Forecasting Solar Power Output to assist with the integration of solar energy into the power grid

Aarush Garg [*α,β]

[α] German Swiss International School, Hong Kong

[β] Hong Kong University of Science and Technology

Abstract

This paper discusses the integration of solar energy into the power grid by forecasting solar power ahead of time using a time series forecasting problem formulation. The aim is to help reduce the impact of fossil fuels on the environment. The two main objectives are namely to forecast PV power one timestep ahead and to forecast PV power multiple timesteps ahead. Data has been taken from the Hong Kong University of Science and Technology. This paper compares machine learning techniques such as CNN LSTM, RNN LSTM, Dense Neural Network, Convolutional Neural Network, to find the best predictor of PV output. The tuning of hyperparameters such as the learning rate, regularization parameter, activation function, number of iterations, etc. is also an essential part of the discussion. The long short term memory (LSTM) and dense feed forward layers, as well as convolutional layers help to introduce new important features from the original features. This paper gives a summary of the future works to be done to assist for additional research and improved results.

Keywords: Solar power forecasting, machine learning, renewable energy, neural networks, autoregression, power integration.

1 Introduction

Due to the emergence of rising global temperatures, and the dire consequences of global warming, it is of utmost importance to work on reducing greenhouse gas emissions. The largest source of these are fossil fuels, therefore it is important to find alternative energy sources, such as renewable energy sources to reduce the impact of global warming. Finding clean energy and integrating it into the energy grid is a huge step towards solving the UN Sustainable Development Goal Number 7, as shown in [Figure 1]. However, it has been challenging for scientists and policy-makers to harness these renewable energy sources, especially solar energy, to the fullest extent due to the unreliability of such energy sources. Therefore to aid with the integration of renewable energy to the energy grid, it is important to forecast the output of the energy sources. Since solar energy is the most prominent source of renewable energy, with long-lasting advantages, this paper focuses on forecasting Photo-Voltaic (PV) output ahead of time, to make it easier to integrate solar energy into the grid, and reduce the reliance on fossil fuels.

PV cells convert light energy to electric energy using materials that show signs of photovoltaic effect. The main issue is that solar panels, which make up a PV system only work well if the sunlight is directly on to the panel, and energy is lost when tracking system is not established, hence building a PV forecasting system is of utmost importance. PV outputs vary throughout the day, given the varying solar irradiation and other weather temperatures, therefore, to integrate this resource into the energy grid, whilst maintaining consumer demands, forecasting of power output can help to estimate the amount of load that solar power can assist with. During times of high solar power production, batteries can be used to store the excess energy for times when the sun is not out (i.e during the night, or on a rainy day).

Figure 1: Sustainable Development Goal Number 7: Clean and Affordable Energy

2 Related Research

Multiple techniques were used to predict solar power output one hour ahead given hourly data. In [Akhter et al. 2022], techniques such as artificial neural networks (ANN), convolutional neural networks (CNN), multiple linear regression and support vector machines were used for prediction. In [Isaksson and Karpe Conde 2018] and [Alam et al. 2021] mathematical models such as ARIMA (auto regressive integrated moving average) as well as machine learning algorithms were used, due to their ability to factor in seasonality of winter vs summer

[*] gargaarush2006@gmail.com

and the time of day. Other than the ones stated above, the most common techniques include recurrent neural network (RNN), dense neural network and the combination of RNN and CNN with LSTM (long short term memory). From [Alam et al. 2021], CNN LSTM worked best on the data provided, with a mean absolute error of 5 percent. Other deep learning techniques, such as Auto-LSTM and MLP (multiple layer perceptron) have been used for forecasting solar PV output, as in [Gensler et al. 2016].

For multiple timestep forecasting, techniques from [Pedregal and Trapero 2021] as well as the similar techniques from single timestep forecasting are very common. In particular, the hybrid Convolutional LSTM model is the most common, as the convolutional layer and LSTM layer work well together, giving good approximations.

In the forecasting of PV output, the most common weather features chosen are solar irradiance, temperature, humidity,visibility, pressure dew and windspeed, as can be seen in [Zheng et al. 2020]. In particular the solar irradiance feature is of great importance, as shown in [Feng et al. 2022]. Another additional feature, as discussed in [Theocharides et al. 2020], which could be useful is the azimuth. This is because it indicates the elevation angle of the Sun above the solar panel system, and theoretically should contribute a weight to the PV solar output.

