Rethinking Solar Forecasting: Why Univariate Transformers Challenge Classical Statistical Models in Multi-Series Data

Beatriz Ribeiro[1], **Didier Vega-Oliveros**[1]

[1]Instituto de Ciência e Tecnologia – Universidade Federal de São Paulo (UNIFESP)

beatriz.cottoboni@gmail.com, didier.vega@unifesp.br

Abstract. *Electricity production must keep up with the growing demand, particularly from renewable sources. However, anticipating generation from renewable sources can be challenging, as they are highly variable and consist of multiple series, one for each plant. To prevent power shortages, prediction techniques can be used. This study compared the performance of statistical techniques that require a model for each series with neural network models that can generalize with a single model. The techniques tested were Seasonal Autoregressive Moving Average (SARIMA), Prophet, Long-Short Term Memory (LSTM), LSTM with Convolutional Neural Network (CNN), and Patch Time Series Transformer (PatchTST). The neural network models included only power generation models, as well as models using solar radiation and temperature. The best models were SARIMA and PatchTST, showing that even a single model can perform as well as neural networks.*

1. Introduction

Electricity consumption continues to grow in line with population growth [Gao et al. 2025]. On the other hand, energy generation depends on resource availability, which is often polluting and non-renewable [Gao et al. 2025]. Greater energy demand raises environmental concerns, such as increased emissions of polluting gases and global warming [Gao et al. 2025]. However, renewable energy generation can be challenging due to significant uncertainty [Walther and Weigold 2021]. Solar power generation depends on sunlight availability at the location where the photovoltaic panels are installed and is affected by climatic conditions, particularly solar radiation, making it variable and uncertain [Monjoly et al. 2017]. Another challenge is the presence of multiple series collected from various sources, which may have different time horizons due to limited data access and logistical and methodological changes [Rick and Berton 2022]. Thus, to ensure energy supply from renewable sources, it is necessary to anticipate generation using data-driven methodologies that capture relationships in the data and make predictions [Gao et al. 2025].

Forecasting techniques for this type of prediction can be either linear or nonlinear. Among linear methods, classical statistical models such as the Autoregressive Integrated Moving Average (ARIMA) are commonly used because of their high accuracy, low processing requirements, and ease of implementation [Hewamalage et al. 2021]. Despite their good performance, linear models are not capable of generalizing with a single model to multiple series [Hewamalage et al. 2021], [Rick and Berton 2022]. Thus, although statistical models perform well, they may not be the most suitable for energy generation forecasting, as it often requires forecasting multiple series with nonlinear behavior [Gao et al. 2025].

Artificial Neural Networks (ANNs) have strong generalization and nonlinear capture capacity [Hewamalage et al. 2021], [Liu et al. 2021], but do not account for temporal dependence in the series. Recurrent Neural Networks (RNNs) and their special case, Long Short-Term Memory (LSTM), introduce temporal dependence between network cells and use the previous state as input [Wu et al. 2020], [Chen et al. 2021].

In addition, RNNs can be combined with Sequence-to-Sequence (Seq-2-Seq), composed of three components: an encoder, an intermediate vector, and a decoder [Wu et al. 2020]. However, this architecture may struggle to handle long sequences, as the data is reduced to a single vector [Wu et al. 2020]. Transformers, on the other hand, add the self-attention mechanism to the encoder-decoder architecture, which can identify the most important relationships and show predictive gains over other architectures [Wu et al. 2020].

As presented, several neural network techniques applied to forecasting improve predictive capacity for nonlinear series without requiring multiple models when multiple series are present. To better understand the performance differences between classical methodologies and neural networks, this work demonstrates their practical application using data on solar power generation and weather conditions in Brazil. The main contributions of this article are:

- To test methodologies for forecasting solar energy generation in Brazil, using both classical forecasting methods and neural networks;
- To evaluate the performance of neural networks in many-to-many series with univariate and multiple variable models.

The remainder of the article is organized as follows: Section II related work, Section III methodologies used, Section IV presents the results obtained, and Section V provides a discussion comparing the results presented in Section IV. Finally, Section VI concludes the article with final considerations.

