energies

MDPI

Article

Forecasting Solar PV Output Using Convolutional Neural Networks with a Sliding Window Algorithm

Vishnu Suresh *, Przemyslaw Janik, Jacek Rezmer and Zbigniew Leonowicz

Faculty of Electrical Engineering, Wroclaw University of Science and Technology, 50-370 Wroclaw, Poland; przemyslaw.janik@pwr.edu.pl (P.J.); jacek.rezmer@pwr.edu.pl (J.R.); zbigniew.leonowicz@pwr.edu.pl (Z.L.)
* Correspondence: vishnu.suresh@pwr.edu.pl

Received: 25 December 2019; Accepted: 5 February 2020; Published: 7 February 2020

Abstract: The stochastic nature of renewable energy sources, especially solar PV output, has created uncertainties for the power sector. It threatens the stability of the power system and results in an inability to match power consumption and production. This paper presents a Convolutional Neural Network (CNN) approach consisting of different architectures, such as the regular CNN, multi-headed CNN, and CNN-LSTM (CNN-Long Short-Term Memory), which utilizes a sliding window data-level approach and other data pre-processing techniques to make accurate forecasts. The output of the solar panels is linked to input parameters such as irradiation, module temperature, ambient temperature, and windspeed. The benchmarking and accuracy metrics are calculated for 1 h, 1 day, and 1 week for the CNN based methods which are then compared with the results from the autoregressive moving average and multiple linear regression models in order to demonstrate its efficacy in making short-term and medium-term forecasts.

Keywords: convolutional neural networks; multi-headed CNN; CNN-LSTM; forecasting; solar output; sliding window; renewable energy

1. Introduction

Global efforts to keep the increase in average temperature below 2 °C, with the possibility of keeping it lower than 1.5 °C, was agreed upon in the Paris agreement of 2015. In the recent "Climate action and support trends—2019" report, it was mentioned that current greenhouse gas emission levels and reduction efforts are not in line with meeting the targets that were set out [1].

Due to such environmental concerns and ambitious targets, there has been an increasing penetration of renewable energy sources in the power sector, especially in the form of solar photovoltaic panels. One of the biggest concerns connected with solar energy is its stochastic nature and variability, which threatens grid stability. A well-known approach to mitigate such uncertainty is the use of accurate forecasts [2].

The motivation for this study is the need to build a forecasting algorithm for a stochastic energy management system for the microgrid present at the Wroclaw University of Science and Technology. The microgrid currently employs a system that is deterministic, but, considering the stochastic nature of the solar panels, it was considered necessary. Convolutional neural network-based architectures used in forecasting are mainly used to study images of the sky, as explained later, and are used in tandem with statistical techniques for forecasting. This microgrid facility does not possess a device to record images of the sky but a deep learning approach to forecasting was decided upon. Hence, a data level approach using the sliding window algorithm for forecasting was adopted and the results were analyzed.

The area of forecasting is widely researched and is an age-old concept, aiming to predict solar PV outputs, wind turbine power outputs and loads in an electrical power system. A short literature review reveals numerous approaches, some of which are described as follows. In [3], short-term forecasts

for PV outputs were obtained using Support Vector Regression models wherein the parameters of the models were optimized using intelligent methods, such as the Cuckoo Search and Differential Evolution algorithms. In this study, the authors had used data from an inhouse rooftop solar PV unit at Virginia Tech. In [4], multiple linear regression was employed to make forecasts for solar energy output. This study had used extensive data obtained from the European Centre for Medium-Range Weather forecasts, including as many as 12 independent variables. The study described in [5] presents a generalized fuzzy logic approach in order to make short-term output forecasts from measured irradiance data. The input data in this case was for one particular month (October 2014) and the inputs and outputs were normalized within a range of 0.1–0.9. A comprehensive review and analysis of different methods and associated results regarding the forecasting of solar irradiance and solar PV output is presented in [6].

With regard to the application of Convolutional Neural Networks (CNN) for solar PV output forecasts, there is little available literature. One of the approaches as seen in [7,8] is to use a combination of historical data and sky images. The sky images are crucial in order to capture the effect clouds have on PV output. The study described in [8] used a total sky imager, which provides images of the sky and could, whereas [7] used videos recorded by a 6-megapixel 360 degrees fish eye camera by HiKvision. Other approaches, which do not use images but only historical data, have adjusted the CNN in such a way that it is able to deal with time series data. CNN is, in fact, a machine learning tool that is explicitly used for image detection and classification but based on the method by which data is processed, its ability to understand non-linear relationships between the inputs and outputs can be leveraged for time series data. A hybridized approach where CNN is used for pattern recognition and then a long short-term memory network is used for prediction is seen in [9] and then this framework is applied for 30 min ahead forecasting of global solar radiation. In [2], a method in which suitable data processing is applied before training the CNN is presented. In this case, the time series data is split into various frequencies through variational mode decomposition and it is then converted into a 2D data form that is extracted by convolutional kernels. Finally, the approach used in [10] proposes another hybrid method in which a chaotic Genetic Algorithm/Particle Swarm Optimization is used to optimize the hyper parameters of the CNN, which is then used to make solar irradiance prediction.

This paper's forecasting approach is to be applied in developing a stochastic energy management system for microgrids. Hence, a few contributions in this regard are as follows: p A comprehensive review about weather forecasts, forecast errors, data sources, different methodologies used, and their importance in microgrid scheduling is described in [11]. The focus has been kept on wind energy forecasts, solar generation, and load forecasts. Another popular approach for forecasts using the ARMA (Autoregressive moving average) model, especially for load forecasting followed by solving a microgrid unit commitment problem, is described in [12]. An advanced forecasting method using artificial neural networks, support vector regression, and random forest followed by incorporation into a Horizon 2020 project involving several countries has been described in [13].

This paper utilizes a sliding window approach in order to prepare data in such a way that it can be used to train the CNN with historical data and make accurate predictions.

2. Forecasting Models, Data Processing, and Evaluation Metrics

2.1. Forecasting Models

The data for this study comes from a PV panel installed at a university building of the Wroclaw University of Science and Technology. It is a part of a power plant with a peak power capacity of 5 kW. The input measurements are obtained from associated sensors and are Irradiation (W/m^2), Wind speed (m/s), Ambient temperature (°C), and PV Module Temperature (°C). The output of the panel (W) and all inputs are measured in a 15 min window. The forecasting is also done in steps of 15 min intervals.

The inputs were chosen according to the recommendations of the IEA (International Energy Agency) report on "Photovoltaic and Solar Forecasting" [14] and other reliable sources [15]. The evaluation and

benchmarking techniques to be used for the forecasts were also taken from [14–16] in order to establish the reliability of the results of this study. The metrics are discussed in detail further on.

The structure of the CNN model is shown in Figure 1.

The CNN is a specialized neural network that is explicitly used for image recognition. In such cases, the input images are represented as a two-dimensional grid of pixels. In order to use CNNs for time series data, a 1-D structure is more appropriate. Taking the example of the input time series data used in this study, it is a 175,200 × 4 matrix. The length (number of rows) represents the time step of the input data, whereas the columns (Irradiation, Wind Speed, Ambient temperature, PV Module Temperature) represent the width. This can be equated to the height and width of the pixels that are used as the input data for training CNNs for image recognition.