3 Objectives

This paper discusses 2 main objectives:
1. To predict PV output one timestep ahead
2. To predict PV output multiple timesteps (14 hours) ahead
As discussed, these objectives are both of importance to help with planning, and the integration of renewable energy in the form of solar energy into the power grid.

4 Data Collection

There are many weather variables which may affect the solar power production of a certain hour. Some of these factors include solar irradiance, temperature, windspeed, humidity, time of day and time of year. The data from the past 2 years of hourly readings for these factors was collected from the Hong Kong University of Science and Technology (HKUST) Supersite, and PV data was collected from the HKUST Solar site. The site is located in the New Territories region of Hong Kong, and collects data using various different types of sensors including pressure sensors, temperature sensors and anemometer to measure windspeed.

5 Notations

LSTM	Long Short Term Memory
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
ARIMA	Auto-Regressive Integrated Moving Average
PV	Photo-Voltaic
ANN	Artificial Neural Network
AR LSTM	Auto-Regressive Long Short Term Memory
α	learning rate
λ	regularization parameter
tanh	Hypberbolic Tangent Activation Function
relu	Rectified Linear Unit
Adam	Adaptive Moment Optimizer
Epochs	Number of Iterations
ConvWindow	Convolutional Window Width
MAE	Mean Absolute Error
val	cross-validation set
d	length of day in seconds
t	timestamp in seconds
x	data feature
$\bar{x}$	normalized feature

6 List of Features used in Modelling

- Windspeed
- Azimuth
- Solar Irradiance
- Pressure
- Visibility
- Humidity
- Temperature
- PV output
- Day sin
- Day cos

7 Data Preparation

In this section, the preparation stages of the data will be outlined.

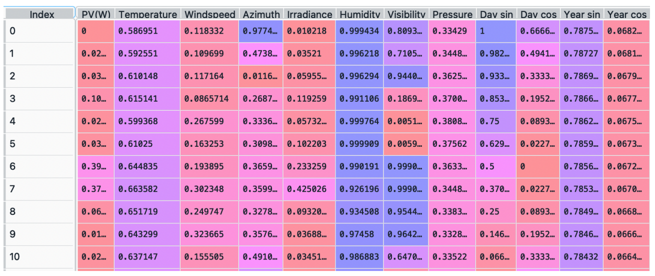
7.1 Removing Anomalous Data from Weather Set

Before merging the data from weather features, it was essential to remove the anomalous data from the dataset. For example, the data for certain hours for temperature happened to be -99999 degrees celsius. This indicated that an error had occurred, and the sensor was unable to give a reading for that hour, and such anomalous data was removed from the dataset. After that, the data was merged into a dataframe as shown below, with the outputs of 2 year of weather data, collected hourly.

7.2 Feature Normalization

For techniques involving machine learning, it is essential to do feature normalization, before running the model. This is because certain features may have different data ranges, and assigning weights to these features will be meaningless. Instead, if all features are normalized, then the weights can be compared against each other, and thus the importance of each feature can be compared. This is highly important for analysis of models and methods to increase performance of the model, as the data can be easily understood. [Figure 2] below shows a snapshot of the first 10 rows of the normalized dataframe.

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



Index	PV(W)	Temperature	Windspeed	Azimuth	Irradiance	Humidity	Visibility	Pressure	Day sin	Day cos	Year sin	Year cos
0	0	0.586951	0.118332	0.9774…	0.010218	0.999434	0.8093…	0.33429	1	0.6666…	0.7875…	0.0682…
1	0.02…	0.592551	0.109699	0.4738…	0.03521	0.996218	0.7105…	0.3448…	0.982…	0.4941…	0.78727	0.0681…
2	0.03…	0.610148	0.117164	0.0116…	0.05955…	0.996294	0.9440…	0.3625…	0.933…	0.3333…	0.7869…	0.0679…
3	0.10…	0.615141	0.0865714	0.2687…	0.119259	0.991106	0.1869…	0.3700…	0.853…	0.1952…	0.7866…	0.0677…
4	0.02…	0.599368	0.267599	0.3336…	0.05732…	0.999764	0.0051…	0.3808…	0.75	0.0893…	0.7862…	0.0675…
5	0.03…	0.61025	0.163253	0.3098…	0.102203	0.999909	0.0059…	0.37562	0.629…	0.0227…	0.7859…	0.0673…
6	0.39…	0.644835	0.193895	0.3659…	0.233259	0.990191	0.9990…	0.3633…	0.5	0	0.7856…	0.0672…
7	0.37…	0.663582	0.302348	0.3599…	0.425026	0.926196	0.9990…	0.3448…	0.370…	0.0227…	0.7853…	0.0670…
8	0.06…	0.651719	0.249747	0.3278…	0.09320…	0.934508	0.9544…	0.3383…	0.25	0.0893…	0.7849…	0.0668…
9	0.01…	0.643299	0.323665	0.3576…	0.03688…	0.97458	0.9642…	0.3328…	0.146…	0.1952…	0.7846…	0.0666…
10	0.02…	0.637147	0.155505	0.4910…	0.03451…	0.986883	0.6470…	0.33522	0.066…	0.3333…	0.78432	0.0664…

