POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Numerical Analysis for Machine Learning Project

Author(s): **Martina Riva - 10756775**

Group Number: **72**

Academic Year: 2023-2024

Contents

1 | Introduction

The main goal of this project is to replicate and evaluate the findings from the paper: *"Forecasting Spot Electricity Prices: Deep Learning Approaches and Empirical Comparison of Traditional Algorithms"*. The paper explores whether deep learning algorithms can achieve better predictive accuracy in electricity price forecasting. Additionally, it provides a comparative analysis of a wide range of forecasting methods proposed in the literature, aiming to highlight the superiority of deep learning models.

The **motivation behind the paper** is the increasing integration of renewable energy sources (RES) into the electricity market. This has led to greater price volatility and more frequent price spikes in electricity markets. As a result, accurate electricity price forecasting has become crucial, as it can help mitigate the negative impacts of price uncertainty.

The **literature** highlights that both, statistical and machine learning methods, are effective for forecasting electricity prices. However, machine learning methods tend to outperform statistical methods, especially for high-frequency data, where linear statistical methods may struggle due to their limitations.

The **findings of the paper** demonstrate that:

- Deep learning models outperform traditional statistical methods and machine learning algorithms, in terms of predictive accuracy;
- Machine learning methods typically perform better than statistical models;
- Hybrid models, combining different techniques, do not necessarily outperform their simpler counterparts.

In this project, I will replicate these findings, with a focus on **Deep Learning models**, by implementing the following:

- A **Deep Neural Network** (DNN) as an extension of traditional MLPs.
- A **Hybrid Long Short-Term Memory** (LSTM)-DNN hybrid model.
- A **Hybrid Gated Recurrent Unit** (GRU)-DNN hybrid model.
- A **Convolutional Neural Network** (CNN) model.

Additionally, I will implement some benchmark models:

- **Multi-Layer Perceptron** (MLP) and **Support Vector Regression** (SVR) as examples of machine learning models;

- The **fARX** model as a statistical approach for electricity price forecasting.

All the models developed in this project aim to predict the 24 hourly electricity prices for the next day, based on information available before that day begins. This approach is known as day-ahead forecasting. An effective forecasting model should accurately predict these 24 hourly prices using the information available before the start of the day.

2 | Dataset

The dataset used in this project differs slightly from that employed by the original paper's authors due to the unavailability of their data. The dataset considers the day-ahead market in Belgium, and includes historical electricity prices along with two relevant exogenous inputs:

- **Date**: timestamp including date and hour;
- **Prices**: electricity price at the given hour;
- **Generation forecast**: day-ahead forecast of the available generation in the EPEX-Belgium.
- **System load forecast**: day-ahead forecast of the grid load in the EPEX-Belgium.

An example of the dataset is shown in the Figure 2.1.

Date	Prices	Generation forecast	System load forecast
2011-01-09 00:00:00	32.54	63065	63000
2011-01-09 01:00:00	21.55	62715	58800
2011-01-09 02:00:00	15.71	61952	58500
2011-01-09 03:00:00	10.58	59262	54300

Figure 2.1: Dataset

2.1. Data division

The data are divided into three sets:

1. **Training set (09/01/2011–31/03/2015)**: These data are used for training and estimating the different models.
2. **Validation set (01/04/2015–14/02/2016)**: These data are used for selecting the optimal hyperparameters.
3. **Test set (15/02/2016–31/12/2016)**: These data are employed as the out-of-sample data to compare the models.

This division results in 37032 samples for the training set, 7680 for the validation set, and 7704 for the test set.

3 DL models

The main goal of this project is to propose a DL modeling framework as a forecasting tool for day-ahead electricity prices.

3.1. Library

To implement the proposed deep learning framework, I use the *Keras* DL library.

3.2. Data pre-processing

For the DL models, I handle the **outliers**, by replacing them with the median of the column, to improve accuracy.

The data are **normalized to the intervals [-1,1]** with the MinMaxScaler. This transformation is applied because these pre-processing steps help achieve more accurate models. It's important to note that this transformation is only applied during model estimation and not when computing performance metrics.

