further investigations. First, the effect in forecasting accuracy of more advanced DL techniques, e.g. autoencoders, will be analyzed. Second, the usage of expert advice to combine the individual benchmark models will be studied. Third, the benchmark comparison will be extended to other markets. Fourth, the effect of the dataset size for each model will be extensively analyzed using a large number of experiments.

Acknowledgment

This research has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 675318 (INCITE).

References

[1] M. Shahidehpour, H. Yamin, Z. Li, Market overview in electric power systems, in: Market Operations in Electric Power Systems, John Wiley & Sons, Inc., New York, USA, 2002, Ch. 1, pp. 1–20.

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

[3] J. Lago, F. De Ridder, P. Vrancx, B. De Schutter, Forecasting day-ahead electricity prices in Europe: The importance of considering market integration, Applied Energy 211 (2018) 890–903.

[4] C. Brancucci Martinez-Anido, G. Brinkman, B.-M. Hodge, The impact of wind power on electricity prices, Renewable Energy 94 (2016) 474–487.

[5] R. Baldick, Wind and energy markets: A case study of texas, IEEE Systems Journal 6 (1) (2012) 27–34.

[6] I. Milstein, A. Tishler, Can price volatility enhance market power? the case of renewable technologies in competitive electricity markets, Resource and Energy Economics 41 (2015) 70–90.

[7] R. Green, N. Vasilakos, Market behaviour with large amounts of intermittent generation, Energy Policy 38 (7) (2010) 3211–3220.

[8] R. Weron, A. Misiorek, Forecasting spot electricity prices: A comparison of parametric and semiparametric time series models, International Journal of Forecasting 24 (4) (2008) 744–763.

[9] A. Cruz, A. Muñoz, J. Zamora, R. Espínola, The effect of wind generation and weekday on Spanish electricity spot price forecasting, Electric Power Systems Research 81 (10) (2011) 1924–1935.

[10] A. Misiorek, S. Trueck, R. Weron, Point and interval forecasting of spot electricity prices: Linear vs. non-linear time series models, Studies in Nonlinear Dynamics & Econometrics 10 (3) (2006) 1–36.

[11] J. Crespo Cuaresma, J. Hlouskova, S. Kossmeier, M. Obersteiner, Forecasting electricity spot-prices using linear univariate time-series models, Applied Energy 77 (1) (2004) 87–106.

[12] Z. Yang, L. Ce, L. Lian, Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods, Applied Energy 190 (2017) 291–305.

