# Comparing Neural Network Models for Photovoltaic Power Generation Prediction

1st Carlos Henrique Torres de Andrade
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
chta@ic.ufal.br

2nd Tiago Figueiredo Vieira
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
tvieira@ic.ufal.br

3rd Ícaro Bezzera Queiroz de Araújo
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
icaro@ic.ufal.br

4th Gustavo Costa Gomes de Melo
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
gustavocosta@ic.ufal.br

5th Erick de Andrade Barboza
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
erick@ic.ufal.br

6th Davi Bibiano Brito
*Computing Institute (IC)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
davi@ic.ufal.br

7th Igor Cavalcante Torres
*Engineering and Agricultural Sciences Campus (CECA)*
*Federal University of Alagoas (UFAL)*
Maceió, Brazil
igor.torres@ceca.ufal.br

***Abstract*—Research on alternative energy sources has been increasing for the past years due to environmental concerns and the depletion of fossil fuels. Since photovoltaic generation is intermittent, one needs to predict solar incidence or solar power generation to alleviate problems due to demand surges in conventional distribution systems. Many works have used Long Short-Term Memory (LSTMs) to predict generation. However, to minimize computational costs related to retraining and inference, LSTMs might not be optimal. Therefore, in this work, we compare the performance of MLP (Multilayer Perceptron), Recurrent Neural Networks (RNNs), and LSTMs for the task mentioned above. We used the energy produced by a photovoltaic system measured throughout 2020 in the city of Maceió (Brazil), taking into account periods of 2 hours for training to predict the next 5-minutes. Hyperparameters were fine-tuned using an optimization approach based on Bayesian inference to promote a fair comparison. Results showed that the MLP has satisfactory performance, requiring much less time to train and forecast. Such results indicate that MLP models can be better when dealing with a short-term forecast in specific contexts, for example, in embedded systems for improved response time.**

***Index Terms*—Multi-Layer Perceptrons, Recurrent Neural Networks, Renewable energy sources, PV power forecasting**

## I. Introduction

The natural depletion of fossil fuels worldwide boosts a search for alternative energy sources to meet increasing demands. In this context, new research and investments aimed at improving renewable energy efficiency are increasing. Among all the options, photovoltaic energy (PV) stands out as a clean, inexhaustible, and environmentally friendly power source. According to the Ministry of Mines and Energy of Brazil [1], renewable sources represented 83.6% of installed generation capacity in Brazilian's power generation matrix in April 2021 (hydraulic, biomass, wind, and PV), with PV responsible for 4.9% of total production. In addition, PV presented an installed capacity growth, between April 2020 and April 2021, of 74.1%, reiterating its growing importance in energy generation in Brazil.

The authors are grateful for the support of the National Electric Energy Agency (ANEEL) and the Equatorial Group of Energy.

Electric power generated by a PV panel depends on environmental factors such as solar irradiation and PV's cell temperature, which causes energy production to become uncertain, generating doubts about its capacity, availability, and reliability. Hence, using prediction models is essential to make PV systems competitive against more traditional means of energy production because they allow for better planning, distribution, and economic return, bringing a vision of stability and reliability. In addition, the output power of PV systems forecast enables systems to perform control actions, compensate for fluctuations in solar radiation and increase the efficiency of the solar power plant.

Variable forecasting is already widespread in the literature, where computer models seek to forecast the most diverse types of variables, from the price of a house (Lim, 2016 [2]) to the air quality in a room (Xie, 2009 [3]). Regarding energy production, Sabino (2018) [4] conducted a study of several techniques of autoregressive models for temperature prediction, seeking to help the generation of photovoltaic energy. Another type of model that is quite powerful for this type of problem is the so-called support vector machine (SVM). Sun (2015) [5] and Yang (2013) [6], worked with this type of model for the prediction of temperature and solar irradiation, respectively, both essential climatic variables for the electrical production by the photovoltaic module.

Still related to the production of solar energy, one of the techniques that have been standing out is artificial neural networks (ANN). One of the most popular ANN is the feedforward multilayer perceptron (MLP), whose efficiency is demonstrated by Mellit (2010) [7] and Watetakarn (2015) [8] for the prediction of solar irradiation, and Malki (2004) [9] for the short-term prediction of the electrical power generated by a photovoltaic module. Another ANN that has been gaining much attention is the recurrent neural network (RNN), more specifically the so-called long short-term memory (LSTM), which can be attested by: Mahmoud (2017) [10], De (2018) [11] and Liu (2021) [12].

Often works have tackled the problem of PV power generation using sophisticated recurrent neural networks (RNN) for accurate prediction [13] [14] [15]. Since the computational cost of a deep recurrent neural network with too many parameters can be prohibitively expensive to be embedded on a small device, in this work, we aim at answering the following question;

1) How complex does the network architecture need to be to predict short-term power production demands satisfactorily?

