sciendo

DOI: 10.2478/ijasitels-2020-0009

Modeling Electricity Consumption and Production in Smart Homes using LSTM Networks

Miroslav-Andrei BACHICI[1], Arpad GELLERT[1]

[1]Computer Science and Electrical and Electronics Engineering Department, Faculty of Engineering, "Lucian Blaga" University of Sibiu, Romania
{miroslav.bachici, arpad.gellert}@ulbsibiu.ro

Abstract

This paper presents a forecasting method of the electricity consumption and production in a household equipped with photovoltaic panels and a smart energy management system. The prediction is performed with a Long Short-Term Memory recurrent neural network. The datasets collected during five months in a household are used for the evaluations. The recurrent neural network is configured optimally to reduce the forecasting errors. The results show that the proposed method outperforms an earlier developed Multi-Layer Perceptron, as well as the Autoregressive Integrated Moving Average statistical forecasting algorithm.

Keywords: electricity prediction, Long Short-Term Memory, smart home, energy management system, photovoltaics

1. Introduction

The worldwide energy consumption has seen a substantial growth over the past century and ensuring that everyone has access to energy is still a major and ongoing challenge. In this work we will focus on the electricity consumption and production at household-level, however the modelling of a larger area is also of interest, as we can see in [11] and [12]. In order to cope with the environmental degradation caused by the usage of fossil fuels, the demand of alternative, "renewable energy" sources are undoubtedly increasing. Solar energy is the most important source of this kind, mainly because of the Sun's lifespan which is approximately 5 billion years. Therefore, taking into consideration human timescale, it is safe to say that this is an inexhaustible source of energy. Photovoltaic solar energy, one of the Solar energy's subcategories is captured by photovoltaic panels in the process of producing electric energy. The panels are composed of multiple arrays of photovoltaic cells, usually made of silicon, which in contact with the light from the Sun causes agitation in the electrons of the cells, and therefore, produce

electricity. To analyze how efficient this method of producing electricity in a household is and whether it is feasible or not, different prediction methods have been applied in order to evaluate both the energy production and consumption on such a smaller scale.

In the current stage of the research, we analyze the efficiency of a Long Short-Term Memory (LSTM) recurrent neural network (introduced by Hochreiter and Schmidhuber in [10]) in forecasting the electricity consumption and production in a smart house. In our use case, the electricity production is assured by two photovoltaic panels. We already evaluated some stochastic methods (Markov chains, prediction by partial matching) and some statistical algorithms (ARIMA, TBATS). Other methods will be further implemented and evaluated. We are interested in determining the method with the lowest prediction error, which then will be integrated into a smart energy management system whose role is to keep a balance between the consumption and production of a certain household.

The rest of this paper is organized as follows. Section 2 presents the relevant related work. Section 3 describes the proposed LSTM-based modelling of the electricity consumption and production in a smart house. Section 4 discusses the evaluation results in a comparative view with other existing methods. Finally, Section 5 concludes the paper and presents some further work possibilities.

2. Related Work

In [3], Stefan Feilmeier presented the software architecture of the FENECON energy management system. The author recorded the electricity production of two photovoltaic panels and the loads on three phases in a household during five months, with a step of 5 minutes. The obtained dataset was used later for evaluations in different researches (including our current work). In [3], the dataset was used to evaluate a Multi-Layer Perceptron (MLP) in predicting the electricity consumption and production. The author reported a mean absolute error of 211.07 Watts. Other works which are applying neural networks to predict the electricity consumption of buildings are [13] and [9]. Sensor-based forecasting of the electricity consumption in a large entertainment venue with neural networks and support vector regression is investigated in [8].

In [4], we adapted a Markov model to be able to work with the electricity consumption and production levels and to provide a short-term prediction of the upcoming levels. To reduce the state-complexity of the model, we downscaled the input data by dividing all the values to the same scaling factor. The value returned by the model is upscaled to obtain the predicted electricity level, by multiplying it with the same scaling factor. The scaling factor, the length of the input vector, as well as the order of the Markov Model, were the main parameters varied in the experiment. The mean absolute error, measured

on the same dataset which is used in our current work, was 34.43 Watts. A stride predictor and a hybrid predictor composed of the Markov model and the stride predictor have been also evaluated. However, the stride predictor proved to be inefficient and, thus, the more complex hybrid predictor could not provide better results than its Markov prediction component.

