Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

A Multivariate Prediction Model for Short-Term Photovoltaic Plant Generation Using Bi-LSTM and CNN

Master Thesis on Renewable Energies

Supervised by:
Cecilio Hernández Rodríguez[1]
David Cañadillas Ramallo[2]

Author:
Kiril Ivanov Kurtev

September 6, 2022

[1]ULL, Department of Applied Physics. Renewable Energies and Optics
[2]Energy Research and Intelligence Solutions (EnergyRIS)

Abstract

The short-term prediction of the energy produced by a photovoltaic plant is a widely studied topic, and it is an important issue for the stability of the grid and its correct operation, as well as for reducing the operating costs and increasing the lifetime of the elements that make up it. The creation of a tool to more accurately predict the solar generation of the PV plant, specifically the prediction of ramps 5-10 minutes in advance. In this work, a multivariate prediction model is presented that combines images, historical production data, and solar position at each moment. The model consists of two parts: image processing, with a convolutional neural network (CNN) and time series processing using a Bidirectional Long Short-Term Memory (Bi-LSTM) capable of detecting long-term nonlinear features. CNNs will be trained to automatically detect the relationship between the images taken of the sky and cloud movement and the current power of the solar array. Then, the recurrent neural networks (RNNs) created will be used to give rise to a 5-minute prediction and a 10-minute prediction. The prediction results are compared using different error metrics, like skill score and the mean squared error (RMSE).

Contents

List of Figures

1 Introduction

Despite the economic impact of COVID-19, the installed capacity of global renewable energy increased by more than 260 GW during 2020, and by around 257 GW during 2021. During the latter, the installed record solar energy capacity was reached with 133 GW, representing almost 20% of the installed capacity worldwide [1]. However, the increase in the capacity of renewable energies, specifically photovoltaic energy, leads to taking into account some problems related to their integration. According to K.N. Nwaigwe et al. [2] large-scale PV projects are mostly located far from urban centers and often require transmission lines to transport electricity over long distances to where it is actually used. This requires greater investment in transmission line construction and often results in "line losses" as some of the energy in transit is converted to heat and lost. Some of the most significant challenges associated with solar power integration are the issues of voltage stability, frequency, and power quality.

As mentioned above, in addition to voltage and frequency power quality problems, another important problem is harmonics. The harmonics problem stems mainly from the power inverters used to convert the renewably generated DC voltage to AC. Harmonics are created by certain loads that introduce frequencies that are multiples of 50 or 60 Hz and can cause the equipment to malfunction.

Photovoltaics is also the only technique for generating solar power that does not result in inertial power generation. Inertia in electrical systems refers to the energy stored in large rotating generators and some industrial motors, which gives them the tendency to keep turning. This stored energy can be especially valuable when a large power plant fails, as it can temporarily compensate for the power loss of the failed generator. This temporary response allows the mechanical systems that control most power plants to detect and respond to failures. The absence of this stored energy proves to be a problem for large-scale grid integration.

Another important challenge is the variability of irradiance, which is the focus of this paper. The amount of PV system generation depends on the amount of irradiance at a given location and time. [3]

Thus, both under- and over-generation can lead to instability in the grid, producing an abrupt decrease and increase, respectively. A sequence of solutions to this challenge involves [4]:

- Install solar power in a wide geographic area to minimize the impact of generation variability due to local cloud cover.
- Shifting Electricity Supply and storing Excess Energy for Later Use
- Shift electricity demand by encouraging customers to use electricity when it is available.

- Use better forecasting tools to more accurately predict when solar generation might decline to the minimum penetration capacity.

The present work will focus on the latter, the creation of a tool to more accurately predict the solar generation of a photovoltaic plant, specifically the prediction of ramps 5-15 minutes in advance.

The problem of forecasting the future of photovoltaic energy has been extensively studied in the literature. [5, 6]. A wide variety of solutions have been proposed for medium- and long-term forecasting. But according to Zhang et. al [7] while medium- and long-term forecasting are useful tasks, it is also important that we consider the problem of forecasting energy production on a minute scale. Thus, short-term forecasting is critical when managing smart grid operations, such as system integration, ensuring power continuity and managing ramp rates, etc.

A fundamental pillar of most techniques is the need for meteorological data, which are obtained from weather stations or satellites capable of providing the necessary data for prediction. The problem in obtaining data from numerical models and satellites lies in the low spatial and temporal resolution of such data. Most meteorological stations are usually too far away to be reliable, whereas geostationary satellites have relatively limited resolution. [7]. Lipperheide et al. [8] have pointed out this limitation, in addition to assessing the importance of cloud wind speed for the prediction of irradiance.

One solution to this problem is to capture local weather conditions in the solar array with high spatial and temporal resolution using a regular video camera pointed at the sky and installed near the photovoltaic plant.

But although they are cheap and easy to install, they do not explicitly provide relevant meteorological information: rather, they provide images of the sky that must be analyzed to determine what the relationship is between the images and the PV output.

For the analysis of the images, artificial neural networks (ANN) will be trained to be able to automatically detect the relationship between the images taken of the sky and the current power of the photovoltaic plant. For this purpose, convolutional neural networks (CNNs) will be used. CNNs can learn to distinguish details in an image, such as the area occupied by clouds in the sky dome or the position of the sun [9]. This property of this ANN is of special interest in the field of irradiance and photovoltaic energy prediction. [7, 10].

Another important challenge of the artificial neural network is to capture the temporality of historical data. Due to its complexity and nonlinearity it is a difficult task to achieve using regression methods, such as linear regression and the autoregressive integrated moving average model (ARIMA), which were the first to be used for time series prediction. [11, 12].

To solve the temporality problem, recurrent neural networks (RNNs) are used. Feedforward neural networks pass data from input to output, whereas recurrent networks have a feedback

loop in which data may be returned to the input at some point before being fed back for further processing and final output. According to Youru et al. [13] "in the case of a longer sequence, problems such as the disappearance of the gradient in the training of the RNN" usually appear due to the use of common activation functions, such as the sigomoid and hyperbolic tangent, limiting the accuracy of the models. To avoid such problem, in the present work we will use an ANN based on long short-term memory (LSTM) cells, which include some characteristic gates to forget and remember information, and thus modify the state of the cells depending on the information flowing through them. In this type of neural network, the information flows only in one direction; in this case, a bidirectional layer has been added to each of the cells so that the flow of information, as the name of the layer itself indicates, flows in both directions, improving the learning process and showing better results. [14].

Thus, the present work will use CNNs combined with another bidirectional LSTM (Bi-LSTM) that will process historical production data and the solar position at each moment to give rise to a short-term prediction (5 - 10 minutes) of a PV plant. Therefore, it is a multivariate model combining images, historical production data, and solar position that will focus on short-term prediction of photovoltaic energy but with emphasis on the ramps produced by the fall or rise of production in a short period.

2 Model

The multivariate prediction model proposed in this work consists of the combination of CNN and Bi-LSTM. Firstly, the prediction process of the model will be explained, and secondly, the convolutional model and the Bi-LSTM model will be explained in greater depth. In addition, the way to combine both models to obtain the final result will be explained in detail.

2.1 Model Framework

The model consists of two parts: image processing, with a CNN, and time series processing using a Bi-LSTM capable of detecting long-term nonlinear features.

The two-dimensional convolution will allow the extraction of features from the images, which will have a size of 64x64 pixels and three channels (RGB). To give a temporal sense to the "N" images to be used as input for the prediction, they will be concatenated to obtain something similar to a video. Thus, the final input size for the CNN will be Nx64x64x3. The CNN layers will have an envelope layer called the "Time-Distributed Layer" applied to them, which is applied to the video sequences to give a temporal sense to each video frame. After each convolution block, a batch normalization layer is applied that applies a transformation that keeps the mean of the output close to 0 and the standard deviation of the output close to 1. The layer normalises

its output using the mean and standard deviation of the current batch (see Section 3.2). After the normalization layer, a maximum pooling layer is added, which is commonly used after a convolution (see Section 2.3.2). The convolution layer, the normalisation layer and the maximum pooling layer make up the convolution block.

The model of historical production data and solar position, as mentioned above, will be composed of LSTM layers, which are able to give temporality to the input data. In addition, the LSTM blocks will have a bidirectional envelope layer, which allows the information flow, as the name itself indicates, to flow in both directions, decreasing the learning time. Each Bi-LSTM block is followed by dropout layers.

The term *dropout* refers to the elimination of units (both hidden and visible) in a neural network. Simply put, dropout refers to ignoring units (i.e., neurons) during the training phase of a certain set of neurons that are chosen at random. This is done to eliminate the noise present at the output of each Bi-LSTM layer or block.

Finally, both models are connected to a dense layer with as many neurons as output we are going to want for prediction. That is: if we want to predict 5 values at the output we will have 5 neurons in the dense layer.

The idea of attention has been around for a long time in machine learning and artificial intelligence, particularly in computer vision [15]. The concept of attention in how human brains process an enormous amount of visual and auditory input served as the inspiration for the term "attention", just as it did for the term "neural network". The deep learning layers, called attention layers, inspire the concept of attention. A useful summary on attention in deep learning is given by Luong et al. [16]. According to empirical evidence, attention layers are absolutely necessary when modeling sequences, such as language [17]. Attention layers are most frequently found in transformer neural networks, which simulate sequences.

Fundamentally, attention layers are a weighted mean reduction. Simply computing a mean involves giving each factor that affects the mean some sort of weight. Attention lowers the rank of an input tensor because it is a mean. Unlike other layers, attention requires three inputs, whereas other deep learning layers just require one or possibly two. The query, the values, and the keys are the names of these inputs. When the query has one key and the keys and values are the same, these inputs are frequently identical.

2.2 Long short-term memory

Long-term memory networks, usually called "LSTMs", are a special type of RNNs, capable of learning long-term dependencies. They were introduced by Hochreiter et al. [18].

LSTM networks were specifically designed to overcome the problem of long-term dependence faced by RNN recurrent neural networks (due to the leakage gradient problem). LSTMs have

feedback connections that differentiate them from more traditional neural networks. This property allows LSTMs to process entire sequences of data (e.g., time series) without treating each point in the sequence independently, but retaining useful information about previous data in the sequence to help process new data points. Thus, LSTMs are especially good for processing data sequences such as text, speech, and time series in general.

