




Photovoltaic Generation Forecast: Model Training and Adversarial Attack Aspects

Everton J. Santana¹, Ricardo Petri Silva², Bruno B. Zarpelão¹,
and Sylvio Barbon Junior¹

¹ Computer Science Department, Londrina State University, Londrina, PR, Brazil
{`evertonsantana,brunozarpelao,barbon`}@uel.br

² Electrical Engineering Department,
Londrina State University, Londrina, PR, Brazil
`petri@uel.br`

Abstract. Forecasting photovoltaic (PV) power generation, as in many other time series scenarios, is a challenging task. Most current solutions for time series forecasting are grounded on Machine Learning (ML) algorithms, which usually outperform statistical-based methods. However, solutions based on ML and, more recently, Deep Learning (DL) have been found vulnerable to adversarial attacks throughout their execution. With this in mind, in this work we explore four time series analysis techniques, namely Naive, a baseline technique for time series, Autoregressive Integrated Moving Average (ARIMA), from the statistical field, and Long Short-term Memory (LSTM) and Temporal Convolutional Network (TCN), from the DL family. These techniques are used to forecast the power generation of a PV power plant 15 minutes and 24 hours ahead, having as input only power generation historical data. Two main aspects were analyzed: i) how training size influenced the performance of the forecasting models and ii) how univariate time series data could be modified by an adversarial attack to decrease models' performance through cross-technique transferability. For i), the mentioned methods were used and evaluated with monthly updates. For ii), Fast Gradient Sign Method (FGSM), along with a logistic regression substitute model and past data, were used to perform attacks against DL models at test time. LSTM and TCN decreased the error as the training sample size increased and outperformed Naive and ARIMA models. Adversarial samples were able to reduce the performance of LSTM and TCN, particularly for short-term forecasts.

Keywords: Time series forecast · Solar photovoltaic generation · Deep learning · Adversarial attack · Smart grid

The authors would like to thank the financial support of Coordination for the Improvement of Higher Education Personnel (CAPES) - Finance Code 001 -, the National Council for Scientific and Technological Development (CNPq) of Brazil - Grant of Project 420562/2018-4, and Fundação Araucária.

1 Introduction

Intelligent generation and distribution of electrical energy are beneficial for systems operators, plant managers and consumers [2]. A key aspect in this process is the accurate forecast of produced energy, which is fundamental to enable the integration of several plants to the grid, save costs, make power grids more reliable amid the variation in the demand, avoid power outage, and prevent plant managers from penalties. It is also advantageous for the sake of the environment [10], particularly when renewable sources are employed.

For photovoltaic (PV) generation, the focus of our work, forecasting is challenging due to the dependence on meteorological factors such as clouds covering solar panels and variations of solar radiation [22]. Thus, building accurate forecasting models based only on historical power data is a complex task.

Many works attempted to predict future PV power output [1, 9, 18, 22, 23] making use of statistical models, artificial intelligence, or a combination of them. However, the models are often built in batch, which assumes that data distribution does not change over time and, as a consequence, the model is not updated. Cerqueira et al. [7] compared the performance of machine learning (ML) and statistical methods when dealing with univariate time series (observing a single variable throughout time) forecast. The authors updated the models according to new incremental observed samples and found out that ML tends to provide better results as the training sample size increases.

In this context, we explored a novel real-life dataset collected from the generation history of a newly built PV power plant, and compared the performance of traditional time series forecasting techniques (Naive and Autoregressive Integrated Moving Average (ARIMA)) with Deep Learning (DL) methods (Long Short-term Memory (LSTM) and Temporal Convolutional Network (TCN)) to forecast power generation 15 minutes and 24 hours ahead. Since the amount of data about power generation increases over time, these models were updated and assessed monthly with the goal to evaluate the influence of the training sample size in their performance.

Another factor that may influence the power generation forecasting is its susceptibility to different attacks, which seek to interrupt the grid's safe operation or obtain financial gains, for instance [17]. Tampering with the results of the forecasts is among the possible attacks, since wrong forecasts on power generation may drive operators or automated control to make harmful decisions towards grid balancing. This shows that besides obtaining accurate models, their reliability should be investigated.

Considering that anticipating the potential model vulnerabilities and analyzing the impact of the possible attacks are the first steps of a proactive protection mechanism [5], and also that adversarial attacks against time series regression models have been overlooked by the literature, we performed attacks during the test time of the DL models, which obtained the best prediction results in the first part of this study.

