Deep Neural Networks

Data Science and Information Technologies

Fall 2019-20

Semester Project:

Evaluation of deep learning architectures for short-term price forecasting in the Spanish electricity market

HELLENIC REPUBLIC

National and Kapodistrian University of Athens

Dimitrios Roussis (ID: DS1.19.0017)

E-mail: droussis@outlook.com.gr

Supervisors:

Dr. Haris Papageorgiou

Dr. Athanasia Kolovou

Table of Contents:

1. Introduction

Electricity has been one of the most important factors in the progress of our society and it now constitutes a basic human need. In the recent decades however, the power industry of many European countries has been deregulated and thus, a new market has emerged which is based solely on the electricity price. In Spain, this change was implemented when the Electric Power Act 54/1997 was enacted and since then, all of the stakeholders have been exposed to high levels of uncertainty, due to the inherent difficulty of predicting the price of a commodity which cannot be stored in large quantities efficiently and which is affected by countless factors [1]. With the emergence of this new market, the need for reliable forecasting methods at all scales (i.e. hourly, daily, monthly, etc.) has also emerged and has become a large area of research. Among the many methods and models that have been applied and evaluated for the particular task of short-term electricity price forecasting, some of those that stand out are the auto regressive integrated moving average (ARIMA) models and their variants, the exponential smoothing methods, the support vector machines (SVM) regression models, and of course, the artificial neural networks (ANNs) [2]. In this project, we will only be concerned with the latter and in particular, with convolutional neural networks (CNNs), long short-term memory networks (LSTMs), multilayer perceptrons (MLPs) and some of their combinations.

1.1. Problem Statement

The aim of this project is to implement different deep learning architectures and evaluate them based on their performance on the hour-ahead electricity price prediction task. More specifically, we will evaluate (i) LSTM, (ii) stacked LSTM, (iii) CNN, (iv) hybrid CNN-LSTM and (v) MLP architectures, using the *root mean squared error* (RMSE), one of the most widely used metrics along with the *mean absolute percentage error* (MAPE) for point forecasts [2]. Furthermore, we will experiment on different task formulations and types of frameworks, alongside the two following dimensions:

- We will compare the performance of *univariate time series forecasting* and *multivariate time series forecasting*. Univariate time series forecasting is a framework on which the predicted quantity (i.e. electricity price) is the sole feature that is used by the models, whereas the multivariate variant of the task also uses other features which may prove important for the prediction, such as the load of the energy grid, the temperature, etc.
- We will compare the performance of using different time-steps (3, 10 and 25 time-lags) as a way of reframing the time-series prediction task into a supervised learning problem, i.e. using the past 3, 10 and 25 values of the features which are fed into our models.

1.2. Case Study

In order to compare the deep learning models in the way that we mentioned above, we decide to use two datasets which contain 4 years (from 01/01/2015 up to 31/12/2018) of information about the electrical consumption, generation, pricing and temperature in Spain. In particular, the two datasets that we used are: (and which can found on Kaggle in the following link: https://www.kaggle.com/nicholasjhana/energy-consumption-generation-prices-and-weather)

- 'weather_features.csv': This dataset contains hourly data (178,396 rows with 17 columns; 35,064 rows for each city) about the weather conditions (e.g. temperature, wind speed, humidity, rainfall, etc.) of 5 major cities of Spain. The uploader of the dataset claims that this dataset has been purchased from the Open Weather API.
- 'energy_dataset.csv': This dataset contains hourly data (35,064 rows with 29 columns) about the electricity price (in €/MWh), the total electricity load of the national grid (the total amount of the energy demand in MW) and the amount of electricity generated (in MW) by various sources such as fossil gas, wind energy, nuclear energy, etc. This dataset has been retrieved by the ENTSO-E Transparency Platform and the Red Electrica de España.

The weather information contained in the first dataset concerns the cities of Madrid, Barcelona, Valencia, Seville and Bilbao which comprise approximately 1/3rd of the total population of Spain and which are geographically distributed across most of the territory of Spain in a uniform manner, as we can see in Fig. 1 below (the cities are highlighted by a red star on the map).

Fig. 1: Map of the country of Spain with the 5 major cities for which weather information is available

2. Data Analysis and Preparation

In this part of the project, we present the most important steps that we did in order to: (i) analyze and preprocess the data, (ii) gain various insights, (iii) detect and remove the outliers, (iv) fill the missing data, (v) delete the features that we deemed to be unreliable or unusable, (vi) merge the two datasets into one, (vii) conduct statistical tests on the predicted quantity, (viii) generate new features manually, (ix) select the most important features for the multivariate forecasts, (x) normalize the variables and (xi) convert the time-series into a supervised learning problem.