To predict the next values of time series, several machine learning techniques can be considered as good candidates, a design pattern for an efficient implementation being presented in [6]. The efficiency of the prediction model is highly dependent on the type of the input data. Therefore, we evaluated different prediction methods with the goal of finding the most appropriate one for the electricity consumption and production datasets. In [5], we used statistical methods to predict the electricity consumption and production. The Autoregressive Integrated Moving Average (ARIMA) algorithm can describe time series based on the past values or lags and the forecast errors. The other statistical method is based on Trigonometric Seasonal, Box-Cox Transformation, ARMA Residuals, Trend and Seasonality components (for short TBATS), is decomposing seasonal time series into trend, seasonal and irregular components. The evaluations have shown a mean absolute error of 198.27 Watts for the ARIMA algorithm and 73.62 Watts in the case of the TBATS model.

In [14], Monteiro et al. evaluate short-term statistical prediction of photovoltaic electricity production. Two models are proposed: one of them is analytical and the other one is using a MLP. The prediction relies on weather forecasting tools focused on the location of the photovoltaic plant, as well as on hourly recorded photovoltaic electricity production. The analytical model computes the sky irradiation based on hourly radiation forecasts and adjusts it with irradiation attenuation index and photovoltaic production attenuation index. The neural network was selected and configured using genetic algorithms and is using weather forecasts as input information. The proposed models were evaluated and compared on the same data collected from a grid-connected photovoltaic plant. The authors concluded that the two models have similar results, and both are usable in the sight of selling electricity to the markets.

In [2], Fan et al. evaluated the hybrid prediction through data mining techniques of the next-day energy consumption in buildings. The proposed method has three steps. In the first step, an outlier identification and removal is performed. In the second step, a recursive feature elimination is applied in order to use the optimal inputs for the eight different predictors. In the final step, an ensemble model is optimized for the predictors through a genetic algorithm. The authors concluded that the proposed ensemble model can be efficient in fault detection.

3. LSTM-Based Forecasting of Electricity Consumption and Production

This section focuses on describing the proposed model of this paper and the functionalities that it has. For our paper, we implemented and used an LSTM, which is a model of Recurrent Neural Network (RNN). RNNs represent one of the most optimal choices when working with data organized in time series models. Their work principle is based on combining nonlinear activation functions in a recurrent structure, which makes prediction possible and provides improved prediction accuracy, as stated in [1]. In contrast to the standard Neural Networks, which are usually represented using feedforward architectures, RNNs allow the information to be transferred both forward and backward, with the help of their feedback connections. Therefore, these neural networks benefit from the ability to work with dynamic data. An analysis regarding the applicability of RNNs for prediction purposes is presented in [15].

An RNN can have multiple layers, steps or stages. Their work principle is described in Fig. 1. Each stage from the above schema corresponds to a given time T. The RNN at the time T+1 will use the RNN from the time T as one of its inputs. Each stage will send its output to the next stage. The key mechanism which makes the RNN to work well is represented by the hidden state information propagated from a certain stage to the next. The hidden state works as a memory capable of retaining information of the current stage. A layer from the RNN is processing the input data and is returning its internal state which is going to be used as an input in the next stage. More specifically, each stage is trained to transform the target sequence from a moment T into the input sequence but with a T+1 timestep offset. To achieve this, a backpropagation algorithm is used. The value of the loss obtained for each parameter is used to change the parameter values in the reverse direction with the purpose of minimizing the loss. As this movement is time based, each timestep contains its own loss value. In the process of modeling the dependencies between value sequences, the gradient of the timestep T depends on the gradient of the timestep T-1 and so on, and because of this, the further we progress with the timesteps, the gradient of the latter timestep matters less and less. This is known as the "vanishing gradient problem" whose effect is that the network cannot learn from long term dependencies, because the gradients of the early stages become smaller. LSTM networks are a solution to this problem.