For efficient and quick training of all networks, the min–max scaling algorithm was used. This is necessary since the distribution and scale of the data varies for every variable. Moreover, the units of measurement for every variable are also different, which could lead to large weight values, and models assuming such large weight values often perform poorly while learning and are sensitive to changes in input values [17]. It was applied to normalize the data within the range of [0,1]. The formula for the same is described in (1).

$$\frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{1}$$

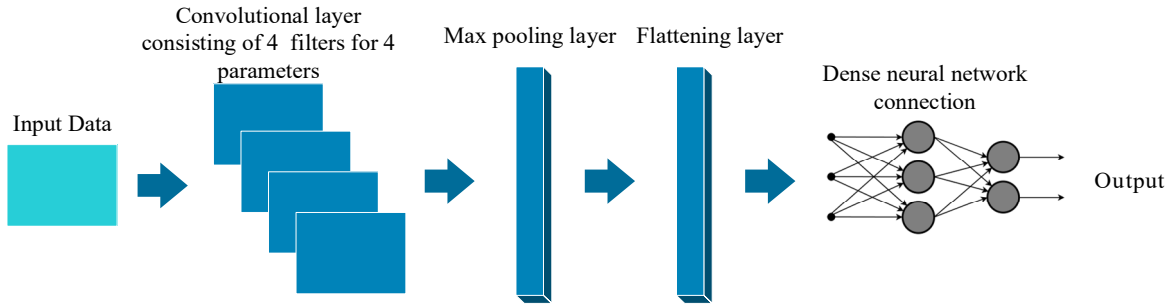


Figure 1. Convolutional Neural Network (CNN) structure.

The convolutional layer that follows input data processing is responsible for feature extraction [18]. The layer is made up of as many filters (neurons) as there are variables (4). These filters carry out convolution, which, by definition, is a function that is applied to the input data to obtain specific information from it. These filters are moved across the entire input data in a sliding window-like manner. In case of 2-D images the sliding window is moved horizontally and vertically but since this study employs a 1-D data the window is made to move vertically. The function used in this case is the Rectified Linear Activation Function (RLAF), which is described below, and the sliding window algorithm is described later.

The RLAF is a function that behaves like a linear function but is actually non-linear in nature, which enables the learning of complex relationships in the input data. It is widely used and can be defined in an easy manner. When the input is greater than 0.0, the output value remains the same as the input value, whereas if the input is less than 0.0 the output is 0.0. Mathematically, it is defined as:

$$g(z) = \max\{0, z\} \tag{2}$$

where z is the input value and g is the RLAF function. The advantage of this function includes computational ease, sparsity, and the ease of implementation to neural networks due to its linear behavior despite non-linearity [19].

The output of the filters in the convolutional layers are called feature maps. The feature maps hold relationships and patterns from the input data. These feature maps from each filter put together complete the convolutional layer. This layer is followed by the pooling layer, the objective of which is

to reduce the feature maps of the convolutional layer (it summarizes the features learnt in the previous layer). This is done in order to prevent overfitting. It also reduces the size of the input data, which results in increased processing speeds and reduced memory demand. While there are numerous pooling functions, such as max, average, and sum [18], this study employs the max function, hence the max pooling layer.

The flattening layer succeeding the max pooling layer converts the output into a 1-D input vector that can be given to the dense or fully connected layer. The dense layer in this case is a regular neural network that has a non-linear activation function.

The model in this case is fit by the Adam optimization algorithm. The advantage of using this optimizer is that the learning rate is adjusted as the error is reduced.

It is in fact a combination of two well-known extensions of stochastic gradient descent, which are the Adaptive gradient algorithm (AdaGrad) and Root mean square propagation (RMSProp). Adam is discussed in detail in [20].

The second CNN structure used in this study is the multi-headed CNN. This approach involves handling every input series by its own CNN. This approach has shown some flexibility. While there is no significant proof in the literature behind the advantages of multi-headed CNN over the regular CNN using multiple filters, a multi-headed CNN with 3 convolutional 2-D nets has been used for enhanced image classification as shown in [21]. This paper uses a similar, yet different, architecture. The structure of the multi-headed CNN is shown in Figure 2.

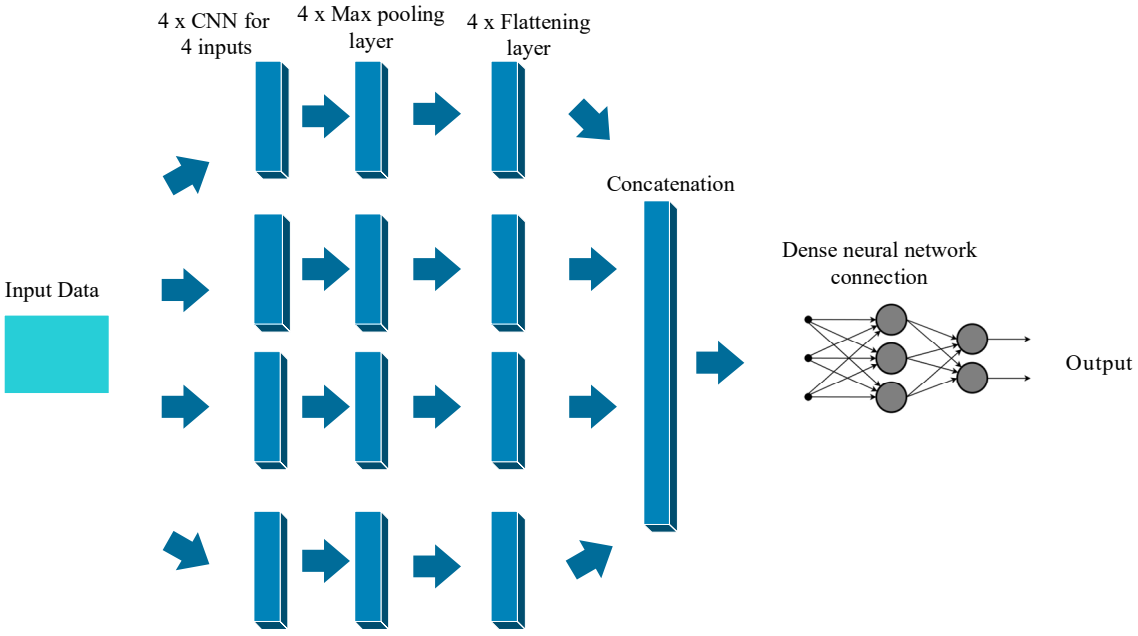


Figure 2. Multi-headed CNN structure.

In this study, as described in Figure 2, the multi-headed CNN has 4 CNNs, one for each input. This is followed by 4 max pooling layers and then by 4 flattening layers, and then the results from these layers is combined before the information is fed to the dense neural network, which makes the final prediction.

The third approach for forecasting is the CNN-LSTM (CNN-Long Short-Term Memory) network. Recently, the CNN-LSTM has been implemented in many areas for time series predictions. Study [22] presents a problem where water demand in urban cities is predicted. The correlation between water demand and changes in temperature and holiday periods is obtained using CNN-LTSM networks, and an improvement in predictions was observed. Similarly, an improvement in weather predictions was demonstrated in [23] by using such a hybrid CNN-LSTM architecture.

