International Journal of Advanced Engineering Research and Science (IJAERS)
Peer-Reviewed Journal
ISSN: 2349-6495(P) | 2456-1908(O)
Vol-9, Issue-8; Aug, 2022
Journal Home Page Available: https://ijaers.com/
Article DOI: https://dx.doi.org/10.22161/ijaers.98.32

Hybrid Artificial Neural Networks for Electricity Consumption Prediction

Ricardo Augusto Manfredini

IFRS - Instituto Federal de Educação, Ciências e Tecnologia do Rio Grande do Sul – Campus Farroupilha, Brazil
Email: ricardo.manfredini@frarroupilha.ifrs.edu.br

Received: 26 Jun 2022,

Received in revised form: 14 Jul 2022,

Accepted: 22 July 2022,

Available online: 19 Aug 2022

©2022 The Author(s). Published by AI Publication. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

Keywords*— *Artificial Neural Network, Artifial Inteligence, eletricity consumption predictions, time series.

***Abstract*—** *We present a comparative study of electricity consumption predictions using the SARIMAX method (Seasonal Auto Regressive Moving Average eXogenous variables), the HyFis2 model (Hybrid Neural Fuzzy Inference System) and the LSTNetA model (Long and Short Time series Network Adapted), a hybrid neural network containing GRU (Gated Recurrent Unit), CNN (Convolutional Neural Network) and dense layers, specially adapted for this case study. The comparative experimental study developed showed a superior result for the LSTNetA model with consumption predictions much closer to the real consumption. The LSTNetA model in the case study had a rmse (root mean squared error) of 198.44, the HyFis2 model 602.71 and the SARIMAX method 604.58.*

I. INTRODUCTION

In recent decades, the world population is increasing rapidly and, due to this increase, the global energy demanded and consumed is also growing more and more [6]. Concerning , residential or commercial buildings, are identified as major energy consumers worldwide, accounting for about 30% of the global electricity demand related to energy consumption in the residential sector [7]. Buildings are responsible for a significant share of energy waste as well. Energy waste and climate change represent a challenge for sustainability, and it is crucial to make buildings more efficient [11]. Therefore, the development and use of clean products and renewable energy in buildings have gained wide interest [6]. In the residential and commercial sectors, photovoltaic (PV) systems are the most common distributed generation, minimizing demand dependence on traditional power plants and maximizing household self-sufficiency [8].

Due to PV's dependence on weather conditions, the intermittent nature of the power generated brings some uncertainty [24]. Similarly, the electricity consumption of these buildings also has inherent uncertainties due to seasonality. The easiest way to manage the risk of solar power and harness this power is to forecast the amount of power to be generated [15] as well as the consumption. A reliable forecast is key for various smart grid applications such as dispatch, active demand response, grid regulation and smart energy management [12].

The energy consumption of a building and the PV generation can be represented by a time series with trends and seasonality [14]. There are numerous prediction studies on time series, from classical linear regressions to more recent works using machine learning algorithms, which are powerful tools in predicting electricity consumption and PV generation [21]. Recently, many PV power forecasting techniques have been developed, but there is still no complete unit versal forecasting model and methodology to ensure the accuracy of predictions. Concerning this, Artificial Neural Networks (ANNs) are very popular machine learning algorithms for object prediction and classification and are based on the classical *feed-forward* neural network approach [23]. ANNs are computing systems inspired by the biological neural

networks of the brain, how neurons work, pass and store information [13; 24].

Due to the accelerated development of computing technology, ANN has provided a powerful framework for supervised learning [5]. Deep learning allows models composed of multiple layers to learn data representations [11]. Deep Neural Networks (DNN[1]) are inspired by the structure of mammalian visual systems and they are also an important machine learning tool that has been widely used in many fields [25]. DNN employs an architecture of multiple layers of neurons in an ANN and can represent functions with higher complexity [5].

This work aimed at predicting the electricity consumption of a commercial building using ANN in its various architectures. Several ANN architectures were used and tested and a hybrid architecture (Dense, Convolutional and Recurrent), originally described by Lai, G. et al. [4] and adapted for this case study, was selected.

II. FOUNDATION

2.1 Time Series

Time series are sets of observations ordered in time [14]. A temporal series can be defined as a class of phenomena whose observational process and consequent numerical quantification generate a sequence of observations distributed over time.