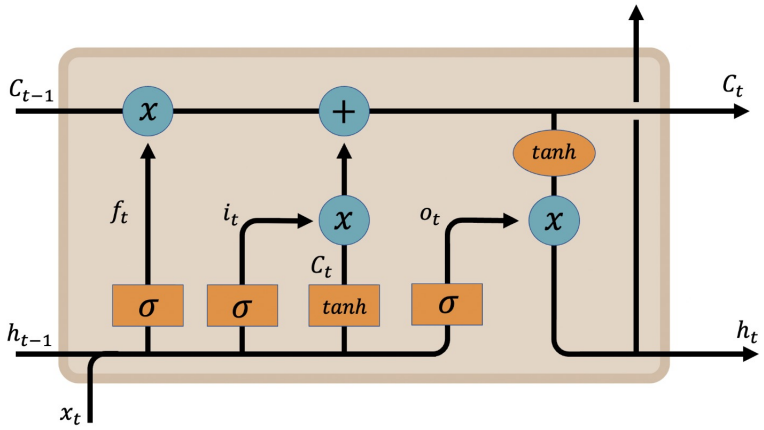

Figure 1: LSTM structure.

As shown in Figure 1, the core of this structure is the horizontal line across the top, which represents the state of the cell. This state can be changed by a variety of actions, which are described in more detail below. Cell gates are used that regulate these actions by modifying the state of the system based on the values they traverse. The first cell gate found in the structure is the so-called "forget gate layer", which is controlled by a sigmoid function (only values between 0 and 1 are found at its output) that will change the state of the system by looking at the values of h_{t-1} y x_t, representing the hidden state $t-1$ (responsible for remembering short-term features) and the input in t, respectively.The sigmoid function's output will be a value between 0 and 1, where 0 indicates that the cell's state does not change and 1 indicates that it remembers all of the shown input. The forget-gate operation is thus given by:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{1}$$

where W_f and b_f, are the weight and bias of the forget layer, which will be updated at each iteration to minimize the cost function.

The LSTM's next phase will be to determine what data will be saved to update the cell state. This process may be split into two parts: a hyperbolic tangent layer and a sigmoid layer. The value to be updated will be chosen in the sigmoid layer, while a vector of potential candidates is formed in the hyperbolic tangent layer. Then, these two parts are combined to create an update

to the cell state. It should be noted that the inputs for the sigmoid layer, the hyperbolic tangent and the forget gate have the same input: hidden previous state and new input data. Thus, the mathematical expression that models this behavior is as follows.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned} \tag{2}$$

where W_i, W_C and b_i, b_C are the weights and bias, respectively, to be modified in each iteration in these two layers, similarly to the forget layer.

Thus, the next thing will be to update the state of the system with the output of the previous layers, which is given by:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{3}$$

Finally, once the cell state has been updated, it remains to update the hidden state, which will be given by the step of the previous hidden state, the current input, and the updated system state:

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t \cdot \tanh\left(C_t\right) \end{aligned} \tag{4}$$

2.3 Convolutional neural network

The great utility of CNNs is due to their ability to reduce the number of system parameters in neural network image processing. To put that into perspective, there would be 64x64x3 weight parameters for a single neuron in a Multilayer Perceptron (MLP) neural network if we wanted to link a 64x64-pixel picture with a depth of three colors (RGB) to that neuron. Only two neurons would have 12,288 parameters if they were coupled as the input for the same picture. As has been done in several studies, as in [19], it would make sense to link the picture to a total of 64x64 neurons to assess height and width to analyze the complete image. This would result in a total of 12,582,912 connections.

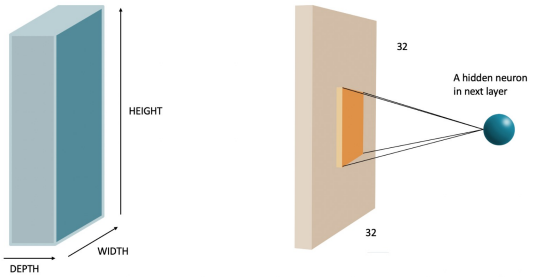

Figure 2: Image representation (left) and kernel representation (right).

Therefore, a more efficient method is instead of looking at every pixel of an image, to look at local regions (see Fig. 2) [20, 21].

It is still possible to make another consideration to reduce the number of parameters, that is, to include another layer that shares the wieghts of the previous layer. This drastically reduces the number of parameters [22].

Thus, with these considerations two benefits are obtained: firstly, the number of parameters is reduced; secondly, an even more interesting concept is that setting the weights for the 5x5x3 local connections is similar to sliding a window over the image mapping the values to the next layer. In essence, this process is a convolution. It allows features to be detected and recognised regardless of their position in the image [9].

2.3.1 Convolutional layers

In convolutional neural networks, the filters are not decided, but only the number of kernel filters in each convolutional layer is provided. Filter values are automatically learned by the neural network through the training process, and filters that result in features that are most efficient for the particular classification or detection are automatically learned. The values of the kernel filters are the weights in the particular CNN and those values are learned rather than decided.

In Fig. 3 an image of 200x200 pixels is shown, although in practice 64x64 pixels will be used, on which the convolution will be performed.

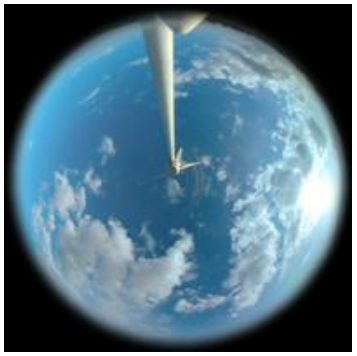

Figure 3: Image on which the two-dimensional convolution is performed. Resolution of 200x200 pixels.

Fig. 4 shows 32 filters with different weights for a single channel (there are three channels: red, green and blue (RGB)), these will be applied to the previous image for cloud detection.

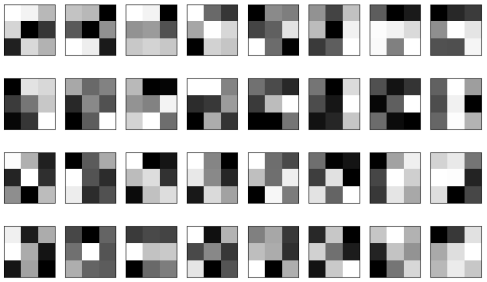

Figure 4: First 32 filters (3x3) with different weights applied in the first convolution block. The weights have random values between 0 (black) and 1 (white) when the training is initialized.

But what is more interesting to observe is the output produced by these filters at the output of the convolution block. Fig. 5 shows the response to the different filters applied after the first convolution. As mentioned above, the weights of the kernel will be adjusted during training. In Fig. 5 it is also observed that some filters have been able to detect the cloud covered part, as, for example, in the fifth filter in the first row.

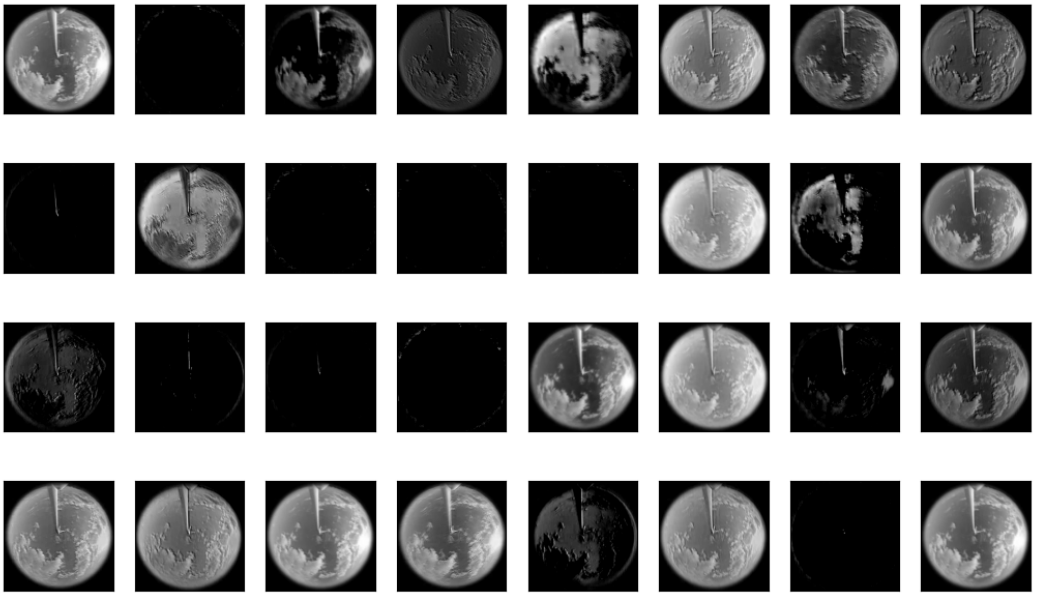

Figure 5: Response after the first convolution for each one of the filters mentioned above. In the first convolutional block 32 filters will be applied. Resolution 200x200.

2.3.2 Pooling layer

A common practice in convolutional neural networks is to use pooling layers after convolution layers. This strategy allows reducing the image size, in addition to being a translationally invariant operation, just like convolutions. For max pooling and average pooling, Boureau et al. [23] provided a detailed theoretical analysis of their performances. Max pooling has also been determined to have a faster convergence speed and better generalization than average pooling [24].

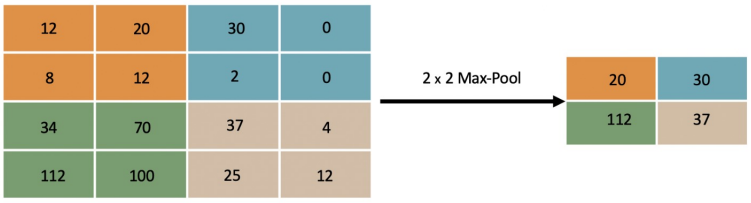

Figure 6: Max pooling layer operation.

As shown in Figure 6, the operation applied by this layer is to extract the maximum value of the smaller windows on the convolution output, reducing the input size for the following layers. On a 4x4 pixel window, if the 2x2 max pooling operator is applied to it, it reduces the final image to 2x2 with a stride of 2. The stride is nothing more than the displacement of the operator window over the window to which it is applied.

An example of the current model can be seen in Fig. 7. A reduction in resolution is observed after three convolution blocks. With a max pooling operator of 2x2 with stride of 2, the original 200x200 image has been reduced to 25x25.