In order to answer the research question, we compared the performances of MLP, simple RNN, and LSTM to predict five minutes of power generation using the past 120 minutes, as per İzgi (2012) [16] indicates that a five-minute horizon presents the best results for short-term predictions. The hyperparameters were fine-tuned to ensure we had the optimal (minimum) error for each given architecture. Furthermore, the results were compared between each other and to a naive baseline, showing no significant reduction in error levels for PV power generation prediction provided by models with recurrence. Therefore, we conclude that lighter MLP is sufficient to perform the prediction satisfactorily.

This work presents the following sections: Section 1, the introduction is shown. Section 2 presents an overview of the topologies of recurrent neural networks and the evaluation metrics used. Section 3 presents the methodology, and the results obtained are exposed and discussed in section 4. Finally, in section 5, the conclusions about the work are presented.

## II. Theoretical Foundation

### A. Artificial Neural Network (ANN)

According to Haykin (2007) [17], the artificial neural network is a parallel distributed processor formed by simple processing units capable of acquiring knowledge and storing it for future use. Artificial neural networks are often presented as systems of interconnected neurons that can compute input values simulating the behavior of biological neural networks. Currently, there are several types of neural network architectures but, in this work, we will address only the multilayer perceptron (MLP), the simple recurrent networks (RNN), and the long short-term memory (LSTM).

*1) Multilayer Perceptron (MLP):* MLP is a feedforward artificial neural network. In other words, the signal flow crosses the network from left to right and from layer to layer. It can map non-linear relationships between input variables and the output goal, thus solving classification and regression problems.

In this topology, the network can have one or more hidden layers, in which every neuron present is connected to all the neurons in the previous layer, as shown in Fig. 1. The behavior of each neuron in this architecture can be described by (1), where $x_j$ are the inputs, $W_{kj}$ their respective weights, $b_{kj}$ are the bias, $\phi$ its activation function and $y_k$ its output.

$$\overline{y}_k = \phi(\sum_{j=1}^{m}(W_{kj} \cdot \overline{x}_j) + b_k) \tag{1}$$

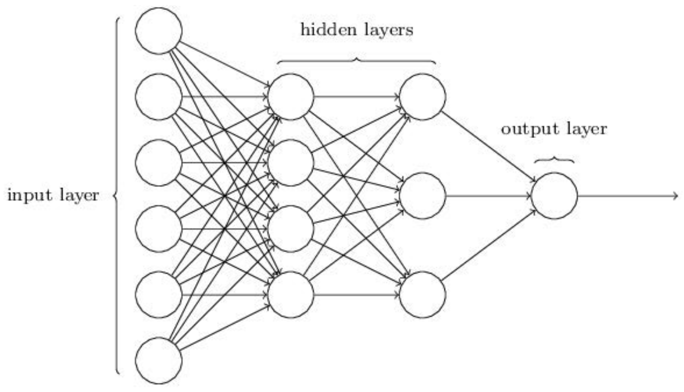


Fig. 1. Structure of an MLP network [18].

*2) Recurrent Neural Network (RNN):* A simple recurrent neural network differs from a feedforward neural network because it presents feedback [17]. This feedback makes the activation go through the entire model in a loop, which allows the networks to perform temporal processing, making them capable of remembering context. The behavior of this recurrence is shown by (2) which the hidden state $\overline{h}_t$ in each layer at the instant of time $t$ does not depend only on the input $\overline{x}_t$ but rather the combination of the input and the value of the previous state. Then, after the bias $b_h$ is summed, it passes through the activation function, usually the hyperbolic tangent. Fig. 2 briefly illustrates the whole process, where, finally, we have the output, which is given by (3).

$$\overline{h}_t = \phi(W_{xh} \cdot \overline{x}_t + W_{hh} \cdot \overline{h}_{t-1} + b_h) \tag{2}$$

$$\overline{y}_t = W_{hy} \cdot \overline{h}_t \tag{3}$$

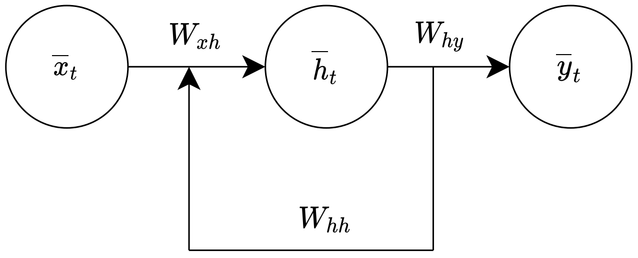


Fig. 2. Simple recurrent neural network [13].

*3) Long Short-Term Memory (LSTM):* Introduced by Schmidhuber (1997) [19], the LSTM network is a modification of the simple RNN, which seeks to solve the problem of gradient dissipation (Vanishing Gradient Problem). This new architecture makes it capable of updating the weights corresponding to the initial states of the sequences, thus learning long-term dependencies.