We examined an adversary with restricted knowledge about the training data and the victim's forecasting models. To degrade the DL model's performance,

the attacker modifies the algorithm input data by using Fast Gradient Sign Method (FGSM). In view of this, we evaluated whether FGSM is suitable to generate adversarial test samples which are almost visually imperceptible and, concomitantly, able to increase test error.

The rest of this paper is organized as follows. In Sect. 2, we provide a brief background related to time series and the forecast methods adopted in this work. Section 3 describes experimental details, while results are presented and discussed in Sect. 4. Section 5 concludes the work with the main highlights and future work proposal.

2 Background

2.1 Time Series

A time series is given by the data sequence in a particular time period, and this data can produce different values at distinct moments in time. Formally, it can be defined as an ordered set $X = [x_1, x_2, \dots, x_T]$ in which T corresponds to the length of the series.

The forecasting task consists of finding a function f that predicts the h -th future value of X , i.e., \hat{x}_{t+h} based on i past values:

$$\hat{x}_{t+h} = f(x_{t-i-1}, x_{t-i-2}, \dots, x_{t-1}, x_t) \quad (1)$$

where i represents the input window size and h , the forecast horizon. When the latter is equal to one, the forecasting task is referred to as a one-step-ahead forecast. Otherwise, it is known as a multi-step ahead forecast.

In addition, time series can present seasonality. This occurs when regular patterns are captured in the series. Seasonal events are phenomena that occur, for instance, daily at a certain time, every day, or in a certain month every year.

The Naive method to forecast the future value in a time series, also known as the persistence model, consists of supposing that the next value of a time series will be the same as the current value:

$$\hat{x}_{t+h} = x_t \quad (2)$$

Given the complexity involving PV power generation, more sophisticated models are generally required.

2.2 Statistical Methods

Understanding the different factors of a time series is important to extract information that can be used to predict future points in this series. Many statistical methods were applied in time series for this purpose and one of the most adopted, ARIMA, is presented next.

ARIMA method is essentially exploratory and seeks to fit a model to adapt to the data structure [6]. With the aid of the autocorrelation and partial autocorrelation functions, it is possible to obtain the essence of the time series so that

it can be modeled. Then, information such as trends, variations, cyclical components, and even patterns present in the time series can also be obtained [13]. This allows the description of its current pattern and predictions of future series values [20].

This model is defined by the values (p, e, q) , where p is the number of auto-regressive terms, e is the number of differences, and q is the number of moving averages. Auto-regressive (AR) indicates that the evolution variable of interest is returned to its own previous values. The Moving average (MA) part indicates the regression error consisting of a linear combination of values at various times in the past. The Integrated (I) part indicates the process of differentiating between current values and previous values. In some cases, the ARIMA model is applied to non-stationary data. To solve this problem, the integrated part is applied, where differentiation processes are carried out and can be applied more than once until stationarity is obtained.

2.3 Machine Learning Methods

ML, particularly DL, models have been showing to be adequate for dealing with time series made up of power related data, from both demand and generation sides. Among them, TCN and LSTM achieved relevant results for this type of data [16, 22, 24].

TCN [4] is a specific Convolutional Neural Network (CNN) that has the capability of dealing with time series. For this purpose, it uses causal convolutions, which performs the convolution operation depending only on past values.

This kind of convolution may be submitted to dilation. In TCN, dilated convolutions in one dimension are used seeking to explore long-term patterns. The procedure for doing this is skipping d values between the inputs of convolution. The dilations will be denoted in this work as $[d_1, d_2, \dots, d_n]$, where d_1 corresponds to the dilation rate of the layer that is the closest to the input, and d_n to the layer that is the closest to the output.

Aiming to increase the receptive field of the network, b convolutional blocks can be stacked. Since it increases the number of parameters, the learning process becomes more complex. Figure 1 shows a dilated causal network with two stacked blocks.

The parameters b , k and d are factors that define the receptive field of the network by the following Eq. 3. For an adequate use of TCN, the receptive field should cover past history, which, in turn, should cover seasonality.

$$\text{receptive field} = b \times k \times d_n \quad (3)$$

LSTM. A classical neural network for dealing with time series is Recurrent Neural Networks (RNN). In this network, the information is propagated throughout a chain of repeating units of neural network located in the hidden layer.

However, standard RNN suffers from error backflow problems as gradient vanishing and explosion, which restraint long-term dependency learning. The first may lead to very slow learning, and the latter, to weight oscillation.