[13] J. M. Vilar, R. Cao, G. Aneiros, Forecasting next-day electricity demand and price using nonparametric functional methods, International Journal of Electrical Power & Energy Systems 39 (1) (2012) 48–55.
[14] C. R. Knittel, M. R. Roberts, C. Knittel, M. Roberts, An empirical examination of restructured electricity prices, Energy Economics 27 (5) (2005) 791–817.
[15] R. C. Garcia, J. Contreras, M. Van Akkeren, J. B. C. Garcia, A GARCH forecasting model to predict day-ahead electricity prices, IEEE Transactions on Power Systems 20 (2) (2005) 867–874.
[16] A. K. Diongue, D. Guégan, B. Vignal, Forecasting electricity spot market prices with a k-factor GIGARCH process, Applied Energy 86 (4) (2009) 505–510.
[17] F. J. Nogales, J. Contreras, A. J. Conejo, R. Espínola, Forecasting next-day electricity prices by time series models, IEEE Transactions on Power Systems 17 (2) (2002) 342–348.
[18] A. Conejo, M. Plazas, R. Espinola, A. Molina, Day-ahead electricity price forecasting using the wavelet transform and ARIMA models, IEEE Transactions on Power Systems 20 (2) (2005) 1035–1042.
[19] Z. Tan, J. Zhang, J. Wang, J. Xu, Day-ahead electricity price forecasting using wavelet transform combined with ARIMA and GARCH models, Applied Energy 87 (11) (2010) 3606–3610.
[20] N. Amjady, M. Hemmati, Energy price forecasting - problems and proposals for such predictions, IEEE Power and Energy Magazine 4 (2) (2006) 20–29.
[21] B. Szkuta, L. Sanabria, T. Dillon, Electricity price short-term forecasting using artificial neural networks, IEEE Transactions on Power Systems 14 (3) (1999) 851–857.
[22] J. P. S. Catalão, S. J. P. S. Mariano, V. M. F. Mendes, L. A. F. M. Ferreira, Short-term electricity prices forecasting in a competitive market: A neural network approach, Electric Power Systems Research 77 (10) (2007) 1297–1304.
[23] L. Xiao, W. Shao, M. Yu, J. Ma, C. Jin, Research and application of a hybrid wavelet neural network model with the improved cuckoo search algorithm for electrical power system forecasting, Applied Energy 198 (2017) 203–222.
[24] D. Wang, H. Luo, O. Grunder, Y. Lin, H. Guo, Multi-step ahead electricity price forecasting using a hybrid model based on two-layer decomposition technique and BP neural network optimized by firefly algorithm, Applied Energy 190 (2017) 390–407.
[25] S. Fan, C. Mao, L. Chen, Next-day electricity-price forecasting using a hybrid network, IET Generation, Transmission & Distribution 1 (1) (2007) 176–182.
[26] J. Che, J. Wang, Short-term electricity prices forecasting based on support vector regression and auto-regressive integrated moving average modeling, Energy Conversion and Management 51 (10) (2010) 1911–1917.
[27] W.-M. Lin, H.-J. Gow, M.-T. Tsai, An enhanced radial basis function network for short-term electricity price forecasting, Applied Energy 87 (10) (2010) 3226–3234.
[28] S. K. Aggarwal, L. M. Saini, A. Kumar, Electricity price forecasting in deregulated markets: A review and evaluation, International Journal of Electrical Power & Energy Systems 31 (1) (2009) 13–22.
[29] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554.
[30] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[31] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS'12, Curran Associates Inc., USA, 2012, pp. 1097–1105.
[32] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, Signal Processing Magazine 29 (6) (2012) 82–97.
[33] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv eprint (2014). arXiv:1409.0473.
[34] H. Wang, G. Wang, G. Li, J. Peng, Y. Liu, Deep belief network based deterministic and probabilistic wind speed forecasting approach, Applied Energy 182 (2016) 80–93.
[35] I. Coelho, V. Coelho, E. Luz, L. Ochi, F. Guimarães, E. Rios, A GPU deep learning metaheuristic based model for time series forecasting, Applied Energy 201 (2017) 412–418.
[36] C. Fan, F. Xiao, Y. Zhao, A short-term building cooling load prediction method using deep learning algorithms, Applied Energy 195 (2017) 222–233.
[37] H.-Z. Wang, G.-Q. Li, G.-B. Wang, J.-C. Peng, H. Jiang, Y.-T. Liu, Deep learning based ensemble approach for probabilistic wind power forecasting, Applied Energy 188 (2017) 56–70.
[38] X. Kong, X. Xu, Z. Yan, S. Chen, H. Yang, D. Han, Deep learning hybrid method for islanding detection in distributed generation, Applied Energy.
[39] C. Feng, M. Cui, B.-M. Hodge, J. Zhang, A data-driven multi-model methodology with deep feature selection for short-term wind forecasting, Applied Energy 190 (2017) 1245–1257.
[40] I. P. Panapakidis, A. S. Dagoumas, Day-ahead electricity price forecasting via the application of artificial neural network based models, Applied Energy 172 (2016) 132–151.
[41] D. Keles, J. Scelle, F. Paraschiv, W. Fichtner, Extended forecast methods for day-ahead electricity spot prices applying artificial neural networks, Applied Energy 162 (2016) 218–230.
[42] M. Shafie-Khah, M. P. Moghaddam, M. Sheikh-El-Eslami, Price forecasting of day-ahead electricity markets using a hybrid forecast method, Energy Conversion and Management 52 (5) (2011) 2165–2169.
[43] V. Sharma, D. Srinivasan, A hybrid intelligent model based on recurrent neural networks and excitable dynamics for price prediction in deregulated electricity market, Engineering Applications of Artificial Intelligence 26 (5) (2013) 1562–1574.
[44] S. Anbazhagan, N. Kumarappan, Day-ahead deregulated electricity market price forecasting using recurrent neural network, IEEE Systems Journal 7 (4) (2013) 866–872.
[45] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.
[46] K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv eprint (2014). arXiv:1409.1259.
[47] A. Graves, Generating sequences with recurrent neural networks, arXiv eprint (2013). arXiv:1308.0850.
[48] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv eprint (2014). arXiv:1412.3555.
[49] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14, MIT Press, 2014, pp. 3104–3112.
[50] N. Amjady, A. Daraeepour, F. Keynia, Day-ahead electricity price forecasting by modified relief algorithm and hybrid neural network, IET Generation, Transmission & Distribution 4 (3) (2010) 432–444.
[51] A. J. Conejo, J. Contreras, R. Espínola, M. A. Plazas, Forecasting electricity prices for a day-ahead pool-based electric energy market, International Journal of Forecasting 21 (3) (2005) 435–462.
[52] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: Advances in Neural Information Processing Systems, 2011, pp. 2546–2554.
[53] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in: International Conference on Learning and Intelligent Optimization, Springer, 2011, pp. 507–523.
[54] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global Optimization 13 (4) (1998) 455–492.