3.3. Techniques for training

All the neural networks are trained by minimizing the **Mean Absolute Error (MAE)**:

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

where y_i is the true value, $\hat{y}_i$ is the predicted value, and n is the total number of samples. The choice of using MAE is due to the fact that the traditional root mean square error (Euclidean norm) would put too much importance on the spiky prices.

For training the model, all neural network models implement **Early Stopping**, a technique that monitors performance on the validation set during training. If the model's performance stops improving on the validation set, early stopping halts the training process (if the validation accuracy does not improve for ten consecutive epochs). This prevents the model from overfitting, ensuring better generalization to new data.

Together with early stopping, the DL models also **Adam**, a **stochastic gradient descent method** that uses adaptive learning rates. The advantage of using this optimization method is that the learning rate does not need to be tuned online.

Additionally, the "ReduceLROnPlateau" technique is used in LSTM-DNN, GRU-DNN, CNN to reduce the learning rate of a neural network during training, further improving the convergence of the model.

In conclusion the data are **shuffled** during training.

3.4. Deep Neural Network (DNN) model

The first model for predicting day-ahead prices is a Deep Neural Network (DNN), which extends traditional Multi-Layer Perceptrons (MLPs) with two hidden layers.

The **input features** considered for the model are the following:

1. **Day of the Week:** the day of the week of day D, represented as an integer (0 for Monday, 1 for Tuesday, and so on up to 6 for Sunday).
2. **Price of the Previous Day ($D-1$):** the electricity price for each hour of the previous day ($D-1$), resulting in 24 features (one for each hour of the day).
3. **Price of the Two Previous Days ($D-2$):** the electricity price for each hour of the two days before day D, resulting in 24 features.
4. **Price of the Previous Week ($D-7$):** the electricity price for each hour of the same day of the previous week ($D-7$), resulting in 24 features.
5. **Generation Forecast for Day D:** the forecasted generation for each hour of day D, resulting in 24 features.
6. **Load Forecast for Day D:** the forecasted load for each hour of day D, resulting in 24 features.

Thus, the input X has 121 features, where each row represents a single day. The output Y is the vector of day-ahead prices that we intend to forecast.

The network architecture includes **two hidden layers**:

- The first hidden layer consists of 239 neurons.
- The second hidden layer consists of 162 neurons.

Both hidden layers use the ReLU activation function.

The model can be seen in Figure 3.1.

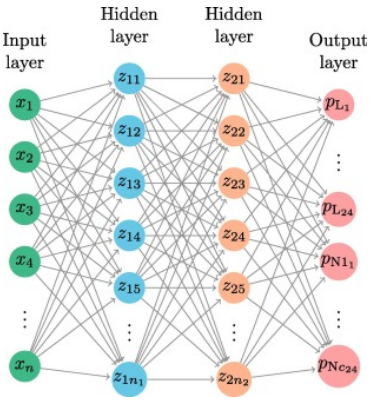

Figure 3.1: DNN model

3.5. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are designed to model time series data by maintaining information from past inputs. Advanced RNNs like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are used.

3.5.1. LSTM-DNN hybrid model

The second deep learning model for predicting day-ahead prices is a hybrid forecaster that combines **Long Short-Term Memory (LSTM)** and **Deep Neural Network (DNN)** architectures.

- **LSTM** networks are designed to capture and model sequential data.

  This model handles **Sequential Inputs** $\mathbf{X}_S$: a collection of input vectors, each one representing past day-ahead prices (with a timestep of 2 weeks).

  The LSTM network has a hidden layer with 83 neurons and uses **hyperbolic tangent** (**tanh**) activation function.

- **DNN** handles non-sequential data.

  The input vector is $\mathbf{X}_F$ representing **Future Data**. It has 49 features: day of the week, and the generation and load forecast of the day we want to predict (24h).

  The DNN has a hidden layer with 184 neurons and uses **ReLU** activation function.

Then, the output of these two networks is concatenated into one vector $\mathbf{Y}$ of day-ahead prices that we intend to forecast.

The model can be seen in Figure 3.2.