The LSTM is in fact an RNN (Recurrent neural network), which is efficient in working with time series data and is known to be a powerful tool for classification and forecasting associated with time series data. The uniqueness of LSTM comes from the memory cell, which behaves as a collector of state information. Whenever new information is obtained if the input gate is triggered it will be accumulated in the cell and past information would be forgotten if the forget gate is triggered. The latest cell obtained in such a process would be propagated to the final stage only if the output gate is triggered. This kind of cell behavior prevents the gradients trapped in the cell from vanishing quickly and is characteristic of LSTM, which makes it better suited to handle time series data and make predictions compared to other RNN structures [24].

The advantage of using a hybrid CNN-LSTM architecture is that the CNN is used to extract features from the raw input time series and then these features are given as an input to the LSTM, which is efficient with time series data.

Figure 3 provides the CNN-LSTM architecture. It can be noticed that, overall, the structure is similar to the CNN structure in Figure 1, with the exception of the LTSM layer, which enables the whole network to process the time series data more efficiently.

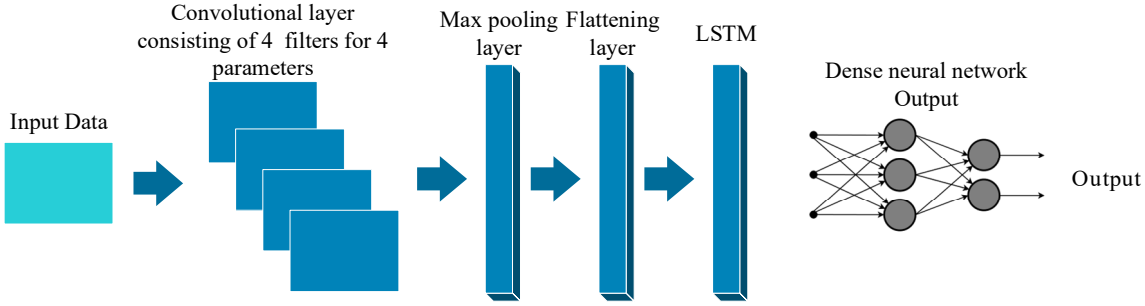


Figure 3. CNN-LSTM structure.

In order to provide a benchmark with an established technique for forecasting, the ARMA model is proposed. The ARMA model is utilized mainly for stationary time series data. In this method, the predicted variable is calculated on the basis of a linear relationship with its past values [25,26]. In cases when the data is non-stationary and has seasonal characteristics, as will be explained in the next section, it has to be transformed into a stationary one before an ARMA model is fit. The model consists of two parts, AR (Autoregressive) and MA (Moving Average), and is defined as ARMA (m, n) where m, n represent the orders of the model.

$$y_t^{'AR} = \sum_{i=1}^{m} \varnothing_i x_{t-i} + \omega_t = \varnothing_1 x_{t-1} + \varnothing_2 x_{t-2} + \ldots + \varnothing_m x_{t-m} + \omega_t \quad (3)$$

$$y_t^{'MA} = \sum_{j=0}^{n} \theta_j \omega_{t-j} = \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \ldots + \theta_n \omega_{t-n} \quad (4)$$

$$y_t^{'ARMA} = \sum_{i=1}^{m} \varnothing_i x_{t-i} + \sum_{j=0}^{n} \theta_j \omega_{t-j} \quad (5)$$

where $y_t^{'AR}$, $y_t^{'MA}$, and $y_t^{'ARMA}$ represent the time series values of the autoregression (AR), the Moving average (MA), and the Autoregression moving average (ARMA), respectively. $\varnothing_i$ is the autoregressive coefficient and θ_j is the moving average coefficient. ω_t is the noise.

The autoregressive (AR) part involves representing the current value as a result of a linear combination of the previous values and the noise ω_t. It is represented in Equation (3). The Moving average part is a combination of previous individual noise components, which is used to create a time series, as shown in Equation (4). ARMA is a combination of both AR and MA [27].

The parameters of the model m and n are chosen on the basis of an auto correlation function (ACF) and a partial auto correlation function (PACF). The ACF provides a correlation between a value of a given time series with past values of the same series, whereas the PACF provides a correlation between a value of the time series with another value at a different lag. If the ACF is reduced to a minimum value after a few lags and PACF depicts a large cut-off after the initial value, the time series is said to be stationary. This is then finally confirmed by the Augmented Dickey Fuller (ADF) test, which is explained in [25]. A confidence level of 95% is assumed for this study, hence a p-value of less than 0.05 is a confirmation of stationarity.

The analysis of the time series data according to the ACF, PACF, and ADF, in addition to its conversion to a stationary time series followed by the fitting of an ARMA model, is discussed in the next section.

Finally, the same data is also fit with a linear regression model. The linear regression model is explained below. A comprehensive study on the use of linear regression along with an improved model for hourly forecasting can be found in [28].

$$Y = \beta_o + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k + \epsilon \quad (6)$$

where Y is the dependent variable, X_k are the independent variables, β_0 is the constant term, β_k is the coefficient corresponding to the slope of each independent variable, and ϵ is model's error, also known as residuals

2.2. Data Processing (Sliding Window)

While using the sliding window data processing approach for CNNs, a time series dataset is split as follows. The input data column is split into vectors consisting of an equal number of time steps. So, assuming the input data has 10 time steps, it is split into 5 vectors consisting of 2 time steps each. Then, these vectors are mapped to a label that is an output value from the training data. In this way, 5 vectors are mapped to 5 output values and 5 values are dropped, resulting in a reduced computational burden during the training of the model. The algorithm for the sliding window approach is presented in Algorithm 1.

Algorithm 1 sliding window

```
Procedure Variables (X, V, t)
i = 0, n = 0;                  # number of windows = n
K = [];                        # K is the set of windows extracted
While i + V ≤ length (X) do    #V is the length of the sliding window
K[n] = X [i . . . . (i + V − 1)};
i = i + t; n = n + 1;
end While
return F
end Procedure
```

While a general definition of the sliding window algorithm is presented here, every CNN model needs data to be prepared according to its structure. The sliding window for the CNN model in this study is applied to multivariate (the presence of more than one variable for every time step) time series data. In this case, every window determined by the algorithm has 2-time steps and its associated variables mapped to one output. The multi-headed CNN has 4 convolutional layers for every available input variable, hence the input time series is split into 4 univariate (one variable per time step) time series for each convolutional layer. Then, the sliding window algorithm is applied to each univariate series, and every window determined by the algorithm has 2-time steps and its associated variable mapped to an output.

The CNN-LSTM model reads input data in a different manner. In this case, the first step involves the application of the sliding window, where every window determined has 4-time steps, and then it is reshaped into 2 sub sequences containing associated variables and is mapped to outputs. The window is applied to a multivariate time series data.