Electricity consumption histories over time are univalued time series [20] with trends, cycles, seasonality and randomness. Trends are long-term characteristics related to a time interval. Cycles are long-term oscillations, more or less regular, around a trend line or curve. Seasonalities are regular patterns observed from time to time. Finally, randomness is effects that occur randomly and that cannot be captured by cycles, trends and seasonalities.

Thus, the time series prediction models most used in the literature are those of linear and polynomial regressions. Among the regression models, we can mention the SARIMAX method [19]. This statistical model is a variant of the autoregressive moving average model (ARMA), adding derivations to make the model stationary (I), adding seasonality (S) and finally adding the effect of eXogenous (X) or random variables over time. In this work, the SARIMAX model was used as a baseline to compare its results, its application to the test case and the results obtained from other prediction models.

2.2 Convolutional Artificial Neural Networks

Convective Artificial Neural Networks (CNN[2]) are a type of DNN that is commonly applied to analyse images. One of the main attributes of CNN is to drive different processing layers that generate an effective representation of the features of image edges. The architecture of CNN allows multiple layers of these processing units to be stacked, this deep learning model can emphasize the relevance of features at different scales [24].

Fig. 1 demonstrates a typical architecture of a CNN, composed of at least, a convolution layer, a *pooling layer*, a *flattening layer* and dense layers.

1 DNN - *Deep Neural Network*

2 CNN - *Convolutional Neural Network*

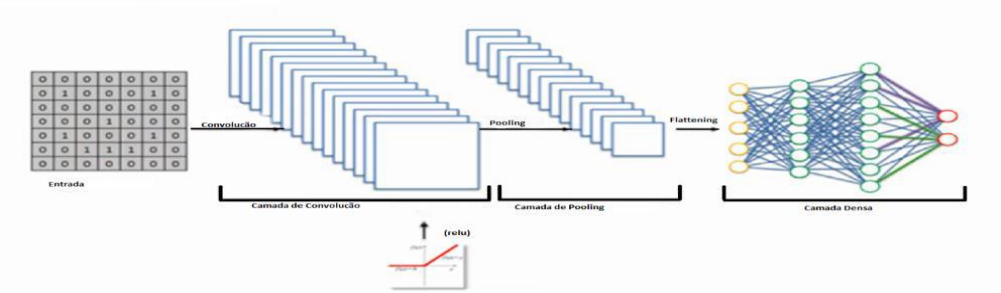

Fig. 1. Basic CNN.

Source: The author

In the convolution layer, a filter (*kernel*, which is also a matrix) is applied to the input matrix aiming at its reduction while maintaining its most important characteristics. Fig. 2 represents, step by step, the application of the convolution function where g(x,y) represents the element of the convolution matrix, that is the matrix product of the matrix colored in Fig. 2 by the *kernel*, at each step it shifts one position to the right until the last column of the input matrix after it shifts one line down and continues the process until it runs through the whole input matrix. In the example of Fig. 2, a 7X7 input matrix was reduced to a 5X5 convolution matrix. The

whole process represented in Fig. 2 is repeated for each of the *kernels* used, generating several convolution matrices.

$$g(x,y) = \omega \ast f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} \omega(dx,dy) f(x+dx, y+dy)$$

For the *pooling* layer, it is usual to apply the activation function *relu* $f(x) = max(0, x)$ for example, generating a new reduced matrix as shown in Fig. 3.

Finally, the *flattening* layer is nothing more than transforming the matrices of the *pooling* layers into vectors, which will be the inputs of the dense layer.