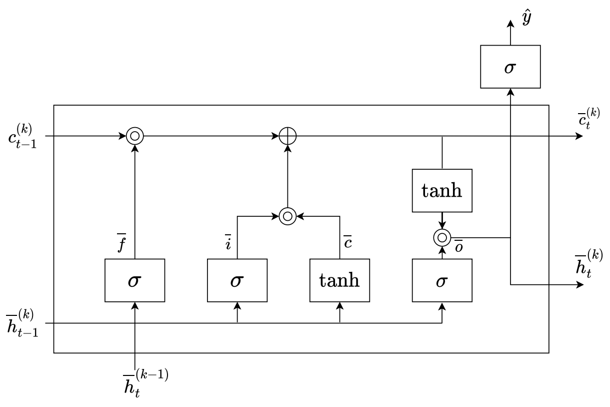


Fig. 3. Basic unit of a long short-term memory (LSTM). [13]

At each time step $t$, it tries to decide whether to forget the current information, approaching it to zero or not, according to (4). This happens mainly due to the operation of the forgetting and updating gates that seek to decide which information is kept in the cell state and which input values to be updated by the LSTM blocks, respectively, where (5) and (6) describe the elements that constitute them, being $\overline{h}$ the concatenation of $\overline{h}_t^{(k-1)}$ and $\overline{h}_{t-1}^{(k)}$; $W^{(k)}$ is the weight matrix; $\sigma$ the sigmoid function and $\tanh$ the hyperbolic tangent function. Finally, we have the output calculation of the current cell $\hat{y}_t$, given by (7) and (8).

$$\overline{c}_t^{(k)} = \overline{f} \odot \overline{c}_{t-1}^{(k)} + \overline{i} \odot \overline{c} \tag{4}$$

$$\overline{i}, \overline{f}, \overline{o} = \sigma(W^{(k)}\overline{h}) \tag{5}$$

$$\overline{c} = \tanh(W^{(k)}\overline{h}) \tag{6}$$

$$\overline{h}_t^{(k)} = \overline{o} \odot \overline{c}_t^{(k)} \tag{7}$$

$$\hat{y}_t = \sigma(\overline{h}_t^{(k)}) \tag{8}$$

## B. Evaluation Metrics

The metrics mean absolute error, root mean square error, and standard deviation are used to evaluate the prediction models developed in this work. The mean absolute error is used as the loss function in the training phase of all ANN presented in this work. In addition, the number of trainable parameters and the inference time as ways to evaluate the complexity and usability of the ANN. All metrics are described below.

*1) Mean Absolute Error (MAE):* It is a model evaluation metric used primarily with regression models. The MAE of a model, (9) is the average of the absolute values of the individual prediction errors across all instances of the test dataset, where each prediction error is the difference between the actual value and the predicted value.

$$MAE = \frac{1}{N}(\sum_{i=1}^{N} |y_i - \hat{y}_i|) \tag{9}$$

*2) Root Mean Squared Error (RMSE):* It is the square root of the mean square of all errors, as shown in (10). The use of RMSE is widely used in the literature and is considered an excellent general purpose error metric for numerical predictions [20].

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{10}$$

*3) Standard Deviation (STD):* The standard deviation is given by (11) and tells us the average amount of variability in the prediction error and can be used to measure the confidence in the statistical conclusions of the models.

$$STD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(e_i - \overline{e})^2} \tag{11}$$

*4) Number of Trainable Parameters (NTP):* As the name describes, it represents the number of parameters that will be calculated during the training process of the artificial neural networks. It is a way to estimate how complex the network is and how time-consuming the training will be.

*5) Inference Time (IF):* It stands for the time a model takes to produce its output. In the context of this work, to ensure that it is correctly calculated, the inference time consists of the average of the time it took to predict one hundred random samples.

## III. Methodology

### A. Database

The database used comprises thirteen variables captured throughout the year 2020, during day and night, sampled every 60s. This data was acquired by a solarimetric station equipped with: a CR1000 data logger (an electronic device that records data over time), pyranometer, humidity sensor, rain gauge, wind direction indicator, anemometer, thermistors, current sensors, voltage sensors, and power sensors. However, as the forecasts in the project are based on time series, only the variable of interest is needed. Thus, the power generated by the photovoltaic module was the only one used.

The Tab. I shows the main characteristics of the 523,058 samples of electrical power generated by the photovoltaic module obtained. The main characteristics of the 523,058 samples of electrical power generated by the photovoltaic module were obtained. Another critical factor of the database is that it has a reasonable amount of outliers, which was expected due to sensor measurement errors. This fact can be observed in Fig 4.