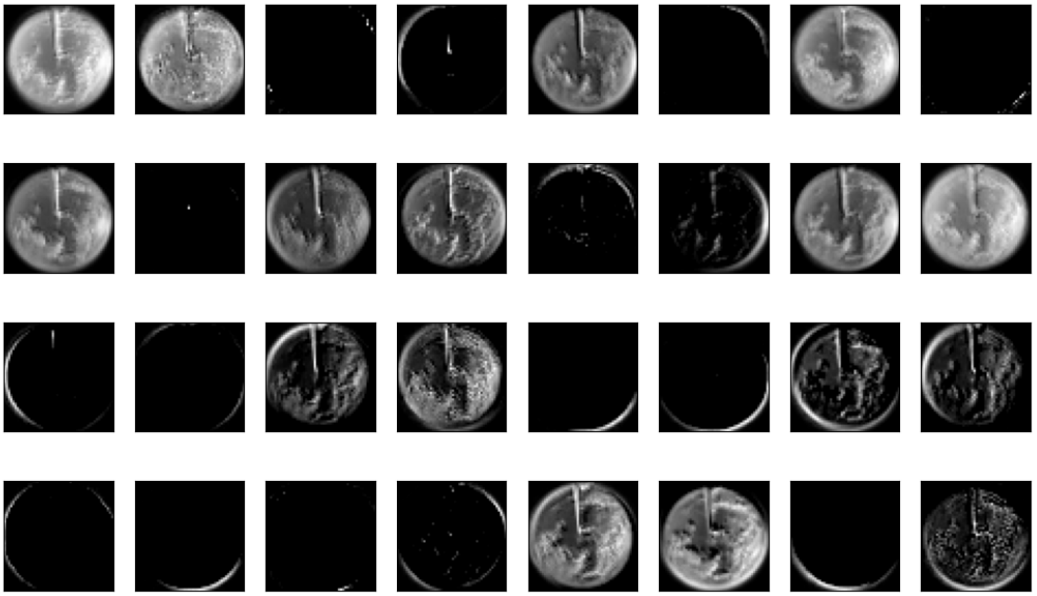

Figure 7: Response after the third convolution for each one of the filters mentioned above. In the third convolutional block 128 filters will be applied. Resolution 50x50.

2.3.3 Normalization layer

One strategy that accelerates the training of any neural network is normalization of the output to the output of any network architecture. It is also a technique to reduce the covariate shitf, which causes a reduction in the accuracy of the model when it is run to predict data that varies greatly from the training data. Models are typically trained in local or off-line environments on a sample of labeled training data. It is not uncommon for the input distribution in a dynamic environment to be different from that in a controlled training environment. As commented by Ioffe et al. [25], "training deep neural networks is complicated by the fact that the distribution of each layer's input is different from the controlled training environment. That the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating non-linearities. Our method draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch."

3 Experimental setup

Model training and data preprocessing has been carried out on an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz with 40 CPUs and 128 GB of RAM.

3.1 Data

- Historical data

  Historical data used for model training have been taken every 10 seconds since 2020. These are active, reactive, and apparent power data, voltage at different points of the line, and potential difference between the different points. Finally, the data on final, reactive and apparent power, voltage at different points of the line, and the potential difference between the different points have been discarded. Only active power has been used and elevation and azimuth angles have been added for the multivariate LSTM model, as can be seen in Figure 8.

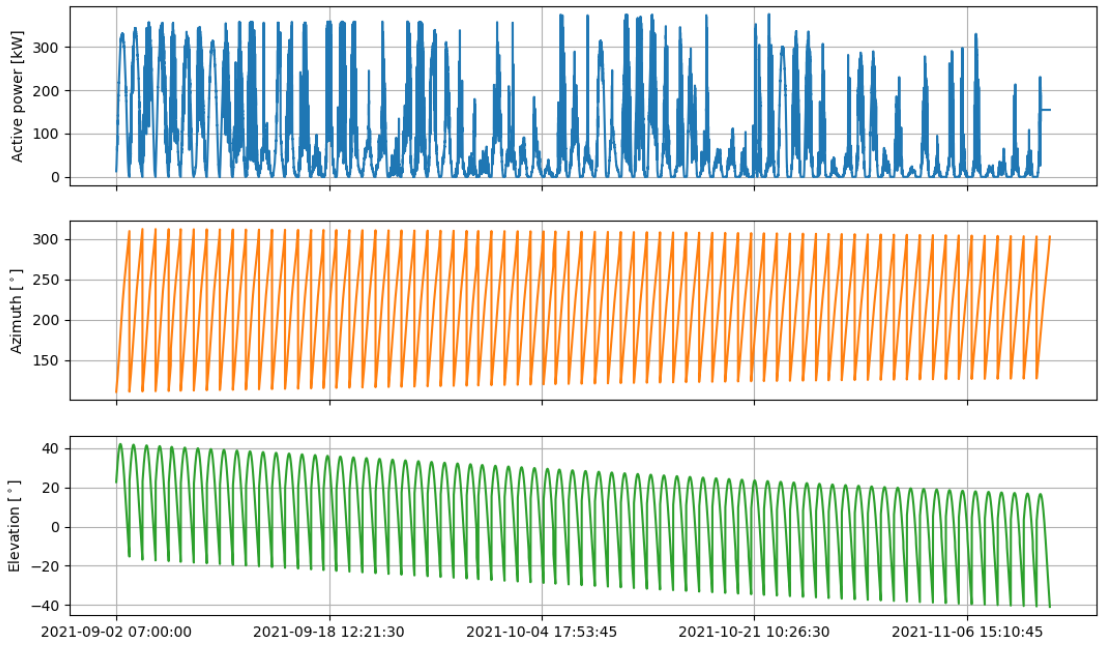

Figure 8: Data used for training. Active power (up). Azimuth (middle). Elevation (down).

It is worth noting that the power drop/raise can be significant between two consecutive time steps, as can be seen the Figure 9. In Figure 10, a single day drop / increase is shown

between consecutive time steps, where a drop of 200 kW can be seen in a 15-second interval, which justifies the need to predict the ramps produced by the irradiance drops.

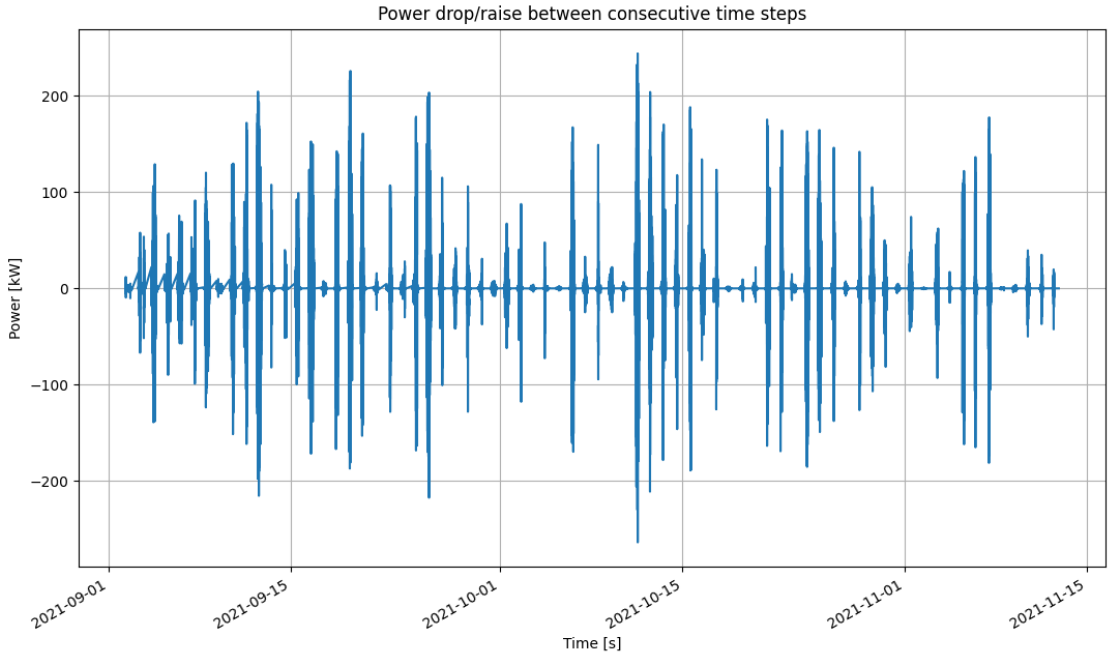

Figure 9: Power drop/raise of the whole data set.

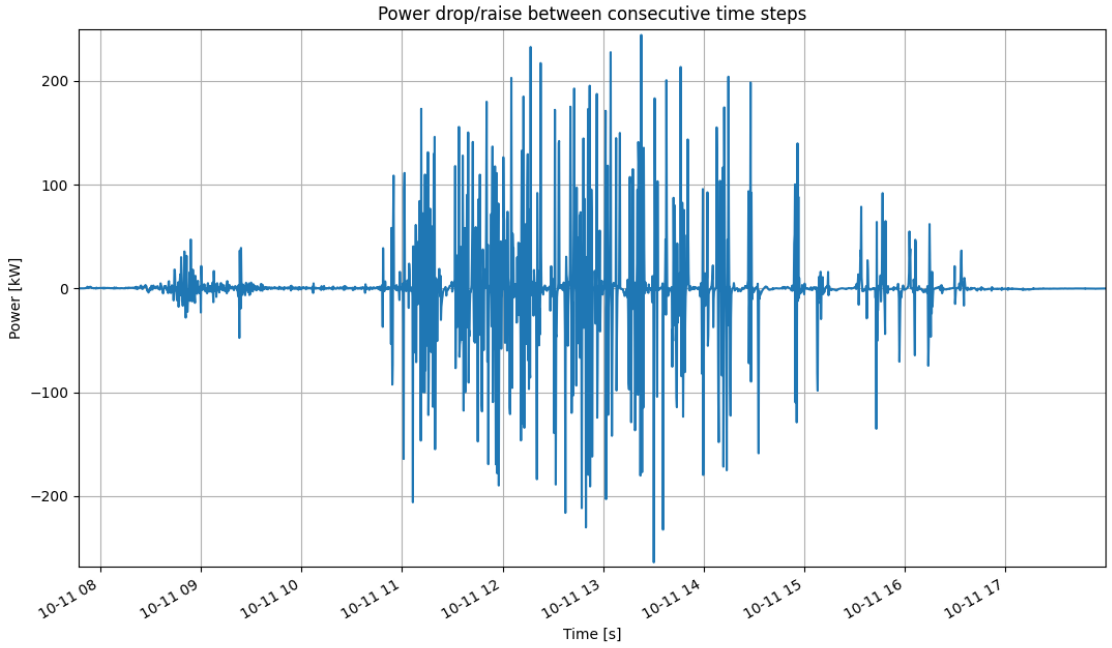

Figure 10: Power drop/raise for a single day.