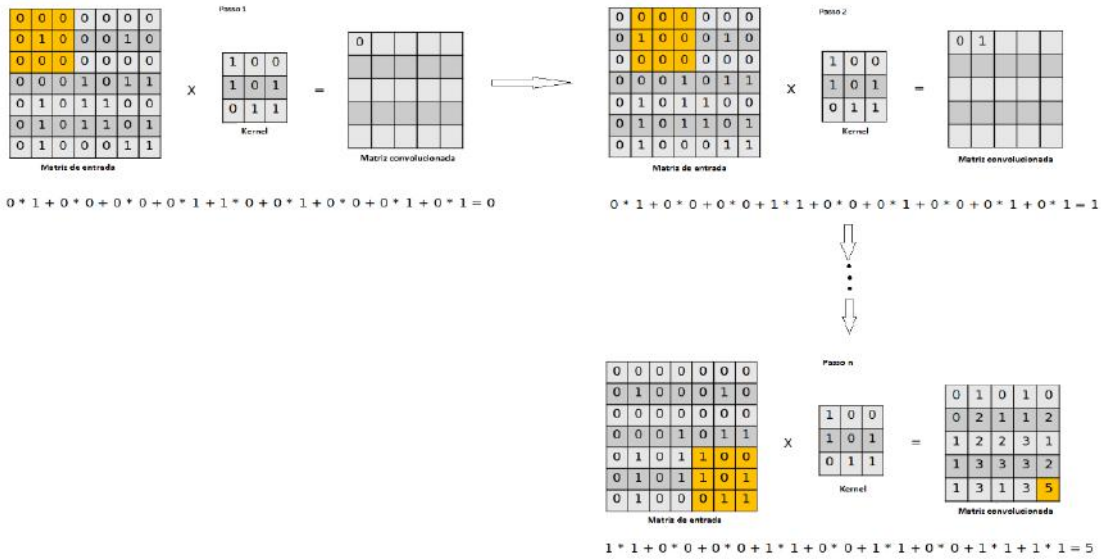

Fig. 2: Convolution process

Source: The author.

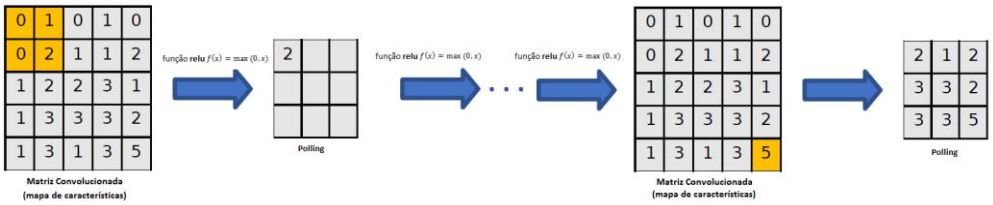

Fig. 3. Pooling Process.

Source the Author.

2.3 Recurrent Artificial Neural Networks

In traditional ANNs, the inputs (and outputs) are independent of each other, making it difficult to use them, for example, in natural language processing where a word in a sentence depends on previous words in the same sentence, or in time series where we need to know the values over time for better projections.

In contrast, recurrent artificial neural networks (RNN[3]) [8] store their previous state and also use it as input to the current state for calculations of new outputs. Another way of thinking about RNNs is that they have a "memory" that captures information about what has been calculated so far. In theory, RNNs can make use of information in arbitrarily long sequences, but in practice, they are limited to looking back only a few steps. Fig. 4 is a typical representation of an RNN.

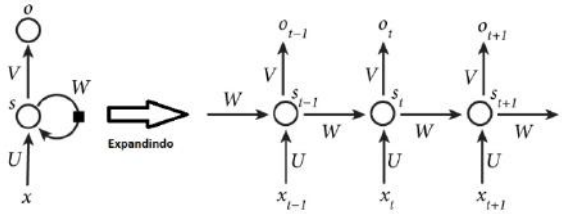

Fig. 4: Basic RNN.

Source: The Author.

3 RNN - *Recurrent Neural Network*

Fig. 4 shows an RNN being expanded into a complete network. Where x_t is the input in time step t. For example, x_1 could be a *one-hot* vector corresponding to the second word of a sentence, s_t is the hidden state in the time step t. It is the "memory" of the network. s_t Is calculated based on the previous hidden state and the input in the current time step: $s_t = f(U x_t + W s_{t+1})$. The function f is usually a nonlinearity, such as $\tanh$ or relu. S_{-1}, which is needed to compute the first hidden state, is usually initialized with zeros. o_t is the output in step t. For example, if we wanted to predict the next word in a sentence, it would be a probability vector in our vocabulary. $o_t = softmax(V s_t)$. By expanding, we simply mean that we write the network for the complete sequence. For example, if the sequence we are interested in is a 5-word sentence, the network would be unfolded into a 5-layer neural network, one layer for each word.