TABLE I
MAIN CHARACTERISTICS OF THE DATABASE.

| | **characteristic** | | | |
|---|---|---|---|---|
| | **Maximum** | **Minimum** | **Average** | **Standard deviation** |
| value | $7,807$ W | $-110.1$ W | $1,006.517$ W | $1,437.447$ W |

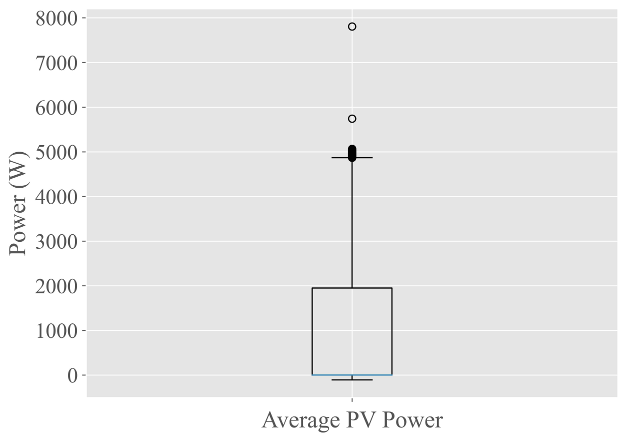


Fig. 4. Boxplot of the database. It is noticeable that the median is close to zero due to many samples acquired during the night.

## B. Data Analysis and Preprocessing

Initially, an analysis and preprocessing of the available data were carried out, which were obtained from the aforementioned solarimetric station. It was observed that some given instances had missing or negative values resulting from technical problems and sensor noise. Such missing values were removed, and negative were truncated to zero. Outliers associated with measurements performed during the night (moments with zero solar irradiation and consequently no power generation) were left in the database because the models must operate under such conditions in day-to-day use.

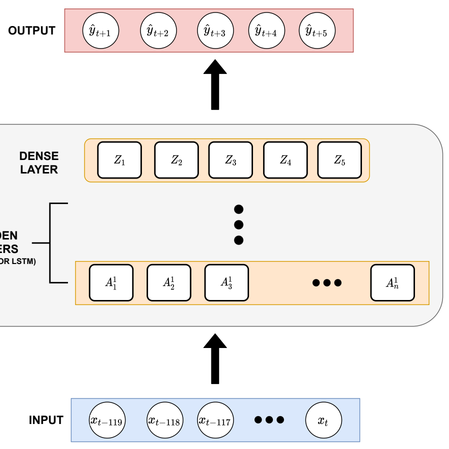
For each sample, the 120 previous ones were grouped as a sequence to form the input units and the subsequent five samples as the respective output. Afterward, the database was divided by reserving the last 10% of the data for testing. Then, the first 90% data were grouped into eight distinct sets for hyperparameter tunning, with 7 for training and 1 for validation. In the end, the test set enables a vision into the network's performance on data that were not used during the training and validation process, also simulating the network's performance in an actual operating scenario.

## C. Prediction Models

For the implementation of all neural networks in this work, the programming language Python 3.8 was used, as well as open-source libraries Keras [21] and scikit-learn [22] as they facilitate the implementation of well-performing neural networks.

The first model created was a "naïve" prediction model, or *baseline*. In baseline, the forecast is given by the variable value in the day before at the same time. Once its construction was completed, forecasts were made throughout the database. The three metrics: MAE, RMSE, and STD, were used, together with actual values, to evaluate the method's performance.

Three neural network architectures were used: MLP, simple RNN, and LSTM. As the purpose is to predict five minutes of power generation, using the values of the previous two hours, all developed networks receive vectors of one hundred and twenty consecutive time steps and output a vector with the five following time steps through the output layer (Dense). The main structural differences between the models were in their hidden layers, where each one has specific layers of its type and varies in the amount of them. Figure 5 describes the general structure of the created models.



OUTPUT
$\hat{y}_{t+1}$ $\hat{y}_{t+2}$ $\hat{y}_{t+3}$ $\hat{y}_{t+4}$ $\hat{y}_{t+5}$
DENSE LAYER
$Z_1$ $Z_2$ $Z_3$ $Z_4$ $Z_5$
MODEL
HIDDEN LAYERS (MLP, RNN OR LSTM)
$A_1^1$ $A_2^1$ $A_3^1$ $A_n^1$
INPUT
$x_{t-119}$ $x_{t-118}$ $x_{t-117}$ $x_t$

Fig. 5. General neural network architecture developed for the present work.

Initially, choosing the number of hidden layers and how many cells would compose them were performed empirically, using the results obtained from the cross-validation. To this end, Keras Tuner[1], a library that helps to choose the ideal set of hyperparameters for the neural network, was used to find

[1]https://keras.io/keras_tuner/

a final model close to the optimum for each architecture. The following setup was used:

- MLP:
  - Number of hidden layers between 1 to 6;
  - Neurons from 32 to 256 with 16 increase at each step, and;
  - 5 Neurons in the output layer;
  - Learning rate between 1e-3 and 1e-4;
  - Rectified Linear Unit (ReLU) as hidden layers activation function, and;
  - Linear as output layer activation function.
- Simple RNN:
  - Number of hidden layers between 1 to 6;
  - Neurons from 32 to 256 with 16 increase at each step;
  - 5 Neurons in the output layer;
  - Learning rate between 1e-3 and 1e-4;
  - Hyperbolic tangent (TanH) as hidden layers activation function, and;
  - Linear as output layer activation function.
- LSTM:
  - Number of hidden layers between 1 to 3;
  - Neurons from 32 to 256 with 32 increase at each step;
  - 5 Neurons in the output layer;
  - Learning rate between 1e-3 and 1e-4.
  - Hyperbolic tangent (TanH) as hidden layers activation function, and;
  - Linear as output layer activation function.