[55] S. Makridakis, Accuracy measures: theoretical and practical concerns, International Journal of Forecasting 9 (4) (1993) 527–529.
[56] F. X. Diebold, R. S. Mariano, Comparing predictive accuracy, Journal of Business & Economic Statistics 13 (3) (1995) 253–263.
[57] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958.
[58] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv eprint (2014). arXiv:1412.6980.
[59] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of COMPSTAT'2010, Physica-Verlag HD, Heidelberg, 2010, pp. 177–186.
[60] Y. Yao, L. Rosasco, A. Caponnetto, On early stopping in gradient descent learning, Constructive Approximation 26 (2) (2007) 289–315.
[61] N. Amjady, F. Keynia, Day-ahead price forecasting of electricity markets by mutual information technique and cascaded neuro-evolutionary algorithm, IEEE Transactions on Power Systems 24 (1) (2009) 306–318.
[62] V. Vahidinasab, S. Jadid, A. Kazemi, Day-ahead price forecasting in restructured power systems using artificial neural networks, Forecasting next-day price of electricity in the Spanish energy market using artificial neural networks 78 (8) (2008) 1332–1342.
[63] H. M. I. Pousinho, V. M. F. Mendes, J. P. S. Catalão, Short-term electricity prices forecasting in a competitive market by a hybrid PSO–ANFIS approach, International Journal of Electrical Power & Energy Systems 39 (1) (2012) 29–35.
[64] R. Weron, A. Misiorek, Forecasting spot electricity prices with time series models, in: Proceedings of the European Electricity Market EEM-05 Conference, 2005, pp. 133–141.
[65] J. W. Taylor, Short-term electricity demand forecasting using double seasonal exponential smoothing short-term electricity demand forecasting using double seasonal exponential, Journal of Operational Research Society 54 (2003) 799–805.
[66] A. M. De Livera, R. J. Hyndman, R. D. Snyder, Forecasting time series with complex seasonal patterns using exponential smoothing, Journal of the American Statistical Association 106 (496) (2011) 1513–1527.
[67] B. Uniejewski, J. Nowotarski, R. Weron, Automated variable selection and shrinkage for day-ahead electricity price forecasting, Energies 9 (8) (2016) 621.
[68] R. Cao, J. Hart, A. Saavedra, Nonparametric maximum likelihood estimators for AR and MA time series, Journal of Statistical Computation and Simulation 73 (5) (2003) 347–360.
[69] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) (1996) 267–288.
[70] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67 (2) (2005) 301–320.
[71] D. C. Sansom, T. Downs, T. K. Saha, Evaluation of support vector machine based forecasting tool in electricity price forecasting for australian national electricity market participants, Journal of Electrical and Electronics Engineering Australia 22 (3) (2003) 227–234.
[72] D. Niu, D. Liu, D. D. Wu, A soft computing system for day-ahead electricity price forecasting, Applied Soft Computing 3 (2010) 868–875.
[73] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.
[74] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.
[75] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 785–794.
[76] G. E. P. Box, D. R. Cox, An analysis of transformations, Journal of the Royal Statistical Society. 26 (2) (1964) 211–252.
[77] ENTSO-E transparency platform, `https://transparency.entsoe.eu/`. Accessed on 15.05.2017.
[78] RTE, Grid data, `https://data.rte-france.com/`. Accessed on 15.05.2017.
[79] Elia, Grid data, `http://www.elia.be/en/grid-data/dashboard`. Accessed on 15.05.2017.
[80] F. Chollet, Keras, `https://github.com/fchollet/keras` (2015).
[81] Theano Development Team, Theano: A Python framework for fast computation of mathematical expressions, arXiv eprint (2016). arXiv:1605.02688.
[82] R. J. Hyndman, forecast: Forecasting functions for time series and linear models (2017).
[83] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.
[84] J. Andersson, A General-Purpose Software Framework for Dynamic Optimization, PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium (October 2013).
[85] F. Ziel, R. Steinert, S. Husmann, Forecasting day ahead electricity spot prices: The impact of the EXAA to other European electricity markets, Energy Economics 51 (2015) 430–444.
[86] J. Nowotarski, E. Raviv, S. Trück, R. Weron, An empirical comparison of alternative schemes for combining electricity spot price forecasts, Energy Economics 46 (2014) 395–412.
[87] R. Pino, J. Parreno, A. Gomez, P. Priore, Forecasting next-day price of electricity in the spanish energy market using artificial neural networks, Engineering Applications of Artificial Intelligence 21 (1) (2008) 53–62.
[88] N. Amjady, F. Keynia, Day ahead price forecasting of electricity markets by a mixed data model and hybrid forecast method, International Journal of Electrical Power & Energy Systems 30 (9) (2008) 533–546.
[89] F. Keynia, A new feature selection algorithm and composite neural network for electricity price forecasting, Engineering Applications of Artificial Intelligence 25 (8) (2012) 1687–1697.
[90] S. K. Aggarwal, L. M. Saini, A. Kumar, Electricity price forecasting in ontario electricity market using wavelet transform in artificial neural network based model, International Journal of Control Automation and Systems 6 (5) (2008) 639–650.
[91] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[92] Y. LeCun, L. Bottou, G. B. Orr, K.-R. Müller, Efficient BackProp, in: G. B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, no. 1524 in Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 9–50.
[93] S. Ruder, An overview of gradient descent optimization algorithms, arXiv eprint (2016). arXiv:1609.04747.
[94] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166.
[95] A. Borovykh, S. Bohte, C. W. Oosterlee, Conditional time series forecasting with convolutional neural networks, arXiv eprint (2017). arXiv:1703.04691.