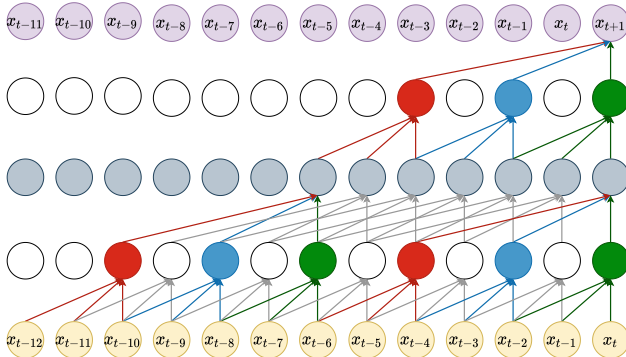


Fig. 1. Example of dilated causal networks with $b = 2$, $k = 3$ and $d = [1, 2]$.

The backflow calculation drawback was one of the main motivations for the development of LSTM [14]. This network, a particular case of RNN, uses a gating mechanism for learning long-term dependencies without losing the short-term capability. These gating mechanisms are inside memory cells, which are located in the hidden layer. Each unit in traditional LSTM has the aspect presented in Fig. 2.

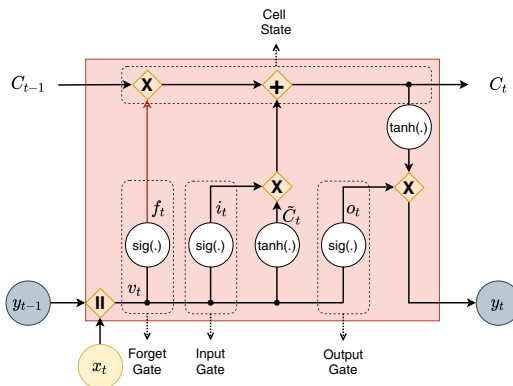


Fig. 2. LSTM cell.

Hyperbolic tangent function ($\tanh(\cdot)$) transforms the values to the range from -1 to 1 , and the sigmoid function ($\text{sig}(\cdot)$) to values between 0 and 1 . This property makes $\text{sig}(\cdot)$ act as a gate, since the values that are transformed to 0 are forgotten by the network and the values that are transformed into 1 are kept.

Each cell is mainly composed of three gates (forget, input and output gates) and a cell state. The forget gate (f_t) is the main gate to select which information should pass forward to the next cell or be eliminated. The input gate (i_t) is used

to update the cell state and to select what will be written to the cell. The output gate (o_t) decides the values that will be part of the output. The cell state (C_t) allows the gradient flow. Considering v_t the concatenation of x_t and y_{t-1} , the equations related to these components are:

$$f_t = sig(W_f.v_t + b_f) \quad (4)$$

$$i_t = sig(W_i.v_t + b_i) \quad (5)$$

$$o_t = sig(W_o.v_t + b_o) \quad (6)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (7)$$

\tilde{C}_t and y_t are also calculated to select the new candidate values that can be added to regulate the network and to actually calculate the output of the cell, respectively:

$$\tilde{C}_t = tanh(W_c.v_t + b_c) \quad (8)$$

$$y_t = o_t * tanh(C_t) \quad (9)$$

W_f , W_i , W_o and W_c correspond to the weights matrices related to forward, input, output gates and cell state; b_f , b_i , b_o and b_c correspond to bias of the respective gates and cell state.

2.4 Related Work

Related works in the literature, which address time series forecasting and electricity energy consumption prediction, are presented next.

Regarding statistical methods, Atique et al. [3] used ARIMA to forecast the total daily solar energy generated in a specific solar panel. In most of the recent works, this method is compared to DL. For instance, Jaihuni et al. [15] compared ARIMA and LSTM to a hybrid version to predict 5 minutes and one hour ahead. The hybrid version had better performance in predicting the longest forecast horizon, whereas LSTM and ARIMA outperformed for 5 minutes ahead.

Another example of solar energy generation prediction through DL methods is seen in Torres et al. [22]. In this work, the application of DL consists of predicting the generation of energy for the next day. According to the authors, the proposed method was capable of handling big time series data. In [23], Wang et al. evaluated CNN, LSTM, and a hybrid model with CNN and LSTM for modeling a PV system. The results showed robustness, stability, and great performance. TCN is also a very suitable DL method, as shown in [24], in which Yen et al. verified that TCN is capable of satisfactorily predicting PV generation.

Relating to adversarial attacks in time series, Favaz et al. [11] adapted FGSM and Basic Iterative Method to univariate time series classification and performed attacks to DL models. These attacks achieved an average reduction in the model's accuracy of 43.2% and 56.89%, respectively, and pointed out that FGSM allows real-time adversarial sample generation.