As shown, the possibilities for the LSTM were reduced to 1 to 3 hidden layers with neurons from 32 to 256 with 32 increase for each step. This reduction was due to the more significant time required for the tuning process in this case

### D. *Training and Testing*

For training, using the backpropagation through time (BPTT) algorithm, of all ANN, sequences of the previous 120 minutes were used as inputs, and sequences of the next five minutes were used as outputs, both normalized between 0 and 1. In addition, Adam was used as the optimizer as it is well-suited to a wide range of non-convex optimization problems in the field of machine learning [23], the MAE was the loss function, and the batch size was 512. Two stopping criteria were used to prevent overfitting: when training reaches 128 epochs or when the loss function has an improvement smaller than $10^{-9}$ for ten consecutive epochs (Early Stoping).

As stated previously, 8-fold cross-validation was used. Thus, eight training sessions were performed for each model using a separate subset for validation and the rest for training. Afterward, the performance was evaluated on the test set to assess the model's generalization capabilities at predicting unseen data. This approach is a common one since the validation dataset is considered during training for hyperparameter adjustment.

Optimizing hyperparameters is a particularly important problem in Deep Learning due to its tendency to overfitting as a consequence of a high capacity provided by the usage of many hidden layers. Traditional approaches are grid search, random search and to ensure a balanced comparison between different models, we applied a model search based on Bayesian inference which maps the hyperparameters to a probability score on the objective function. Unlike Random Search and Hyperband models [24], Bayesian Optimization keeps track of its past results and uses it to improve the probability model to guide the search for better values.

## IV. Results and Discussions

Several configurations were considered for the MLP, RNN, and LSTM networks, as discussed in the previous section. Thus, given many evaluated models, the best-performing models were chosen of only one hidden layer of each type to compare with the baseline and the models found by the hyperparameter optimizer. The best configuration found empirically for the one-layer model were 120 neurons, 60 and 120 neurons for the MLP, RNN, and LSTM networks, respectively, all with a learning rate of $10^{-3}$. In the case of networks found by Keras Tuner, the final configuration found for MLP contains four hidden layers of 96, 96, 224, and 32 neurons, respectively, with a learning rate of $10^{-4}$. In comparison, the final configuration found for the simple RNN contains six hidden layers of 32, 32, 256, 32, 256, and 32 neurons, respectively, and also with a learning rate of $10^{-4}$. At the same time, for the LSTM, it presented two hidden layers of 128 and 64 neurons, respectively, and a learning rate of $10^{-4}$.

The performance of each model can be observed in Tab. II. One can see that the neural network models are better than the baseline. The optimized MLP, for example, showed an improvement of approximately 71.026% in MAE and 70.423% in RMSE compared to the baseline. Among the neural network models, the results are very similar. The RNN network optimized by Keras Tuner presented the best metric values in the test phase, followed close by the optimized MLP.

Given the proximity of reliability of all ANNs, it is crucial to analyze their usability and complexity. Table III presents the NTP, IF, and MAE for each neural network model. One can see that the optimized simple RNN, despite the best metrics, is the one with the worst inference time and the most significant number of trainable parameters. Much of this is because the configuration of RNN is the deepest of all. On the other hand, the optimized MLP presented the best inference time, about thirteen milliseconds faster than the optimized simple RNN, and with a considerably smaller number of trainable parameters, about 114.816 parameters of difference, with a very close MAE. Thus, it can be said that the optimized MLP was the network model that delivers the best trade-off between performance and simplicity.

About LSTM, we observed a performance similar to the other two, even with its optimization in Keras Tuner having a more limited search space. Since the great advantage of LSTM networks is their ability to learn extended temporal contexts, in our prediction model, which only uses one hundred and twenty minutes elapsed, this advantage may not be well explored.

Then, to ensure that the prediction enhancements that the ANNs models provide are statistically significantly different from the baseline model, t-tests (Welch (1947) [25]) of the MAE were performed using random samples of the test dataset. The result of the optimized MLP and the baseline can be observed in the Tab. IV, and it shows that the p-value is less than 0.05. Consequently, we can deny the hypothesis that the mean of the model's MAE is equal to the baseline. In other words, the forecast results made by the ANN models and the baseline model are significantly different. Furthermore, all the tests from the other models gave out similar results.

In the analysis of the forecasts of each minute, it was noticed that the MAE grows the farther away the predicted minute is, which can be seen in Fig. 6 that shows the evaluative metrics of every minute of the optimized MLP. An MAE increase of 54.56% between the first and fifth minute can be observed, in addition to a notorious increase in the STD as the forecast horizon increases, demonstrating that the error dispersion around the mean grows. Figure 7 confirm this statement by showing the error dispersion graph for the first minute (Fig. 7a) and the error dispersion graph for the fifth minute (Fig. 7b). In these graphs, the horizontal axis represents the real values generated by the solar panels. The vertical axis is the values predicted by the network. The red line represents the ideal case in which the neural network can predict all values without error.