- Images

  The images taken consist of images of the sky with a fisheye camera capable of taking information about the entire sky. The images have been taken every 15 seconds since 2020. In Figure 11 a sunny and a partly cloudy day are observed, respectively. The images were taken in Simrishamn, Sweden next to a photovoltaic plant.

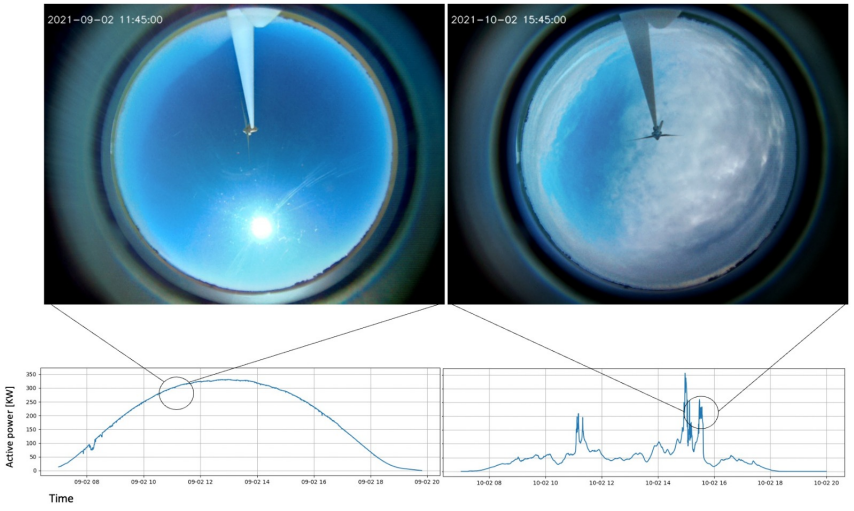

Figure 11: Sky images during a sunny and cloudy day (up), and active power for the same days (down).

3.2 Data preproccesing

In order to train the neural network models created, it is necessary to preprocess the data, whether images or historical data.

- **Training, test and validation data.**

  The data will be divided into three distinct blocks: training, test and validation. Training data will be the data that will be used to adjust the hyperparameters of the neural network. The validation data will be used to test the model in each training step. Each training step is called an epoch, at each epoch all the training data is passed to the model, and ends by passing the validation data to get an unbiased view of the model. The test data will be used for a final test of the model. Typically, 80% of the data is taken for training. Of the remaining 20%, another 20% are used for validation and the rest for test data.

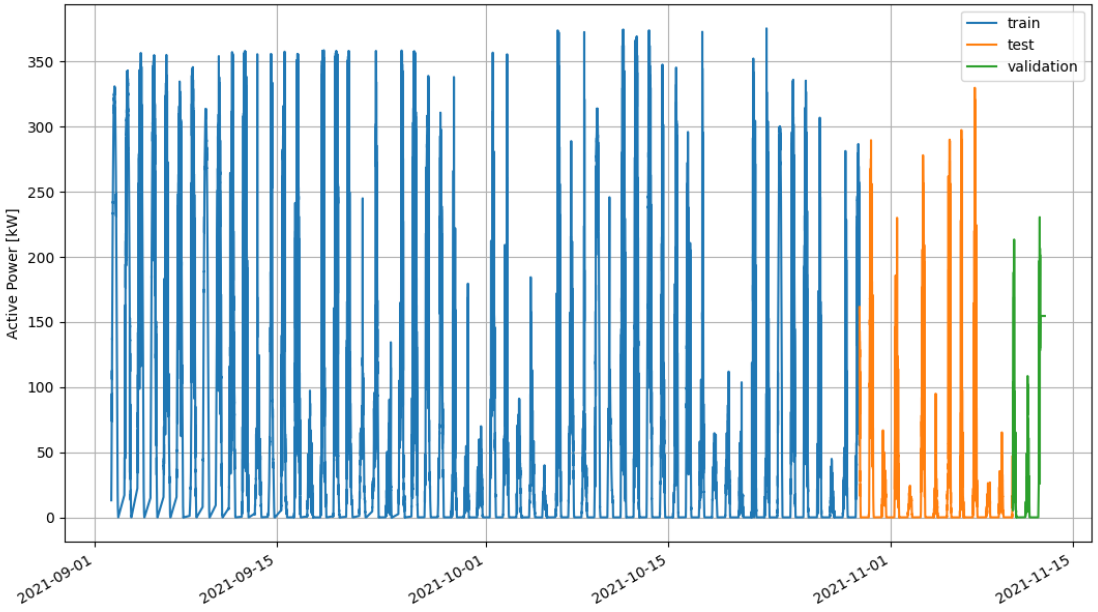

Figure 12: Test, train and validation data sets.

- **Data normalization**

  Normalisation of the data is done to speed up the learning process of the neural networks. We ensure that different features have similar ranges of values (feature scaling) so that gradient descents can converge faster. Thus, we normalise the training data to solve the challenge of model learning.

  The procedures for normalisation of images and historical data have been different, but the end result, in essence, is the same. Historical data have been normalized following equation (1). While the normalization of the images has been performed with the help of a Python library called "Pillow", which allows the reading of images in manageable formats facilitating the normalization.

$$X_{sc} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{5}$$

- **Prediction window**

  Before training the model, a prediction window has to be created which will indicate the inputs that the model needs for the outputs that are expected to be obtained. That is, if

you want to predict 5 minutes of production from data of the previous 5 minutes, with data every 15 seconds, you will obtain a window of 40 values with 20 inputs and 20 outputs. Thus, the model must be trained with the same instructions as the prediction window created.

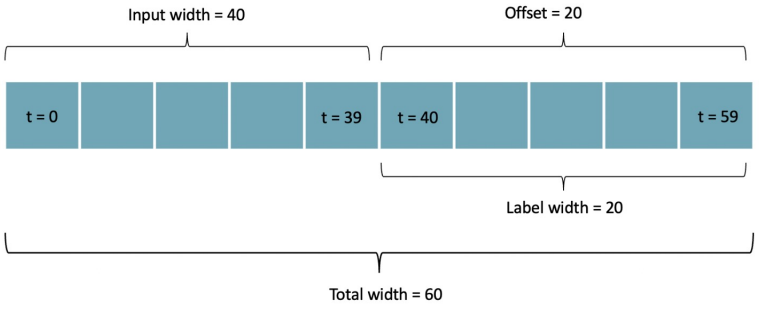

Figure 13: Prediction window

- **Batch size**

  Finally, before starting the training of the model the data of each epoch is divided into smaller parts called "batch". The batch size has influence on the learning of the neural network. If the batch size is too large, this may imply poor generalization, while a small batch size implies less memory usage, but greater difficulty in obtaining the absolute minimum of the system.

  The batch size is the number of iterations that occur in each epoch; for each iteration, the parameters of the neural network are readjusted; therefore, the smaller the batch size, the more readjustments occur in the neural network.

4 Results

The prediction results will be divided into two parts: a 5-minute and a 10-minute prediction. Both parts will show sunny, partly cloudy, and cloudy days, with the calculation of the RMSE (Eq. 6) between the actual value and the one predicted by the model. It will also show the detected ramps and specific cases in which the ramps are not detected correctly.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N}} \tag{6}$$

Finally, what will be done is to compare the forecast skill of the 5- and 10-minute forecasts. The forecast skill (Eq. 7) is a metric used in time series forecasting to check the quality of the

models used for forecasting. A forecast skill greater than zero is usually considered a good model. Forecast skill is defined by:

$$SS = 1 - \frac{RMSE_{\text{forecast}}}{RMSE_{\text{ref}}} \tag{7}$$

where MSE_{ref} is the mean squared error of a reference model. The reference model in our case is the baseline model, which consists of using the last N values used for the prediction as the prediction itself, and comparing these with the values predicted by the model.

A table with the hyperparameters used for training the model with the different approaches is shown below. The 5 and 10 minute predictions differ in the batch size, as well as the number of data used for the prediction of the following minutes. It is worth noting that the number of input images used is not equal to the number of historical data used, this is due to the fact that the consecutive images are quite similar, in addition to occupying a much larger space than the historical data, which could saturate the memory of the computer used for model training and processing.

Table 1: Hyperparemter for both forecasting approaches.

	5-minute forecast	**10-minute forecast**
Input time [minutes]	10	20
Output time [minutes]	5	10
Input timesteps	40	80
Output timesteps	20	40
Sampling rate time series [seconds]	15	15
Sampling rate images [seconds]	60	60
Image resolution	64	64
Number of images	10	20
Batch size	128	32
Epochs	200	200

4.1 5 minutes forecast

As mentioned above, in this section, the 5-minute forecast will be shown. In Figures 14, 15 a linear fit of the prediction against the real values is observed; therefore, the closer the value of R^2 is to the unity, the better the predictions. Although it should be noted that the main objective of the prediction is not to minimize the error, but to predict the ramps, the ideal would be to detect them with the smallest possible error.

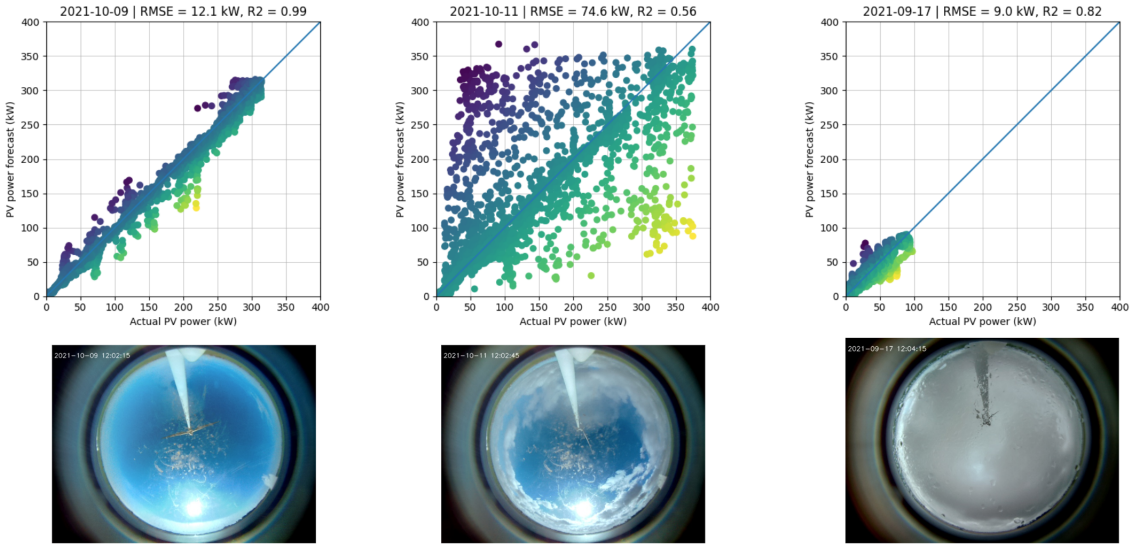

Figure 14: Linear regression between real and predicted data: 1) Sunny day 2) Partially cloudy day 3) Cloudy day.