LSTMs are RNNs that work with data that varies through time or sequentially, like language, stock market prices, weather recording sensors, etc. The way they work is similar to other RNNs, by using the outputs of a layer at a timestep T as inputs for the same layer at a timestep T+1. They have a component that acts as a memory which helps to transfer information learned at the timestep T to the next timesteps, and they can also forget irrelevant information from the preceding state and update the current state, allowing

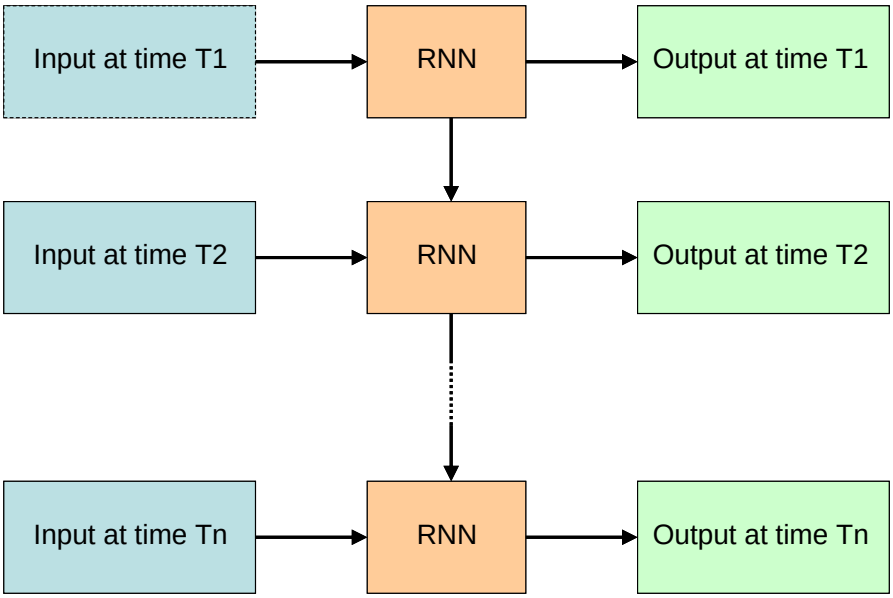

Figure 1. RNN

only important parts of the state to reach the output. The networks use activation functions to induce nonlinearity to the data.

Among the most used activation functions are the sigmoid and the hyperbolic tangent. As in our datasets we had no negative values, we decided to use the sigmoid function in our LSTM network, as the interval of this function is [0,1]:

$$f(x) = \frac{1}{1+e^{-x}} \tag{1}$$

As mentioned before, LSTMs are capable of remembering and choosing which data is relevant as future inputs. They do this by using three gates that release data between hidden state and cell state. These gates are called "forget gate", "input gate" and "output gate". An LSTM neuron incorporates a cell, an input gate, an output gate, as well as a forget gate. The transformation process of information passing through a cell is described in [7] as follows. All the gates of the cell are collecting activations from the block and from the outside. A recurrent connection with the weight 1 keeps the current internal state of a cell. The input and output gates scale the input and output of the cell by using activation functions. The forget gate decides which information must be eliminated from the cell state. That is a sigmoid layer, which provides output values between 0 and 1, and scales the internal state, so as the values exiting the gate are ranged in the interval mentioned above.

For our experiment, we implemented the LSTM network using Python 3 and the TensorFlow framework with the Keras API. As input data, we used the datasets provided by the FENECON Energy Management System (FEMS) described in [3]. Fig. 2 shows the schema of this system, with "PV1" and "PV2" being the producers and "Ph1", "Ph2" and "Ph3" being the consumers.