Figure 2: Dataset with PV and all weather features normalized

7.3 Data Transformation - Day sin, Day cos features

As discussed in [Elfeky et al. 2005], it is very important to detect the periodicity and pattern of data, and hence creating features to detect this is highly important. One key step required was to create a feature which captures the time of day in a numerical method, and gives a repeating pattern of values for each new day. Hence, each date and time was give a timestamp and then multiplied by 2π and divided by the duration of each day. The cosine and sine of these values were taken, to form a periodic curve for every 24 hours. A similar process could be used to find the seasonality of the year, by replacing the length of day with length of year.

$$DaySin = sin(\frac{2\pi t}{d}) \tag{2}$$

$$DayCos = cos(\frac{2\pi t}{d}) \tag{3}$$

7.4 Training Set - Test Set - Validation Set Split

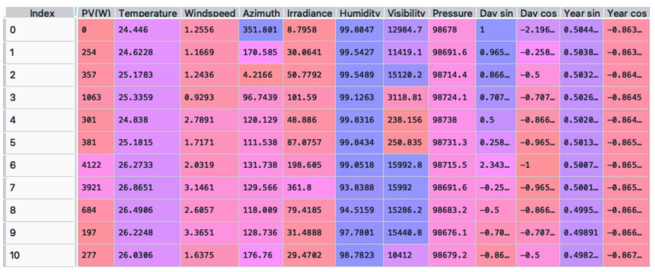
In this data, the split of data has been done 75 percent training, 15 percent validation and 10 percent test data. It is essential to ensure that the data is chosen in a time-series manner, such that the training set data has to be together, as the data being used is in chronological order (date and time wise). Hence, the algorithm being run is a time-series forecasting model. Due to the fact that this problem is a time-series formulation, the machine learning algorithm would not give proper weights if the order of the data were to be changed. Auto-regression of the PV output would fail as the data would not be in a time-series order, if the data were split randomly. The weather and solar output of the previous timestep does impact the weather and solar output of the next timestep, due to the physical constraints on how quickly the weather can change.

7.5 Cleaning the PV Output Data

Data for the PV output for each day was available for every 5 minutes between 5:00 am to 8:00 pm. This is because at any time after 8:00pm and before 5:00 am, the sun was not out and therefore the PV output data was not collected during these times. Accurate data was collated into a dataframe, as shown below.

7.6 Merging PV and Weather Data

Merging the PV and weather data was tricky as the PV data had been collected every 5 minutes whereas the weather data had been collected hourly. Therefore, when merging the PV and weather data, the largest common subset of both sets of times had to be chosen along with the label data (PV output) and the input features (weather data). In [Figure 3] shown below, a sample of the dataframe can be seen with all the PV and Weather data together.



Index	PV(W)	Temperature	Windspeed	Azimuth	Irradiance	Humidity	Visibility	Pressure	Day sin	Day cos	Year sin	Year cos
0	0	24.446	1.2556	351.801	8.7958	99.8047	12984.7	98678	1	-2.196…	0.5044…	-0.863…
1	254	24.6228	1.1669	170.585	30.0641	99.5427	11419.1	98691.6	0.965…	-0.258…	0.5038…	-0.863…
2	357	25.1783	1.2436	4.2166	50.7792	99.5489	15120.2	98714.4	0.866…	-0.5	0.5032…	-0.864…
3	1063	25.3359	0.9293	96.7439	101.59	99.1263	3118.81	98724.1	0.707…	-0.707…	0.5026…	-0.8645
4	301	24.838	2.7891	120.129	48.886	99.8316	238.156	98738	0.5	-0.866…	0.5020…	-0.864…
5	381	25.1815	1.7171	111.538	87.0757	99.8434	250.835	98731.3	0.258…	-0.965…	0.5013…	-0.865…
6	4122	26.2733	2.0319	131.738	198.605	99.0518	15992.8	98715.5	2.343…	-1	0.5007…	-0.865…
7	3921	26.8651	3.1461	129.566	361.8	93.8388	15992	98691.6	-0.25…	-0.965…	0.5001…	-0.865…
8	684	26.4906	2.6057	118.009	79.4185	94.5159	15286.2	98683.2	-0.5	-0.866…	0.4995…	-0.866…
9	197	26.2248	3.3651	128.736	31.4888	97.7801	15440.8	98676.1	-0.70…	-0.707…	0.49891	-0.866…
10	277	26.0306	1.6375	176.76	29.4702	98.7823	10412	98679.2	-0.86…	-0.5	0.4982…	-0.867…