2.1. Cleaning and Preprocessing

Many challenges that we faced in this project had to do with preprocessing and cleaning the two datasets, especially before merging them into a single, final dataset.

After a brief statistical analysis of the *energy dataset*, we removed quite a large number of features which were constituted solely by zero-valued observations (or NaNs, i.e. 'Not Any Number') and we parsed the dates correctly, by setting them as the index for each row of the observations. The energy dataset also contained day-ahead forecasts (made by Red Electrica de España) for the total load of the energy grid, the amount of electricity that will be generated by wind and solar energy, as well as a day-ahead forecast for the electricity price, the variable that we are most interested in. However, the forecast proved to have an RMSE of 12.332, which is a lot worse than a simple "naive" forecast (see section 3.1.1. for a more detailed explanation) and thus, was removed from the dataset. Finally, with the utilization of visualizations, we were able to spot the rows on which empty values (NaNs) appeared and we filled them using linear interpolation, the simplest form of interpolation which joins two data points with a straight line [3].

As regards the *weather features dataset*, we also had to parse the dates correctly and convert the types of many of the features (i.e. columns), but some problems and outliers were noticeable from the beginning of our analysis. We will not go into further details about it and the interested reader is referred to the source code of the project as some data in the weather features had been -probably- converted into the wrong units or had obvious outliers; a notable example is the column with information about the observed pressure (in hPa) which had a maximum of 1,008,371 hPa, which is roughly the pressure at the bottom of Mariana Trench, 11 km below ocean surface. Moreover, we grouped the weather features by each of the 5 major cities, detected a large number of duplicates and removed the features/columns on which there were differences between the duplicate values for the same particular observation on a given hour. Thus, all categorical features which contained qualitative information about the weather were not used in our analysis. The final step in preprocessing this particular dataset was to employ visualization techniques (box plots) and manually remove some of the outliers, which we later filled using linear interpolation.

2.2. Analysis of the Electricity Price

After merging the two datasets into one, we decided to further analyze the time-series of the electricity price. The values of the price seem to follow -roughly- a normal distribution and the behavior of the time-series reveals many patterns at different scales:

- From year to year, there are seasonal patterns at the *monthly scale*, meaning that certain "spikes" in the time-series appear in approximately the same months.
- From week to week, there are periodic patterns at the *daily scale*. In particular, the electricity price tends to be higher from Monday to Friday and lower at the weekends and especially on Sundays, as most businesses are closed.
- From day to day, there are periodic patterns at the *hourly scale*. More specifically, the electricity price tends to be higher during the day and lower during the night. What is more, a certain pattern manifests during business hours, as the electricity price drops for a few hours. The last pattern is probably due to the "siesta", the traditional lunch break between 01:30PM and 04:30PM, as many types of businesses in Spain do not follow the typical 9AM-5PM working day.

Many time-series exhibit long-term trends, i.e. systematic time-dependent structures, which cannot be detected by an artificial neural network as it takes sequences of a specific number of time-steps as input. A common transformation which alleviates this problem is *differencing*, i.e. the subtraction of the previous value from each value in the series (i.e. first-order differencing, which can be followed by the application of higher order differencing). However, such a transformation would add unneeded complexity into our framework and may actually be unnecessary. Thus, we decided to conduct two relevant statistical tests, the *Augmented Dickey - Fuller* (ADF) test and the *Kwiatkowski - Phillips - Schmidt - Shin* (KPSS) test:

The ADF test is a type of unit root test, which means that it determines how strongly a time-series is determined by a trend. Its hypotheses are the following:

- *Null Hypothesis* H_o: There is a unit root in the series, i.e. the series is autocorrelated with a time dependent structure.
- *Alternate Hypothesis* H_1: There is not a unit root in the series and thus, the time-series is either stationary or can be made stationary using first-order differencing.

The KPSS test follows the reverse logic and checks for stationarity, i.e. whether the statistical properties of the time-series do not change over time. Its hypotheses are the following:

- *Null Hypothesis* H_o: The time-series is level, i.e. it is stationary around a constant.
- *Alternate Hypothesis* H_1: There is a unit root in the time-series and thus it is not stationary.

Test	Statistic	p-value	Critical value (1%)	Null Hypothesis
ADF	-9.147	<< 0.001	-3.431	Rejected
KPSS	7.957	0.01	0.739	Cannot be rejected

Table 1: Results of the ADF and KPSS tests for stationarity

From the results of the tests above, we can conclude that the time-series is stationary or could be made stationary with first-order differencing. Nonetheless, we decided to leave the time-series in its original form, as we judged that the added complexity would not be of any significance.