Supplementary Material

The following appendices act as supplementary material to this research paper. They provide a deeper understanding of the methods used in the research and they extend the results provided in the case study.

Appendix A. Deep Learning

In this appendix, we give a description of each of the DL structures considered in the modeling framework. For a deeper understanding we refer to [30].

Appendix A.1. DNN

A DNN [30] is the natural extension of the traditional MLP. In particular, defining by $\mathbf{X} = [x_1, \dots, x_n]^\top \in \mathbb{R}^n$ the input of the network, by $\mathbf{Y} = [y_1, y_2, \dots, y_m]^\top \in \mathbb{R}^m$ the output of the network, by n_k the number of neurons of the k^{th} hidden layer, and by $\mathbf{z}_k = [z_{k1}, \dots, z_{kn_k}]^\top$ the state vector of the k^{th} hidden layer, a general DNN with two hidden layers can be represented by Figure A.7.

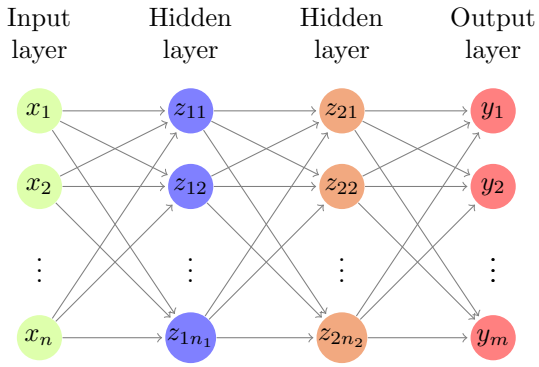

Figure A.7: Example of a deep neural network.

A general neuron i in the k^{th} layer can be represented by an equation of the following form:

$$z_{ki} = f_{ki}\Big(\mathbf{W}_{ki}^\top \cdot \mathbf{z}_{k-1} + b_{ki}\Big), \tag{A.1}$$

where f_{ki} represents the activation function of the neuron, e.g. a sigmoid function, $\mathbf{W}_{ki}$ the mapping weights from the $(k-1)^{\text{th}}$ layer to neuron i in the k^{th} layer, and b_{ki} the bias of the neuron. Note that in the above convention the input layer is considered as the 0^{th} layer.

Appendix A.2. RNN

While DNN structures are successful in many applications, they might fail to capture the nature of time series data. In particular, given a sequence of inputs $\mathbf{X}_1, \dots, \mathbf{X}_N$ that correspond to successive time steps, DNNs assume that an input $\mathbf{X}_k$ is independent from the others. Nevertheless, in time series data, inputs are usually correlated; therefore, to obtain a correct map of the output $\mathbf{Y}_k$, we might have to consider an input time sequence $\mathbf{X}_1, \dots, \mathbf{X}_k$ instead of a single input $\mathbf{X}_k$. To model this sequential dependence, RNNs [30] build additional mappings between the neurons of the same hidden layer to hold relevant information from past inputs. An example of a RNN is given in Figure A.8.

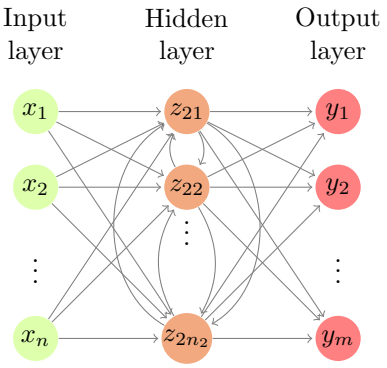

Figure A.8: Example of a recurrent neural network.