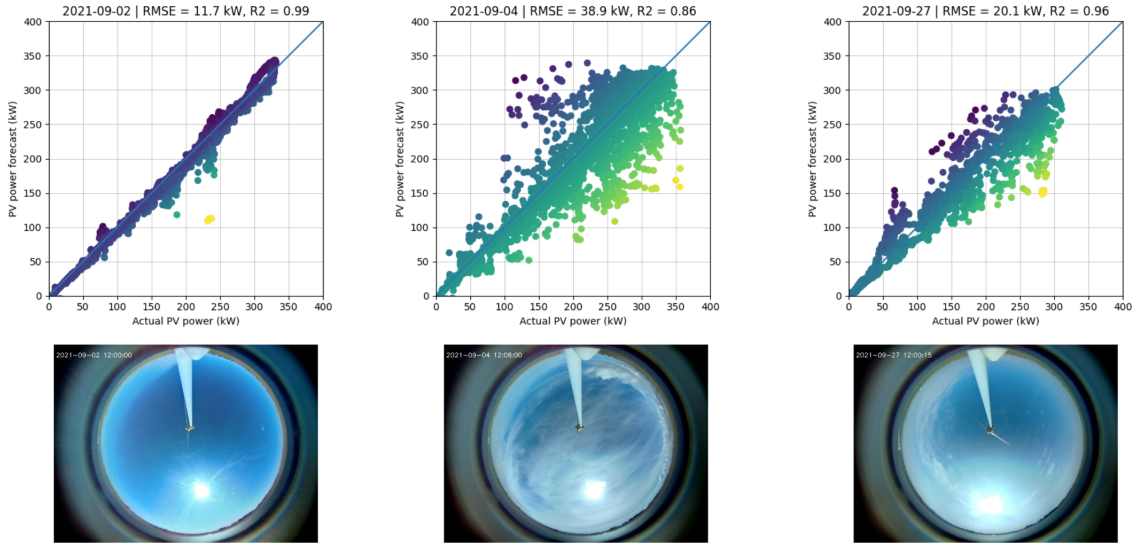

Figure 15: Linear regression between real and predicted data: 1) Sunny day 2) Partly cloudy day 3) Cloudy day.

Moreover, predictions of clear days and cloudy days are relatively easy to predict, since the presence of clouds is null in the first case and totally cloudy in the second case. But the real difficulty in the predictions is presented on partly cloudy days and this is where the study is most relevant, as shown in Figures 14.2 and 15.2, as expected, the greatest error is presented on partly cloudy days. However, as mentioned above, although the error is relatively large, the detection of ramps can be good, as shown in Figures 16 and 17.

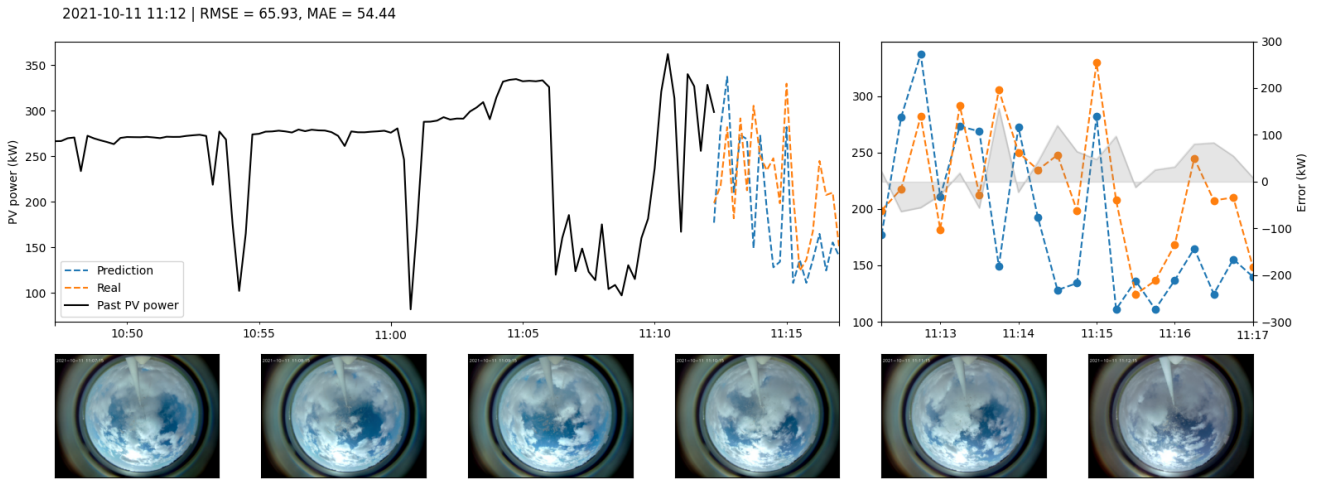

Figure 16: Forecast vs. actual active energy production values for 2021-10-11.

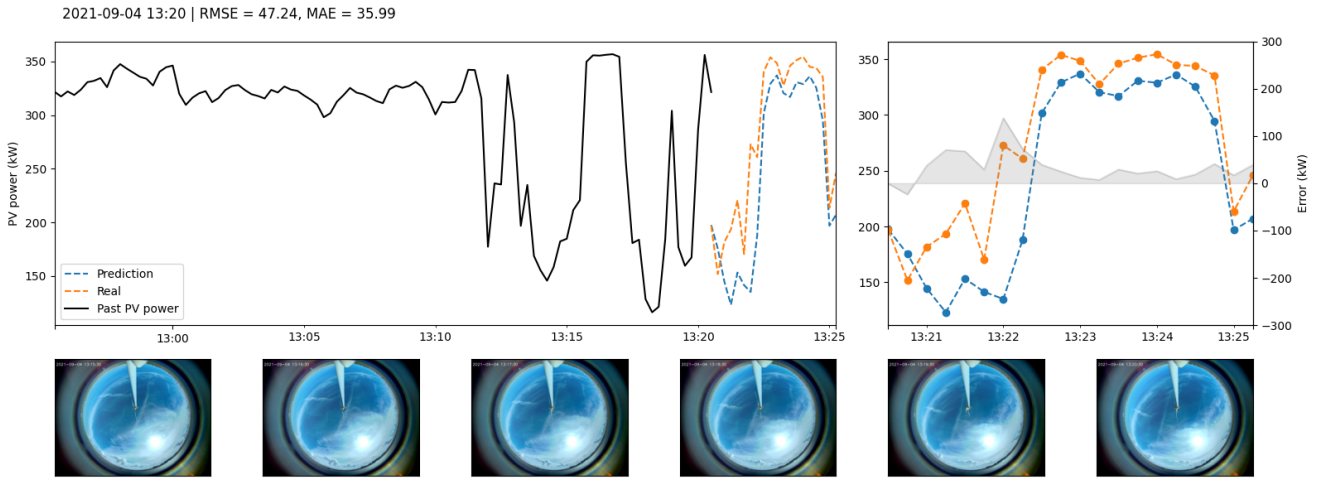

Figure 17: Forecast vs. actual active energy production values for 2021-09-04.

Figure 16 shows a particularly interesting case; although the prediction detects the ramps quite accurately, some ramps are displaced over time and do not correctly evaluate the amplitude or real value of the irradiance. In this case, the difficulty of predicting partly cloudy days is clearly seen, and where the high RMSE values come from, in addition to the relatively low R^2. This may be due, in part, to the accumulated dirt in the camera due to the previous rainy days.

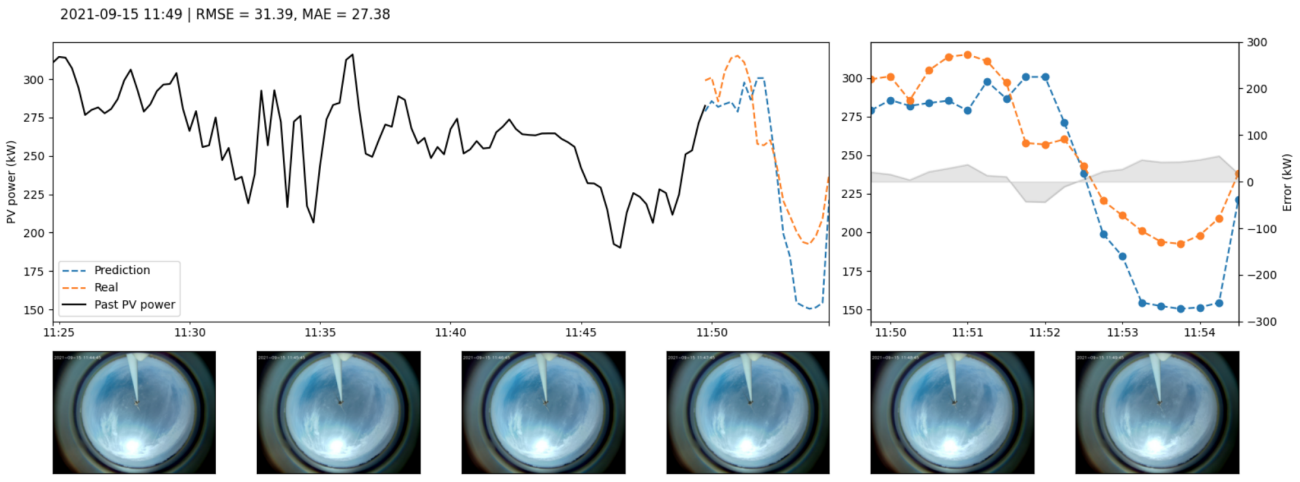

Figure 18: Forecast vs. actual active energy production values for 2021-09-15.

4.2 10 minutes forecast

This section will show some relevant results from the 10-minute forecasts. As was done in the previous section, here we will show the linear fits of the predictions to the actual data for the same selected days as in the case of the 5-minute predictions.