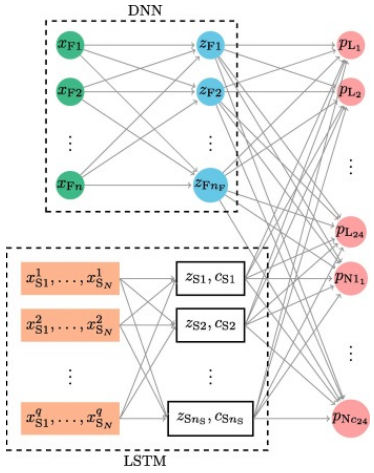

Figure 3.2: LSTM-DNN hybrid model

3.5.2. GRU-DNN hybrid model

The third DL model for predicting day-ahead prices is a hybrid forecaster that combines **Gated Recurrent Unit (GRU)** and **Deep Neural Network (DNN)** architectures. It's similar to the LSTM-DNN hybrid structure, but it uses a GRU instead of LSTM to model the time data sequences. The sequence length considered in this case is 3 weeks.

In this model:

- **GRU** has a hidden layer with 132 neurons and uses (hyperbolic tangent (tanh)) activation function.
- **DNN** has a hidden layer with 166 neurons and uses **ReLU** activation function.

Additionally, a Dropout rate of 0.32 is applied. This means that 32% of the neurons are randomly set to zero during each training iteration, which helps to prevent overfitting.

3.6. CNN model

The fourth deep learning model for predicting day-ahead prices is a Convolutional Neural Network (CNN). This model divides the data into two distinct parts:

- **Future Data $\mathbf{X}_F$**: includes information about the day-ahead, such as the day of the week, load forecasts, and generation forecasts.
- **Sequential Data $\mathbf{X}_S$**: consists of past input sequences representing historical prices (1 week).

The model uses **two Parallel CNNs**:

- **Day-ahead CNN**: processes the future data $\mathbf{X}_F$ as input channels;
- **Past CNN**: processes the sequential data $\mathbf{X}_S$.

Each CNN performs the following operations:

- **Convolution** to extract features from data arrays using sliding filters.
- **Max Pooling** to reduce data size.
- **Batch Normalization** normalizes the inputs of each layer, to improve training stability and performance.

After these operations, the feature maps from both CNNs are merged and fed into a fully connected layer to predict the day-ahead prices.

Each CNN uses **ReLU** activation function and applied a Dropout of 0.31.

4 | Benchmark Models for Electricity Price Forecasting

In addition to the deep learning models, several other forecasting techniques have been implemented. For comparison purposes, I have selected a statistical method and two machine learning methods that are particularly significant:

- Statistical Method: fARX Model
- Machine Learning Methods: MLP and SVR

These methods are included to offer a broad comparison and evaluate the performance of deep learning models against traditional approaches.

4.1. Libraries

I used the "$scikit-learn$" library for fARX and SVR models.

The MLP is modeled using the same frameworks as the other DL models.

4.2. Data processing

To simplify the time series and make them easier to forecast, the data used for the statistical models are processed using a **Box-Cox transformation**.

For the machine learning, the data are **normalized to the intervals [0,1]**, using the MinMaxScaler.

4.3. fARX model

Flexible Auto-Regressive with Exogenous Variables (fARX) model is used to capture both past values of the time series (e.g., electricity prices in Belgium for days D-1, D-2, D-3 and D-7)) and external variable (such as day-ahead load forecasts). This provides a statistical approach to forecasting. The model uses 120 input features, and a linear regressor model is built for each hour of the day.

4.4. Artificial intelligence models

This category includes machine learning algorithms, which incorporate exogenous inputs and can capture more complex relationships, such as non linear one, compared to statistical models.

4.4.1. MLP model

The **Multilayer Perceptron** (MLP) is a standard neural network with a single hidden layer (containing 117 neurons). It's a simpler version of deep learning algorithms. The input features are the same as the DNN model: the day of the week, electricity prices of previous day, two days before, and a week before, as well as the generation and load forecast for day D.

4.4.2. SVR model

Support Vector Regressor (SVR) performs a nonlinear mapping of the data into a higher-dimensional space, where linear functions are used to perform regression. The input has the following features: 24 lagged prices from the previous day; 24 lagged prices from one week before; the day-ahead grid load forecast for Belgium at the prediction hour. A SVR model is build for each hour of the day.