3 Materials and Methods

3.1 Dataset

The data was collected from a PV power generation system installed in a parking lot at the State University of Londrina. It started to operate in November 2019 with a total capacity of 300 kW distributed over 6 inverters. For being representative of the other inverters, inverter 1 was chosen for the analysis. Since the PV plant relies on solar energy to operate, it is usually turned off between 7 pm and 6 am (of the next day). Table 1 shows the data description for each month.

Table 1. Main monthly information about power generation in inverter 1.

Month	Maximum value [W]	Minimum value [W]	Average [W]	Number of samples
2019-11	41521.48	0	8749.14	2880
2019-12	41148.77	0	7582.21	2976
2020-01	42189.33	0	8062.55	2976
2020-02	41194.93	0	7485.98	2784
2020-03	43521.93	0	8506.43	2976

Samples were collected every 15 minutes, implying that each day is composed of 96 samples. Considering this and that the data behavior mostly repeats every day, the input window size was set to 96 ($i = 96$) to cover seasonality.

The interest in energy production forecast can be part of different strategies in very short and short-term time horizons. Thus, we evaluated the performance of these methods when forecasting the production in the next 15 minutes ($h = 1$) and 24 hours ($h = 96$).

Two experiments (Setup 1 and Setup 2) were performed using Intel Xeon CPU @2.3 GHz and Tesla K80 GPU made available by Google Colab. The source code can be found at¹.

3.2 Setup 1 - Obtaining the Forecasting Model

Setup 1 is dedicated to assess the prediction performance of Naive, ARIMA, LSTM, and TCN methods. Naive was picked to be a baseline for our experiment. ARIMA is likely the most adopted option when it comes to statistical methods for time series analysis. Lastly, LSTM and TCN are DL methods, which represent the state-of-the-art of ML methods for time series forecasting.

Prequential evaluation was used to show the evolution of performance as the sample size grows and to simulate a situation in which the model is updated monthly. In this case, each month (except the first and the last) was used for test before being incrementally used for training, making the most use of available data, as observed in Fig. 3. Furthermore, for hyper-parameter tuning, 20% of

¹ http://www.uel.br/grupo-pesquisa/remid/?page_id=145.

training data was reserved for validation. A preprocessing step with z-score was performed to improve convergence. The normalization parameters were obtained for each new training set.

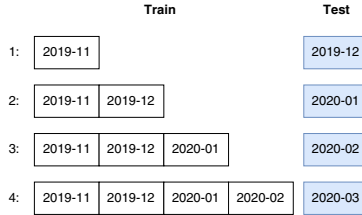


Fig. 3. Train and test sets for Setup 1.

To apply the ARIMA model, *auto-arima* from `pmdarima` library² was used since it auto-tunes its parameters. This approach was applied because the preparation of the parameters ends up being a time-consuming task. The *auto-arima* technique performs several procedures automatically, making the process simpler and faster, finding the best parameters for each data entry.

Both deep methods share some training parameters. We selected Adam as optimizer, Mean Absolute Error (MAE) as loss function, 25 epochs and evaluated batch sizes of 32 and 128. The specific hyper-parameters of LSTM and TCN are shown in Tables 2 and 3, respectively.

Table 2. Specific hyper-parameters of LSTM.

Parameter	Experimental choice
Number of stacked layers (l)	1, 2, 3
Units	32, 64
Dropout	0

Table 3. Specific hyper-parameters of TCN. *The possible dilations followed Eq. 3.

Parameter	Experimental choice
k	2, 3
d^*	[1, 4, 12, 48], [1, 2, 4, 8, 12, 24, 48], [1, 4, 16, 32], [1, 2, 4, 8, 16, 32], [1, 3, 6, 12, 24], [1, 2, 6, 12, 24], [1, 2, 4, 8, 16], [1, 4, 16], [1, 2, 4, 8], [1, 4, 8]
b	1, 2
Number of filters	32, 64
Dropout rate	0

² <https://pypi.org/project/pmdarima/>.

For TCN, Rectified Linear Unit (ReLU) was adopted as activation function and each block output has a residual connection. For this method, we adopted the keras-tcn library³. The hyper-parameters not specified previously assumed the default configuration of Keras-TensorFlow⁴.

3.3 Setup 2 - Evaluating the Impact of the Adversarial Attacks

In Setup 1, the objective is to find the models with the best predictive performance. In Setup 2, we evaluate the impact of attacks against these models at test time. For this purpose, we consider that the attacker has limited computational and knowledge capabilities, as follows.