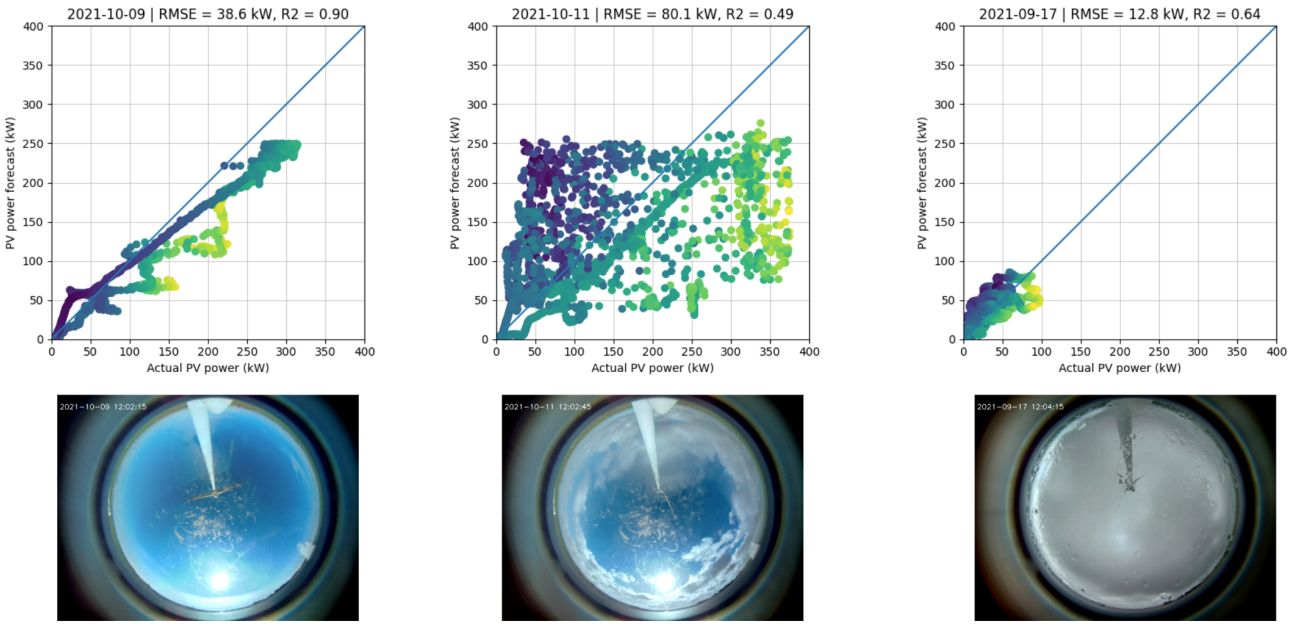

Figure 19: Linear regression between real and predicted data: 1) Sunny day 2) Partially cloudy day 3) Cloudy day.

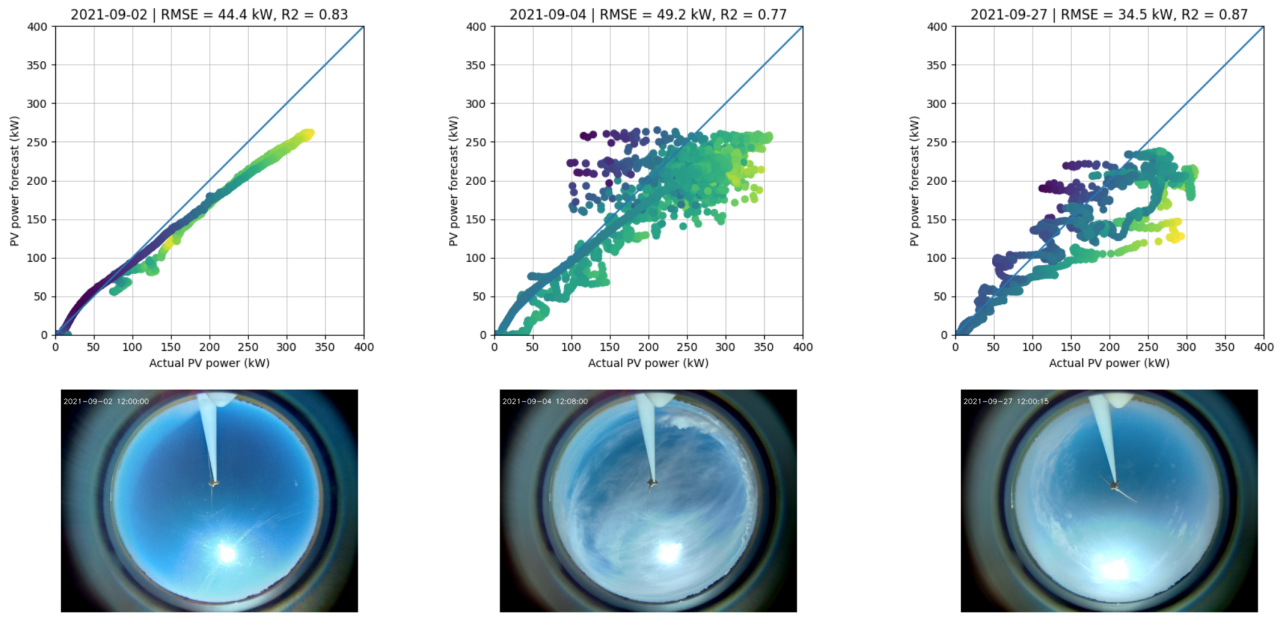

Figure 20: Linear regression between real and predicted data: 1) Sunny day 2) Partially cloudy day 3) Cloudy day.

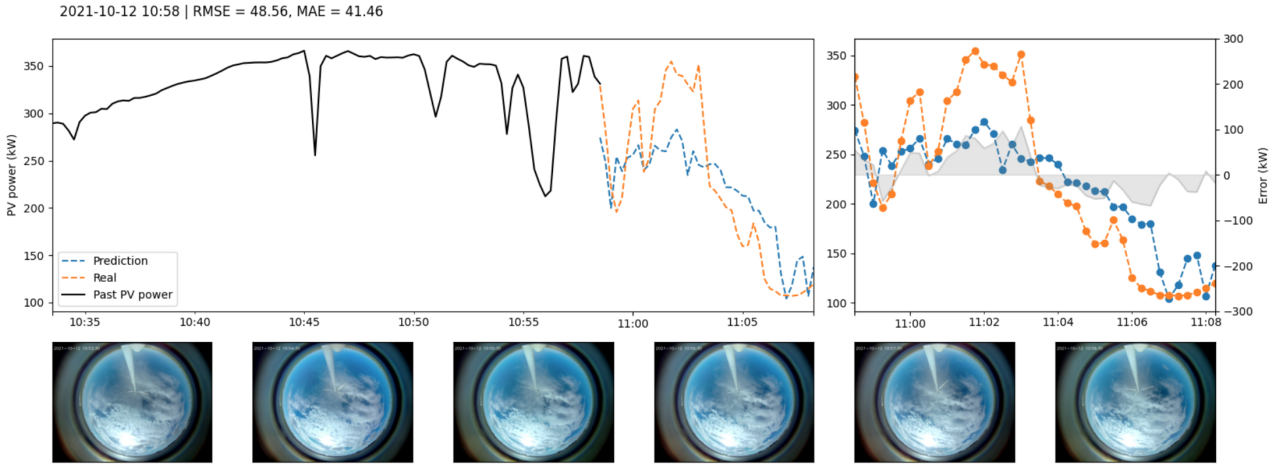

Figure 21: Forecast vs. actual active energy production values for 2021-10-12.

It can be seen that the 5-minute forecast exceeds the accuracy of the current forecast for the three cases studied: sunny, partly cloudy and totally cloudy. It can also be seen in Figure 23 that the 10-minute forecast has a greater difficulty in detecting ramps, or at least those forming local minima of short duration. Although its skill score is similar to that of the 5-minute prediction, as will be seen in the following sections. This prediction is clearly outperformed by the previous and, therefore, it will be necessary to modify hyperparameters, layers in the model or even improve the

dataset to improve this final results.

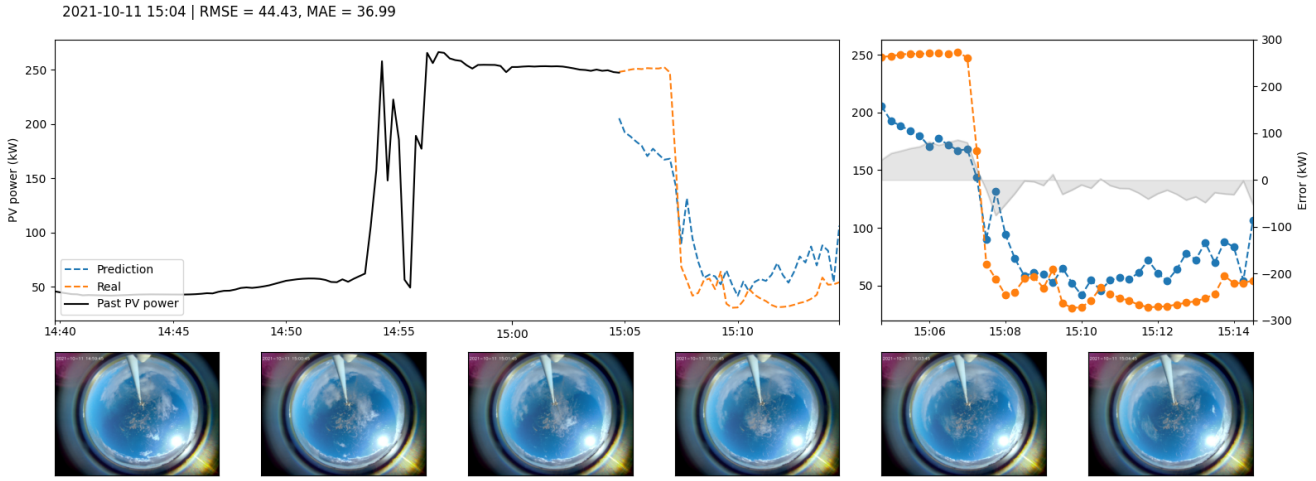

Figure 22: Forecast vs. actual active energy production values for 2021-10-11.