Figure 3: Clean dataframe with PV Output and Weather Features

7.7 Violin Plot

After the normalization of data, a violin plot was created to show the distribution of data for each feature, and evaluate the split of data for the entire dataset. [Figure 4] was used to check for large anomalies in the data, and if so, to understand the cause behind this anomalous data, and whether any modification would be required.

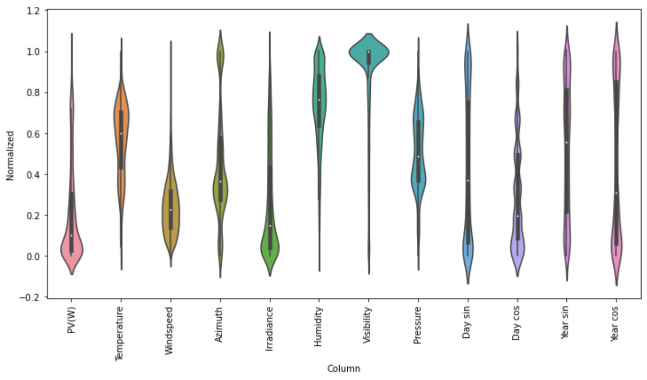


Figure 4: Violin Plot of Different Features

8 Methodology

8.1 Single-Step Model

The single step model involves taking inputs of a certain window length, and producing a single output from these inputs. This is then compared against the actual output/label, and the MAE is found, which is fed back to the model. The figure below [Figure 5] gives a visual representation of the single-step forecasting model.

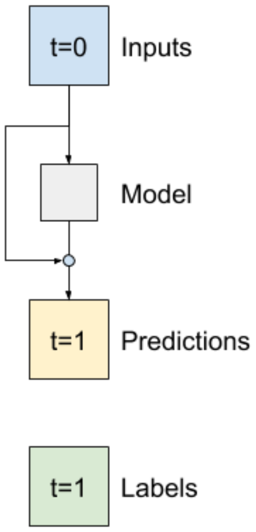


Figure 5: Single Step Forecasting Model

8.2 Multi-Step Model

The multi output model involves taking inputs of a certain window length, and producing a multiple timestep ahead output from these inputs. These outputs are compared against the actual outputs, and the MAE is found, which is fed back to the model. The figure below [Figure 6] gives a visual representation of multi-step forecasting model.

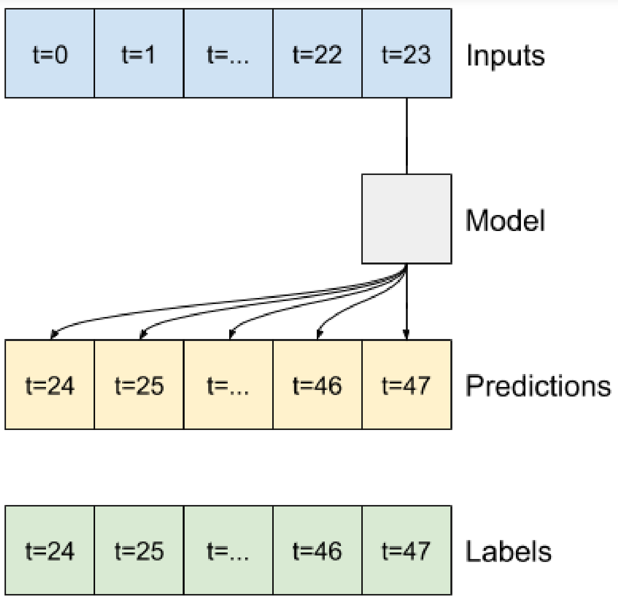


Figure 6: Multi-Step Forecasting Model

8.3 Performance Metric

The metric being used is the Mean Absolute Error (MAE), as it is the most appropriate metric, after the parameters have been normalized. Since the features are normalized from 0-1, MAE depicts the Mean Absolute Percentage Error in the prediction, and is easily interpretable for evaluating performance every time a hyperparameter has been changed.