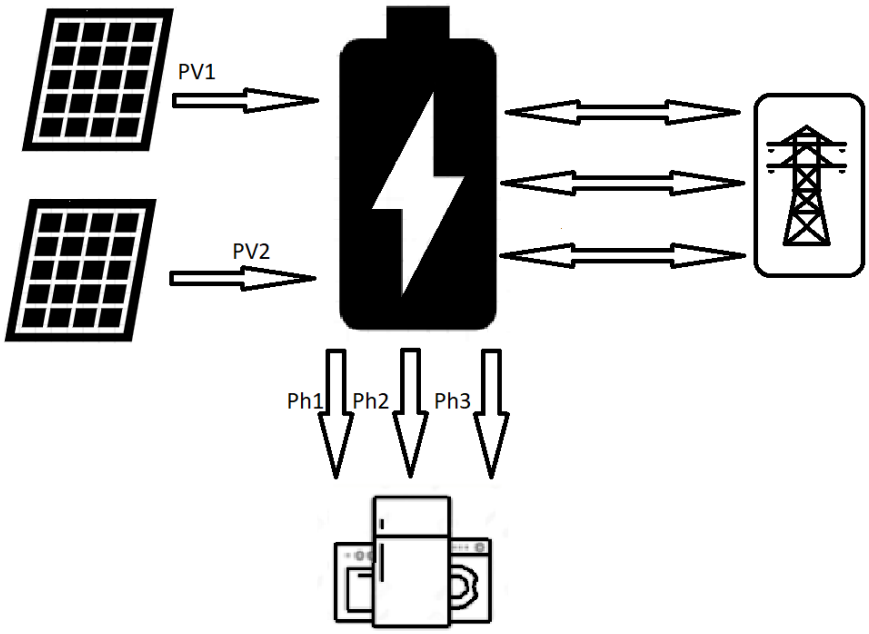

Figure 2. FEMS

The datasets from the system consist in 5 files, two of them containing values from the energy produced by "PV1" and "PV2" photovoltaic panels, and the other three files ,each having recordings for "Ph1" , "Ph2" and "Ph3" consumers. The recording interval for each value is 5 minutes and the period is 5 months. Each file consists of a 1D array of values.

To determine the accuracy of our LSTM network, we used the Mean Absolute Error (MAE) metric:

$$MAE = \frac{\sum_{i=1}^{N} |R_i - P_i|}{N} \tag{2}$$

where R_i is the real value at time i, P_i is the predicted value (the output provided by the LSTM) at time i, whereas N is the total number of evaluated electricity levels. All the values from our datasets are measured in Watts and therefore, the MAE values which we are going to report further are also represented in Watts.

4. Experimental Results

In this section we focus on the results we obtained from our experiment. We tuned the LSTMs parameters in an effort to try and find the best configuration that would produce the smallest value for the MAE. We started with a standard configuration of 5 inputs, two hidden layers each containing 50 neurons, a learning rate of 0.01 and 30 epochs. The first parameter that we varied was

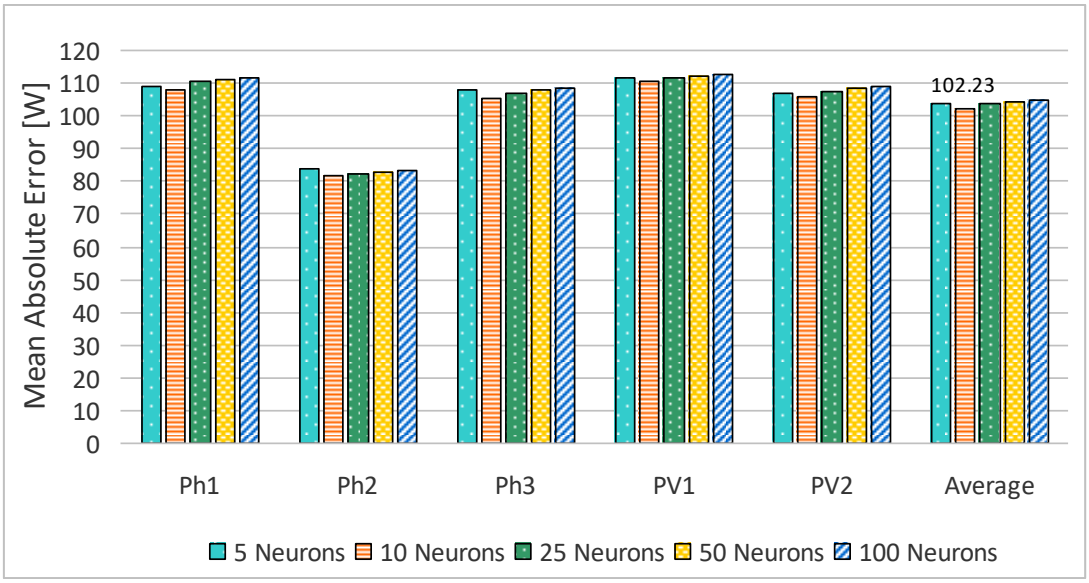

Figure 3. The influence of the number of neurons from the first hidden layer

the number of neurons in the first hidden layer, going from 5 values to 10, 25, 50 and 100, leaving the rest of the configuration unchanged. Due to the fact that the LSTM provides slightly different results in different runs because of its random initialization, ee ran each dataset through the network 5 times for each changed parameter, and calculated the average of the MAE values we obtained. By increasing the input number, we noticed the MAE value was increasing. After a series of experiments towards this direction, we concluded with 10 being the optimal value, and thus we obtained the MAE equal to 102.23 for this configuration. Fig. 3 shows a graph with the values obtained following the tests.