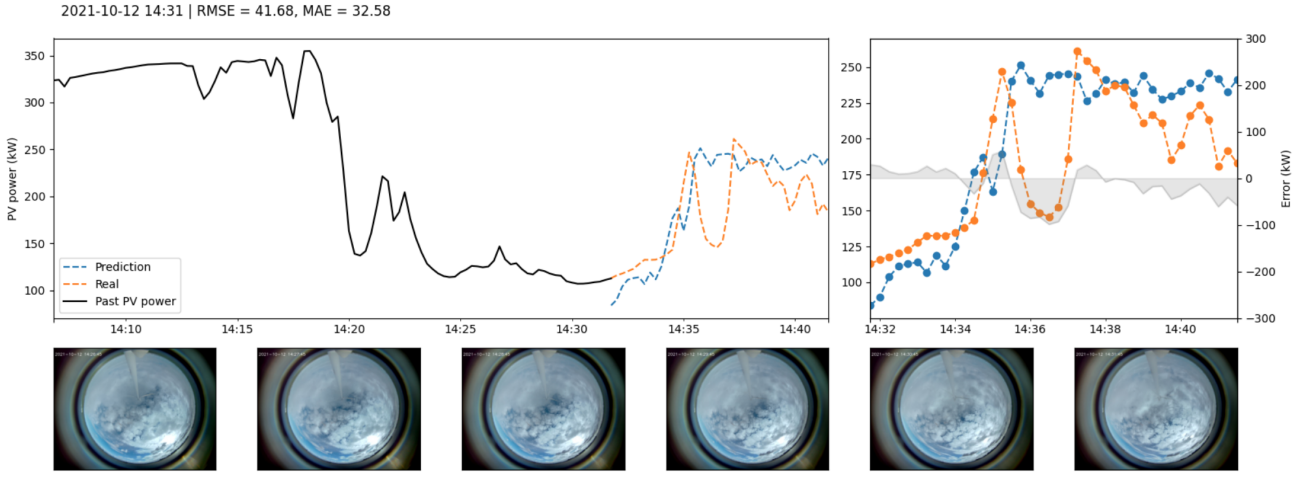

Figure 23: Forecast vs. actual active energy production values for 2021-10-12.

Finally, in Appendix A more R^2 and RMSE values for the 5-minute and 10-minute predictions for partly cloudy days can be seen. For each of the days it can be seen how the 5-minute prediction is better than the 10-minute prediction.

4.3 Forecast skill and RMSE

This section compares the forecast skill of the 10-minute and 5-minute forecasts, see Figures 24 and 25. As mentioned before, a value higher than zero implies that the model improves the persistence, or baseline model, which consists of no other strategy than to extend the last real N

values as the N values to be predicted by our model. Thus, in the images below, it can be seen that both approaches improve the persistence model. On the one hand, the 5-minute prediction outperforms the persistence model for predictions after 75 seconds; that is, before that time, the persistence model has a better prediction. However, the 10-minute prediction outperforms the persistence model for predictions after 225 seconds. Both predictions reach an approximate skill score value of 0.5. A positive value of the skill score means that the forecast is an improvement over the standard forecast.

Morover, the root mean squared error will be shown in this section (Eq. 6) for each time step. As shown in Figures 24 and 25, the error for the 5-minute prediction is smaller than the 10-minute prediction for each of the time steps, reaching a maximum of 31 kW and a minimum of 26 kW for the 5-minute prediction, and a maximum of 45 and a minimum of 31 kW for the 10-minute prediction. In the case of the 10-minute prediction, a nonlinear decrease of the error is observed, with some local minima occurring over the predicted preiod; in the case of the 5 minutes, a linear decrease is clearly observed until the last value. It should be noted that these differences are due to the difference between the hyperparameters used (see Table 1) for each of the predictions: in the case of the 5-minute prediction we used the previous 10 minutes and a batch size of 128, while for the 10-minute prediction we used data from the previous 20 minutes with a batch size of 32, these parameters have been adjusted to obtain the minimum error for each of the cases studied.

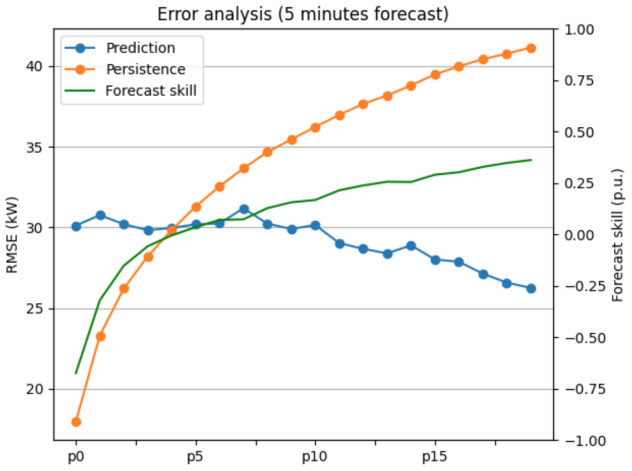

Figure 24: Forecast skill for prediction (green), RMSE for prediction (blue) and persistance RMSE (orange) for the 5-minute prediction.

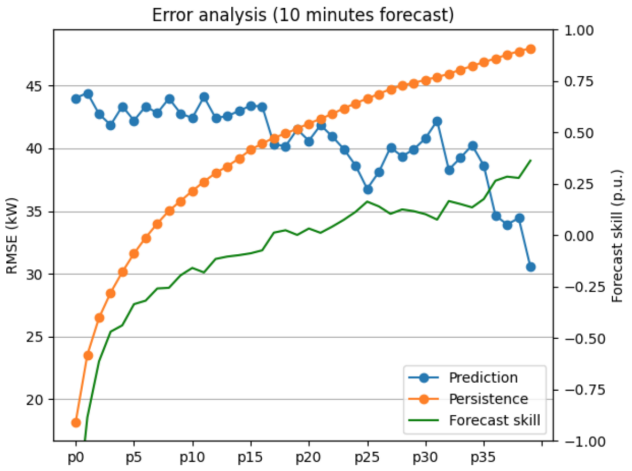

Figure 25: Forecast skill for prediction (green), RMSE for prediction (blue) and persistance RMSE (orange) for the 10-minute prediction.

5 Conclusions

Short-term prediction of the energy produced by a photovoltaic plant is a widely studied topic, and, as has been previously mentioned, it is an important issue for the stability of the grid and its correct operation, as well as for reducing the operating costs and increasing the lifetime of the elements which it is composed.

In this work, two approaches have been presented: a 5-minute prediction and a 10-minute prediction are compared using different error metrics. As can be seen, the model used is able to reach a positive skill score in a shorter time for the 5-minute prediction than for the 10-minute prediction, although both end up converging to a final value of approximately 0.5, which is an improvement compared to the baseline model.

The use of neural networks for the prediction of photovoltaic energy seems to be an efficient solution that shows good results for the short-term power forecasting. Furthermore, it shows very good results in the detection of ramps produced by the irradiance drop due to cloud movement, which is the main goal of the present study.

As mentioned above, there are cases where ramp detection is very good, but there are also cases where detection is not correctly detected. This could be due to the presence of dirt in the camera (see Figure 14.1). It also appears to be affected by the type of cloud that emerges, as some are denser and easier to detect by the convolution neural network than others, as seen in Figure 15.2.

An effective way to improve predictions would be to create a network of cameras across a wide region and at different latitudes. This would allow the first cameras in the wind direction to have

the accuracy of the current model, however later cameras would be able to learn from the data of earlier cameras obtaining better results. This, coupled with historical production data, would allow much better ramp prediction for partly cloudy days than the present present work as neural networks are able to learn non-linear dependencies with a suitable dataset.

Another improvement for the current study would be the creation of an error metric for the detection of ramps, in order to quantify the quality of the detected ramps, although the skill score gives a good notion of the quality of the prediction, it is still a comparison between the RMSE between two models, which, as we have seen, is not the most optimal method for evaluating the quality of the prediction of the ramps.

The reason why there is an overestimation of production in the predictions may be because the wind turbine is captured at all times by the cameras. As observed in the convolutions and the applied filters, the turbine tower and its blades are detected in some filters as part of the clear sky, so it would be interesting to have images of the sky dome without interference from other objects that may influence the accuracy of the predictions.

Another interesting conclusion is that the Attention Layer, which is not intended to be used as a merging layer, shows very satisfactory results for combining two very different models such as historical production data and solar positions and sky images. Although its use is not intended, it is essentially an averaging layer that assigns weights to each value, just as it would do with language processing, which is its normal use.

For future work, it is possible to work on the classification of ramps according to their duration in order to see how the model adapts to different ramps, both in terms of magnitude and duration.

Finally, the last improvement proposed is a higher sampling frequency for both images and production data. With a higher sampling frequency, more accurate predictions could be made for even shorter periods (between 1 and 5 minutes), and could also lead to an improvement in the predictions mentioned above.

References

[1] IRENA, *Renewable Energy Statistics 2022*. Abu Dhabi: The International Renewable Energy Agency, 2022.

[2] K. N. Nwaigwe, P. Mutabilwa, and E. Dintwa, "An overview of solar power (PV systems) integration into electricity grids," *Materials Science for Energy Technologies*, vol. 2, no. 3, pp. 629–633, 2019.

[3] B. Belcher, B. J. Petry, T. Davis, and K. Hatipoglu, "The effects of major solar integration on a 21-bus system: Technology review and psat simulations," in *SoutheastCon 2017*, pp. 1–8, 2017.

[4] E. Mulenga, *Impacts of integrating solar PV power to an existing grid. Case Studies of Mölndal and Orust energy distribution (10/0.4 kV and 130/10 kV) grids*. Chalmers University of Technology / Department of Energy and Environment, 2015.

[5] G. M. Tina, C. Ventura, S. Ferlito, and S. De Vito, "A State-of-Art-Review on Machine-Learning Based Methods for PV," *Applied Sciences*, vol. 11, p. 7550, Jan. 2021. Number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[6] H. Ye, B. Yang, Y. Han, and N. Chen, "State-of-the-art solar energy forecasting approaches: Critical potentials and challenges," *Frontiers in Energy Research*, vol. 10, 2022.

[7] J. Zhang, R. Verschae, S. Nobuhara, and J.-F. Lalonde, "Deep photovoltaic nowcasting," *Solar Energy*, vol. 176, pp. 267–276, Dec. 2018.

[8] M. Lipperheide, J. Bosch, and J. Kleissl, "Embedded nowcasting method using cloud speed persistence for a photovoltaic power plant," *Solar Energy*, vol. 112, pp. 232–238, 2015.

[9] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, 2017.

[10] T. ZHU, H. ZHOU, H. WEI, X. ZHAO, K. ZHANG, and J. ZHANG, "Inter-hour direct normal irradiance forecast with multiple data types and time-series," *Journal of Modern Power Systems and Clean Energy*, vol. 7, pp. 1319–1327, Sept. 2019.

[11] G. E. P. Box and D. A. Pierce, "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models," *Journal of the American Statistical Association*, vol. 65, no. 332, pp. 1509–1526, 1970.

[12] G. U. Yule, "On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 226, pp. 267–298, 1927.

[13] Y. Li, Z. Zhu, D. Kong, H. Han, and Y. Zhao, "EA-LSTM: Evolutionary Attention-based LSTM for Time Series Prediction," Tech. Rep. arXiv:1811.03760, arXiv, Nov. 2018.