A great disadvantage of RNNs is that, while in theory they are able to model any sequential dependence, in practice they are incapable of learning long-term dependencies due to the vanishing gradient problem [94]. More specifically, given a training sequence $\{(\mathbf{X}_k, \mathbf{Y}_k)\}_{k=1}^N$, traditional RNNs have the problem that, due to the recurrence in the structure, the network gradient with respect to an input $(\mathbf{X}_k, \mathbf{Y}_k)$ depends on the multiplication of the gradients w.r.t. the previous inputs. As a consequence, as the length of the training sequence increases, the gradient contribution later training pairs either becomes 0 or grows unbounded. In the first case, only earlier inputs of the training sequence are effectively used, and thus, the training becomes slow and hard. In the second case, the training runs into numerical issues. In both scenarios, the end result are RNNs that are unable to hand long-term dependencies.

Appendix A.3. LSTM

LSTM networks [45] are a type of recurrent networks that avoid the vanishing gradient problem. Whereas in a standard RNN each neuron is represented by a simple neural unit, i.e. a single nonlinear mapping, an LSTM consists of a cell with four neural units. The key idea is that, by using four units per neuron, an LSTM is able to model a memory cell state c with a selective forget-remember behavior. In more detail, as depicted in Figure A.9, each LSTM cell consists of three gates: an input gate I, an output gate O, and a forget gate F. Together with a hyperbolic tangent function, these gates represent the four neural units. Then, at a time step t, the cell is characterized by its hidden state z_t, its cell state c_t and the input

state x_t; moreover, the output of the cell is represented by the hidden state z_t. In Figure A.9, the blue ellipses represent vectorial element-wise operations and the $\odot$ symbol represents the Hadamard or element-wise product.

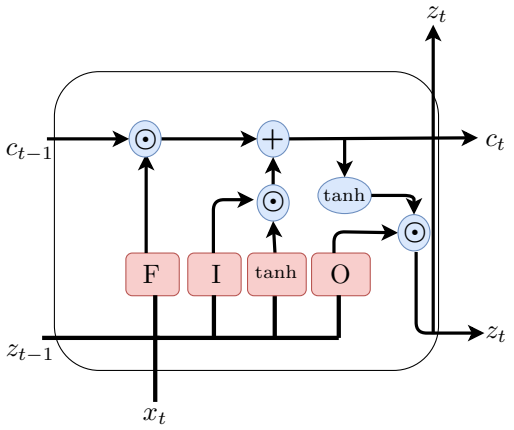

Figure A.9: Basic LSTM Cell. The $\odot$ symbol represents the Hadamard or element-wise product.

The working principle of the cell is as follows: at any time step t, the neuron regards z_{t-1} and x_t as decision variables. Based on them, it uses the neural units to build F_t, I_t and O_t, three vectors of real numbers between 0 and 1 that select which information from x_t, c_{t-1}, and z_{t-1} is used to build c_t and z_t. Defining the parameters of an LSTM cell as the matrices $W_F, W_I, W_O, W_c, b_F, b_I, b_O$ and b_c, the neuronal mapping consists of four steps:

1. The forget gate decides which information from the old cell state c_{t-1} is kept in c_t by building the decision vector
$$F_t = \sigma\Big(W_F \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_F\Big), \tag{A.2a}$$
with σ representing the sigmoid function.
2. Next, the input gate and the tanh unit select which new information is added. Particularly, the tanh unit creates a vector $\bar{c}_t$ with the relevant new information:
$$\bar{c}_t = \tanh\Big(W_c \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_c\Big). \tag{A.2b}$$
Then, the input gate builds the vector
$$I_t = \sigma\Big(W_I \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_I\Big), \tag{A.2c}$$
which selects which of the new information in $\bar{c}_t$ is kept in c_t:
3. Using F_t and I_t, the new cell state is built:
$$c_t = F_t \odot c_{t-1} + I_t \odot \bar{c}_t, \tag{A.2d}$$
4. Finally, the output gate builds the last decision vector
$$O_t = \sigma\Big(W_O \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_O\Big), \tag{A.2e}$$
which decides which information of c_t is used for the new hidden state z_t:
$$z_t = O_t \odot \tanh(\bar{c}_t). \tag{A.2f}$$

Appendix A.4. GRU

GRU networks [48] are RNNs that use a cell structure very similar to the LSTM case. However, in contrast with an LSTM cell, a GRU cell does not distinguish between the memory cell c and the hidden state z; instead, it uses a single state variable z. In addition, while an LSTM cell uses a three-gates structure, the GRU cell only requires two: an update gate U and a reset gate R. A representation of an GRU cell is given in Figure A.10; as before, the red boxes represent the three neural units, the blue ellipses vectorial element-wise operations, and the $\odot$ symbol represents element-wise product.