2. Related Work

As presented, ANNs have a high capacity for self-adaptation and generalization due to their ability to identify relationships between data, especially nonlinear ones [Hewamalage et al. 2021], [Liu et al. 2021]. Studies show that ANNs can outperform ARIMA in time series forecasting [Alameer et al. 2019]. RNNs introduce temporal dependencies between cells to better adapt to sequential data [Wu et al. 2020], [Chen et al. 2021]; however, they can suffer from gradient-vanishing and exploding gradients, which can degrade performance. On the other hand, LSTM introduces a mechanism to forget information without the RNN gradient problem [Wu et al. 2020]. The Gated Recurrent Unit (GRU) architecture, similar to LSTM, has gate components with a simplified architecture [Gao et al. 2025]. GRU architecture, combined with trend decomposition and data augmentation, can capture patterns with limited data and outperforms other models, such as ARIMA [Zhang et al. 2023].

The seq2seq architecture consists of an encoder, an intermediate vector, and a decoder [Wu et al. 2020]. However, because the data are combined into a single vector, some information may be lost. Hence, attention mechanisms were added to the architecture, which use weights to increase or decrease the importance of observations for

each computed point [Hewamalage et al. 2021]. Gao et al. [2025] propose a combination of encoder-decoder architectures with attention mechanisms, using Convolutional Neural Networks (CNN) and GRU to predict energy consumption in buildings [Gao et al. 2025].

Rick and Berton [2022] propose a similar architecture that combines a CNN with an LSTM and an encoder-decoder. The authors use the architecture to predict energy generation and consumption across multiple series of varying sizes. The authors' model predicted both energy consumption and generation and performed similarly to the statistical methods ARIMA and Prophet, which used a distinct model for each series [Rick and Berton 2022]. Succetti et al. [2020] also use architectures that combine LSTM networks with CNNs to predict photovoltaic energy generation and include multivariate models with solar radiation, temperature, and wind speed data in the tests. The authors show that the most accurate model uses multiple LSTM layers and multiple variables [Succetti et al. 2020].

Transformers also use an encoder-decoder architecture, but incorporate a self-attention mechanism that operates on both the encoder and the decoder [Wu et al. 2020]. Wu, Green, and O'Banion [2020] apply the transformer methodology to predict influenza cases in the United States, achieving greater accuracy than ARIMA, LSTM, and seq2seq with attention mechanisms. Nie et al. [2022] propose the Patch Time Series Transformer (PatchTST), which introduces patching and a channel independence structure into the transformer architecture. The proposed model showed superior performance to other transformer architectures across different datasets [Nie et al. 2022].

3. Methodology

For the empirical test, we used the solar energy generation database in Brazil, provided by the National Electric System Operator (ONS)[ONS 2025a, ONS 2025b], along with climate condition data from the National Institute of Meteorology (INMET) [Instituto Nacional de Meteorologia 2025]. The data collected hourly in both databases covers the period from April 2017 to October 2025. To merge the databases, we selected the location closest to each power plant with climate information for a given time. The tests considered only 24 plants, which had the longest data periods and the least amount of missing data. The data were split into training and validation sets until 2024, with 20% of this period for validation, and five months were used for testing, from January to May 2025. The network model tests considered both univariate models and models with climate variables to improve accuracy [Succetti et al. 2020].

For training traditional models, the incremental windowing technique [Montieri et al. 2021] was used, in which, for each round, the forecast is made for the immediately following period, and this period is added to the training base for the next forecast. For neural network models, the sliding window technique was used to train the neural network models [Gao et al. 2025], in which training is performed with fixed windows that move over time, and the predicted value is the one immediately after the window. The data were normalized using the standard normalization function, which sets the feature mean to zero and the standard deviation to 1 [Pedregosa et al. 2011]. We used Mean Squared Error (MSE) [Chen et al. 2021] to optimize the models.

For traditional models, we selected statistical models that do not use neural networks. The selected models were Seasonal Autoregressive Integrated Moving Average

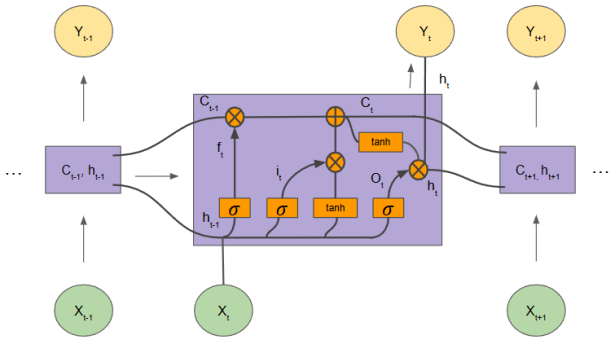
(SARIMA) [Mirboluki et al. 2022] and Prophet [Saeed et al. 2023]. SARIMA is a univariate model based on autoregressive, moving average, differencing, and seasonal components [Mirboluki et al. 2022]. The Python library pmdarima.auto_arima was used to implement the model, which automatically selects the best parameters given the possible ranges.