5 | Results and Observations

5.1. Loss function: MAE

Each Deep Learning model was trained using **Mean Absolute Error (MAE)** as the loss function. Below, I present the loss curves for each model, comparing the training and validation loss during the training process. The loss curves help illustrate how the model's performance evolves and when training is stopped.

An important aspect of the training process is the use of **Early Stopping**. This technique monitors the *validation loss* during training and halts the training if the validation loss stops improving for 10 consecutive epochs. By doing so, we avoid overfitting the model to the training data. In the graphs below, we can observe how early stopping prevents excessive training and ensures optimal generalization performance.

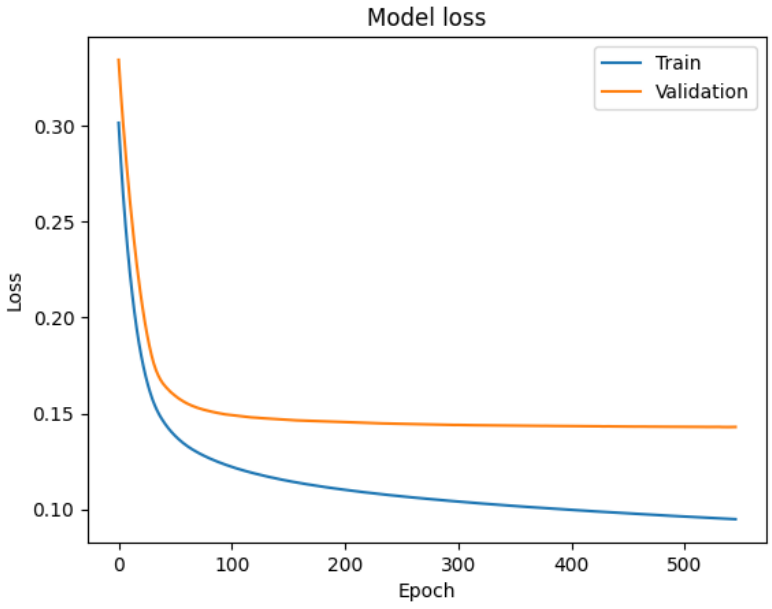

Figure 5.1: Training and Validation Loss Curves for DNN model

The DNN model stopped after 546 epochs with a training loss of 0.0948 and a validation loss of 0.1429. This suggests that the model was able to fit well to the training data, although there is some difference between the training and validation losses, indicating slight overfitting.

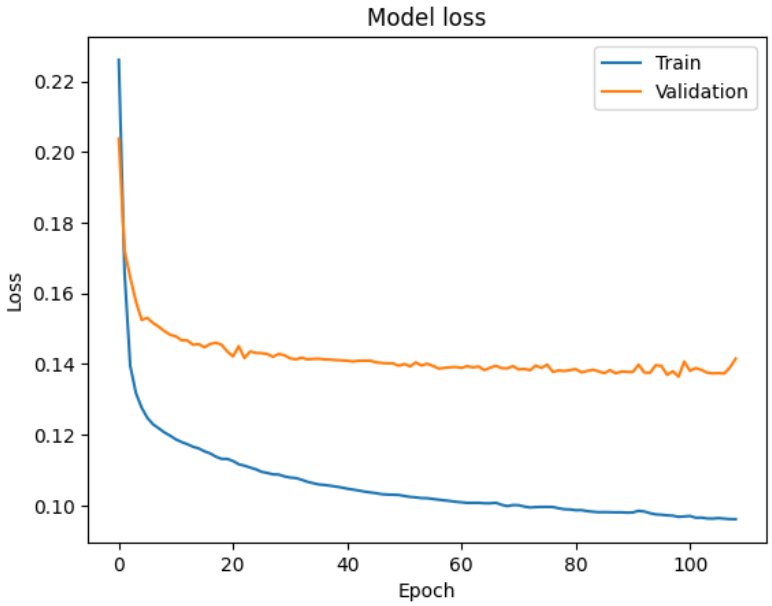

Figure 5.2: Training and Validation Loss Curves for LSTM-DNN model

The LSTM-DNN model stopped at epoch 109 with a training loss of 0.0949 and a validation loss of 0.1415. The slight difference between training and validation losses suggests minimal overfitting.