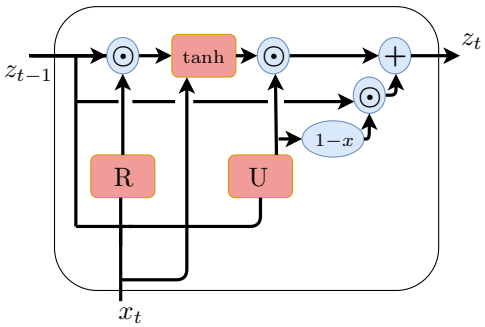

Figure A.10: Basic GRU Cell. The $\odot$ symbol represents the Hadamard or element-wise product.

The working principle of this cell resembles the one of an LSTM. Particularly, defining the parameters of an GRU cell as the matrices W_U, W_R, W_c, b_U, b_R and b_c, the neuronal mapping consists of three steps:

1. The reset gate builds the decision vector
$$R_t = \sigma\Big(W_R \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_R\Big), \tag{A.3a}$$
which selects which information from z_{t-1} is kept in the vector of new information $\bar{z}_t$.
2. Simultaneously, the tanh unit builds
$$\bar{z}_t = \tanh\Big(W_c \begin{bmatrix} R_t \odot z_{t-1} \\ x_t \end{bmatrix} + b_c\Big), \tag{A.3b}$$
which contains the relevant new information in x_t and in the reset gate selection $R_t \odot z_{t-1}$.
3. Finally, the update gate builds
$$U_t = \sigma\Big(W_U \begin{bmatrix} z_{t-1} \\ x_t \end{bmatrix} + b_U\Big), \tag{A.3c}$$
a decision vector that models the new state z_t as a trade-off between the old state z_{t-1} and the new relevant information $\bar{z}_t$:
$$z_t = U_t \odot \bar{z}_t + (1 - U_t) \odot z_{t-1}. \tag{A.3d}$$

Compared with an LSTM network, the GRU has a simpler structure and it is easier and faster to train. In addition, for some applications, it has been empirically shown that it can outperform the LSTM counterpart [48].

Appendix A.5. CNN

Another prominent family of DL structures are CNNs. The core idea of a CNN is to analyze an array of data by performing local operations in different areas of the array and outputting the result of these operations to a new layer. As a result, unlike other network structures, a CNN does not have all the layers fully connected.

CNNs are modeled using three building blocks: a convolution operation followed by a nonlinear function, a pooling operation, and a fully connected layer. Given an array of data, the convolution operation slides a filter across the data array and computes an element-wise cross product between the filter and the areas where the filter goes over. Then, for each of the convolved values, a nonlinear map is used and a new array of data is outputted. As different filters capture different properties, CNNs typically use different filters over the input data to output several data arrays. These output arrays are called feature maps, and each one of them represents a distinctive characteristic of the original data array.

As a second building block, the CNN performs a pooling operation. The basic idea is to reduce the size of the feature maps by reducing large areas into single values. Typical operations are the maximum pooling (maximum value of an area) or the average pooling (average value in the area).

Finally, after the CNN subsequently performs several convolutions and pooling operations, the values of the final set of feature maps are used as inputs for a fully connected layer. This layer can be used as the output of the network or be followed by more fully connected layers.

It is important to note that, while the above description considered a single input data array, a general CNN can have many; particularly, each of these input arrays is called a channel. A typical 3-channel example is an RGB image, where the blue, red and green layers represent the 3 input channels.

An example of a CNN structure is given in Figure A.11. In the example, the CNN considers three channels represented by 50×50 data arrays. Then, in a first layer, it computes 24 features maps using 8 different filters per channel. Next, it reduces the size of the maps to 11×11 arrays via a pooling operation. Then, it performs a second convolution and pooling operations that lead to 72 feature maps of size 6×6. Finally, the network becomes fully connected using a DNN with two hidden layers.

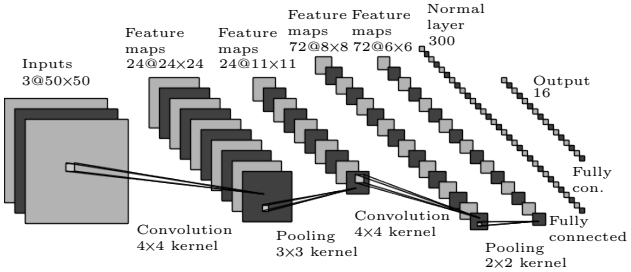

Figure A.11: Example of a CNN network.