Adversary’s Goal. The adversary aims to carry out attacks at test time, in the sense that the attacker modifies input data during operation to increase the prediction error of the victim’s forecasting model F (obtained during training time). In this case, F could be either TCN or LSTM.

Adversary’s Knowledge. We simulated a gray-box attack [21] in which the attacker has limited knowledge about training data and no knowledge about the model adopted by the victim. Particularly, in our scenario, the attacker has access to the data collected during the first 2 months of operation.

Adversary’s Capability. The attacker is able to read the legitimate input data during the operation phase, craft new malicious input based on this legitimate data and a substitute model F' , and, finally, feed the victim’s forecasting model with this malicious data. The attacker has to define a substitute model F' because they do not know the model F built by the victim.

To create the adversarial input, the attacker uses an adaptation of the Fast Gradient Sign Method [12] to the time series regression context. The goal is to add perturbations in the input of testing data. The two main requirements of the perturbation are being not easily visually detected and, at the same time, degrading the performance of the victim’s ML model. The perturbation η generated by FGSM is given by:

$$\eta = \epsilon * \text{sign}(\Delta_{\mathbf{x}}J(\theta, \mathbf{x}, y)) \quad (10)$$

where ϵ corresponds to the coefficient that controls the attack magnitude, \mathbf{x} to the input to the model, y to the output associated to \mathbf{x} , θ to the weights of the adversarial model and $J(\cdot)$ to the loss function.

The attacker also has limited computational resources. This means that the attacker is not able to use complex models to generate adversarial examples, so that simpler models should be adopted as F' and rely on cross-technique transferability, i.e., adversarial examples generated by the model F' can affect the

³ <https://pypi.org/project/keras-tcn/>.

⁴ https://www.tensorflow.org/api_docs/python/tf/keras/.

performance of another model, trained using a different learning technique [8]. For being simpler than DL, successful in classification scenarios [19] and still differentiable, logistic regression (LR) was adopted to build the substitute model.

Figure 4 shows, in practical terms, how the available data is used to perform the attack.

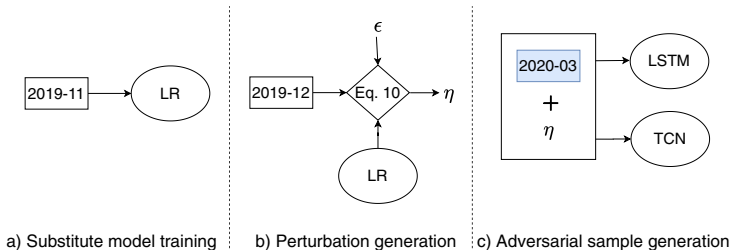


Fig. 4. Illustration of the adversary capability and knowledge.

To obtain $F'(a)$, the first collected month is used as training data. For building the perturbation (b), along with the LR substitute model, the second month is used as \mathbf{x} and y in Eq. 10. Then, at test time (c), the adversarial sample x'_{test} for the corresponding legitimate test input x_{test} is computed by:

$$x'_{test} = x_{test} + \eta \quad (11)$$

and inputted to the victim's model. The ϵ value was varied from 0.05 to 2 with steps of 0.05.

3.4 Evaluation Metrics

To compute the model performance, the root means squared error (RMSE) was assessed for the test sets as:

$$RMSE = \sqrt{\frac{\sum_{j=1}^n (\hat{x}_{j+h} - x_{j+h})^2}{n}}, \quad (12)$$

where n corresponds to the number of samples of the test set.

4 Results and Discussion

4.1 Setup 1

Figure 5 presents the test error obtained by the assessed models. As observed, the Naive and ARIMA models had the worst performances, incurring an error of around 16 and 11 kW, respectively. Moreover, for the last test set, the error increased for both methods. During the operation of the plant, the derivatives between 2 intervals are very high, which probably led to a worse performance.

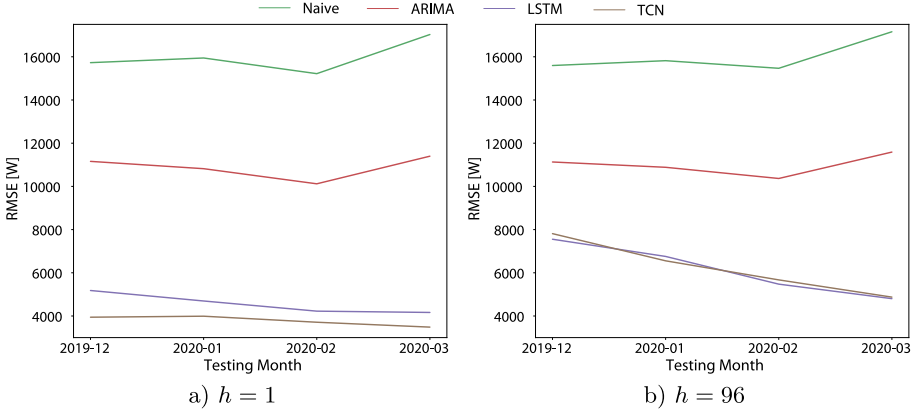


Fig. 5. RMSE of the methods using one month as testing period and the two experimented forecasting horizons. For LSTM and TCN, the mean values for the different hyper-parameters configurations are shown.