2.3. Evaluation Metrics

The evaluation metrics chosen for this study were chosen based on recommendations of studies and reports in the field of solar PV output forecasting [6,14]. The metrics are the Root Mean Square Error (RMSE), Mean absolute error (MAE), and Mean Bias Error (MBE). RMSE is a metric that is widely used in forecast studies. According to [29], it is suitable for such data since it has the tendency to punish the largest errors with the largest effect, which the MAE and the MBE are unable to do. MAE is calculated as the average of the forecast errors. The MBE also calculates the average forecast errors but does not take in the absolute magnitude alone, this gives information regarding whether the model has a tendency to over or under forecast. The metrics are as follows:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2} \tag{7}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |e_i| \tag{8}$$

$$MBE = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{9}$$

$$e_i = y_{i(forecast)} - y_{i(observed)} \tag{10}$$

where $y_{i(forecast)}$ and $y_{i(observed)}$ represent the forecasted and observed observations at the i^{th} time step. e_i is the error at i^{th} time step. $i = 1, \ldots\ldots, N$ represents all the time steps within the data.

The evaluation metrics presented in the results section were calculated on the basis of original data after normalized prediction values were converted back using the inverse of the min–max scaling algorithm presented in Equation (1).

3. Results

All models were built on PYTHON using jupyter notebook. The deep learning tools that were used are TensorFlow and KERAS where the models were assembled. Additionally, Sci-kit learn and other basic Python libraries were used for data processing and data handling. The computer used for this purpose was equipped with an Intel®Core™ i5-4210 U CPU@ 1.70 GHz 2.40 GHz processor with an installed 8 GB of RAM operating Windows 10. It was also equipped with a 2048 MB GeForce 840M Nvidia graphics card. The training times for the CNN, Multi-CNN, and the CNN-LSTM models were 1364 s, 1657 s, and 3534 s, respectively. All architectures used the same data stretching over 6 years for model training and were trained for 100 epochs. The ARMA and MLR models were fit quite instantaneously, providing an advantage over the CNN based models with regard to the computational cost involved in model fitting. Once the models are fit, they are quite easy to use for the purposes of predictions. There is not any significant difference in terms of ease of usage amongst the statistical and CNN based techniques. Both models would need re fitting from time to time in order to take into account the changes in climate.

The data used for training the models were 6+ years' worth of data recorded from 1 March 2012 up to 31 December 2018. The validation split (test/train split) used was 20%, meaning that 80% of the data was used to train the CNN models and 20% was used to test them. The evaluation metrics obtained for 1 h, 1 day, and 1 week for both summer and winter months were obtained by testing

the model for the months of July and December in 2019, which was unknown to the training models. There was no validation split for the MLR and ARMA models. They were fit on to the whole data and were tested with the July and December data of 2019, same as for the CNN models.

Figure 4 represents the time series data used in this study for the ARMA model without the validation split, it is quite evident that data has seasonality where the peaks in power output are observed during the summer. Hence, the periodicity for this study would be taken as 12 months. A look at the ACF with 20 lags indicates significant correlation. In fact, a clear pattern is visible when the lags are further increased to 60 and above. The PACF of the data also does not show any large cut-offs after the initial value hence the time series is non-stationary and has to be converted to a stationary time series before the ARMA model is fit to the data.

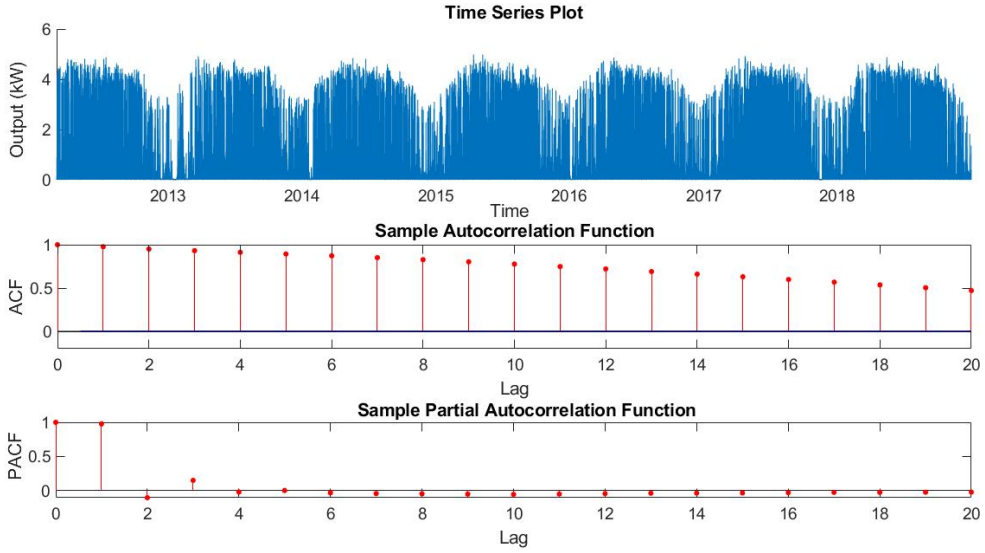


Figure 4. Solar panel output data with an auto correlation function (ACF) and a partial auto correlation function (PACF) analysis.

Figure 5 presents the differentiated time series. It can be seen from its characteristic that it fluctuates around zero, which is a defining characteristic for a stationary signal. Furthermore, in comparison with Figure 4, it can be seen that the ACF is not significant and also does not possess a trend, which is also the case for the PACF. In both cases, there is a sharp cutoff at 12, indicating seasonality at 12, which is in line with the selection of seasonality or periodicity at 12.

The ADF test made with the differentiated signal resulted in a p-value of 0.001, which confirms that the signal is stationary. Now the ARMA model parameters can be determined since the ACF and PACF are negligible beyond lag 2 therefore m and n could have a maximum value of 2. In this study, the m and n are taken as 1 and 2 and the following ARMA model is obtained.

Table 1 presents the ARMA model parameters that are used to predict solar output values for an hour, 1 day, and 1 week. The model ignores the constant value due to its high p-value. The evaluation metrics for the model predictions are presented in Table 2, and a comparison of the predictions as a result of model application with other methods is shown later on. Figure 6 presents the manner in which an appropriate forecasting model is obtained by different CNN architectures used. Figure 6a represents the loss value that is optimized in every epoch for the multi-headed CNN structure. It can be observed that for this model there is not any improvement in reduction of the loss function over many epochs of training. After an initial drop in the loss value it remains a constant, which means that training the architecture for a small number of epochs is sufficient for an accurate model. Figure 6b represents the loss value minimization for a simple CNN structure. In contrast to the multi-headed CNN structure, the loss minimization is more gradual, yet in a small number of epochs, a satisfying model is obtained. It has been noticed during several trials that, in the simple CNN structure, the loss

minimization keeps improving up to a 1000 epochs and more. However, the improvement in forecast accuracy is not significant vis-à-vis the time it takes to train the model for a high number of epochs.