Prophet is an automated forecasting model that uses an additive model that sums the annual, monthly, and daily nonlinear trends and is robust to outliers and changes in the time series [Saeed et al. 2023]. Prophet is given by Eq. 1, where $y(t)$ is the forecast value, $g(t)$ is the trend component, $s(t)$ is seasonality, $h(t)$ is holidays and events, and ε_t are effects not captured by the model.

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \tag{1}$$

Among the neural network models, LSTM, LSTM-CNN, and PatchTST were chosen. For each model, we conducted three tests: one with the lagged energy generation variable, one with the lagged energy generation and radiation variables, and one with the lagged energy generation, radiation, and temperature variables.

The LSTM architecture used in the tests is a special type of RNN that incorporates gates that control the flow of information between hidden states [Wu et al. 2020]. In Fig. 1 we depict the LSTM cell architecture, where f_t is the forget gate, i_t is the input gate, o_t is the output gate, h_t is the hidden states vector, $\tilde{C}_t$ is the candidate cell in state t, W are the weight vectors, and b are the bias vectors calculated by the neural network, x_t is the input, y_t the output, σ is the sigmoid activation function of the input, and $\tanh$ is the hyperbolic tangent activation function of the output.



$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$
$$h_t = y_t$$

Figure 1. Long Short-Term Memory Architecture. Adapted from [Wu et al. 2020]

CNNs are commonly used for image recognition, but they can also be used to identify patterns in time series [Rick and Berton 2022]. The main components of CNNs are kernels, which perform operations on a window and allow data propagation; strides, which define how much of the window is considered at each step; the pooling layer, which propagates the most relevant data within a given neighborhood; and the connected layer, as shown in Fig. 2.

The LSTM predictions were made using the Python TensorFlow Sequential library. For all predictions, a sequence length of 60 days, a time window step of 1, a

prediction horizon of 1 period, and a batch size of 1 to 10 epochs. The fine-tuning was performed with BayesianOptimization, which is a built-in tuner from the keras_tuner library and uses a Gaussian process for hyperparameter search [O'Malley et al. 2019]. During optimization, the hyperparameters for the number of layers (1 to 3), the number of neurons (16 to 48 with a jump of 16), and the learning rate (0.01 or 0.001) were set for each model (single, two, and three variables) with 5 epochs. For the CNN models, in addition to the same parameters from LSTM tuning, the number of CNN layers, from 1 to 3, the number of CNN units, from 16 to 48, and the kernel size, 5 or 7, were set.

For the univariate prediction with power generation data, a network with 2 layers, 16 LSTM units, 1 dense layer, and a learning rate of 0.001 was used. For the prediction with two variables, a network with 2 layers with 48 LSTM units, 1 dense layer, and a learning rate of 0.001 was selected. For the three-variable test, a network with 1 layer of 32 LSTM units and 1 dense layer, with a learning rate of 0.001, was used. For the univariate test that combined CNN and LSTM, the model had a CNN 1 layer with 48 filters and a kernel of 3, a pooling layer of 2, and an LSTM layer with 16 units, with a dense layer and a learning rate of 0.001. For the tests with radiation and temperature, a CNN layer with 48 filters with a kernel of 7, a pooling layer of 2, and 2 LSTM layers with 48 units, a dense layer, and a learning rate of 0.001 were used.

Patch Time Series Transformer (PatchTST), which uses the Transformer architecture with temporal step aggregation within patch sub-series, has smaller inputs; the attention map is decreased, and the model can retain long-term dependencies [Nie et al. 2022]. The architecture encoder maps the patches to the Transformer latent space with a trainable linear projection and an additive position encoding; then the multi-head attention transforms them into query, key, and value matrices that are used for the output and the predictions.