9 Different Techniques Used

There are many different techniques that can be used for this time series forecasting problem.

9.1 Linear

This technique is a simple multiple linear regression, with the algorithm assigning weights to all weather parameters and predicting the PV output of the next timestep. This technique involves auto-regression of weather parameters and PV output, so that weights can be given to the weather parameters for the next timestep for the multiple timesteps ahead forecasting objective.

9.2 CNN

This technique involved the usage of 1 convolutional layer and 2 hidden dense layers with 64 units, as well as 1 output layer. The convolutional layer contained 32 filters and a 7x7 kernel, connected to a dense neural network with 2 hidden layers of 64 units each. The window size used was 7 timesteps, as this was the most suitable window size, as discussed in the section discussing hyperparameters. The convolutional layer is used to help split each of the data features to help the dense neural network easily predict each part and then give weights to the original inputs of the convolutional layer. The figure below [Figure 7] shows an example of convolutional window.

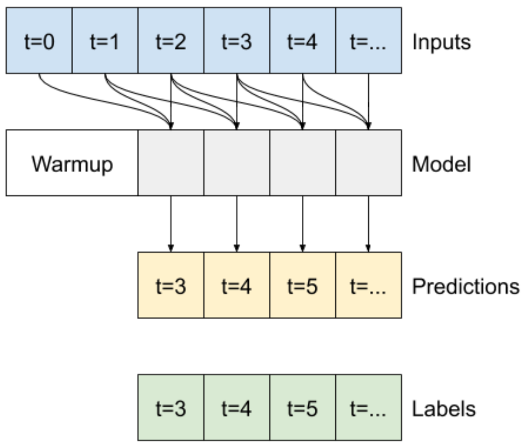


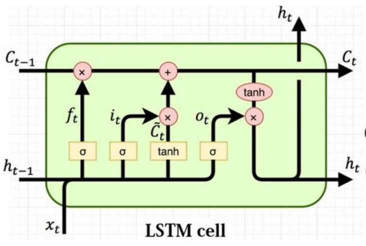
Figure 7: Convolution Window - CNN Model

9.3 Multi-Step Dense

This technique was similar to the single step dense neural network, as it involved 2 dense layers consisting of 64 units each and one output layer, however the window-size was 3 timesteps. This meant that the previous 3 timesteps were being used to predict the next timestep for the single step output. This technique is used to check how important autoregression of weather parameters is to the model. With three timesteps of given input data, it gives the algorithm better knowledge of the dependence of the next timestep on the previous timestep (i.e autoregression).

9.4 CNN LSTM

This technique involved the usage of 1 convolutional layer, 1 LSTM layer, 1 dense layer and 1 output layer. The convolutional layer contains 32 filters and a 7x7 kernel, with the tanh activation function. The window size used was 7 timesteps, as this was the most suitable window size, as discussed in the section discussing hyperparameters. The LSTM layer contains 32 filters as well, whilst the dense neural network consisted of 1 hidden layer of 64 units. First the convolutional layer was used to extract deeper features from the original weather data, and then the LSTM layer, as seen in [Figure 8] is used to help with finding long term time series features. This gives the dense layer more features, which it was able to use to good effect. Then, backward propagation occurred, which helps to find the weights of the original inputs. With multiple timesteps ahead forecasting, the CNN LSTM model also has autoregression taking place, such that the results of the first output, as well as the previous 6 given inputs are used to predict the results of the next output.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 8: LSTM Model with tanh activation function

9.5 RNN LSTM

This techniques involves the usage of one LSTM layer, to help serve as a building block for the other layers. The LSTM helps assign certain weights which helps the RNN let new information in, forget information or assign it importance to the output. As in [Luo et al. 2021], this technique is used instead of a normal recurrent neural network, due to the issue of vanishing gradients. The figure below [Figure 9] gives a visual representation of the RNN model.

9.6 Dense Neural Network

This technique involved using 3 dense layers consisting of 64 units each, and one output layer. The window size for this technique is only 1 timestep, implying that only the previous timestep is used to predict the next timestep. This technique is used as it is important to strike a comparison between the other neural network techniques and the conventional neural network.