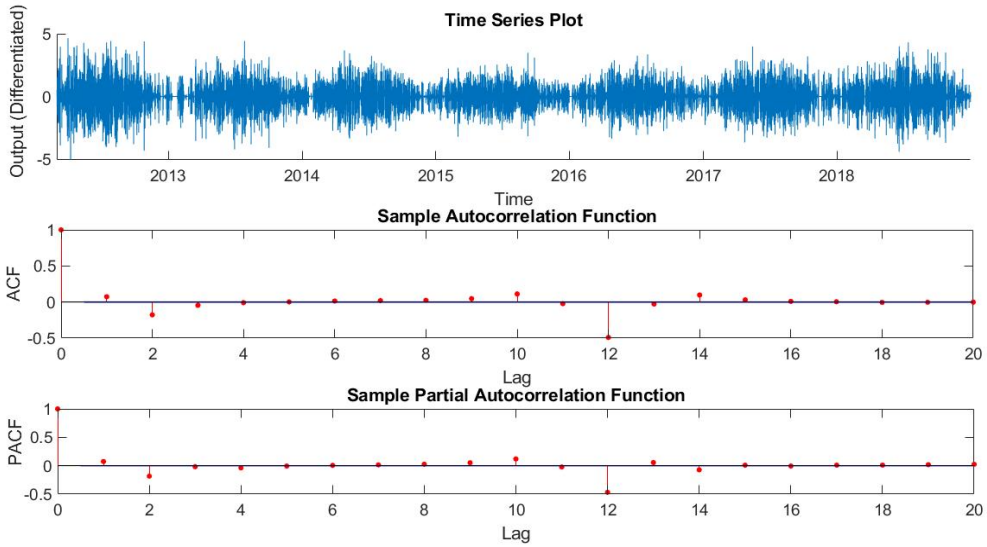
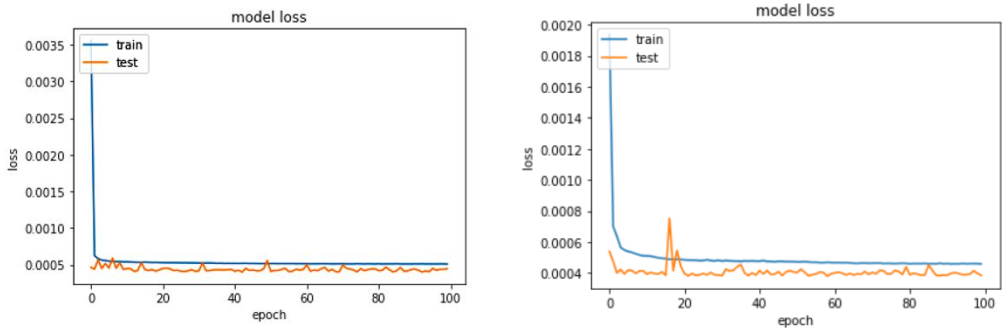


Figure 5. Differentiated output with ACF and PACF analysis.

Table 1. Autoregressive moving average (ARMA) model Parameters.

Parameter	Estimated Value	*p*-Value
$\varnothing_1$	0.208	0.000
θ_1	−0.125	0.000
θ_2	−0.197	0.000
Constant term	0.000	1.000
Variance	0.107	0.000

$\varnothing_1$—AR coefficient 1, θ_1—MA coefficient 1, θ_2—MA coefficient 2.



(**a**) (**b**)

Figure 6. Model fitting test and train loss minimization for (**a**) Multi-headed CNN (**b**) and Simple CNN structure.

Figure 7 represents the loss value minimization for the CNN-LSTM architecture. In comparison with Figure 6a,b, it can be observed that the model fitting takes slightly longer, yet it is completed with sufficient accuracy within 20 epochs. The model keeps improving with an increasing number of epochs, but it has been observed that, with a higher number of epochs (>500), the model tends to overfit with the loss curves of the test and train the data crossing over one another. For comparison purposes, keeping in mind the time for model fitting, 100 epochs was considered to be sufficient for all models.

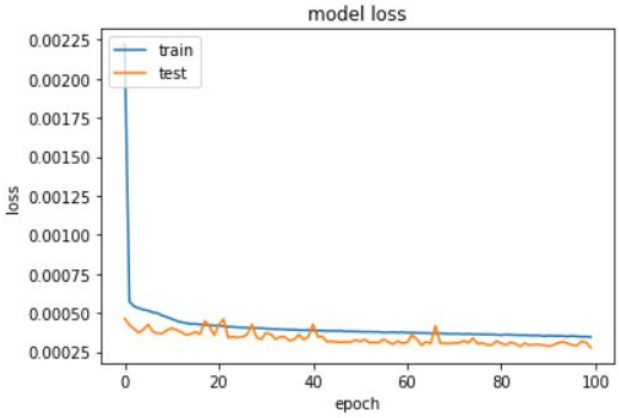


Figure 7. Model fitting test and train loss minimization for the CNN-LSTM network.

Table 2 presents the various metrics, as described in the previous section, which help understand the accuracy of the forecasts. The metrics are calculated for one hour (h), 1 day (D), and 1 week (W) in order to understand its consistency over the short and medium term. Table 2 is specifically for the summer months and the week in question is the 1st week of July 2019.

Table 2. Forecast metrics for the short and long term during the summer months.

Methods Used	RMSE (1 h)	MAE (1 h)	BIAS (1 h)	RMSE (1 D)	MAE (1 D)	BIAS (1 D)	RMSE (1 W)	MAE (1 W)	BIAS (1 W)
CNN-Simple	0.068	0.066	−0.066	0.051	0.033	−0.017	0.056	0.031	−0.016
Multi-headed CNN	0.169	0.169	−0.169	0.081	0.053	−0.036	0.080	0.050	−0.038
CNN-LSTM	0.053	0.053	−0.053	0.051	0.035	−0.025	0.045	0.030	−0.019
ARMA	0.046	0.043	+0.043	0.192	0.153	+0.153	0.244	0.134	+0.880
Multiple linear regression	0.477	0.474	−0.474	0.258	0.179	−0.149	0.258	0.146	−0.120

The RMSE, MAE, BIAS values are all in kW, hour—h, day—D, week—W.

It can be noticed that the values of RMSE, MAE, and the BIAS for the 1h forecast for all CNN-based methods are nearly the same. This is because the number of observations within an hour is just limited to four (due to 15 min time step) and the neural network methods take multiple inputs in order to make one prediction. In fact, for the CNN-LSTM model, the inputs needed are four for one prediction, hence the RMS, MAE, and the BIAS are the same for the 1 h forecast. It can be noticed from the BIAS value that, in the case of all methods being used except the ARMA model, there is a slight tendency to overpredict. The ARMA model has performed as good as any used CNN method and has the most accurate prediction followed by the CNN-LSTM.

For the 1-day forecasting (2 July 2019), it can be seen that the CNN-simple and the CNN-LSTM perform in a similar manner. The RMSE being around 0.051 kW. All methods have shown better performance than the MLR. The BIAS still indicates a tendency to over predict except for the ARMA model. It can be noticed that while the ARMA model provided very accurate results for the 1-h predictions, the RMSE value has increased considerably for the 1-day forecasts. The multi-headed CNN performs the worst amongst the CNN models.

For the 1-week forecasting (1st week of July) it can be noticed that the CNN-LSTM makes more accurate forecasts than the CNN-simple and multi-headed CNN models. In general, it can be noticed that with longer forecasts the accuracy metrics improve, indicating a general improvement or at least consistency in predictions for the CNN-based models. On the contrary for the ARMA model, the RMSE value has increased considerably from the 1-h and 1-day predictions. The BIAS in this case is to overpredict, except for the ARMA model.