III. MATERIAL and METHODS

This work was carried out at the *Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development* (GECAD[4]), a research centre located at the Instituto Superior de Engenharia do Porto of the Instituto Politécnico do Porto ISEP/IPP, Porto, Portugal. Similarly to the HyFIS2 model (Josi et al.; 2016), the posited model uses the actual electrical consumption data of sectors of Building N of ISEP/IPP where GECAD is located. The building has five energy meters that store the electrical energy consumption data of specific sectors of the building, with a time interval of 10 seconds. This information, as well as meteorological data, are stored in a SQL server automatically, through agents developed in Java.

To validate the model described below, tests were performed using the same consumption data applied to the SARIMAX model and HyFIS2. The N Building laboratories sector was not computed as it has a large variation in consumption due to the experiments conducted there, which generate many *outliers* in the consumption history. For the experiment tests, it was performed an hourly average of the consumption stored every ten seconds, due to the need of predicting the next hour of consumption.

3.1 The Long and Short Time series Network Adapted (LSTNetA) Model

The model developed for energy consumption prediction was based on the model proposed by Lai [4], represented in Fig. 4, which consists of a hybrid ANN with three distinct layers, initially has a convolutional layer for the extraction of short-term patterns of the time series, has as input the time series, the output of this layer is the input of the recurrent layer that memorizes historical information of the time series, which in turn its output is the input of the highly connected dense layer. Finally, the output of the highly connected layer is combined with the output of the autoregressive linear regression (ARMA) [26] ensuring that the output will have the same scale as the input, thus composing the prediction.

4 http://www.gecad.isep.ipp.pt/GECAD/Pages/Pubs/PublicationsPES.aspx

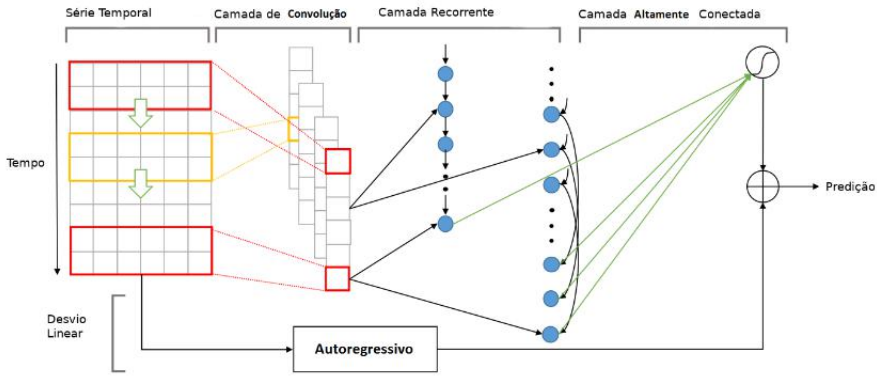

Fig. 5. Architecture of the LSTNetA model.

Source: adapted from Lai [4].

Fig. 6 summarizes the implementation of the LSTNetA network. The convolution layer is represented by the **Conv2D class**, the recurrent layer is represented by the **GRU** classes, the dense layer is represented by the **Dense** classes, the auto-regression is represented in the **PostARTrans class.**

It is important to note that the recurrent layer uses one of the RNN variants the GRU (*Gated Recurrent Unit*)

[1], this ANN model as well as the LSTM (*Long Short-Term Memory*) aims to solve the problem of short-term memory of RNNs that, in long series, have difficulty transporting the results of previous steps to the later ones.

This may be caused by multiline strings or comments not indented at the same level as the code.
Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 24, 24)]	0	
reshape (Reshape)	(None, 24, 24, 1)	0	input_1[0][0]
conv2d (Conv2D)	(None, 19, 1, 100)	14500	reshape[0][0]
dropout (Dropout)	(None, 19, 1, 100)	0	conv2d[0][0]
reshape_1 (Reshape)	(None, 19, 100)	0	dropout[0][0]
pre_skip_trans (PreSkipTrans)	(None, 1, 100)	0	reshape_1[0][0]
gru (GRU)	[(None, 100), (None, 60600		reshape_1[0][0]
gru_1 (GRU)	[(None, 5), (None, 5 1605		pre_skip_trans[0][0]
dropout_1 (Dropout)	(None, 100)	0	gru[0][1]
post_skip_trans (PostSkipTrans)	(None, 95)	0	gru_1[0][1] input_1[0][0]
pre_ar_trans (PreARTrans)	(None, 24)	0	input_1[0][0]
concatenate (Concatenate)	(None, 195)	0	dropout_1[0][0] post_skip_trans[0][0]
flatten_1 (Flatten)	(None, 24)	0	pre_ar_trans[0][0]
flatten (Flatten)	(None, 195)	0	concatenate[0][0]
dense_1 (Dense)	(None, 1)	25	flatten_1[0][0]
dense (Dense)	(None, 24)	4704	flatten[0][0]
post_ar_trans (PostARTrans)	(None, 24)	0	dense_1[0][0] input_1[0][0]
add (Add)	(None, 24)	0	dense[0][0] post_ar_trans[0][0]