9.7 AR LSTM

This technique involves using one LSTM layer, one RNN layer, one dense layer and one output layer. This technique helps with the autoregression of PV through time, and therefore this is used for the multiple time steps forecasting. This

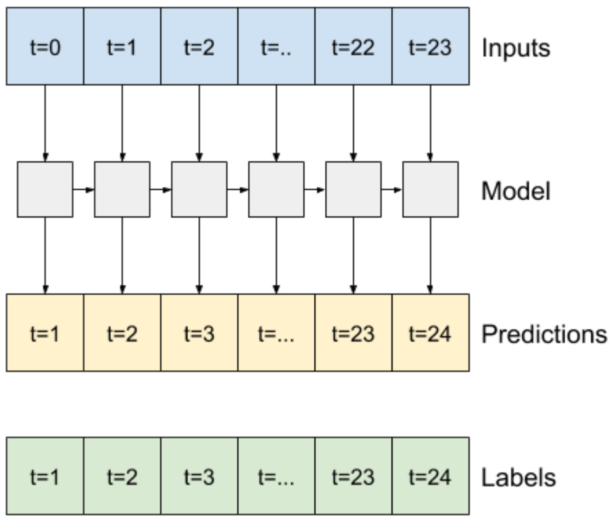


Figure 9: RNN Model Recursion Loop

model has a clear advantage to others, as it can be adjusted easily to give an output length of many timesteps ahead, and due to the autoregression, it has a higher accuracy rate. However, the downside is that this model can only be run well on multiple timesteps forecasting as it as an autoregressive model. For the prediction of one timestep ahead forecasting, it would give no different results to the usual dense network.

9.8 Baseline

The baseline model is used as a metric of comparison, as it shows the error of assuming that the next output is equal to the last PV output. If the baseline error is lower than the error of the other model, it is an indication that either the data is heavily skewed, or the model has been formulated incorrectly.

In a multi-step model, an additional baseline technique is used, which repeats the pattern of the last n given inputs for the next n outputs, where n is the number of timesteps forecasted.

10 Hyperparameters

In this section, the paper discusses the different hyperparameters which need to be tuned and the values they have.

10.1 Regularization Parameter (λ) and Number of Epochs

The number of epochs or iterations the model had to run on the training set is another important hyperparameter. This was highly important to get the optimal weights for the model to ensure that neither underfitting nor overfitting was taking place. In addition to this, another hyperparameter which was used to prevent overfitting was the regularization parameter; λ. This added weight penalized overfitting, but it had to be tuned in order for it not to be too low which would have caused overfitting, but also for it not to be too high, which would have caused underfitting. After testing 4 times, the optimal value for λ turned out to be 1e-2 for the CNN LSTM model and 1e-1 for all the other machine learning models.

10.2 Convolution Window

Another very important hyperparameter for the convolutional neural network was the window width. This would help determine how many previous timesteps would be used to predict the next time step for the various different models. For example, the convolutional window width, used for the CNN and CNN LSTM was 7 timesteps, as it gave results with lower MAE than any of the window sizes greater or smaller than 7.

10.3 Learning Rate(α)

Another hyperparameter which required tuning was the learning rate α. It is necessary to alter the learning rate to an optimum to ensure that neither overshooting nor slow gradient descent was taking place. After 5 iterations of the program, it could be seen that the optimal solution for the learning rate was 8e-4.

10.4 Activation Function

The activation function was another very important hyperparameter to tune. The method used to find the optimal activation function was to take the average of three runs for each of the three activation functions: tanh, relu and sigmoid and compare the mean absolute percentage error for each of the functions. From the results of each of three runs, it could clearly be seen that the optimal function was the tanh activation function. This was a very important hyperparameter, as this caused a significant increase in the model performance.

10.5 Optimizer - Adam

The optimizer being used is the Adam (Adaptive Moment) optimizer, as it is computationally faster than other optimizers, and requires fewer parameters for tuning than other optimizers, as it tunes the parameters by itself.