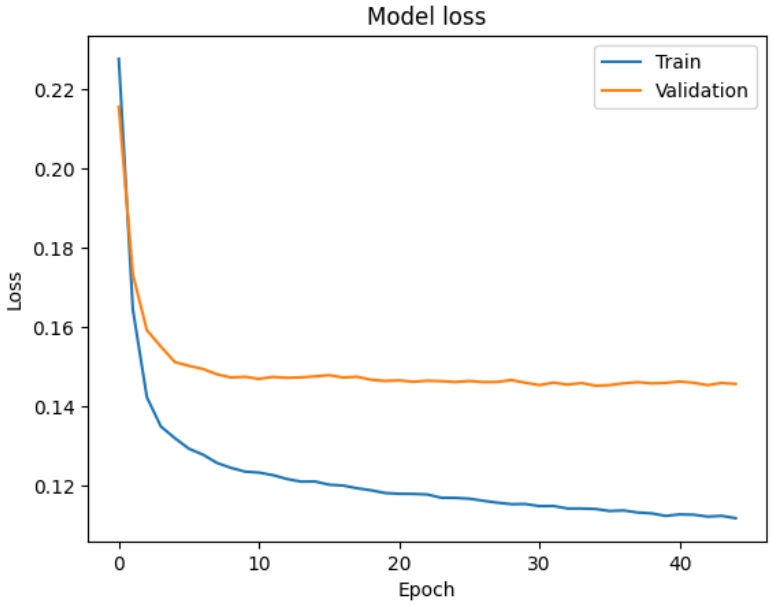

Figure 5.3: Training and Validation Loss Curves for GRU-DNN model

The GRU-DNN model stopped at epoch 45 with a training loss of 0.1126 and a validation loss of 0.1457. The relatively close values of training and validation losses suggest that the model performed reasonably well on both datasets.

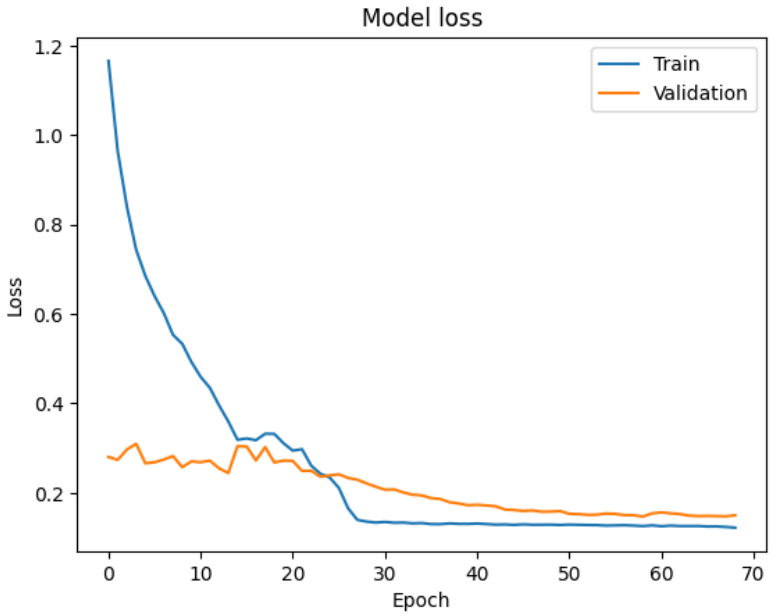

Figure 5.4: Training and Validation Loss Curves for CNN model

The CNN model stopped at epoch 69 with a training loss of 0.1201 and a validation loss of 0.1496.The validation loss shows a trend of increasing or instability, that might suggest that the model is starting to overfit to the training data.