Total params: 81,434
Trainable params: 81,434
Non-trainable params: 0

Fig. 6: Summary of the LSTNet implementation.

Source: The Author.

In the *backpropagation* stage, the learning process of ANNs, the RNNs suffer from the problem of gradient dissipation (*The Vanishing Gradient Problem*). Gradients are values used to update the weights of neural networks. The vanishing gradient problem is when the weights propagated during network training are multiplied by values smaller than 1 for each network layer passed through, arriving at the initial network layers with tiny values. This causes the adjustment of weights, calculated at each iteration of net training, to be too small, and makes net training more expensive.

Thus, in RNNs the layers that receive a small gradient update stop learning, with this the RNNs can forget what was seen in longer sequences, thus having a short-term memory.

Fig. 7 shows a typical architecture of a GRU. Basically what makes it different from a standard RNN are the *reset gate* and *update gate*, which by applying the *Sigmoid* and *tanh* activation functions, it is defined whether the previous output h_{t-1} will be considered or discarded for the calculation of the new output.

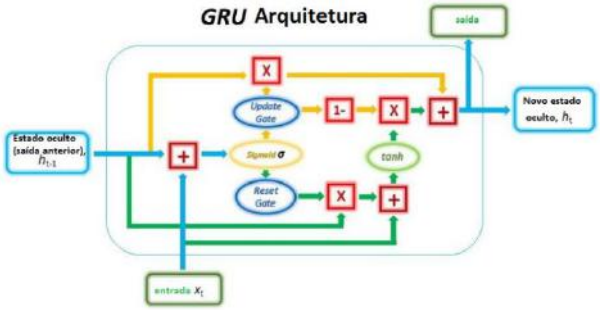

Fig. 7. Typical architecture of a GRU.

Source: The Author

The LSTNetA model was developed in the Python programming language version 3.7 [17] using the machine learning library, developed by Google, TensorFlow version 2.0 [22].

IV. RELATED WORKS

Fig. 8, represents the power consumption time series used by the SARIMAX model to train and test the LSTNetA model and HyFIS2. The top graph represents the historical series of consumption in *watts*, which starts at zero hours on 08/04/2019 to eight hours on 20/12/2019. The middle graph shows the calculated trend of the series and the bottom graph its seasonality.

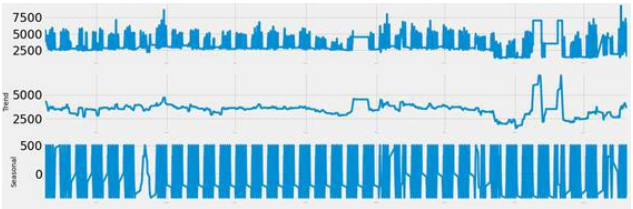

Fig. 8. Historical series of consumption.

Source: The Author.

4.1 SARIMAX

As seen previously, the SARIMAX method is a statistical method of time series analysis, enabling the prediction through linear regressions. Thus, it cannot be characterized as a machine learning algorithm. In the scope of this work, it was applied to obtain prediction data of a widely used model, obtaining results for comparison with the proposed model and with the HyFIS2 model.

To verify the accuracy of all models covered in this work, the last 120 records corresponding to five days of consumption were used for comparison between real and predicted consumption, shown in Fig. 9. To calculate the error used to verify the results of this work, in all models, the *root mean square error* (RMSE - described in

chapter 01) was used, shown in Fig. 10. The application of this model resulted in an average RMSE of 604.72 that was considered as accuracy of this model, in this work.

Fig. 9. Comparison Real Consumption X Sarimax.

Source: The Author.