11 Results

11.1 Single Step Ahead Model

11.1.1 *Linear Weights*

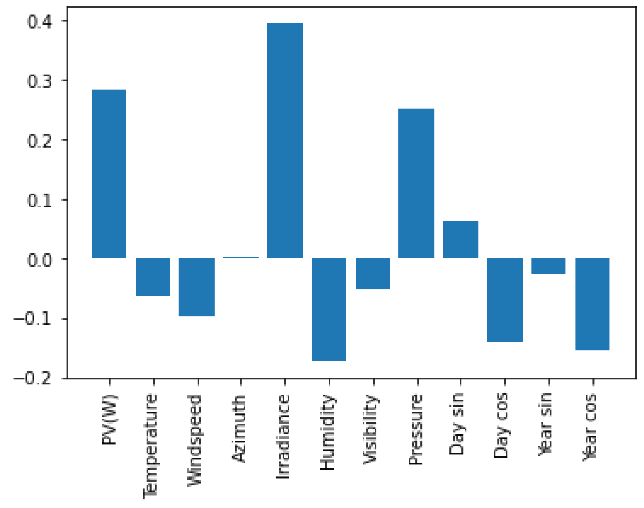


Figure 10: Weights for Linear Regression Model

11.1.2 *Linear*

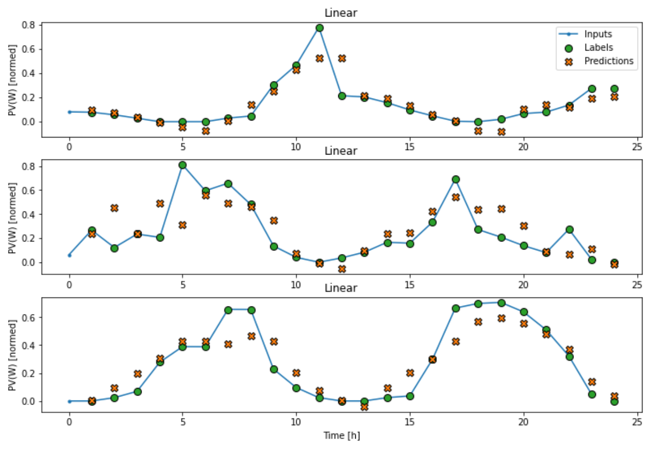


Figure 11: Predicted vs actual values for Linear Regression model

11.1.3 *CNN LSTM*

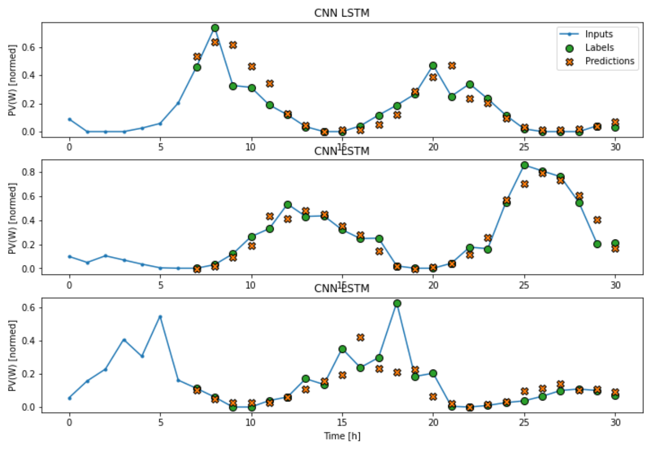


Figure 12: Predicted vs actual values for CNN LSTM model

11.1.4 *RNN LSTM*

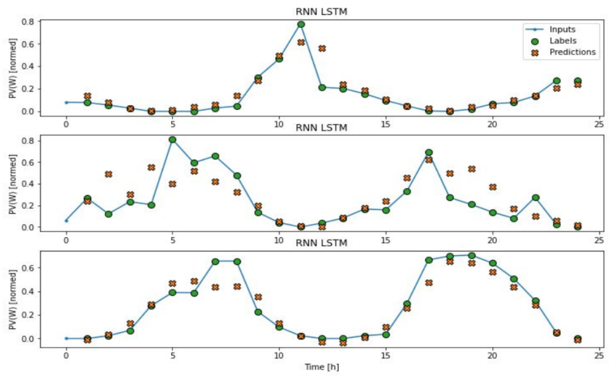


Figure 13: Predicted vs actual values for RNN LSTM model

11.1.5 *CNN*

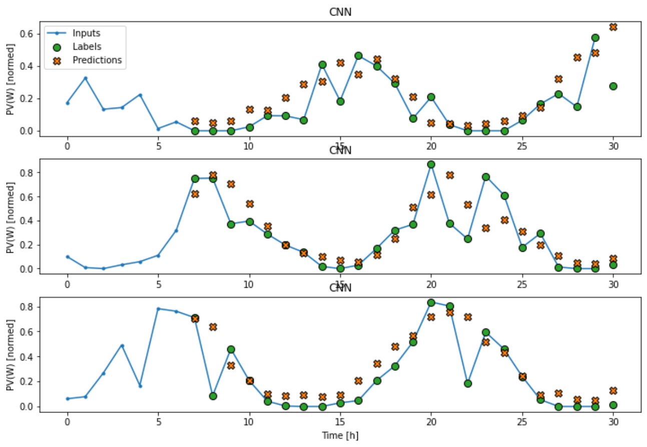


Figure 14: Predicted vs actual values for CNN model

11.1.6 *Comparison*

As can be seen from the figures below [Figure 15] and [Figure 16], the results for the CNN LSTM model are the best, with an error of less than 8 percent for the training set and validation set. Additionally, the RNN LSTM model is also quite successful, giving very similar errors to the CNN LSTM model. This implies that the LSTM layer is highly beneficial in the prediction, and helps with getting the most important features.

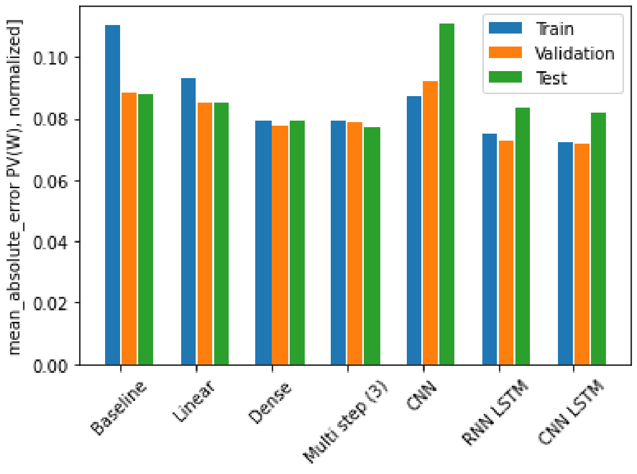
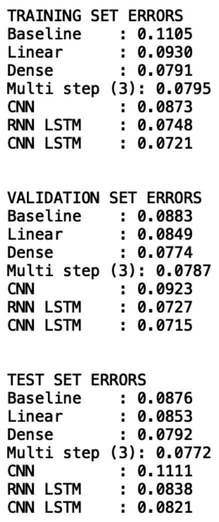


Figure 15: Comparison of MAE for training, test and validation data for all techniques

11.1.7 *Results Table*



```
TRAINING SET ERRORS
Baseline      : 0.1105
Linear        : 0.0930
Dense         : 0.0791
Multi step (3): 0.0795
CNN           : 0.0873
RNN LSTM      : 0.0748
CNN LSTM      : 0.0721

VALIDATION SET ERRORS
Baseline      : 0.0883
Linear        : 0.0849
Dense         : 0.0774
Multi step (3): 0.0787
CNN           : 0.0923
RNN LSTM      : 0.0727
CNN LSTM      : 0.0715