Figure 8 presents the 1-day forecasting (2 July 2019) made by different algorithms explained previously. It can be inferred from the Figure that the most accurate forecasts for the day are made by the CNN and the CNN-LSTM models, wherein they almost overlap the actual values. These are followed by the multi-headed CNN model, which has a higher error in its predicted values, the ARMA model, which, despite being more accurate than MLR, underpredicts at moments of sharp changes.

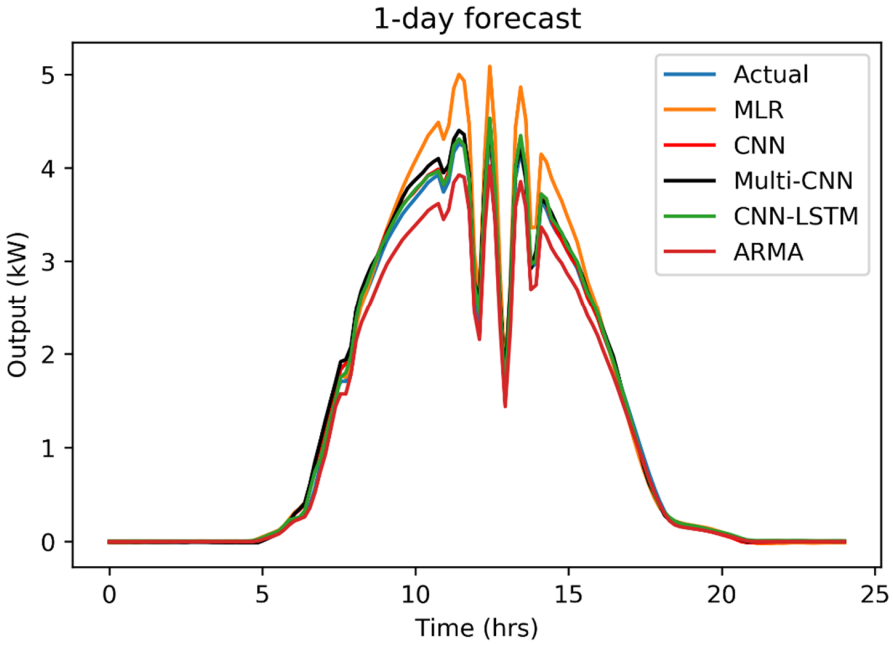


Figure 8. Comparison of different methods for the 1-day forecast (summer).

Figure 9 represents the forecasts made by different approaches used for a week (the 1st week of July). It can be noticed that for most parts the forecasts form CNN-LSTM, CNN-simple (CNN), and Multi-CNN closely match the actual values, though, at the peak, some inaccuracy can be noticed with the multi-CNN forecasts. The ARMA model slightly underpredicts, especially during peak output. The MLR is also inaccurate at the peaks.

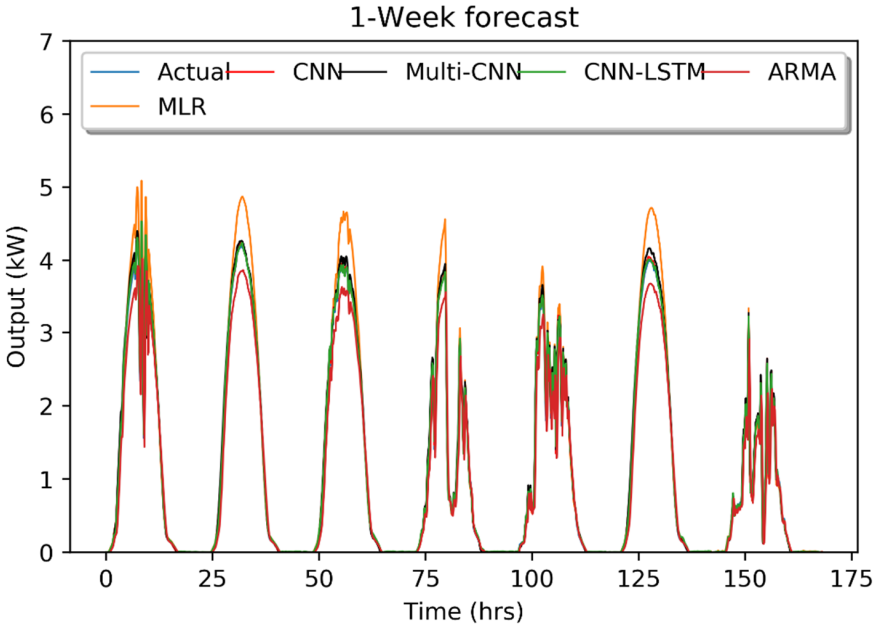


Figure 9. Comparison of different methods for the 1-week forecast (summer).

Table 3 provides the evaluation metrics of the models used during the winter months, it can be seen that the models have less accuracy for the 1-h predictions when compared to their own predictions during the summer. Again, the ARMA model performs with the highest accuracy for the 1-h predictions followed by the CNN-LSTM model. For the longer 1-day (28 December 2019) and 1-week periods (3rd week of December) it can be seen that the performance of all CNN-based models is very consistent with their performance during the summer months. The most accurate forecasting is made by the simple CNN model for 1-day (28 December 2019) forecasting, whereas for the 1-week forecasting it is the CNN-LSTM model. An intriguing observation between the summer and winter models is the fact that the difference between the RMSE and MAE values is higher during the winter period. The RMSE values are two times that of the MAE values and are higher for the CNN-based methods for the 1-day predictions, they are almost 3 times higher during the 1-week predictions, which is an indication that when errors are made during predictions they are higher in magnitude when compared to predictions in the summer because the RMSE has the inherent characteristic to give more weight to bigger errors.

Table 3. Forecast metrics for the short and long term during the winter months.

Methods Used	RMSE (1 h)	MAE (1 h)	BIAS (1 h)	RMSE (1 D)	MAE (1 D)	BIAS (1 D)	RMSE (1 W)	MAE (1 W)	BIAS (1 W)
CNN-Simple	0.466	0.451	+0.450	0.036	0.018	+0.005	0.104	0.039	+0.026
Multi-headed CNN	0.341	0.328	+0.328	0.038	0.019	−0.016	0.092	0.032	+0.023
CNN-LSTM	0.297	0.295	−0.295	0.056	0.029	+0.005	0.100	0.036	+0.022
ARMA	0.188	0.187	+0.187	0.040	0.018	+0.018	0.142	0.050	+0.023
Multiple linear regression	0.465	0.778	+0.778	0.092	0.059	−0.011	0.040	0.098	+0.032

The RMSE, MAE, and BIAS values are all in kW, hour—h, day—D, week—W.

Figure 10 presents the forecasting made for 1 day (28 December 2019) during the winter. In comparison with the 1-day prediction for the summer, the ARMA and MLR forecast values have significantly improved, and all methods predict quite accurately.