For the DL models, the error tended to monotonously decrease as the amount of training data increased. The performance improvement is more evident for $h = 96$: the mean error decreased from almost 8 kW to nearly 5 kW.

Table 4 shows the hyper-parameters configurations that led to the lowest error.

Table 4. Best results of TCN and LSTM for both forecasts horizons and the test datasets. The bold values correspond to the best RMSE values for LSTM and TCN in each forecast horizon.

h	Test	Method	RMSE	Batch	k	d	b	Filters	l	Units
1	2019-12	TCN	3625.71	32	2	[1, 4, 12, 48]	1	64		
		LSTM	4276.75	32					1	32
	2020-01	TCN	3811.69	32	2	[1, 4, 12, 48]	1	64		
		LSTM	4067.07	32					1	32
	2020-02	TCN	3576.19	32	2	[1, 2, 6, 12, 24]	2	64		
		LSTM	3705.22	32					1	64
	2020-03	TCN	3333.10	128	2	[1, 3, 6, 12, 24]	2	32		
		LSTM	3672.89	32					1	64
96	2019-12	TCN	7320.73	32	3	[1, 2, 4, 8, 16]	2	64		
		LSTM	7294.97	128					2	32
	2020-01	TCN	6265.15	32	2	[1, 2, 6, 12, 24]	2	32		
		LSTM	6016.31	128					1	32
	2020-02	TCN	5276.89	128	3	[1, 2, 4, 8, 16, 32]	1	32		
		LSTM	5175.33	32					1	32
	2020-03	TCN	4497.43	128	2	[1, 4, 12, 48]	1	32		
		LSTM	4408.46	128					1	32

For $h = 1$, TCN achieved the best performance, with an error of 3333.10 W. It represented 29.24% and 19.57% in relation to ARIMA and Naive RMSEs for the same test set and 7.66% in relation to the maximum power value of 2020-03.

For $h = 96$, LSTM slightly outperformed TCN, with an error of 4408.56 W. It represented 38.04% of the error presented by ARIMA, 25.69% in relation to Naive and 10.13% in relation to the maximum power value of this set.

As to the hyper-parameters, for $h = 1$, LSTM presented a preference for smaller batch size and a single layer (i.e, no stacking). As the training sample size increased, the number of required units also increased. For the same forecast horizon, the preferred kernel size of TCN was 2 and higher training sample sizes preferred more blocks. The last month preferred the largest batch size and a lower number of filters.

Still analyzing the hyper-parameters, for $h = 96$, LSTM preferred 32 units for all training sample sizes and mostly only one layer. The last month preferred the largest batch size. TCN mostly required 32 filters, and as the training sample size increased, the batch size and the number of blocks increased as well.

Alongside the error values, we also wanted to evaluate qualitatively the models. With this intention, Fig. 6 compares the true output values from 7 days of the last test set to the forecast results that led to the lowest RMSE for each h .