It is important to note that, while most of the applications of CNNs have been with images, they have also been applied to other type of data, e.g. time series forecasting [95].

Appendix B. Optimal Feature Selection

In this appendix, we present the main results obtained during the feature selection process. In particular, we outline which of the input variables are helpful to predict the different day-ahead prices. The content of this analysis is qualitative, not quantitative; in particular, as we evaluate 27 models, each model predicts 24 hours, and as there are more than 750 individual input features available, the individual results would not only be very vast, but might not provide a very helpful insight.

The appendix consists of four parts: in a first section, we present the main feature selection results common to all forecasters. Then, in a second part, we list the results for the forecasters that require an individual model per hour, e.g. ARIMA or SVR. In a third part, we present the results of the benchmark forecasters with multiple outputs, e.g. neural networks. Finally, we will provide an overall discussion based on the listed results.

Appendix B.1. Common Results

Independently of the forecaster and the hour, using the day-ahead generation forecast in Belgium, i.e. g_{B}, decreases the overall accuracy. As discussed in [3], a possible explanation for this effect might be a change in the Belgian generation conditions in mid 2015, which would lead, in our study, to different generation conditions between the training and validation datasets and the test dataset.

Appendix B.2. Forecasters with an individual model per hour

Within the class of forecasters with exogenous inputs and an individual model per hour, we can distinguish between the nonparametric models, i.e. IHMARX and SNARX, and the rest.

In the case of nonparametric models, the optimal features are the same as in the original study: lagged prices at 24, 48, and 168 hours; the minimum price of the previous day; and the day-ahead forecast of the grid load at the prediction hour in the local market, i.e. Belgium. For the rest of the forecasters, we can make a distinction between three groups:

- For the DR, TF, ARX, and TARX models:
    1. Lagged prices in Belgium at 24, 25, 47, 48, 49, 167, 168, 169 hours.
    2. The day-ahead grid load forecast in Belgium at prediction hour.
    3. For ARX and TARX, the minimum price of the previous day.

- For the five machine learning methods, i.e. the three SVR-based models, the RF, and the XGB:
    1. 24 lagged prices of the previous day in Belgium.
    2. 24 lagged prices of one week before in Belgium.
    3. 24 lagged prices of the previous day in France.
    4. The day-ahead grid load forecast in Belgium at prediction hour.
    5. The day-ahead grid load and generation forecasts in France at prediction hour.

- For the three fARX-based forecasters, the optimal features are: the past prices in Belgium and France, the day-ahead load forecast in Belgium and France, and the day-ahead generation forecast in France. For each of these 5 variables, the specific lagged values are the same as in the original paper [67]: a very large combination of past prices at different lags, and the day-ahead forecasts at different future time steps and lags (as it is a total of 107 inputs, we refer to the original paper [67] for full details).

Appendix B.3. Results for forecasters with multiple outputs

The 6 forecasters that predict the 24 prices in a single model have in common, as optimal features, the 48 inputs represented by the day-ahead generation and load forecast in France. With respect to the rest of the features, there is a division into 2 groups:

- The RBF model, which has as optimal features the day-ahead load forecast in Belgium and considers only lagged prices in Belgium: 48 lagged prices representing the previous day and one week before.

- The DL models, which disregard as optimal features the day-ahead load forecast in Belgium, and consider the same lagged prices in France and Belgium:
    - MLP and DNN: the 72 lagged prices of the previous two days and one week before.
    - CNN: the 168 lagged prices of the week before the day of prediction.
    - LSTM and GRU: the 336 lagged prices of the two weeks before.

Appendix B.4. Discussion

If we look at the results, we can make the following observations:

1. The local prices in Belgium are the most important quantity. In particular, the lagged prices of the previous two days and one week before are the most important features.
2. For all statistical models, the load forecast for the prediction hour in the local market is also a very important feature.
3. For machine learning models, the features from the neighboring market, i.e. France, are also important. In particular, the lagged prices in France of the previous day and the load and generation forecasts for the prediction hour play an important role.
4. Except for the three fARX-based models, the effect of market integration, i.e. using features from neighboring markets, can only be observed in machine learning models.

Appendix C. Optimal Hyperparameters for Base Models

In this appendix, we describe the hyperparameters that are optimized for each base model and the result of this optimization for our case study. The hyperparameters and optimization results for the machine learning methods are listed in Table C.9; likewise, the optimization results for the statistical methods are listed in C.10. For a more detailed explanation of the meaning of the different hyperparameters, we refer to the original papers. In addition, for the explanation of the hyperparameters of the SVR-based models and the RF model, the library [83] used for the implementation is also a good reference.