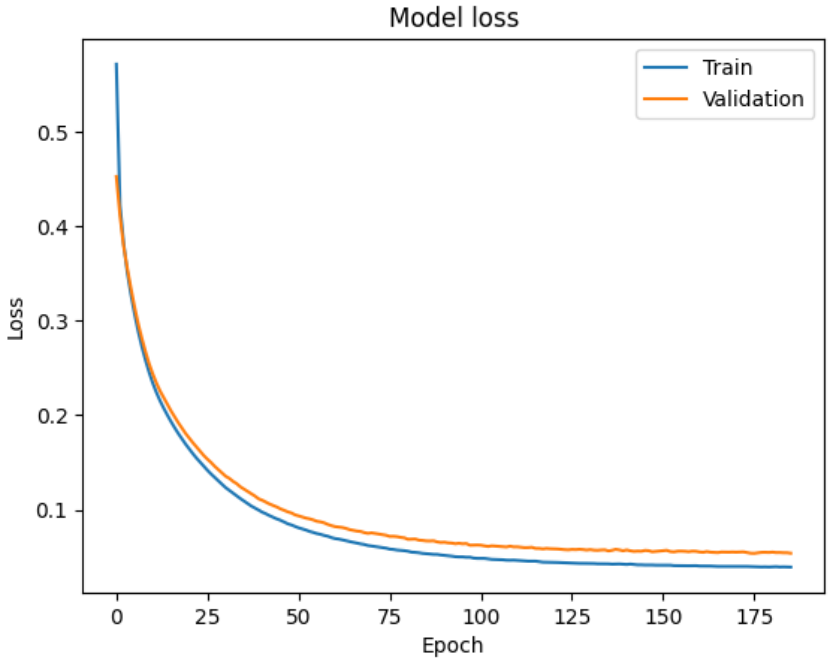

Figure 5.5: Training and Validation Loss Curves for MLP model

The MLP model completed 186 epochs and achieved the lowest training loss of 0.0392 and

validation loss of 0.0541, demonstrating strong performance in comparison to the other models. The small difference between training and validation losses indicates that the model has effectively learned from the data with minimal overfitting.

5.2. Performance Metric: sMAPE

To evaluate and compare forecasting accuracy, the **symmetric mean absolute percentage error** (sMAPE) is used.

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

where:

- $[\hat{y}_1, ..., \hat{y}_N]^T$ are the predicted values;
- $[y_1, ..., y_N]^T$ are the real outputs;
- N is the number of data points.

5.2.1. sMAPE on Test Set

The sMAPE values computed on test set are shown in the following table.

	Model	**sMAPE**
1	DNN	16.28%
2	MLP	18.48%
3	LSTM-DNN	18.59%
4	GRU-DNN	19.73%
5	CNN	21.06%
6	SVR	22.02%
7	fARX	57.01%

Table 5.1: sMAPE Values on Test Set

These results differ from those reported in the paper, potentially due to using a slightly different dataset with fewer samples. Notably, the difference between models is more pronounced in my results. The higher sMAPE values observed in my results compared to the paper highlight the impact of dataset characteristics on model performance.

5.2.2. Key Observations

- **DNN vs MLP**: The first consideration is that although the MLP had the lowest loss during training and validation, it doesn't necessarily translate into better forecasting performance on the test set. The DNN model, with slightly higher training and validation losses (MAE), achieved a better sMAPE score on the test set, meaning it

performed better predicting the actual patterns and trends in the unseen test data. This highlights the importance of considering performance metrics like sMAPE, which better reflect model generalization to new data.

- **MLP vs Hybrid models**: MLP outperformed the two hybrid models (LSTM-DNN and GRU-DNN), in this case. This could be attributed to having fewer input features compared to the paper, and to the complexity added by combining LSTM/GRU with DNN in a smaller dataset. So making a simpler method, like MLP, may be more effective when working with smaller datasets or limited features.
- **Statistical methods**:As expected, the fARX model performed significantly worse than the other models. This aligns with the general trend observed in the paper, where statistical methods tend to underperform compared to more sophisticated models.
- **CNN**: Among the deep learning models, CNN performed the worst, possibly due to its complex architecture, which may not have been well-suited for the dataset used in my analysis.

5.3. Conclusions

In conclusion, the DNN model demonstrates the best performance for predicting day-ahead electricity prices based on the test set, achieving the lowest sMAPE value among all the models analyzed.

The significant discrepancy between my results and those reported in the paper underscores the considerable impact of dataset characteristics on model performance. This suggests that further experimentation with parameters such as the number of epochs, learning rates, and batch sizes could potentially enhance the model's quality and lead to improved results. Additionally, choosing a better architecture, tailored to the specific features of my dataset, could also contribute to further improvements in performance.