Since the models require sequences of values as inputs and the quantity that we want to predict as output, we had to convert the series and reframe its prediction as a supervised learning problem. More concretely, for each given observation in the time-series of the electricity price, we take the previous time-steps, (lag observations: t-1, t-2, etc.) as inputs and use the said observation (t) as the output variable. As we briefly mentioned in section 1.1, we will experiment on different configurations, by taking the previous 3, 10 and 25 observations of the electricity price as inputs for both the univariate and the multivariate frameworks. For the multivariate framework, we will do the same for the time-series of every feature that we will take into consideration and actually use in our models. It should be noted though, that he choice of the number of time-steps on which we experimented was mostly based on the partial autocorrelation plot of the electricity price time-series, which shows that the direct relationship between an observation at a given hour (t) is strongest with the two next time lags (t+1, t+2) and essentially diminishes after 25 time lags (t+25).

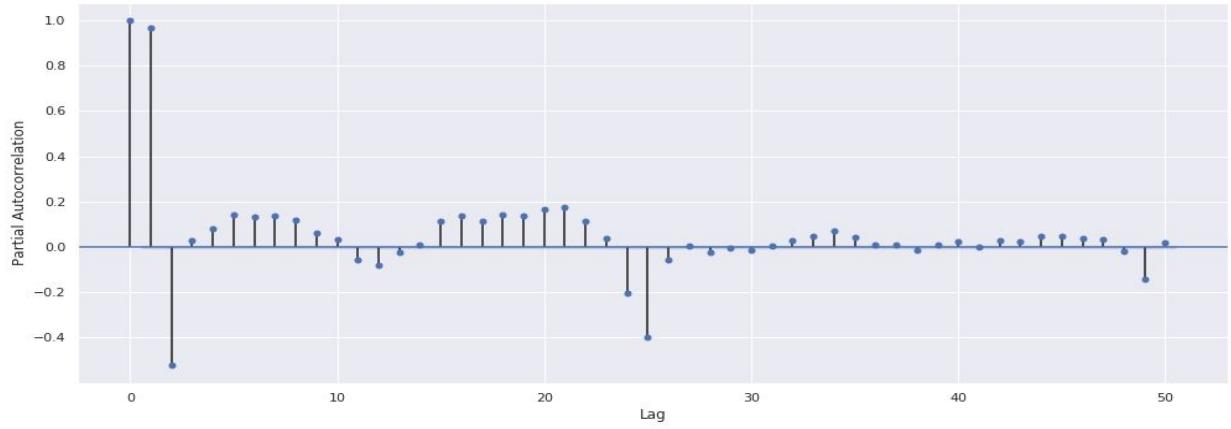

Fig. 2: Partial Autocorrelation function plot of the hourly electricity price

2.3. Feature Engineering and Outlier Removal

In univariate time-series forecasting, the only features that are given as input to our models are the previous values of the output variable. In multivariate time-series forecasting, however, we are able to give lagged observations (i.e. sequences of previous observations) from a plethora of different features which can possibly improve the prediction performance. The use of a disproportionately large number of features compared to the size of the data gives rise to a well-known phenomenon, widely known as the *"curse of dimensionality"*, in which the data become sparse within the feature space. This could create numerous problems in our approach, as the way that we have reframed the problem requires that we use 25 lagged observations of every feature; then the use of 60 of the original features, for example, would translate into 1500 input variables given to the model for the prediction of just a single value.

At the current stage of our analysis, we had actually ended up with more than 60 features. In spite of this large number of features, we need to *generate* additional ones which will help our model capture hourly, weekly, daily and seasonal patterns (see section 2.2.). Additionally, after an analysis of the Pearson correlations between the individual features, we observed that there are strong relationships between some of them; especially between the maximum and minimum temperatures of a certain city and generally among the temperatures of each city with other cities. Hence, we decided to generate the following additional features:

- "*Hour*", "*Weekday*" and "*Month*" features, which simply provide time information.
- A "*Business Hour*" flag, which is aimed at differentiating between hours on which all businesses are open, the hours on which the "siesta" (the traditional lunch break at 1:30PM to 4:30PM) takes place and all the other hours.
- A "*Weekend*" feature, which is aimed at differentiating weekends from all the other days and which assigns a larger weight on Sundays compared to Saturdays.
- The "*Temperature Range*" feature for each city, which is the absolute difference between the maximum temperature and the minimum temperature in a certain city at a given hour.
- The "*Weighted Temperature*" feature, which is a summation of the temperature in each of the five major cities for a given hour, multiplied by an appropriate weight that takes into account the population of each one compared to the total population of all five.
- The "*Generation Coal All*" feature, which aggregates the electricity generation by both energy sources related to coal, hard coal and brown coal/lignite, as they showed a strong correlation as well.