[14] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[15] "Expectation-based selective attention for visual monitoring and control of a robot vehicle," *Robotics and Autonomous Systems*, vol. 22, no. 3, pp. 329–344, 1997. Robot Learning: The New Wave.

[16] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016. Recent Developments on Deep Big Vision.

[20] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition.," in *Interspeech*, vol. 2013, pp. 1173–5, Citeseer, 2013.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," tech. rep., Sept. 2014.

[23] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

[24] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, pp. 92–101, Springer, 2010.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

A Partly cloudy days

A.1 2021-10-29

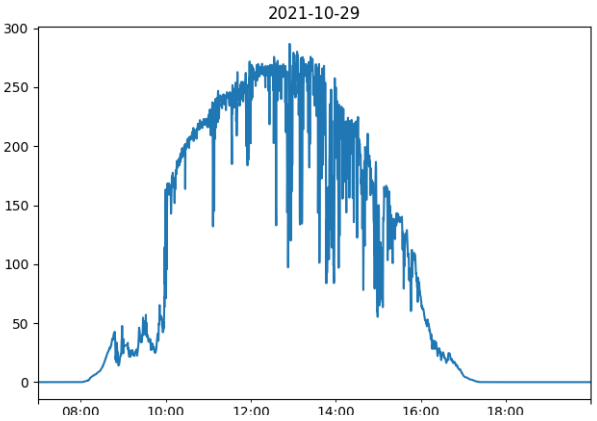

Figure 26: 2021-10-29 real active power production.

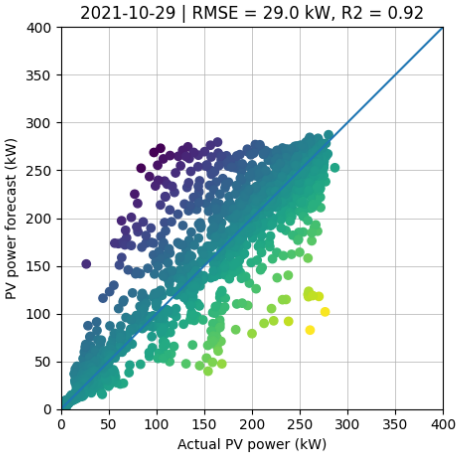

Figure 27: 5-minute R^2 value for 2021-10-29.

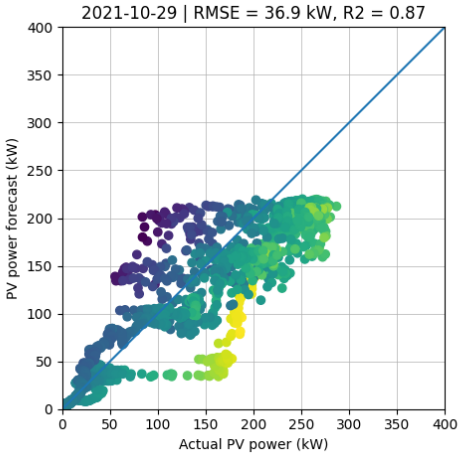

Figure 28: 10-minute R^2 value for 2021-10-29.

A.2 2021-09-20

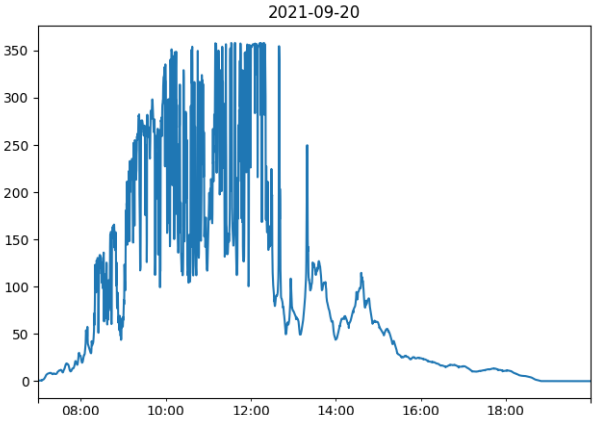

Figure 29: 2021-09-20 real active power production.

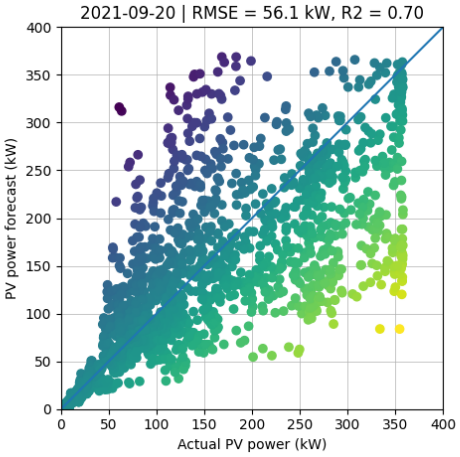

Figure 30: 5-minute R^2 value for 2021-09-20.

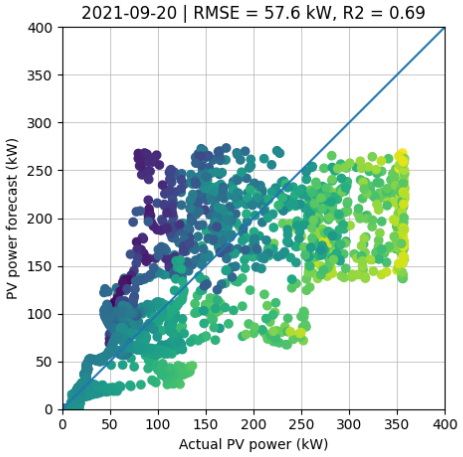

Figure 31: 10-minute R^2 value for 2021-09-20.

A.3 2021-09-12

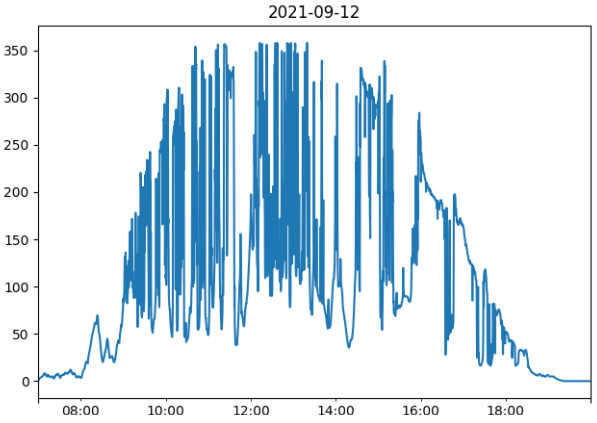

Figure 32: 2021-09-06 real active power production.

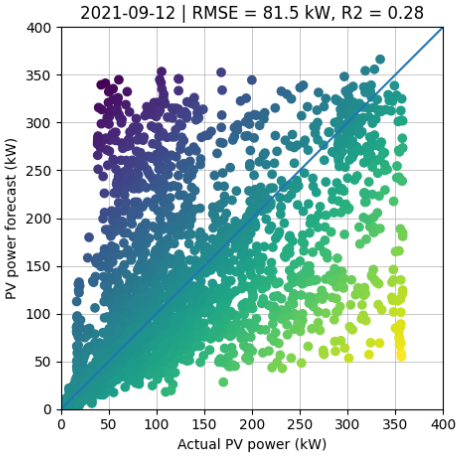

Figure 33: 5-minute R^2 value for 2021-09-12.

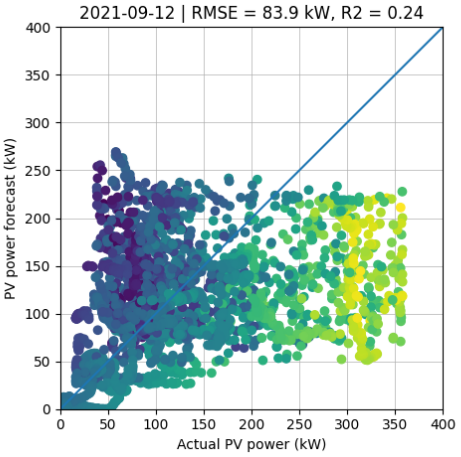

Figure 34: 10-minute R^2 value for 2021-09-12.

A.4 2021-09-06

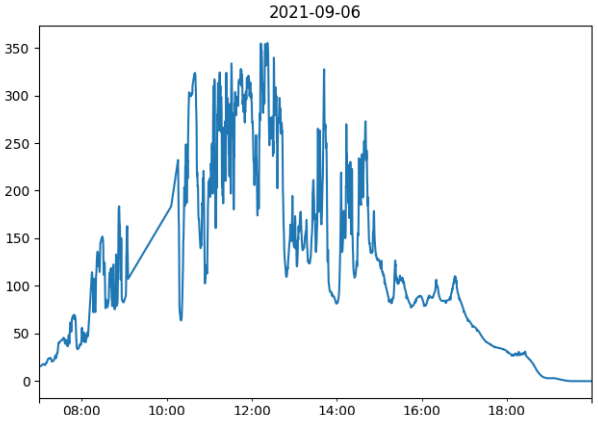

Figure 35: 2021-09-06 real active power production.

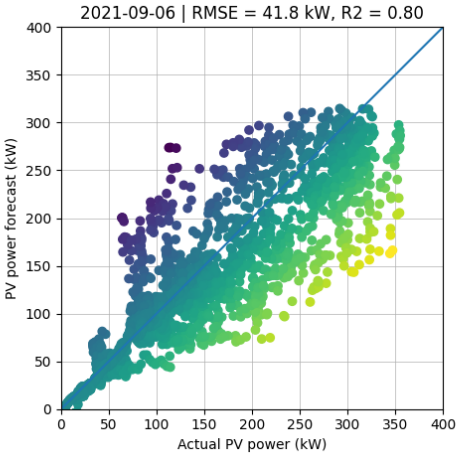

Figure 36: 5-minute R^2 value for 2021-09-06.

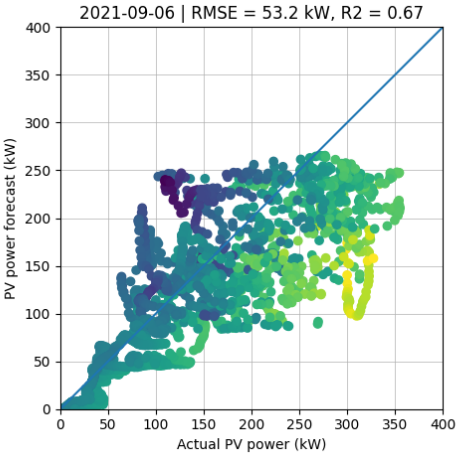

Figure 37: 10-minute R^2 value for 2021-09-06.