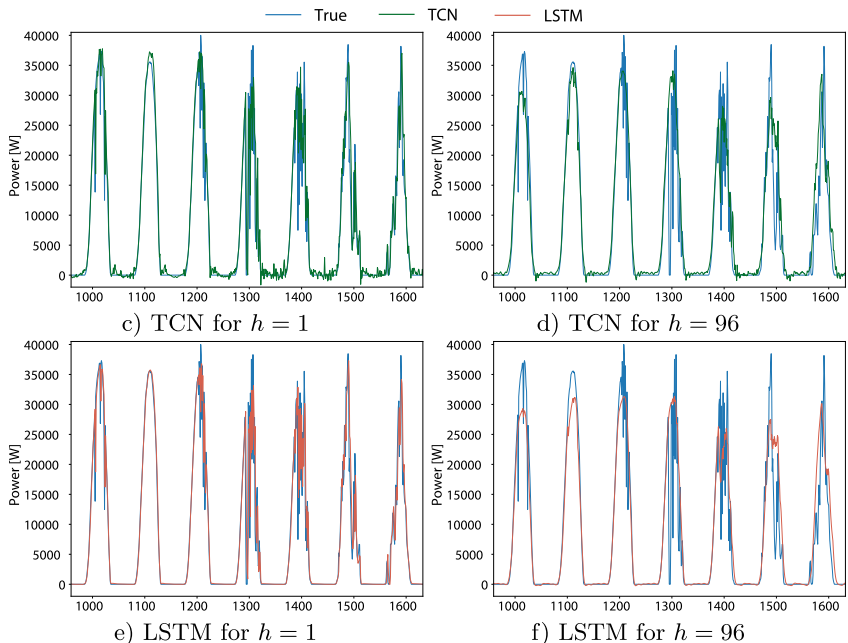


Fig. 6. Comparison between true output and predicted results by the best models.

In general, the predictions followed the true output values. TCN presented lower stability around 0 than LSTM, particularly for $h = 1$. When $h = 96$ and

the true output values are very high, it is observable a conservative forecast of the power generation.

4.2 Setup 2

As previously mentioned, the adversarial input data must be subtle for not being easily visually identified. Considering this, Fig. 7 compares one sample of an original input window that belongs to the test set and its respective sample of a maliciously crafted window input generated by FGSM.

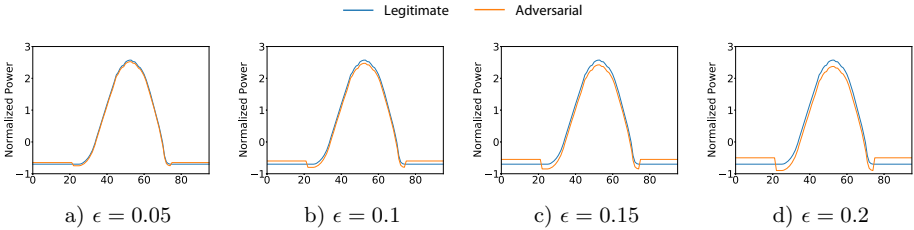


Fig. 7. Sample of legitimate input window and adversarial input window according to variation in ϵ following the procedure described in setup 2.

Given the high similarity between the shape of both curves, it is very difficult to distinguish between the adversarial and the legitimate input, showing that a human would have difficulties to visually detect the attack. Although small, the difference in both curves increases as ϵ increases, and the most noticeable difference occurs when the curve starts to increase more intensively or stops decreasing.

To demonstrate how the adversary perturbation influenced the models' performance, Table 5 shows the error increase caused by the different ϵ values to the best DL models obtained in Setup 1 (when test set was 2020-03).

Table 5. Percentual error increase in relation to the corresponding original dataset/method.

	LSTM				TCN			
	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.15$	$\epsilon = 0.2$
$h = 1$	0.88%	4.55%	10.75%	19.05%	0.22%	3.66%	9.97%	17.86%
$h = 96$	2.70%	17.77%	42.33%	57.06%	4.20%	21.03%	50.24%	77.99%

For $h = 1$ and $\epsilon = 0.05$, nearly no increase in error was found for both methods, but when ϵ slightly varied from 0.05 to 0.1, the error increase was

multiplied by a factor of 5.17 for LSTM and 16.63 for TCN. The biggest error was found for LSTM when $\epsilon = 0.2$.

For $h = 96$, the biggest error increase was found for TCN, reaching almost 78%. Such error increase may cause serious harm to the grid operation. Suppose a situation where the legitimate forecast, i.e., the TCN legitimate output, is 10000 W and there is a fixed demand of 20000 W for the region connected to the studied PV plant. Considering our most extreme setup for an attack ($\epsilon = 0.2$), the TCN output could be changed to 17799 W. Then, based on this wrong forecast, the system operator would assume that it was necessary to deliver only more 2201 W from other plants to meet the demand of that region. However, during the operation, the operator would be actually required to deliver a value closer to 10000 W to properly meet this demand. This difference would unbalance the grid and possibly culminate in a power outage.

Therefore, it was possible to verify that FGSM can generate adversarial samples for the studied models, based only on a limited amount of historical data and using LR as substitute model. Thus, in the context of this work, a defense mechanism should be developed for the adopted forecasting models for preventing them from adversarial attacks.

5 Conclusion and Future Work

Results showed that DL models outperformed Naive and ARIMA. Those models were able to deal with the complexity of PV generation data. Moreover, the bigger the training sample size, the better the TCN and LSTM performances, so updating these models is recommended. As for the attacks at test time, FGSM was able to increase models' error even with small ϵ values. It was also possible to validate the cross-technique transferability of LR as a substitute model for generating adversarial examples for LSTM and TCN.