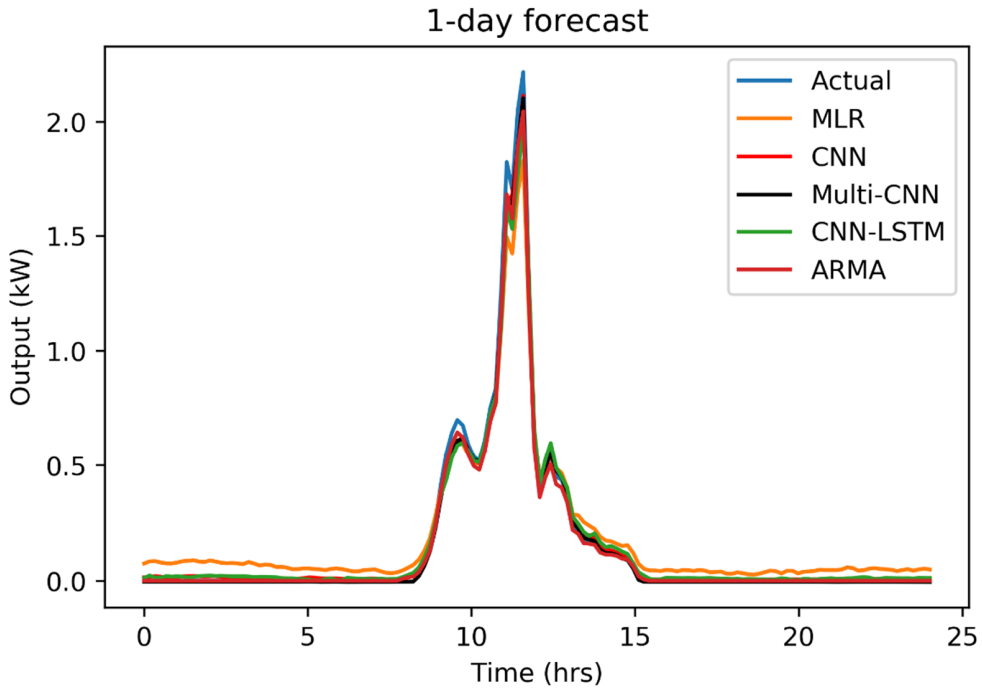


Figure 10. Comparison of different methods for the 1-day forecast (winter).

Figure 11 represents the forecasting made by different approaches used in this study for the 3rd week of December, it can immediately be noticed that the output values are lower when compared with the summer week, with one day having almost no output. Except for the MLR, it can be seen that all models predict quite closely to the real values, with all of them underpredicting a little during peak power output.

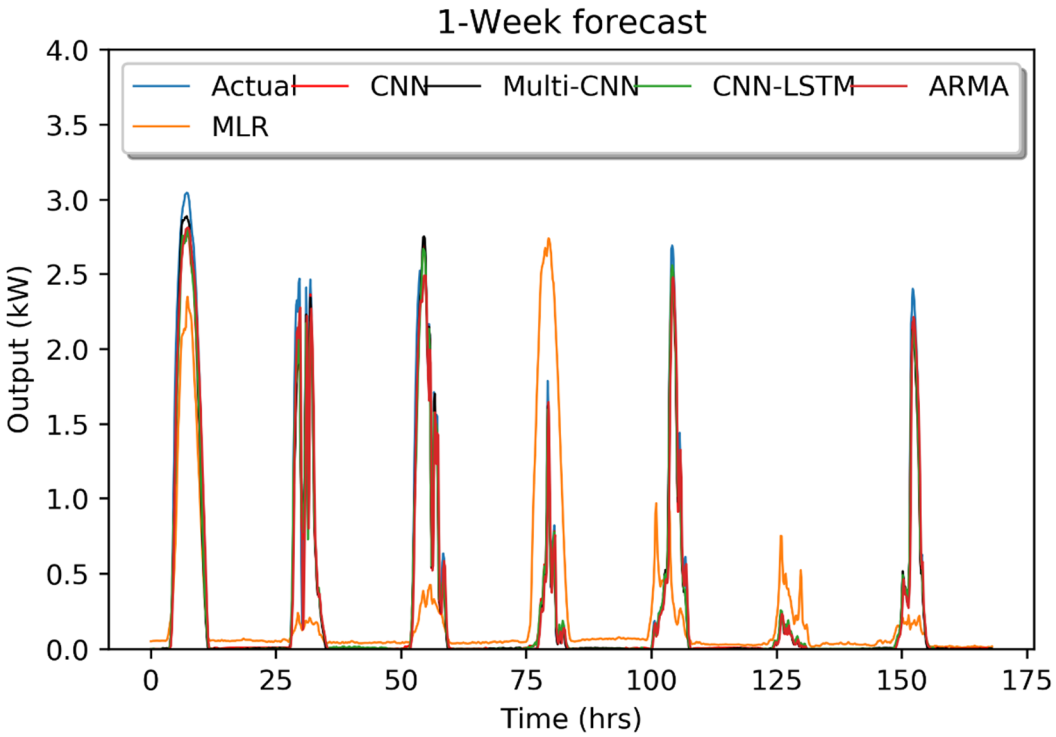


Figure 11. Comparison of different methods for the 1-Week forecast (winter).

4. Discussion

This paper presents a forecasting approach using deep learning neural networks. The neural network structures, used primarily for image recognition, have been adapted to handle time series data with a seasonal characteristic. In order to make this possible, a data processing approach, such as the sliding window algorithm, has been used. A comparison between the performance of different possible structures of the neural network has been carried out along with a multiple linear regression and ARMA model. It has been noticed that the CNN-simple and the CNN-LSTM methods perform best for all 1-h, 1-day and 1-week predictions, with the CNN-LSTM providing better results on certain occasions. The ARMA model performed exceptionally for the 1-h forecasts. The forecasting was carried out for 1 h, 1 day, and 1 week with the function of electricity markets in mind. From the accuracy metrics such as RMSE, MAE, and BIAS it can be concluded that the forecasting algorithms perform satisfactorily. Since its performance has been tested in the short and medium term at the university location, it will be followed by rigorous testing at other locations in order to establish its applicability across geographical regions with different seasonal characteristics. Future work in this regard includes investigating the performance of other architectures that possess more abstraction (the level of complexity of the neural network). Abstraction can be increased by increasing the number of convoluted layers, which may or may not improve the accuracy metrics. It also has an effect on the training time for fitting an appropriate model. Furthermore, a different combination of CNNs and RNNs could be considered. Additionally, the effect of clustering is to be explored. The CNN-based models were fit, and, on the whole, the model performs quite uniformly across all seasons, which we believe is due to the fact that during the training method a set of values from the past is used to train

the model at every step. This enables the model to capture seasonality as the weather-related variables from past values are clearly season-dependent. The approach is similar in the ARMA model since it also uses past values, but it is important to remember that the ARMA class of models are applicable only to univariate data (in this case the output values of the PV plant). Moreover, this study is a part of building a stochastic Energy Management System for microgrids, hence they will be used as inputs for optimization algorithms managing the EMS.