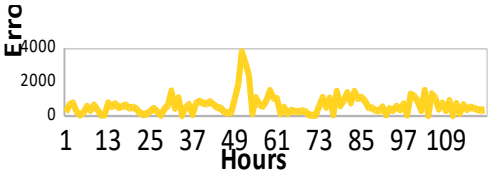

Fig. 10. Verified errors of the SARIMAX method.

Source: The author.

4.2 Model HyFIS2

The HyFIS2 (*Hybrid neural Fuzzy Inference System*) model uses a hybrid approach with the combination of dense ANN and *fuzzy* logic. The system includes five layers, as shown in Fig. 11. In the first layer, the nodes are the inputs that transmit signals to the next layer. In the second and fourth layers, the nodes act as membership functions to express the input-output fuzzy linguistic variables. In these layers, the *fuzzy* sets defined for the input-output variables are represented as: large (L), medium (M) and small (S). However, for some applications, these can be more specific and represented as, for example, large positive (LP), small positive (SP), zero (ZE), small negative (SN) and large negative (LN). In the third layer, each node is a rule node and represents a fuzzy rule. The connection weights between the third and the fourth layer represent certainty factors of the associated rules, i.e., each rule is activated and controlled by the weight values. Finally, the fifth layer contains the node that represents the output of the system.

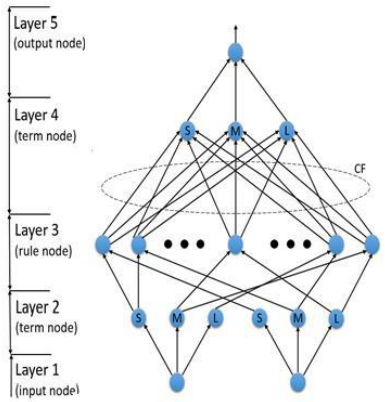

Fig. 11. Neuro-Fuzzy structure of the HyFIS2 model.

Source: Jozi [9]

For prediction of electricity consumption, as in all models tested, the last 120 historical records were used, corresponding to five days of consumption. The comparison between real and predicted consumption is shown in Fig. 12. Fig. 13 shows the RMSE errors calculated. The application of this model resulted in an average RMSE of 602.71 which was considered the accuracy of this model, in this work.

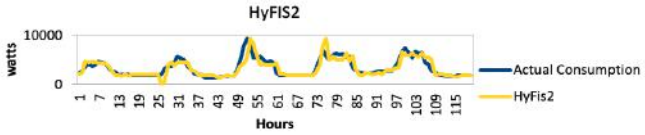

Fig. 12. Real Consumption Comparison X HyFis2.

Source: The Author.

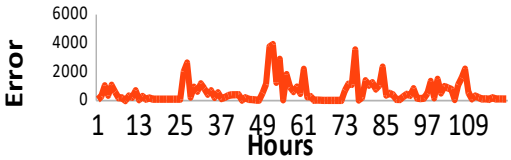

Fig. 13. Verified errors of the HyFIS2 model.

Source: The Author.

V. APPLICATION OF THE LSTNETA MODEL

The training of the LSTNetA ANN was performed as previously described, using the data of the real electricity consumption of the N building of the ISEP/IPP where GECAD is located, except for the laboratory sector. The historical series analyzed was from zero hours on 08/04/2019 to eight hours on 20/12/2019, with measurements every ten seconds, totaled every hour, resulting in 4186 records, containing time and consumption. The training was performed with a learning

rate of 0.0003, using the Adam [10] stochastic method of gradient descent optimization for updating the weights in the *backpropagation* process. For the initial weights of the ANN, the algorithm *VarianceScaling* [3] was used, which generates initial weights with values on the same scale as the inputs. The convolution kernel used was a 6x6 identity matrix and a training loop with 1000 epochs was performed. All these parameters were obtained experimentally and the ones with the best results were selected.

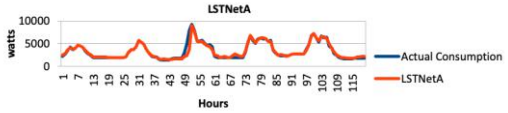

Fig. 14. Comparison Real Consumption X LSTNetA.

Source: The Author.