Fig. 3 shows PatchTST univariate architecture, where d is the dimension, W_p is a trainable linear projection, W_pos is applied to monitor the temporal order of the patches, $x_d^{(i)}$ is the transformer input, h is the multi-head attention head, $Q_h^{(i)}$, $K_h^{(i)}$, and $V_h^{(i)}$ are respectively the query, key, and value matrices of h, and $O_h^{(i)}$ is the output.

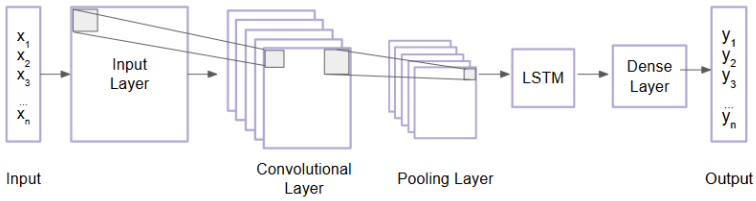
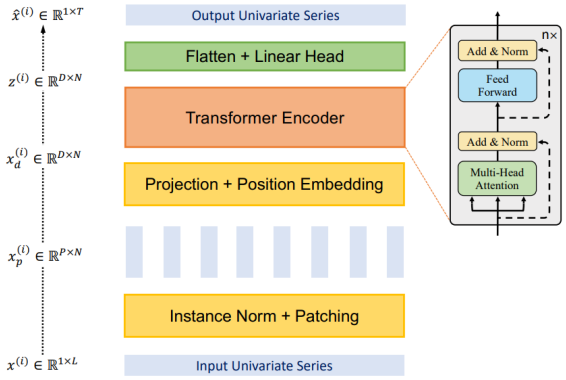


Figure 2. CNN with LSTM Architecture. (Source: adapted from [Rick and Berton 2022])

The tutorial structure provided by Hugging Face [Rasul and et al. 2024] was used to implement the model. The model was optimized with the optuna library, which set the parameters random mask ratio, model dimension, number of attention heads, number of hidden layers, feed-forward dimension, dropout rate, head dropout rate, and learning rate.

The PatchTST models were trained with a sequence length of 60 days, a 1-day



$$
\begin{aligned}
x_d^{(i)} &= W_p x_p^{(i)} + W_{pos} \\
Q_h^{(i)} &= (x_p^{(i)})^T + W_h^Q \\
K_h^{(i)} &= (x_p^{(i)})^T + W_h^K \\
V_h^{(i)} &= (x_p^{(i)})^T + W_h^V \\
(O_h^{(i)})^T &= Attention(Q_h^{(i)}, K_h^{(i)}, V_h^{(i)}) \\
Attention(Q_h^{(i)}, K_h^{(i)}, V_h^{(i)}) &= Softmax\left(\frac{Q_h^{(i)} K_h^{(i)}}{\sqrt{d_k}}\right) V_h^{(i)}
\end{aligned}
$$

Figure 3. PatchTST Univariate Architecture (Source: [Nie et al. 2022])

Table 1. Selected Hyperparameters for each PatchTST Model

Hyperparameter	Model 1	Model 2	Model 3
Random Mask Ratio	0.17	0.39	0.41
Model Dim.	64	64	256
Num. Attention Heads	16	4	4
Num. Hidden Layers	4	3	3
Feed-Foward Dim.	512	512	256
Dropout	0.13	0.25	0.33
Head Dropout	0.13	0.30	0.34
Learning Rate	0.0026	0.0017	0.0008

step between windows, a forecast for 1 subsequent period, 10 epochs, and a batch size equal to 1. Table 1 shows the hyperparameters selected for the models, where Model 1 is the univariate PatchTST model, Model 2 includes power generation and radiation variables, and Model 3 includes power generation, radiation, and temperature variables.

The error metrics used in prior works [Chen et al. 2021, Rick and Berton 2022, Liu et al. 2022] to evaluate the performance of the models are: Symmetric Mean Absolute Percentage Error (sMAPE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared (R^2). The errors are calculated after denormalizing the predictions. The Diebold-Mariano (DM) test is a statistical test described by Eq. 2, where d is the difference in the MSEs of the two methods, $\bar{d}$ is the mean of d, and γ_k is the

autocovariance at lag k. If the $p - value$ is less than 0.05, it is possible to reject the null hypothesis that the predictions are equivalent, with 95% confidence.

$$DM = \frac{\bar{d}}{(Y_0 + 2\sum_{k=1}^{h-1}\gamma_k)/N} \tag{2}$$

The hardware specifications used for training and prediction were Intel(R) Xeon(R) CPU @ 2.20GHz, NVIDIA, CUDA Version 12.4, and Python version 3.12.12.