We now have a pool of even more features than before and need a way to *select* which ones to actually use in the multivariate variants of the prediction models.

In many cases, it is a better idea to apply a feature selection method on the lagged features, i.e. not to just estimate the importance of temperature at time (t) for the prediction of the electricity price at time (t), but to also estimate the importance of each specific lagged temperature (t-1, t-2, etc.). Nonetheless, since we have already made the choice to use specific time-steps for all the relevant features (including the electricity price), we follow a much simpler approach. To be more specific, before converting the problem into a supervised learning problem (see section 2.1.), we calculate (i) the *Pearson correlation* of each input variable with the electricity price and we also make use of (ii) *XGBoost* (a popular gradient boosting framework) to estimate the importance of each individual feature using the *F-score* (how many times each feature is split on by the gradient boosted trees of a depth equal to 2 in 200 rounds) using only the training set (see section 3.1.). In the table below, we present the features with an absolute Pearson correlation greater than 0.20 and those with an F-score larger than 15; as well as the 11 which were finally selected:

Feature	Correlation	F-score	Selected
Month	0.2228	112	**YES**
Hour	0.2482	59	**YES**
Weekday	0.2416	35	**YES**
Business Hour	0.2869	26	**YES**
Total Load Actual	0.4506	23	**YES**
Generation Coal All	0.5361	36	**YES**
Generation Fossil Hard Coal	0.5368	26	NO (duplicate)
Generation Fossil Brown Coal/Lignite	0.4325	< 15	NO
Generation Fossil Gas	0.4590	22	**YES**
Generation Fossil Oil	0.3513	< 15	NO
Generation Wind Onshore	0.2024	< 15	NO
Generation Hydro Pumped Storage Consump.	0.4111	< 15	NO
Generation Hydro Run-of-river & Poundage	0.2425	< 15	NO
Generation Biomass	0.2286	26	**YES**
Generation Other Renewable	< 0.20	21	NO
Temperature Range in Valencia	0.2393	34	**YES**
Pressure in Barcelona	0.2871	34	**YES**
Wind Speed in Madrid	0.2335	16	**YES**
Wind Speed in Bilbao	< 0.20	25	NO

Table 2: Feature selection with Pearson correlation and the F-Score of XGBoost

We are left with *11 features* that will be used in our multivariate models. It has been shown that the removal of outliers in time-series and their replacement with linear interpolation, can greatly enhance the performance of artificial neural networks [3], even if it could be considered wrong from a strict scientific perspective. The reason why this is the case, is that the presence of "anomalous" data can introduce bias in the estimation model, especially when dealing with time-series, as they may appear at adjacent points in time. Consequently, we decided to calculate the mean and the standard deviation of each of the 11 features that will be used, make the assumption that they roughly follow normal -or uniform- distributions (although some clearly do not) and *remove all data points which are 4 standard deviations away from the mean*.

Before describing the framework that we will use to evaluate our models, we *normalize* the electricity price and all the 11 features that will be taken into consideration, so that all of their values are within the range of 0 and 1 (see Eq. 1).

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad \textit{(Eq. 1)}$$

The normalization transformation is applied on all data; however, the minimum and maximum of each feature are computed using -or fitted on- *only the train set* (see section 3.1 for more details on how the data is divided into sets), as in a real setting the test set would not be known beforehand. The normalization transformation is preferred to the standardization (z-score) transformation, as it is more robust for our particular case, since the time-series of the used features do not necessarily exhibit stationarity and may incur large errors in the inverse transformation [4] afterwards. Even though the distribution of the electricity price appears to be approximately normal, we decide to use the same transformation for it as well in order to be consistent.

3. Methodology and Implementation

In previous sections, we have mentioned using a train set, i.e. a part of the dataset that we consider already known and that we use to predict the electricity price. In this part, we will go into detail about the experiments that we conducted, the training, validation and testing framework that we used to evaluate our models, how we set the baseline performance (RMSE), as well as the different architectures and hyper-parameters of the artificial neural networks that we used.

3.1. Evaluation and Experimentation Framework

In order to simulate the actual time-series forecasting task under real conditions, we splitted our dataset into 3 sets which were used for different purposes; training, validation and testing. In particular, from the total 35,064 hour-observations (1,461 days) in our dataset, we used the first 27,048 observations (first 1,127 days) as the *train set*, the next 4,008 observations (167 days) as the *validation set* and the last 4,008 observations (last 167 days) as the *test set*.

The train set is the part of the dataset which is used for the *feature selection process* (see section 2.3.), the calculation of the maximum and minimum values for the *normalization transformation* of the electricity price and the selected features (also see section 2.3.), as well as for the *training process* of all the models. The validation set is used for the determination and fine-tuning of the hyper-parameters of our models, whereas the test set is not used in any way until the end of our analysis, at which stage we use it to evaluate the performance of our models, i.e. their generalization performance with new, previously unseen data.