Next we varied the number of neurons from the second hidden layer, following the same pattern that we used for the previous varied parameter. Starting with the base configuration and adding the optimal value 10 for the first hidden layer, we experimented with the second layer starting with 5 neurons, then 10, 25, 50 and 100, and the smallest MAE value we obtained was 101.47, for 5 neurons on the second hidden layer. We concluded with this value being the optimal tune for this parameter. Fig. 4 describes the results obtained by experimenting with the above mentioned values through all the datasets.

The next parameter we tuned was the learning rate, starting from a value 0.01 and slightly increasing it to 0.02 and 0.03. We noticed that by increasing the learning rate, the MAE value also increased, to the point where we reached the value 107.1 with a 0.03 learning rate value, so we decided to stop increasing it. The optimal configuration here is with a value of 0.01, having a MAE of 101.47, which means that our configuration up until this point was accurate with this parameter already having an optimal value. The graph with the results can be seen in Fig. 5.

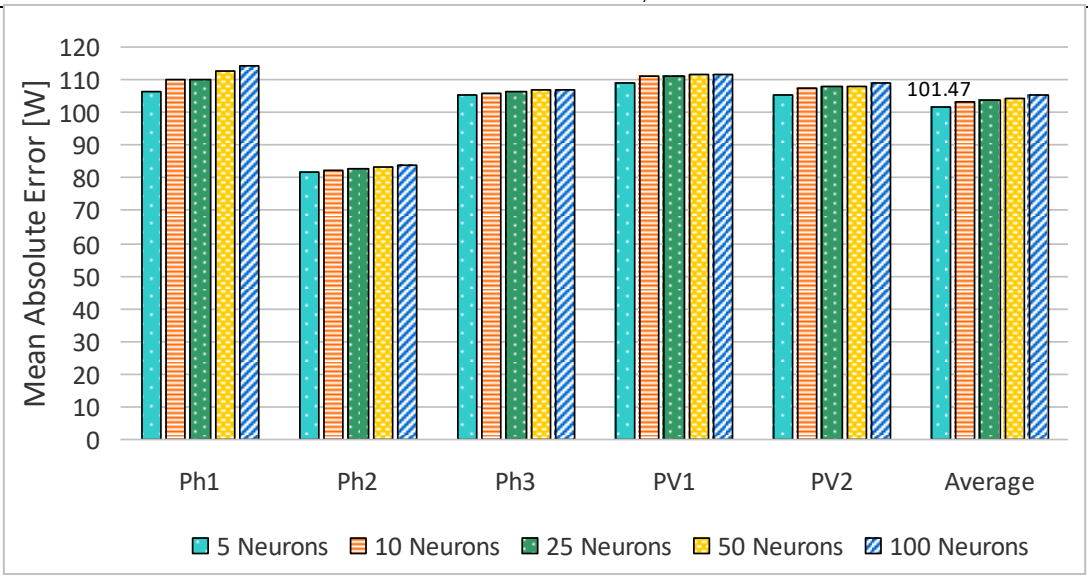

Figure 4. The influence of the number of neurons from the second hidden layer