Fig. 4 summarizes the methodology used in this work. The first step is data collection from ONS and INMET, followed by preprocessing, including filtering for the most relevant power plants, data normalization, and splitting the data into training, testing, and validation. The next step is fine-tuning, training, and forecasting using statistical and neural network models, and the final step is comparing each model's forecast results.

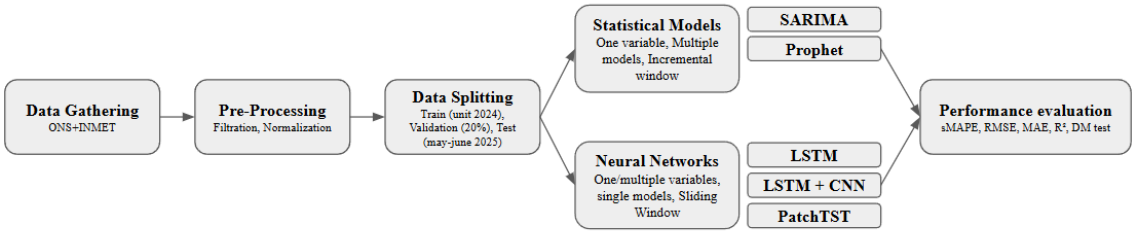


Figure 4. Methodology

4. Results

Fig. 5 shows the evolution of solar energy generation, on average Megawatt (MWMed) units, over time by plant for the Southeast region of Brazil. The generation period differs across plants, likely due to the start date of each plant's operation. It can also be seen that there are cyclical patterns with peaks concentrated at the end of each year and troughs in the middle. Furthermore, the values vary considerably in nearby periods, while for almost all plants they remain within the same range. Fig. 6 shows the evolution of solar power generation in the Northeast, and the series also show different sizes and peaks of power generation at the end of each year. The same behavior of short-term instability and long-term stability is observed.

Table 2 shows the models used in the test, where S1 represents solar power generation, S2 represents solar radiation, S3 represents temperature, and N is the number of plants generating solar power, with N = 24 in the test. The table shows that the statistical models, SARIMA and Prophet, used in the tests were univariate and used a different model for each plant. The table shows that the tests with neural network models were performed with three possible architectures, LSTM, LSTM associated with CNN, and PatchTST, and with three combinations of variables: energy generation, energy generation with solar radiation, and energy generation with solar radiation and temperature. In all neural network tests, we trained a single model across multiple series.

Table 3 shows the consolidated results of the plants by model, RMSE and MAE shows the results on MWMed units. SARIMA has the best result with the lowest values

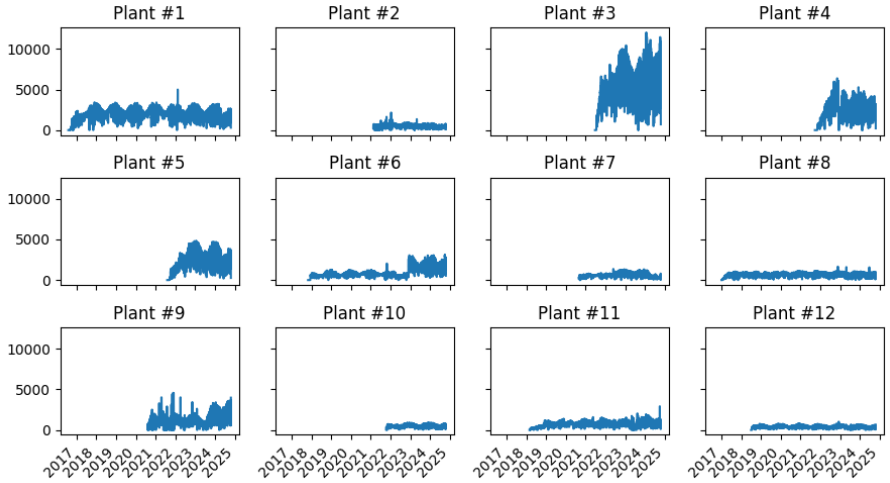


Figure 5. Solar Power Generation by plants in the Brazilian Southeast (MWMEd)