Author Contributions: Conceptualization and methodology were developed by authors V.S. and P.J., software, validation, formal analysis, investigation was done by V.S., resources, data curation and writing—original draft preparation was done by V.S. and J.R., writing—review and editing, visualization, supervision, project administration was done by author P.J. and Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding from the Chair of Electrical Engineering Fundamentals (K38W05D02), Wroclaw University of Technology, Wroclaw, Poland.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. UNCCS. Climate action and support trends, United Nations. *Clim. Chang. Secr.* **2019**, 7.
2. Zang, H.; Cheng, L.; Ding, T.; Cheung, K.W.; Liang, Z.; Wei, Z.; Sun, G. Hybrid method for short-term photovoltaic power forecasting based on deep convolutional neural network. *IET Gener. Transm. Distrib.* **2018**, *12*, 4557–4567. [CrossRef]
3. Alrashidi, M.; Alrashidi, M.; Pipattanasomporn, M.; Rahman, S. Short-Term PV Output Forecasts with Support Vector Regression Optimized by Cuckoo Search and Differential Evolution Algorithms. In Proceedings of the 2018 IEEE International Smart Cities Conference, Kansas City, MO, USA, 16–19 September 2019; pp. 1–8. [CrossRef]
4. Abuella, M.; Chowdhury, B. Solar power probabilistic forecasting by using multiple linear regression analysis. In Proceedings of the IEEE Southeastcon, Fort Lauderdale, FL, USA, 9–12 April 2015; pp. 1–5. [CrossRef]
5. Chugh, A.; Chaudhary, P.; Rizwan, M. Fuzzy logic approach for short term solar energy forecasting. In Proceedings of the 2015 Annual IEEE India Conference (INDICON), New Delhi, India, 17–20 December 2015; pp. 1–6. [CrossRef]
6. Yang, D.; Kleissl, J.; Gueymard, C.A.; Pedro, H.T.C.; Coimbra, C.F.M. History and trends in solar irradiance and PV power forecasting: A preliminary assessment and review using text mining. *Sol. Energy* **2018**, *168*, 60–101. [CrossRef]
7. Sun, Y.; Venugopal, V.; Brandt, A.R. Short-term solar power forecast with deep learning: Exploring optimal input and output configuration. *Sol. Energy* **2019**, *188*, 730–741. [CrossRef]
8. Ryu, A.; Ito, M.; Ishii, H.; Hayashi, Y. Preliminary Analysis of Short-term Solar Irradiance Forecasting by using Total-sky Imager and Convolutional Neural Network. In Proceedings of the 2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia), Bangkok, Thailand, 20–23 March 2019; pp. 627–631. [CrossRef]
9. Ghimire, S.; Deo, R.C.; Raj, N.; Mi, J. Deep solar radiation forecasting with convolutional neural network and long short-term memory network algorithms. *Appl. Energy* **2019**, *253*, 113541. [CrossRef]
10. Dong, N.; Chang, J.-F.; Wu, A.-G.; Gao, Z.-K. A novel convolutional neural network framework based solar irradiance prediction method. *Int. J. Electr. Power Energy Syst.* **2020**, *114*, 105411. [CrossRef]
11. Agüera-Pérez, A.; Palomares-Salas, J.C.; de la Rosa, J.J.G.; Florencias-Oliveros, O. Weather forecasts for microgrid energy management: Review, discussion and recommendations. *Appl. Energy* **2018**, *228*, 265–278. [CrossRef]
12. Alvarado-Barrios, L.; del Nozal, Á.R.; Valerino, J.B.; Vera, I.G.; Martínez-Ramos, J.L. Stochastic unit commitment in microgrids: Influence of the load forecasting error and the availability of energy storage. *Renew. Energy* **2020**, *146*, 2060–2069. [CrossRef]
13. Alamo, D.H.; Medina, R.N.; Ruano, S.D.; García, S.S.; Moustris, K.P.; Kavadias, K.K.; Zafirakis, D.; Tzanes, G.; Zafeiraki, E.; Kaldellis, J.K.; et al. An Advanced Forecasting System for the Optimum Energy Management of Island Microgrids. *Energy Procedia.* **2019**, *159*, 111–116. [CrossRef]

14. Luukkonen, P.; Bateman, P.; Hiscock, J.; Poissant, Y.; Dignard-Bailey, L. International Energy Agency Co-Operative Programme on Photovoltaic Power Systems. 2013. Available online: http://www.cansia.ca/sites/default/files/201306_cansia_2012_pvps_country_report_long.pdf (accessed on 29 October 2019).
15. Paulescu, M.; Paulescu, E.; Gravila, P.; Badescu, V. Weather Modeling and Forecasting of PV Systems Operation. *Green Energy Technol.* **2013**, 325–345. [CrossRef]
16. Lorenz, E.; Remund, J.; Müller, S.C.; Traunmüller, W.; Steinmaurer, G.; Pozo, D.; Ruiz-Arias, J.A.; Fanego, V.L.; Ramirez, L.; Romeo, M.G.; et al. Benchmarking of Different Approaches to Forecast Solar Irradiance. *Eur. Photovolt. Sol. Energy Conf* **2009**, 4199–4208. [CrossRef]
17. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Cambridge, UK, 2006. [CrossRef]
18. Bisong, E. Convolutional Neural Networks (CNN). In *Building Machine Learning Deep Learning Models Google on Cloud Platform*; Apress: Berkeley, CA, USA, 2019; pp. 423–441. [CrossRef]
19. Heaton, J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genet. Program. Evolvable Mach.* **2018**, *2018 19*, 305–308. [CrossRef]
20. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* arXiv:1412.6980, 2014.
21. Mathur, P.; Ayyar, M.P.; Shah, R.R.; Sharma, S.G. Exploring classification of histological disease biomarkers from renal biopsy images. In Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, 7–11 January 2019; pp. 81–90. [CrossRef]
22. Hu, P.; Tong, J.; Wang, J.; Yang, Y.; de Oliveira Turci, L. A hybrid model based on CNN and Bi-LSTM for urban water demand prediction. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation, Wellington, New Zealand, 10–13 June 2019; pp. 1088–1094. [CrossRef]
23. Fu, Q.; Niu, D.; Zang, Z.; Huang, J.; Diao, L. Multi-stations ' Weather Prediction Based on Hybrid Model Using 1D CNN and Bi-LSTM. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 3771–3775.
24. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*; Cornell: New York, NY, USA, 2015; pp. 802–810.
25. 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 Proceedings of the 2019 IEEE 9th annual computing and communication workshop and conference (CCWC), Las Vegas, NV, USA, 7–9 January 2019; pp. 114–119. [CrossRef]
26. Alsharif, M.H.; Younes, M.K.; Kim, J. Time series ARIMA model for prediction of daily and monthly average global solar radiation: The case study of Seoul, South Korea. *Symmetry* **2019**, *11*, 240. [CrossRef]
27. Ghofrani, M.; Suherli, A. Time Series and Renewable Energy Forecasting. *Intech* **2016**, *13*, 77–92. [CrossRef]
28. Trigo-González, M.; Batlles, F.J.; Alonso-Montesinos, J.; Ferrada, P.; del Sagrado, J.; Martínez-Durbán, M.; Cortés, M.; Portillo, C.; Marzo, A. Hourly PV production estimation by means of an exportable multiple linear regression model. *Renew. Energy* **2019**, *135*, 303–312. [CrossRef]
29. Madsen, H.; Pinson, P.; Kariniotakis, G.; Nielsen, H.A.; Nielsen, T.S. Standardizing the performance evaluation of short-term wind power prediction models. *Wind Eng.* **2005**, *29*, 475–489. [CrossRef]