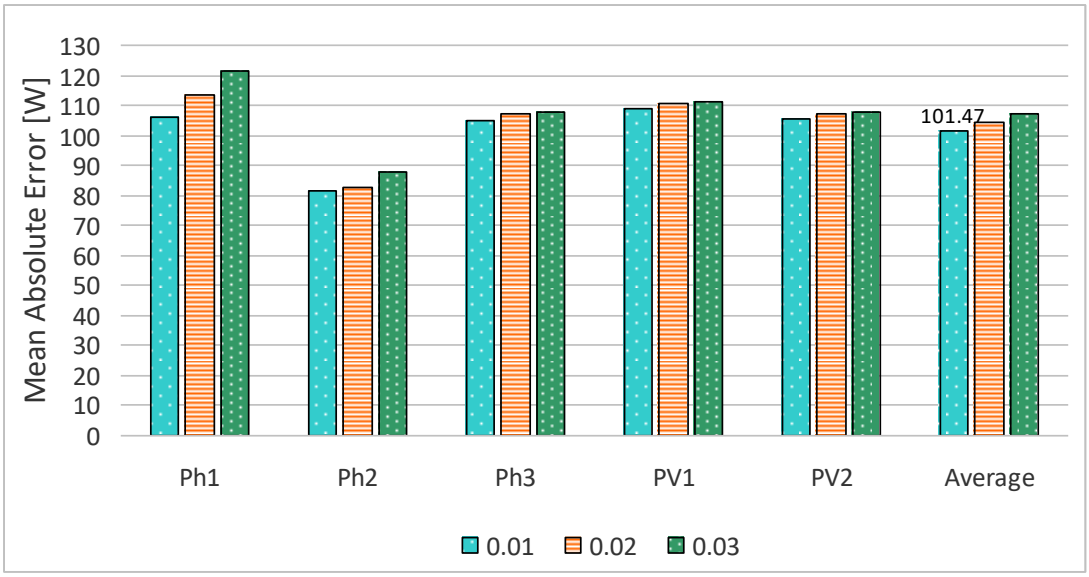

Figure 5. The influence of the learning rate

The next parameter we varied for our configuration was the number of inputs. Having reached the MAE equal to 101.47 with our current configuration using 5 inputs, we increased the number to 10, 15, 20 and 25. We noticed that the higher the number of the inputs, the higher the value of the MAE became. So we also decided to try a smaller number than the starting one and, thus, we went with 4 inputs. This proved to be the right decision, as we reached a MAE equal to 100.99. The results are visible in the graph from Fig. 6.

The last parameter that we decided to vary was the number of epochs. Our base configuration had 30 epochs which achieved the above mentioned MAE, so we decided to increase this number. We varied through 50, 100 and 500 epochs. The results we obtained drove us to the conclusion that 50 epochs was the best configuration, having obtained a MAE equal to 100.77.

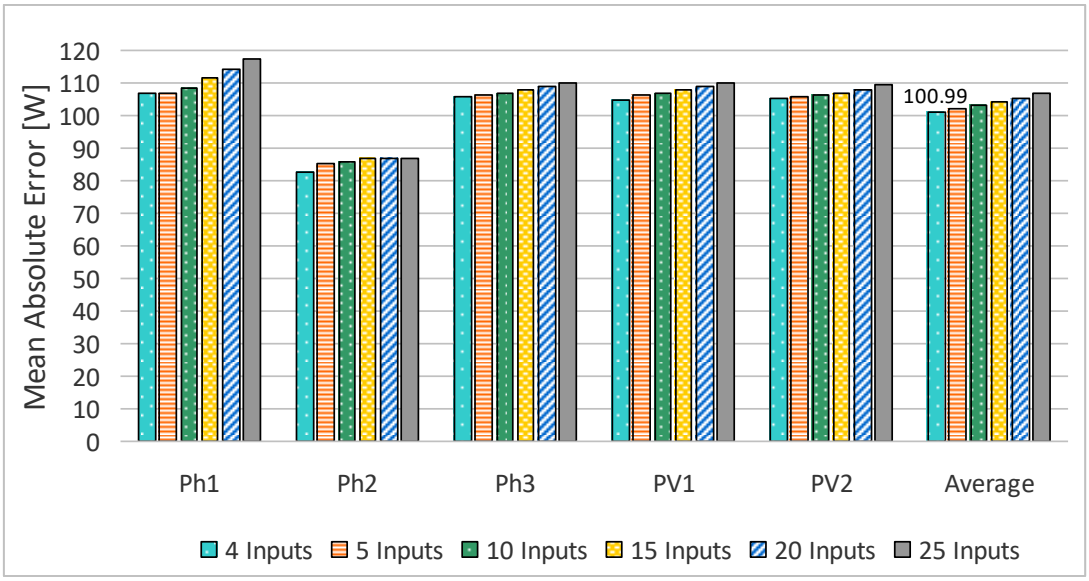

Figure 6. The influence of the input vector size

We also tried to go below our starting value and we decided to run a series of tests with 25 epochs, but as we can see in the graph from Fig. 7, the MAE was higher than the one we obtained with our optimal configuration.