For the prediction of electricity consumption, as in all models tested, the last 120 historical records were used, corresponding to five days of consumption. The comparison between real and predicted consumption is shown in Fig. 14. Fig. 15 shows the RMSE errors calculated. The application of this model resulted in an average RMSE of 198.44 which was considered the accuracy of this model, in this work.

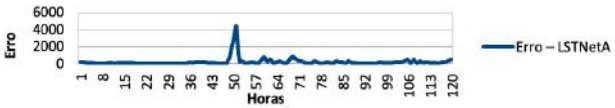

Fig. 15. Verified errors of the LSTNetA model.

Source: The Author.

VI. RESULTS AND CONCLUSION

Table 1 shows a fragment of the results of the three models, the *Date and Time column*, the *Actual column* showing the actual electricity consumption in *watts* at that date and time, the *LSTNetA column* the prediction of this model at that date and time, the *Error - LSTNetA column the* absolute error of this model in the prediction, the column *HyFIS2 the prediction of this model at date* and time, the column Error - *HyFIS2 the absolute error* of this model in the prediction, finally the columns *SARIMAX* and *Error - SARIMAX*, representing the prediction and absolute error, respectively, in the SARIMAX model.

Comparing the results of the SARIMAX, HyFIS2 and LSTNetA models, it can be observed, as shown in Fig. 16, that the LSTNetA method, with the data used for testing, was the one that presented the closest predictions of the real consumption of electricity, where the red line, which represents the predictions of the LSTNetA model, in most of the period overlapped the blue line that represents the real consumption. This demonstrates a prediction very close to the real consumption value, with low errors.

Table 1. Fragment of Predictions and Errors of the 3 models

Date and Time	Actual Consumption	LSTNetA	Error - LSTNetA	HyFis2	Error - HyFis2	SARIMAX	Error - SARIMAX
19/12/2019 09:00	4759,38	4824,27	64,8900	3427,13	1332,2500	4721,76	37,6190
19/12/2019 10:00	6781,51	6685,28	96,2346	6583,38	198,1300	5516,26	1265,2476
19/12/2019 11:00	7279,1	7194,26	84,8373	5798,56	1480,5400	6124,20	1154,8976
19/12/2019 12:00	6332,88	6247,08	85,8038	5798,38	534,5000	5497,10	835,7849
19/12/2019 13:00	5350,34	5569,95	219,6063	6322,98	972,6400	5653,27	302,9276
19/12/2019 14:00	6677,56	6499,50	178,0639	5798,37	879,1900	5197,56	1479,9983

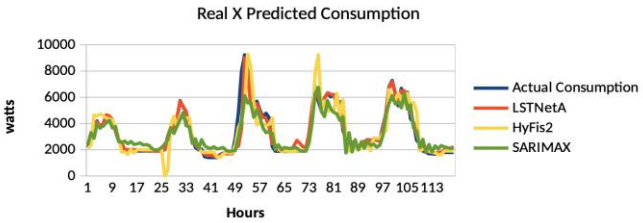

Fig. 16. Comparison of Real Consumption X Prediction Models.

Source: The Author.

Fig. 17 represents the errors (RSME) of the three models, allowing a comparison of the assertiveness of the predictions of each method and also concluding that the LSTNetA method presented a better efficiency in its predictions in comparison to the SARIMAX and HyFIS2 methods. This statement can be corroborated with the data presented in Table 2, where the total average error of the LSTNetA model is significantly lower than the other models.

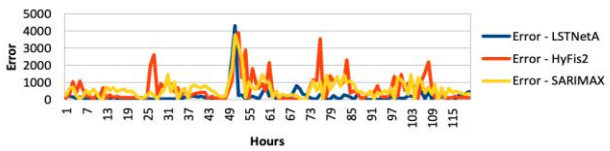

Fig. 17. Comparisons of errors verified in all models.
Source: The Author.

Table 2. RSME of the 3 Models Tested

	Error - LSTNetA	Error - HyFis2	Error - SARIMAX
RSME	198,4496	602,7109	604,5810

REFERENCES

[1] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural network on sequence modeling. in NIPS 2014 Workshop on Deep Learning, December 2014.

[2] Das, U.K., Tey, K.S., Seyedmahmoudian, M., Mekhilef, S., Idris, M.Y.I., Van Deventer, W., Horan, B., and Stojcevski, A. (2018). Forecasting of photovoltaic power generation and model optimization: A review. Renewable and Sustainable Energy Reviews, 81, 912-928.