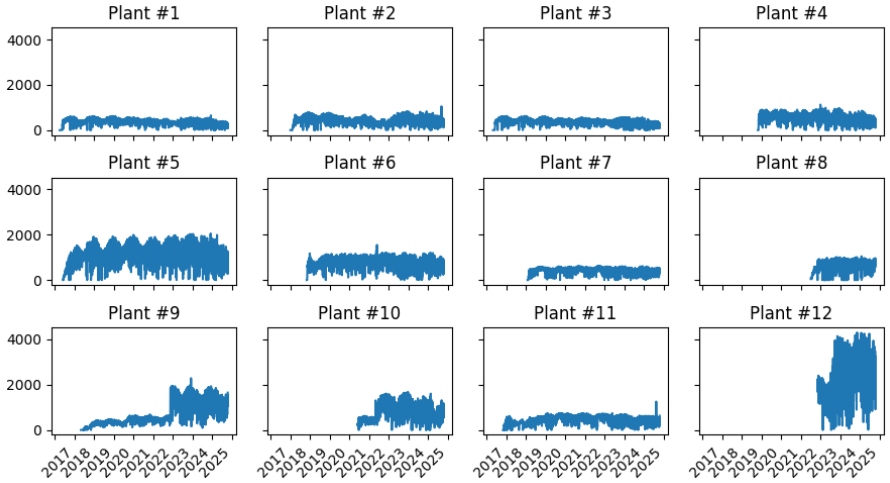


Figure 6. Solar Power Generation by plants in the Brazilian Northeast (MWMEd)

Table 2. Models Details

Model	Variables	Number of Models	Number of Series
SARIMA	S1	N	One Series
Prophet	S1	N	One Series
LSTM	S1	1	Multiple Series
LSTM	S1,S2	1	Multiple Series
LSTM	S1,S2,S3	1	Multiple Series
CNN+LSTM	S1	1	Multiple Series
CNN+LSTM	S1,S2	1	Multiple Series
CNN+LSTM	S1,S2,S3	1	Multiple Series
PatchTST	S1	1	Multiple Series
PatchTST	S1,S2	1	Multiple Series
PatchTST	S1,S2,S3	1	Multiple Series

of sMAPE, 0.35, of RMSE, 704.9, and MAE, 383.4, and the highest R^2 of 0.72, and the second-best result is the univariate PatchTST, which has an sMAPE of 0.37, an RMSE of 727, an MAE of 395, and an R^2 of 0.70, very close to the SARIMA result. The table

Table 3. Results by Model

Model	Variables	sMAPE	RMSE (MWMed)	MAE (MWMed)	R2
SARIMA	**S1**	**0.35**	**704.90**	**383.42**	**0.72**
PatchTST	S1	0.37	727.25	395.05	0.70
LSTM	S1, S2, S3	0.37	751.74	423.45	0.68
LSTM	S1, S2	0.38	750.48	431.50	0.68
Prophet	S1	0.39	775.81	433.50	0.66
LSTM	S1	0.39	755.88	441.14	0.68
CNN+LSTM	S1	0.41	799.87	476.01	0.64
CNN+LSTM	S1, S2, S3	0.45	926.30	520.88	0.52
CNN+LSTM	S1, S2	0.45	946.17	524.85	0.50
PatchTST	S1, S2	0.94	1803.59	1261.58	-0.83
PatchTST	S1, S2, S3	0.94	1681.53	1131.44	-0.59

Table 4. Results by Model - Southeast

Model	Variables	sMAPE	RMSE (MWMed)	MAE (MWMed)	R2
SARIMA	**S1**	**0.32**	**943.26**	**558.21**	**0.65**
PatchTST	S1	0.33	967.18	560.33	0.63
Prophet	S1	0.35	1035.89	613.30	0.58
LSTM	S1, S2, S3	0.36	1000.82	611.30	0.61
LSTM	S1, S2	0.37	993.39	614.01	0.61
LSTM	S1	0.38	999.97	625.99	0.61
CNN+LSTM	S1	0.40	1044.32	664.20	0.57
CNN+LSTM	S1, S2, S3	0.42	1223.49	730.80	0.42
CNN+LSTM	S1, S2	0.45	1273.77	760.82	0.37
PatchTST	S1, S2	1.07	2333.77	1759.25	-1.13
PatchTST	S1, S2, S3	1.07	2141.77	1530.63	-0.79

also shows that the LSTM models performed similarly to each other and were very close to Prophet, with sMAPE around 0.38. The models that include a CNN in the LSTM architecture performed worse than the LSTM-only models, with SMAPEs ranging from 0.41 to 0.45. The PatchTST model with multiple variables performed the worst, with an sMAPE of around 0.94.