TABLE II
MEAN ABSOLUTE ERROR (MAE), ROOTED MEAN SQUARED ERROR (RMSE), AND STANDARD DEVIATION (STD) MEASURED (WATTS) IN THE TRAINING, VALIDATION, AND TEST PHASE OF EACH MODEL.

| | **MAE** | **RMSE** | **STD** |
|---|---|---|---|
| **Baseline** | 359.972 W | 779.223 W | 779.222 W |
| **One Hidden Layer MLP** | | | |
| Training | 111.054 W | 328.128 W | 327.502 W |
| Validation | 111.489 W | 235.701 W | 234.881 W |
| Test | 106.121 W | 297.771 W | 297.643 W |
| **MLP optimized** | | | |
| Training | 107.659 W | 322.565 W | 321.847 W |
| Validation | 107.305 W | 231.406 W | 230.470 W |
| Test | 104.300 W | 307.345 W | 307.220 W |
| **One hidden layer simple RNN** | | | |
| Training | 116.869 W | 342.489 W | 342.148 W |
| Validation | 116.282 W | 245.317 W | 245.109 W |
| Test | 109.304 W | 303.073 W | 302.565 W |
| **Simple RNN optimized** | | | |
| Training | 113.442 W | 332.453 W | 331.984 W |
| Validation | 112.902 W | 238.474 W | 238.104 W |
| Test | **103.077 W** | **294.561 W** | **294.109 W** |
| **One hidden layer LSTM** | | | |
| Training | 115.469 W | 337.309 W | 336.946 W |
| Validation | 114.908 W | 241.901 W | 241.726 W |
| Test | 108.073 W | 303.225 W | 303.162 W |
| **LSTM optimized** | | | |
| Training | 114.731 W | 331.269 W | 332.046 W |
| Validation | 113.902 W | 241.901 W | 241.726 W |
| Test | 106.809 W | 299.220 W | 299.047 W |

TABLE III
NUMBER OF TRAINABLE PARAMETERS (NTP), INFERENCE TIME (IF), AND MEAN ABSOLUTE ERROR (MAE) OF EACH NEURAL NETWORK MODEL.

| | **NTP** | **IF** | **MAE** |
|---|---|---|---|
| MLP | 15.125 | 20.960 ms | 106.121 W |
| MLP Optimized | 50.021 | **20.887 ms** | 104.300 W |
| RNN | **4.025** | 22.806 ms | 109.304 W |
| RNN Optimized | 164.837 | 33.932 ms | **103.077 W** |
| LSTM | 59.165 | 24.693 ms | 108.073 W |
| LSTM Optimized | 116.293 | 27.970 ms | 106.809 W |

TABLE IV
RESULTS OF T-TEST OF THE MAE FROM THE OPTIMIZED MLP AND THE BASELINE MODEL.

| | **t test** | | | |
|---|---|---|---|---|
| | **t value** | **p value** | **Tail** | **Degree of freedom** |
| value | 85.878 | $\leq 0.05$ | Two-sided | $25,309$ |

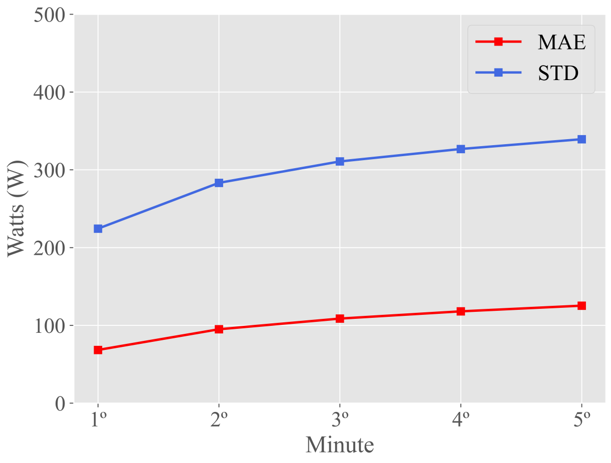


Fig. 6. Mean Absolute Error (MAE) and Standard Deviation (STD) in the test phase of the optimized MLP model as a function of the minute prediction value.