[3] He, K. at al. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.

[4] Lai, G., Chang, W.C., Yang, Y., and Liu, H. (2018). Modeling long- and short-term temporal patterns with deep neural networks. 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2018, 95-104.

[5] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F.E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234, 11-26.

[6] Lo Brano, V., Ciulla, G., and Di Falco, M. (2014). Artificial neural networks to predict the power output of a PV panel. International Journal of Photoenergy, 2014.

[7] Lusis, P., Khalilpour, K.R., Andrew, L., and Liebman, A.(2017). Short-term residential load forecasting: Impact of calendar effects and forecast granularity. Applied Energy, 205, 654-669.

[8] Hammer, Barbara (2007). Learning with Recurrent Neural Networks. London: Springer.

[9] Jozi, A. et al. (2016). Energy Consumption Forecasting based on Hybrid Neural Fuzzy Inference System. 2016 IEEE Symposium Series on Computational Intelligence

[10] Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. Computer Science, Mathematics ICLR 2015 (SSCI). 6-9 Dec. 2016R. Diponsable at: https:// https://arxiv.org/abs/1412.6980.pdf. Accessed on: 01/03/2021.

[11] Marino, D.L., Amarasinghe, K., and Manic, M. (2016). Building energy load forecasting using Deep Neural Networks. IECON Proceedings (Industrial Electronics Conference), 7046-7051.

[12] Massucco, S., Mosaico, G., Saviozzi, M., and Silvestro, F. (2019). A hybrid technique for day-ahead pv generation forecasting using clear-sky models or ensemble of artificial neural networks according to a decision tree approach. Energies, 12(7).

[13] Montavon, G., Samek, W., and M¨uller, K.R. (2018). Methods for interpreting and understanding deep neural networks. Digital Signal Processing: A Review Journal, 73,1-15.

[14] Morete, P.A. e Toloi, C. M. C. Análise de séries temporais. 2. ed. São Paulo: Egard Blucher, 2006.

[15] Mosaico, G. and Saviozzi, M. (2019). A hybrid methodology for the day-ahead pv forecasting exploiting a clear sky model or artificial neural networks. In IEEE EUROCON 2019 -18th International Conference on Smart Technologies, 1-6.

[16] Pelletier, C. at al. Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series. MDPI - Remote Sensinf - Open Access Journal. Available at: https://arxiv.org/pdf/1811.10166.pdf. Accessed on: 01/03/2021.

[17] Python (2021). Python is a programming language that lets you work quickly and integrate systems more effectively. Available at: https://www.python.org. Accessed on: 01/03/2021

[18] Reddy, K.S. and Ranjan, M. (2003). Solar resource estimation using artificial neural networks and comparison with other correlation models. Energy Conversion and Management, 44(15), 2519-2530.

[19] SARIMAX (2021). SARIMAX: Introduction. Available at: https://www.statsmodels.org/dev/examples/notebooks/generated/statespace_sarimax_stata.html. Accessed on: 01/03/2021

[20] Spiegel, Murray Ralph (1974). Statistics. Brasília: McGraw-Hill do Brasil , 1974.

[21] Su, D., Batzelis, E., and Pal, B. (2019). Machine learning algorithms in forecasting of photovoltaic power generation. In 2019 International Conference on Smart Energy Systems and Technologies (SEST), 1-6.

[22] TensorFlow (2021). A complete open source platform for machine learning. Available at: https://www.tensorflow.org. Accessed on: 01/03/2021

[23] Theocharides, S., Makrides, G., Venizelou, V., Kaimakis, P., and Georghiou, G.E. (2017). Pv Production Forecasting Model Based on Artificial Neural Networks (Ann). 33rd European Photovoltaic Solar Energy Conference, 1830 - 1894.

[24] Yang, J. At al. Deep convolutional neural networks on multichannel time series for human activity recognition. in Ijcai, vol. 15, 2015, pp. 3995-4001.

[25] Yi, H., Shiyu, S., Duan, X., and Chen, Z. (2017). A study on Deep Neural Networks framework. Proceedings of 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2016, 1519-1522.

[26] Zhang, G. P. Time series forecasting using a hybrid arima and neural network model. Neurocomputing, 50:159-175, 2003.