Tables 4 and 5 present the forecast results for the Southeast and Northeast regions. For both regions, the best model was SARIMA and, on average, errors are lower in the Northeast, especially for RMSE and MAE, and that R^2 is higher there.

To verify whether there is a statistically significant difference between the predictions, Table 6 presents the SARIMA Diebold-Mariano test results for each model. The univariate PatchTST model showed the highest result for the test, -1.12, and the highest p-value, equal to 0.26. For the Prophet model, the test result was -1.81. The p-value was 0.07; therefore, we do not reject the null hypothesis at the 95% confidence level for either model, and there is no significant difference between the predictions. The LSTM models had p-values of 0.4, 0.3, and 0.2 for the univariate models with 2 and 3 variables, respectively, and the null hypothesis cannot be rejected at the 99% confidence level for all models.

5. Discussion

Table 2 compares the models in terms of flexibility to multiple series. The SARIMA model is less flexible and requires a model for each series in the dataset. This can be computationally costly, since the parameters must be defined at each round, and when they need to be revised, all models must be reviewed [Rick and Berton 2022]. The same

Table 5. Results by Model - Northeast

Model	Variables	sMAPE	RMSE (MWMed)	MAE (MWMed)	R2
SARIMA	**S1**	**0.37**	**322.51**	**208.64**	**0.83**
LSTM	S1, S2, S3	0.39	358.59	235.61	0.78
LSTM	S1, S2	0.40	373.66	248.99	0.77
LSTM	S1	0.41	377.85	256.29	0.76
PatchTST	S1	0.41	349.80	229.78	0.79
Prophet	S1	0.43	361.51	253.70	0.78
CNN+LSTM	S1	0.43	434.73	287.81	0.68
CNN+LSTM	S1, S2	0.45	409.86	288.88	0.72
CNN+LSTM	S1, S2, S3	0.47	468.14	310.96	0.63
PatchTST	S1, S2	0.81	1029.27	763.90	-0.78
PatchTST	S1, S2, S3	0.82	1033.41	732.26	-0.80

Table 6. Diebold-Mariano Test - SARIMA

Model	Variables	DM	p-value
PatchTST	S1	-1.12	0.26
Prophet	S1	-1.81	0.07
LSTM	S1	-2.02	0.04
LSTM	S1, S2	-2.14	0.03
LSTM	S1, S2, S3	-2.40	0.02
CNN+LSTM	S1	-3.01	0.00
CNN+LSTM	S1, S2	-4.17	0.00
CNN+LSTM	S1, S2, S3	-4.37	0.00
PatchTST	S1, S2	-10.94	0.00
PatchTST	S1, S2, S3	-9.03	0.00

is true for Prophet, which cannot aggregate the series into a single model, and when a parameter needs to be revisited, it is done for each series. Neural network models, by contrast, can generalize across multiple series with a single model. The advantage of this format is that when revisiting the parameters, only one model needs to be adjusted. SARIMA and Prophet also benefit from using an incremental window, which adds one additional piece of information at each iteration, whereas neural network models use a sliding window, which maintains the same window size for each iteration.

Even with these disadvantages compared to SARIMA and Prophet, the PatchTST model performed comparably, with no statistically significant difference as seen in Table 6. This indicates that the model, even with multiple series and a sliding window, can predict like statistical models.

6. Conclusion

As presented, energy generation must keep pace with growing demand, preferably using renewable sources to avoid pollution and global warming [Gao et al. 2025]. These sources, however, are highly variable and may consist of multiple series of different sizes [Monjoly et al. 2017], [Rick and Berton 2022]. Thus, to anticipate energy generation and avoid shortages, this study used both statistical and neural network models to predict solar power generation.

Statistical models require one model per series, requiring parameter definition, training, and forecasting for multiple models, one for each solar panel [Rick and Berton 2022]. Neural network models, on the other hand, can generalize across all series using a single model, thereby reducing the number of models and parameters to

be defined and trained [Rick and Berton 2022].