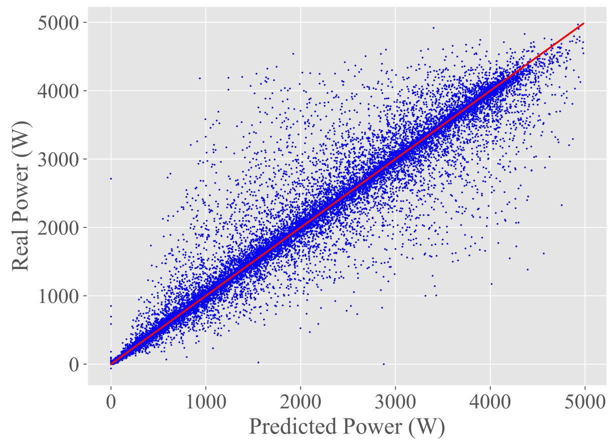
Finally, even with all the reservations made above, it can be said that the models produced good results, presenting a small error if we take into account the power scale of our database, which has values that reach up to seven thousand watts. To illustrate this, we separated the one-day test prediction of the best model, shown in Fig. 8. The series represented by the red line is the actual power values generated by the photovoltaic panels. In contrast, the series represented by the blue dotted line comprises the values predicted by the network. Visually it is possible to see that the network achieved a reliable performance throughout the day, even not tracking perfectly in some points.

## V. Conclusions

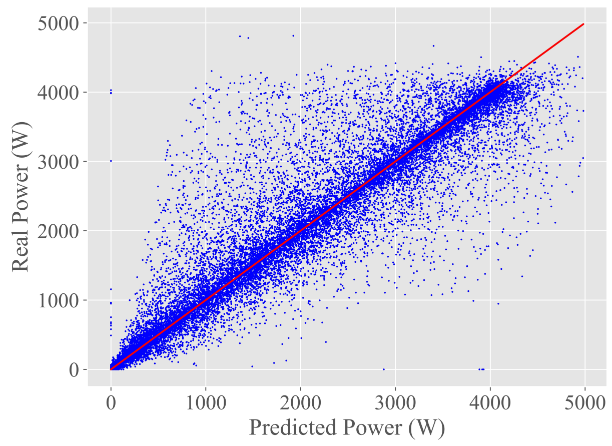
The present work compares MLP, simple RNN, and LSTM models, for the short-term prediction of electric energy generation by a photovoltaic system, through the time series approach. The results presented showed a very close performance among all models, with acceptable mean absolute error values, given the scale of the worked data. Furthermore, the study demonstrated that the models could predict a future five-minute power horizon using the power values of two past hours, even with an increase in error as the forecast horizon increases.

Discussing the error metrics purely, the model that presented the best result was an RNN model, which was expected, given its favorable characteristics for time series. However, the MLP models returned results very close to the best RNN. By considering the usability and complexity of the models, it is possible to observe an advantage of the MLP model to the RNN, as it managed to achieve a shorter inference time and still takes considerably less time to train. In addition, the study made it clear that LSTM was underutilized in the context of short-term forecasting. Thus, it can be concluded that there is no need for very complex architectures for short-term production forecasts, especially in specific contexts where inference time and training time are important, e.g., for embedded systems.

As a continuation of the work, we intend to observe the underutilization of the LSTM, seeking to make predictions using a more extensive temporal context than what was used. In addition, the idea is to apply the forecast models in the solarimetric station and observe how they behave in practice.

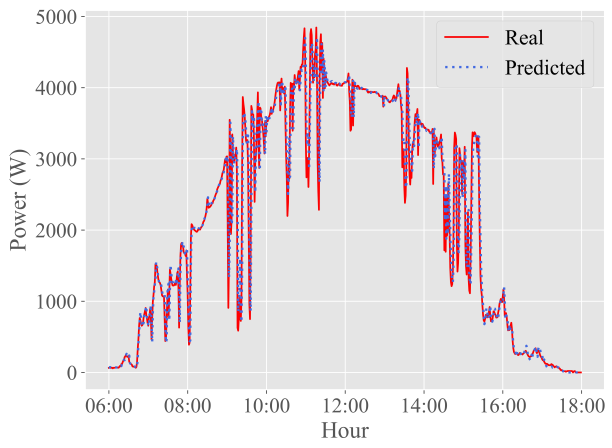


(a) First minute prediction

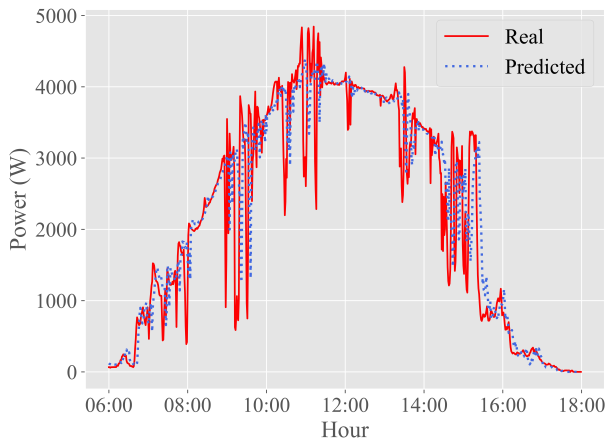


(b) Fifth minute prediction

Fig. 7. Optimized MLP's dispersion graph when prediction (a) one minute ahead and (b) five minutes ahead.



(a) First minute prediction



(b) Fifth minute prediction

Fig. 8. Power generation predicted by the optimized MLP (blue dots) and the real power values (red line) of the day 26/10/20, considering (a) the forecast of the first minute and (b) the forecast of the fifth minute.