Model	Symbol	Value	Definition
SVR	C	9.97	Penalty parameter of the error
	ϵ	0.0038	Epsilon of the epsilon-SVR model
SOM-SVR	C	1.57	Penalty parameter of the error
	ϵ	0.0029	Epsilon of the epsilon-SVR model
	n_{c}	3	Number of clusters
SVR-ARIMA	C	8.54	Penalty parameter of the error
	ϵ	0.0044	Epsilon of the epsilon-SVR model
	p	4	AR order of ARIMA part
	q	2	*Moving average* (MA) order of ARIMA part
	P	3	AR order of the daily seasonality of the ARIMA part
	Q	1	MA order of the daily seasonality of the ARIMA part
	d	1	Differencing order of ARIMA part
	D	0	Seasonal differencing order of ARIMA part
RF	n_{t}	470	Number of trees
	p_{f}	0.49	Percentage of features considered when looking for the best split
	n_{min}	1	Minimum number of samples per leaf node
XGB	n_{t}	105	Number of trees
	d_{max}	4	Maximum tree depth
	lr	0.0491	Learning rate
	γ	0.0071	Minimum loss reduction needed to make a new partition on a leaf node
	α	8.57	Coefficient for L1 regularization
	λ	0.4273	Coefficient for L2 regularization
	r_{sub}	0.7093	Subsample ratio of the training set used for training a tree
	r_{col}	0.3040	Subsample ratio of columns when training a tree
MLP	n	117	Number of neurons on the hidden layer
	nonlin	ReLU	Activation function on the hidden layer
	d	0	Dropout coefficient
	α	0.00032	Coefficient for L1 regularization
RBF	n	247	Number of neurons, a.k.a. kernels or basis functions.
	cluster	Birch	Clustering algorithm to find the centers for the kernels.

Table C.9: Summary of the optimized hyperparameter for the machine learning models (except the DL models).

fARX	N_{window}	40	Data window: number of past months used for estimating the model.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
fARX-Lasso	N_{window}	40	Data window: number of past months used for estimating the model.
	α	0.0040	Coefficient for L1 regularization.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
fARX-EN	N_{window}	39	Data window: number of past months used for estimating the model.
	α	0.0010	Coefficient for L1 regularization.
	r	0.95	Elastic net mixing parameter: $r = 1$ is equal to Lasso.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
IHMARX	N_{window}	32	Data window: number of past months used for estimating the model.
	N_{re}	20	Number of model re-estimations when optimizing the nonparametric model.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
SNARX	N_{window}	43	Data window: number of past months used for estimating the model.
	N_{re}	17	Number of model re-estimations when optimizing the nonparametric model.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
ARX	N_{window}	All	Data window: number of past months used for estimating the model.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
TARX	N_{window}	All	Data window: number of past months used for estimating the model.
	N_{mod}	1	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.
DR	N_{window}	36	Data window: number of past months used for estimating the model.
	$\mathbf{l}_{\text{max}}$	168	Largest lag for the demand.
	$\mathbf{p}_{\text{max}}$	192	Largest lag for the prices.
TF	N_{window}	36	Data window: number of past months used for estimating the model.
	$\mathbf{l}_{\text{max}}$	168	Largest lag for the demand.
	$\mathbf{p}_{\text{max}}$	192	Largest lag for the prices.
	$\mathbf{d}_{\text{max}}$	168	Largest lag for the disturbance term.
WARIMA	N_{window}	16	Data window: number of past months used for estimating the model.
WARIMA-RBF	N_{window}	23	Data window: number of past months used for estimating the model.
	N_{swarm}	350	Swarm size.
	N_{max}	150	Maximum number of iterations for particle swarm optimization.
	ω	0.65	Particle velocity scaling factor.
	ϕ_{p}	0.4	Scaling factor to search away from particle's best known position.
	ϕ_{g}	0.7	Scaling factor to search away from swarm's best known position.
ARIMA-GARCH	N_{window}	12	Data window: number of past months used for estimating the model.
DSHW	N_{window}	All	Data window: number of past months used for estimating the model.
TBATS	N_{window}	16	Data window: number of past months used for estimating the model.
DSARIMA	N_{window}	19	Data window: number of past months used for estimating the model.
AR	N_{window}	All	Data window: number of past months used for estimating the model.
	N_{mod}	24	$N_{\text{mod}}=1$: one model to predict all 24 hours. $N_{\text{mod}}=24$: individual model per hour.

Table C.10: Summary of the optimized hyperparameter for the statistical methods.