The statistical models used were SARIMA and Prophet, univariate and neural network models used were LSTM, LSTM with a CNN, and PatchTST, with univariate and multivariate tests. The results show that SARIMA was the best model, with the lowest sMAPE. However, the univariate PatchTST model performed just as well as SARIMA, since the Diebold-Mariano test did not show statistical differences between them. The LSTM models performed as well as Prophet, while the CNN models and the multivariate PatchTST performed the worst. These results demonstrate PatchTST's highly competitive performance for solar forecasting, as it achieves results on par with SARIMA, even when using a single model.

References

Alameer, Z., Ye, H., Jianhua, Z., Elaziz, M. A., and Ewees, A. A. (2019). Forecasting gold price fluctuations using improved multilayer perceptron neural network and whale optimization algorithm. *Resources Policy*, 61:250–260.

Chen, X., Li, Y., Ye, X., Xiong, X., Zhang, Y., and Zhang, F. (2021). A novel hybrid model based on an improved seagull optimization algorithm for short-term wind speed forecasting. *Processes*, 9(2):121.

Gao, Z., Zhang, X., Xing, J., Deng, Z., Chen, T., and Wang, C. (2025). Research on building energy consumption prediction based on hybrid gru neural network. *Arabian Journal for Science and Engineering*.

Hewamalage, H., Bergmeir, C., and Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427.

Instituto Nacional de Meteorologia (2025). Dados históricos anuais das estações meteorológicas automáticas. `https://portal.inmet.gov.br/dadoshistoricos`. Conjunto de dados.

Liu, M., He, J., Huang, Y., Tang, T., Hu, J., and Xiao, X. (2022). Algal bloom forecasting with time-frequency analysis: A hybrid deep learning approach. *Water Research*, 219.

Liu, Z., Zhu, Z., Gao, J., and Xu, C. (2021). Forecast methods for time series data: A survey. *IEEE Access*, 9:91896–91912.

Mirboluki, A., Mehraein, M., and Kisi, O. (2022). Improving accuracy of neuro fuzzy and support vector regression for drought modelling using grey wolf optimization. *Hydrological Sciences Journal*, 67(10):1582–1597.

Monjoly, S., André, M., Calif, R., and Soubdhan, T. (2017). Hourly forecasting of global solar radiation based on multiscale decomposition methods: A hybrid approach. *Energy*, 119:288–298.

Montieri, A., Bovenzi, G., Aceto, G., Ciuonzo, D., Persico, V., and Pescapé, A. (2021). Packet-level prediction of mobile-app traffic using multitask deep learning. *Computer Networks*, 200.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. (2022). A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., et al. (2019). Kerastuner. https://github.com/keras-team/keras-tuner.

ONS (2025a). Fator de capacidade de geração eólica e solar. https://dados.ons.org.br/dataset/geracao-usina-2. Conjunto de dados do Operador Nacional do Sistema Elétrico, BR.

ONS (2025b). Geração por usina em base horária. https://dados.ons.org.br/dataset/geracao-usina-2. Conjunto de dados do Operador Nacional do Sistema Elétrico, BR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, M., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Rasul, K. and et al. (2024). Patch time series transformer in hugging face. GitHub.

Rick, R. and Berton, L. (2022). Energy forecasting model based on cnn-lstm-ae for many time series with unequal lengths. *Engineering Applications of Artificial Intelligence*, 113:104998.

Saeed, N., Nguyen, S., Cullinane, K., Gekara, V., and Chhetri, P. (2023). Forecasting container freight rates using the prophet forecasting method. *Transport Policy*, 133:86–107.

Succetti, F., Rosato, A., Araneo, R., and Panella, M. (2020). Deep neural networks for multivariate prediction of photovoltaic power time series. *IEEE Access*, 8:211490–211505.

Walther, J. and Weigold, M. (2021). A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry. *Energies*, 14(4):968.

Wu, N., Green, B., Ben, X., and O'Banion, S. (2020). Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint*. arXiv:2001.08317.

Zhang, Y.-L., Bai, Y.-T., Jin, X.-B., Su, T.-L., Kong, J.-L., and Zheng, W.-Z. (2023). Deep fusion prediction method for nonstationary time series based on feature augmentation and extraction. *Applied Sciences*, 13(8):5088.