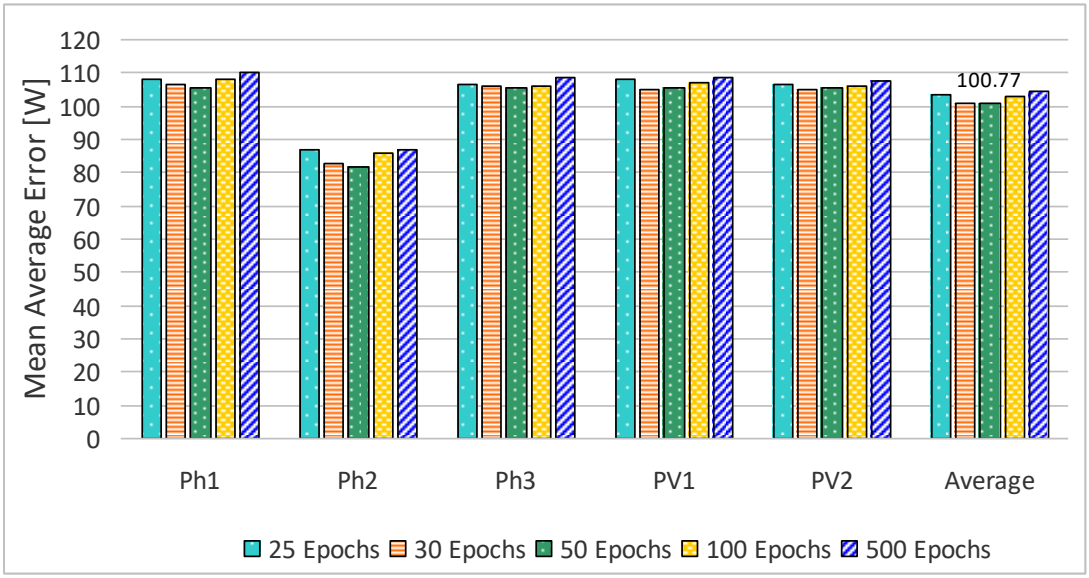

Figure 7. The influence of the number of epochs

After the experiment we concluded that the optimal LSTM configuration has 4 inputs, a first hidden layer with 10 neurons, a second hidden layer with 5 neurons, a learning rate of 0.01 and 50 epochs. Next we made a comparison of our results with other methods used to calculate the MAE on the same datasets.

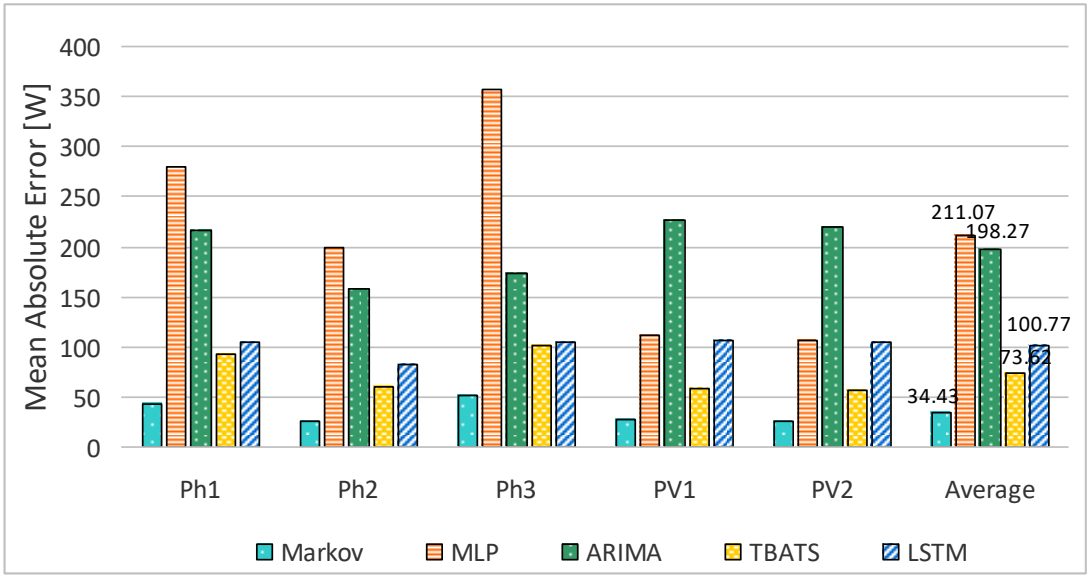

Figure 8. Comparison with other forecasting methods

As the graph in Fig. 8 portraits, with a MAE of 100.77 Watts, our LSTM network outperformed a MLP [3], which had a MAE equal to 211.07, and an ARIMA model [5], with a MAE equal to 198.27. Our LSTM had poorer performance than a TBATS algorithm [5], which had a MAE equal to 73.62, and also than a Markov predictor [4] with MAE 34.43.