Future work can analyze other influencing factors in PV power generation and treat the time series as multivariate aiming at improving the accuracy of forecasting models. Concomitantly, we will investigate defense mechanisms for these attacks since it is a real-world application.

References

1. Akhter, M.N., Mekhilef, S., Mokhlis, H., Shah, N.M.: Review on forecasting of photovoltaic power generation based on machine learning and metaheuristic techniques. *IET Renew. Power Gener.* **13**(7), 1009–1023 (2019)
2. Antonanzas, J., Osorio, N., Escobar, R., Urraca, R., Martinez-de Pison, F.J., Antonanzas-Torres, F.: Review of photovoltaic power forecasting. *Solar Energy* **136**, 78–111 (2016)
3. Atique, S., Noureen, S., Roy, V., Subburaj, V., Bayne, S., Macfie, J.: Forecasting of total daily solar energy generation using ARIMA: a case study. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0114–0119. IEEE (2019)

4. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint [arXiv:1803.01271](https://arxiv.org/abs/1803.01271) (2018)
5. Biggio, B., et al.: Evasion attacks against machine learning at test time. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 387–402. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_25
6. Box, G.E.P., Jenkins, G.: Time Series Analysis Forecasting and Control. Holden-Day Inc., USA (1990)
7. Cerqueira, V., Torgo, L., Soares, C.: Machine learning vs statistical methods for time series forecasting: Size matters. arXiv preprint [arXiv:1909.13316](https://arxiv.org/abs/1909.13316) (2019)
8. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: a survey. arXiv preprint [arXiv:1810.00069](https://arxiv.org/abs/1810.00069) (2018)
9. Das, U.K., et al.: Forecasting of photovoltaic power generation and model optimization: a review. *Renew. Sustain. Energy Rev.* **81**, 912–928 (2018)
10. Diamantoulakis, P.D., Kapinas, V.M., Karagiannidis, G.K.: Big data analytics for dynamic energy management in smart grids. *Big Data Res.* **2**(3), 94–101 (2015)
11. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Adversarial attacks on deep neural networks for time series classification. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint [arXiv:1412.6572](https://arxiv.org/abs/1412.6572) (2014)
13. Ho, S., Xie, M.: The use of ARIMA models for reliability forecasting and analysis. *Comput. Ind. Eng.* **35**(1), 213–216 (1998)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
15. Jaihuni, M., et al.: A partially amended hybrid Bi-GRU–ARIMA model (PAHM) for predicting solar irradiance in short and very-short terms. *Energies* **13**(2), 435 (2020)
16. Lara-Benítez, P., Carranza-García, M., Luna-Romera, J.M., Riquelme, J.C.: Temporal convolutional networks applied to energy-related time series forecasting. *Appl. Sci.* **10**(7), 2322 (2020)
17. Mehrdad, S., Mousavian, S., Madraki, G., Dvorkin, Y.: Cyber-physical resilience of electrical power systems against malicious attacks: a review. *Curr. Sustain./Renew. Energy Rep.* **5**(1), 14–22 (2018)
18. Mellit, A., Massi Pavan, A., Ogliaeri, E., Leva, S., Lughi, V.: Advanced methods for photovoltaic output power forecasting: a review. *Appl. Sci.* **10**(2), 487 (2020)
19. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint [arXiv:1605.07277](https://arxiv.org/abs/1605.07277) (2016)
20. Pena, E.H., Barbon, S., Rodrigues, J.J., Proença, M.L.: Anomaly detection using digital signature of network segment with adaptive arima model and paraconsistent logic. In: 2014 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6. IEEE (2014)
21. Tabassi, E., Burns, K.J., Hadjimichael, M., Molina-Markham, A.D., Sexton, J.T.: A taxonomy and terminology of adversarial machine learning. NIST IR (2019)
22. Torres, J.F., Troncoso, A., Koprinska, I., Wang, Z., Martínez-Álvarez, F.: Deep learning for big data time series forecasting applied to solar power. In: Graña, M., et al. (eds.) SOCO'18-CISIS'18-ICEUTE'18 2018. AISC, vol. 771, pp. 123–133. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-94120-2_12

23. Wang, K., Qi, X., Liu, H.: A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network. *Appl. Energy* **251**, 113315 (2019)
24. Yen, C.F., Hsieh, H.Y., Su, K.W., Leu, J.S.: Predicting solar performance ratio based on encoder-decoder neural network model. In: 2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 1–4. IEEE (2019)