In order to evaluate the performance of our models, we need to establish a *baseline performance* with the use of a simple model so as to have a minimum to compare them with. For this purpose, we use the *persistence model*, also known as the "naive" forecast, which simply uses the value of the current time step (t) as the predicted output of the next time step (t+1). As mentioned in section 1.1., the metric that will be reported for each model is the *root mean squared error* (RMSE), as its value can be roughly translated into the units of the electricity price, i.e. €/MWhr. The persistence model gives an RMSE of **3.111** on the test set, which is probably a very good performance, considering its simplicity and the standard deviation of the electricity price (14.20 €/MWhr). Nevertheless, since we test our models using a different number of past values of the electricity price (univariate and multivariate framework) and the considered features (multivariate framework) that we mentioned in section 2.3., we will also make forecasts with the *XGBoost* algorithm which uses decision tree ensembles (with a maximum depth of 180). In particular, for the XGBoost algorithm, we will use *all the features in our dataset* (75 features) in addition to the previous values of the electricity price. Thus, the XGBoost algorithm will have:

- 228 input features for the configuration which makes use of the 3 previous time-lags.
- 760 input features for the configuration which makes use of the 10 previous time-lags.
- 1,900 input features for the configuration which makes use of the 25 previous time-lags.

Finally, we experiment with five different neural network architectures, different numbers of time-lags used as inputs and the two forecasting frameworks, the univariate and the multivariate. More specifically, for each one of the five architectures (LSTM, Stacked LSTM, CNN, CNN-LSTM, MLP), we experiment on 30 different configurations, based on the following:

- Experiment on the *univariate* variants of the models which use only the previous time-lags of the electricity price, as well as on the *multivariate* variants of the models which use the previous time lags of the electricity price *and* those of the selected features as inputs.
- For both the univariate and multivariate variants of the models, experiment on different lengths of the input sequences for the models. More concretely, the observations of the previous *3, 10 and 25* time-steps of the electricity price will be given as inputs to all of the models, but the multivariate variants will also be given 3, 10 and 25 time-lagged observations of the other 11 selected features.

3.2. Model Architecture

In this part of the project, we describe the five neural network architectures that we used for our experiments, their common implementation details, as well as the chosen and fine-tuned hyper-parameters of each one of them specifically. All the models were implemented with the Keras library running on top of TensorFlow2.

For all the neural network architectures, we use the *AMSGrad* variant of the *Adam* optimizer, which is based on the adaptive estimation of first-order and second-order moments. Furthermore, we use the *mean squared error loss*, which computes the mean of squares of errors between the true values and the predicted values. All models are trained for a maximum of *120 epoch*s, with a batch size of 32 training examples (845 updates per epoch), using the *early stopping* (the training stops automatically if the validation loss does not improve for 10 epochs) and the *model checkpoint* (only the latest model with the lowest validation loss is saved) callbacks.

In order to choose the learning rate of the Adam optimizer, we conduct a preliminary test for each one of the configurations of the models by using the *learning rate scheduler* callback, starting from a learning rate equal to 0.0001 (t=0) and gradually increasing it by a factor of 10 every 10 epochs, until after 50 epochs, we have a learning rate of 1 (see Eq. 2):

$$LR_{t=epoch} = LR_{t=0} * 10^{\frac{epoch}{10}} \qquad (Eq.\ 2)$$

We plot the validation loss for each learning rate on a semi-log plot and finally choose the learning rate which seems to be in the middle of the flat plateau where the validation loss is stable.

3.2.1. LSTM

Long Short-Term Memory (LSTM) networks are designed to learn long sequences with the utilization of the -forget, input and output- gates within their memory cells (their hidden nodes) and have shown to be a very good choice for non-linear time-series, such as electricity prices [5]. Both the univariate and the multivariate variants of this architecture have a LSTM layer with 80 hidden units (the activation function is the *hyperbolic tangent* and the recurrent activation function is the *sigmoid* in all of the LSTM layers that we use, even in the architectures mentioned below) and returns a full sequence which is flattened (Flatten layer) and fed into a fully connected -Dense- layer with 160 neurons and the ReLU activation function. Afterwards, there is a Dropout layer which randomly sets a fraction of the input units to 0 at each update during training (0.05 rate for the univariate variant and 0.10 rate for the multivariate variant), in order to alleviate the issue of overfitting (high variance) and finally, there is the output Dense layer with a single neuron. The multivariate variant (25 time lags) of this architecture can be seen in Fig. 3a.