## References

[1] Brazilian Ministry of Mines and Energy (MME), "Monthly bulletin on monitoring the brazilian

electrical system," Apr. 2021. [Online]. Available: https://www.gov.br/mme/pt-br/assuntos/secretarias/energia-eletrica/publicacoes/boletim-de-monitoramento-do-sistema-eletrico/2021/boletim-de-monitoramento-do-sistema-eletrico-abr-2021.pdf

[2] W. T. Lim, L. Wang, Y. Wang, and Q. Chang, "Housing price prediction using neural networks," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, Aug. 2016. [Online]. Available: https://doi.org/10.1109/fskd.2016.7603227

[3] H. Xie, F. Ma, and Q. Bai, "Prediction of indoor air quality using artificial neural networks," in *2009 Fifth International Conference on Natural Computation*, vol. 2, 2009, pp. 414–418.

[4] E. R. C. Sabino, "Previsão de radiação solar e temperatura ambiente voltada para auxiliar a operação de usina fotovoltaicas," 2018. [Online]. Available: https://repositorio.ufpe.br/handle/123456789/36837

[5] W. T. Lim, L. Wang, Y. Wang, and Q. Chang, "Housing price prediction using neural networks," in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, Aug. 2016. [Online]. Available: https://doi.org/10.1109/fskd.2016.7603227

[6] Y. Sun, F. Wang, Z. Mi, H. Sun, C. Liu, B. Wang, J. Lu, Z. Zhen, and K. Li, "Short-term prediction model of module temperature for photovoltaic power forcasting based on support vector machine," in *International Conference on Renewable Power Generation (RPG 2015)*, 2015, pp. 1–6.

[7] A. Mellit and A. M. Pavan, "A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected pv plant at trieste, italy," vol. 84, no. 5, 2010, pp. 807–821. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038092X10000782

[8] S. Watetakarn and S. Premrudeepreechacharn, "Forecasting of solar irradiance for solar power plants by artificial neural network," in *2015 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA)*, 2015, pp. 1–5.

[9] H. A. Malki, N. B. Karayiannis, and M. Balasubramanian, "Short-term electric power load forecasting using feedforward neural networks," vol. 21, no. 3, 2004, pp. 157–167. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0394.2004.00272.x

[10] C.-H. Liu, J.-C. Gu, and M.-T. Yang, "A simplified lstm neural networks for one day-ahead solar power forecasting," vol. 9, 2021, pp. 17 174–17 195.

[11] V. De, T. T. Teo, W. L. Woo, and T. Logenthiran, "Photovoltaic power forecasting using lstm on limited dataset," in *2018 IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, 2018, pp. 710–715.

[12] M. Abdel-Nasser and K. Mahmoud, "Accurate photovoltaic power forecasting models using deep lstm-rnn," in *Neural Computing and Applications)*. Springer London, Jul. 2019. [Online]. Available: https://doi.org/10.1007/s00521-017-3225-z

[13] I. Gabriel, G. Gomes, I. Araujo, E. Barboza, T. Vieira, and D. Brito, "PrevisÃo de geraÇÃo fotovoltaica a partir de dados meteorolÓgicos utilizando rede lstm," 11 2020.

[14] F. Harrou, F. Kadri, and Y. Sun, "Forecasting of photovoltaic solar power production using lstm approach," in *Advanced Statistical Modeling, Forecasting, and Fault Detection in Renewable Energy Systems*, F. Harrou and Y. Sun, Eds. Rijeka: IntechOpen, 2020. [Online]. Available: https://doi.org/10.5772/intechopen.91248

[15] J. Zhang, Y. Chi, and L. Xiao, "Solar power generation forecast based on lstm," in *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018, pp. 869–872.

[16] E. İzgi, A. Öztopal, B. Durna, M. Kaymak, and A. Şahin, "Short–mid-term solar power prediction by using artificial neural networks," vol. 86, 02 2012, p. 725–733.

[17] S. Haykin, "Redes neurais: Princípios e prática." Artmed, 2007. [Online]. Available: https://books.google.com.br/books?id=bhMwDwAAQBAJ

[18] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[19] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," vol. 9, no. 8, 11 1997, pp. 1735–1780. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[20] S. Neill and M. R. Hashemi, "Fundamentals of ocean renewable energy: Generating electricity from the sea," 06 2018.

[21] F. Chollet, "keras." GitHub, 2015. [Online]. Available: https://github.com/fchollet/keras

[22] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux, "Api design for machine learning software: Experiences from the scikit-learn project," 09 2013.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 12 2014.

[24] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," vol. 18, 04 2018, pp. 1–52.

[25] B. L. Welch, "The generalization of 'student's' problem when several different population variances are involved," vol. 34, no. 1/2. [Oxford University Press, Biometrika Trust], 1947, pp. 28–35. [Online]. Available: http://www.jstor.org/stable/2332510