5. Conclusions and Further Work

In this paper, we analyzed the LSTM used as a predictor of the electricity consumption and production in a smart house. The goal is to integrate such a predictor into a smart energy management system of a household, that might keep a balance between the electricity consumption and production avoiding demands from the grid. The evaluations performed on the datasets collected from a real household have shown that the LSTM's mean average error is 100.77 Watts, which is half of the mean average error encountered by the MLP and by the ARIMA [5] statistical algorithm, respectively. The LSTM proved to be less accurate than the Markov predictor [4] and the TBATS algorithm [5], but we can classify it among the best methods.

Taking into account the constructive and functional differences of the best performing methods (the Markov model-based stochastic predictor, the TBATS statistical predictor and the LSTM neural predictor), we are interested to develop a hybrid prediction mechanism able to exploit all these three predictors as components. Another further work direction is to develop and evaluate a prediction method relying on fuzzy logic.

References

[1] S. Abdulkarim, *Time series prediction with simple recurrent neural networks*, Bayero Journal of Pure and Applied Sciences, Vol. 9, No. 1, 2016.

[2] C. Fan, F. Xiao, S. Wang, *Development of prediction models for next-day building energy consumption and peak power demand using data mining techniques*, Applied Energy, Vol. 127, 2014.

[3] S. Feilmeier, *Loads management based on photovoltaic and energy storage system*, M.Sc. Thesis, Lucian Blaga University of Sibiu, 2015.

[4] A. Gellert, A. Florea, U. Fiore, F. Palmieri, P. Zanetti, *A study on forecasting electricity production and consumption in smart cities and factories*, International Journal of Information Management, Elsevier, ISSN 0268-4012, Vol. 49, pp. 546-556, December 2019.

[5] A. Gellert, U. Fiore, A. Florea, R. Chis, *Forecasting Electricity Consumption and Production in Smart Homes*, Submitted to Pervasive and Mobile Computing, November 2020.

[6] A. Gellert, A. Florea, *Investigating a New Design Pattern for Efficient Implementation of Prediction Algorithms*, Journal of Digital Information Management, Vol. 11, Issue 5, ISSN 0972-7272, pp. 366-377, October 2013.

[7] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Studies in Computational Intelligence, Springer, ISBN: 978-3-642-24796-5, January 2012

[8] K. Grolinger, A. L'Heureux, M. Capretz, L. Seewald, *Energy Forecasting for Event Venues: Big Data and Prediction Accuracy*, Energy and Buildings, Vol. 112, pp. 222-233, 2016.

[9] L. Hernández, C. Baladrón, J. Aguiar, B. Carro, A. Sánchez-Esguevillas, J. Lloret, *Artificial neural networks for short-term load forecasting in microgrids environment*, Energy, Vol. 75, pp. 252-264, 2014.

[10] S. Hochreiter, J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, Vol. 9, No. 8, pp. 1735-1780, 1997.

[11] K. Kavaklioglu, *Modeling and prediction of Turkey's electricity consumption using Support Vector Regression*, Applied Energy, Vol. 88, Issue 1, pp. 368-375, 2011.

[12] T. Khatib, A. Mohamed, K. Sopian, M. Mahmoud, *Solar Energy Prediction for Malaysia Using Artificial Neural Networks*, International Journal of Photoenergy, Vol. 2012, 2012.

[13] H. Khosravani, M.D.M. Castilla, M. Berenguel, A. Ruano, P. Ferreira, *A Comparison of Energy Consumption Prediction Models Based on Neural Networks of a Bioclimatic Building*, Energies, Vol. 9, Issue 1, 2016.

[14] C. Monteiro, L.A. Fernandez-Jimenez, I.J. Ramirez-Rosado, A. Muñoz-Jimenez, Pedro M. Lara-Santillan, *Short-Term Forecasting Models for Photovoltaic Plants: Analytical versus Soft-Computing Techniques*, Mathematical Problems in Engineering, Vol. 2013, 2013.

[15] J. Park, D. Yi, S. Ji, *Analysis of Recurrent Neural Network and Predictions*, Symmetry, Vol. 12, No. 4, 2020.