3.2.2. Stacked LSTM

The *Stacked Long Short-Term Memory (Stacked LSTM)* networks architecture follows a similar logic with the LSTM architecture above, with the difference that two LSTM layers are used; each cell of the first hidden layer passes its own output to the second hidden layer. In both variants of this architecture, the input is fed into a LSTM layer with 250 hidden units which returns a sequence to the second LSTM layer with 150 units and which also returns a sequence; that last sequence is then flattened and given into a Dense layer with 150 hidden units (ReLU activation function) after which there is a Dropout layer (0.05 rate) and finally the output Dense layer with a single neuron (see Fig. 3b).

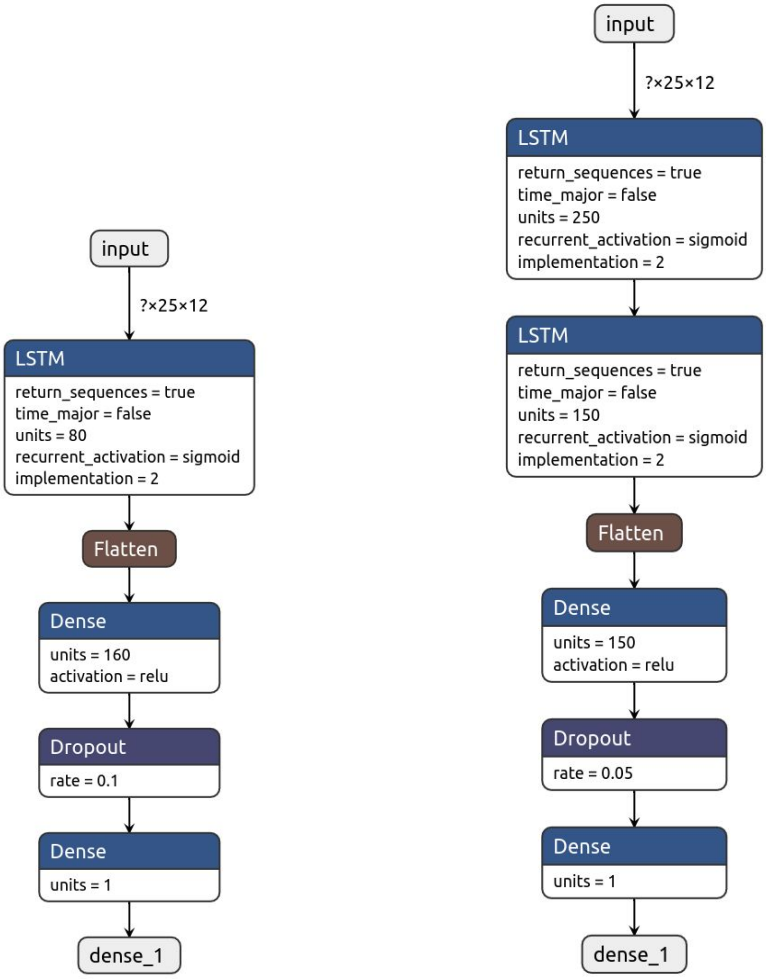

Fig. 3a and Fig. 3b: The multivariate variants of the LSTM (left) and the Stacked LSTM (right) architectures

3.2.3. CNN

The *Convolutional Neural Networks* (CNN) were originally designed for image processing and computer vision tasks; however they have proven quite effective in time-series forecasting and can even outperform LSTM networks [6]. A standard CNN architecture includes a convolutional layer followed by a max-pooling layer. However, in the particular CNN architecture that we use for both the univariate and the multivariate frameworks, the input sequences are fed into an one-dimensional Convolutional layer (Conv1D) with 50 filters, a kernel of size 2, causal padding (so that the model does not violate temporal order) and the ReLU activation function. Its output is then flattened (Flatten layer), fed into a Dense layer (50 hidden units; ReLU) and then finally reaches the last Dense layer with a single neuron which serves as the output layer (see Fig. 4a).

3.2.4. Hybrid CNN-LSTM

The *Hybrid CNN-LSTM* network architecture combines the feature extraction capabilities of CNNs with the sequence analysis capabilities of LSTM networks and has proven effective specifically for the task of predicting the hour-ahead electricity price based on the prices of the previous 24 hours [7]. For both the univariate and multivariate variants of the CNN-LSTM, the input is fed into an 1-D Convolutional layer (100 filters; kernel size 2; causal padding; ReLU activation), followed by a LSTM layer (100 hidden units; returns sequences), a Flatten layer, a Dense layer (50 hidden units; ReLU) and the output -Dense- layer with a single neuron (see Fig. 4b).

3.2.5. Multilayer Perceptron

The last architecture that we experiment with, is the *Multilayer Perceptron* architecture which is constituted by a sequence of *4 Time Distributed Dense layers* with 250, 200, 150 and 100 hidden neurons respectively and the ReLU activation function. The TimeDistributed wrapper in Keras applies a Dense layer to each one of the dimensions which are given to that specific layer. Thus, the first Time Distributed layer applies a Dense layer to each one of the 3, 10 or 25 time-steps that are given as input, the second applies a Dense layer to each one of the 250 time-step dimensions that are given to it by the first and so on. After those 4 Time Distributed layers, a sequence of 100 time-step dimensions is flattened (Flatten layer) and then given to a simple Dense layer (150 hidden units; ReLU), which is followed by a Dropout layer (0.1 rate; the univariate variant has no Dropout layer at all) and finally by the output Dense layer with a single neuron (see Fig. 4c).

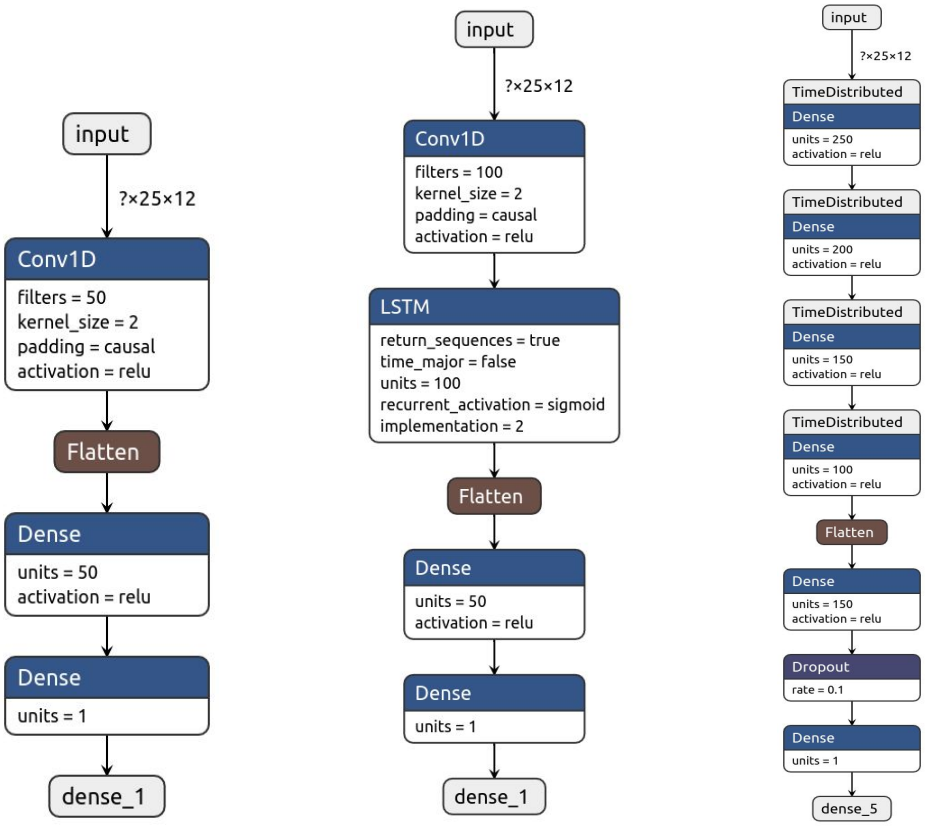

Fig. 4a, Fig. 4b and Fig. 4c. : The multivariate variants of the CNN (left), the Hybrid CNN-LSTM (middle) and the MLP (right) architectures

4. Results

In Table 3, we present the results from all the different configurations that we experimented upon, using the evaluation framework that was described in section 3.1. We should mention yet again that the RMSE of the persistence ("naive") model is 3.111 and that the standard deviation of the electricity price is 14.20 €/MWhr. The main observations to be made from the results can be summarized as follows:

- All models in all configurations were able to outperform the persistence model. However, not all models outperform the baseline RMSE set by XGBoost, as it outperforms all the univariate variants of the models for 3 and 10 time-lags.
- An increase in information given to the models in the form of a larger number of previous time-steps can increase the performance dramatically, although *the opposite is the case* for the multivariate variants of the models if we compare their results for 3 and 10 time-lags; the CNN-LSTM and the MLP architectures are notable exceptions. Surprisingly, an increase in information in the form of additional input features (univariate vs. multivariate)

does not always lead to better predictions and may even lead to a worse performance, as observed for some models with the 25 time-lags configuration. This has been observed in other types of experiments with artificial neural networks for the Spanish energy market [1].

Lags	Framework	LSTM	Stacked LSTM	CNN	CNN - LSTM	MLP	XGBoost
3	Univariate	2.566	2.561	2.556	2.523	2.532	2.373
	Multivariate	2.159	**2.148**	2.170	2.264	2.351	
10	Univariate	2.478	2.370	2.402	2.384	2.400	2.349
	Multivariate	2.277	2.219	2.263	**2.099**	2.253	
25	Univariate	2.052	1.981	1.990	2.003	2.001	2.181
	Multivariate	2.047	2.100	**1.955**	2.034	2.024	

Table 3: RMSE of the electricity price forecast for all experiment configurations

- In all three configurations with a different number of time-lags, the model with the best performance (lower RMSE) is a multivariate variant. Their scores are highlighted in **bold**.
- Overall, the best architecture is the *multivariate CNN*, which scores 1.955 RMSE (~0.138 RMSE to standard deviation ratio) for the 25 time-lags configuration. Its predictions are contrasted with the real electricity prices in Fig. 5. It is also the only architecture that outperforms the corresponding univariate variant in that configuration (25 time-lags). The best univariate model is the *Stacked LSTM* which scores 1.981 RMSE.

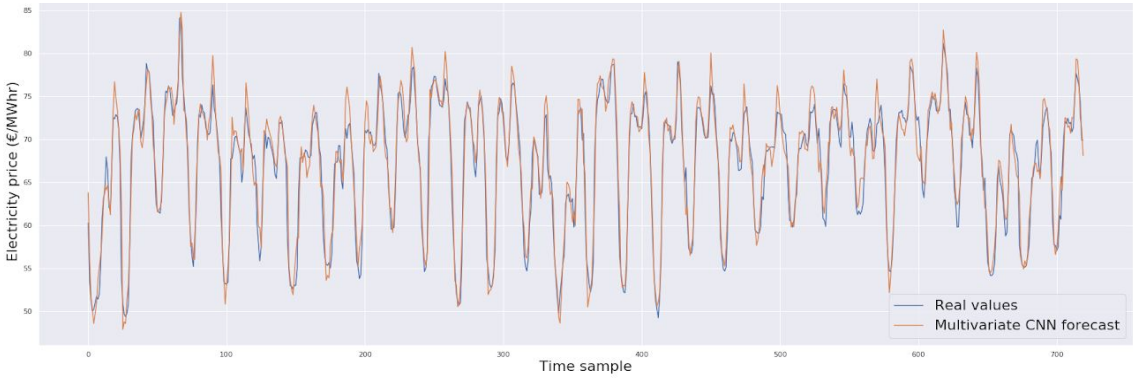

Fig. 5: Real vs. forecasted (multivariate CNN) values for the last 720 samples (last month) of the electricity price

5. Conclusion

In this project we evaluated 5 different deep learning models on the task of predicting the next hour's electricity price in the energy market of Spain. We compared the performance of the models which rely only on the previous values of the electricity price (univariate) with the models which also rely on other, carefully selected and preprocessed, input features (multivariate). Moreover, we experimented on using a different number of previous values for the aforementioned input features and managed to get quite accurate forecasts with an architecture based on convolutional neural networks.

The method that we developed can be used for short-term electricity price forecasting and can prove valuable for the actors (electric power companies, investors, etc.) involved in energy markets, especially if deployed as a forecasting tool which is being updated with new data in real-time.

In future work, the project could be enriched with a different evaluation framework in which walk forward validation would be used to evaluate the robustness of the models for real-time forecasting. Furthermore, an investigation on alternative methods for selecting the most important features that would serve as inputs (e.g. evaluating the importance of each specific previous time-step separately) could prove quite worthwhile, while a more automated framework of tuning the hyper-parameters of the models would certainly provide even better results.

References

[1] Ortiz, M.; Ukar, O.; Azevedo, F. and Múgica, A. (2016). Price forecasting and validation in the Spanish electricity market using forecasts as input data, *International Journal of Electrical Power & Energy Systems*, 77 : 123-127.

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

[3] Wahir, N.; Nor, M.; Rusiman, M. and Gopal, K. (2018). Treatment of outliers via interpolation method with neural network forecast performances, *Journal of Physics: Conference Series*, 995 : 012025.

[4] Zhu, Y.; Dai, R.; Liu, G.; Wang, Z. and Lu, S. (2018). Power market price forecasting via deep learning, *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*: 4935-4939.

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

[6] Koprinska, I.; Wu, D. and Wang, Z. (2018). Convolutional neural networks for energy time series forecasting, *2018 International Joint Conference on Neural Networks (IJCNN)*: 1-8.

[7] Kuo, P.-H. and Huang, C.-J. (2018). An electricity price forecasting model by hybrid structured deep neural networks, *Sustainability*, 10 : 1280.