TEST SET ERRORS
Baseline      : 0.0876
Linear        : 0.0853
Dense         : 0.0792
Multi step (3): 0.0772
CNN           : 0.1111
RNN LSTM      : 0.0838
CNN LSTM      : 0.0821
```

Figure 16: Different techniques - MAE for single time step forecasting

11.1.8 *Discussion of Results*

For the single time step ahead forecasting, the results can be seen in [Figure 15] and [Figure 16]. As expected the CNN LSTM model had the best results with a 6 percent error. This is due to the convolutional layer and the LSTM layer, which help to adapt the given features into different features. Given such a vast amount of features, the dense network was able to assign weights to the original features via backward propagation. The RNN LSTM model also performed quite well, as the LSTM layer was able to help with giving the RNN layer certain weights to help the model remember or forget features depending on their importance. Additionally, the problem of vanishing gradients was not present, hence the model was able to backward propagate to give all the weights.

11.2 Multi Step Ahead Forecasting

In this model, the errors were found to be much higher than in the single step ahead forecasting, and the baseline algorithm gave significantly higher errors, of around 0.38 or 0.39 in comparison to the other techniques which gave errors of close to 0.11 or 0.12. This is because, the autoregression is more useful in multiple step ahead forecasting, whereas it is not as useful in the single step ahead forecasting. Therefore, the AR LSTM model has done the best on this data, whereas the other models have not done as well.

11.2.1 *Linear Weights*

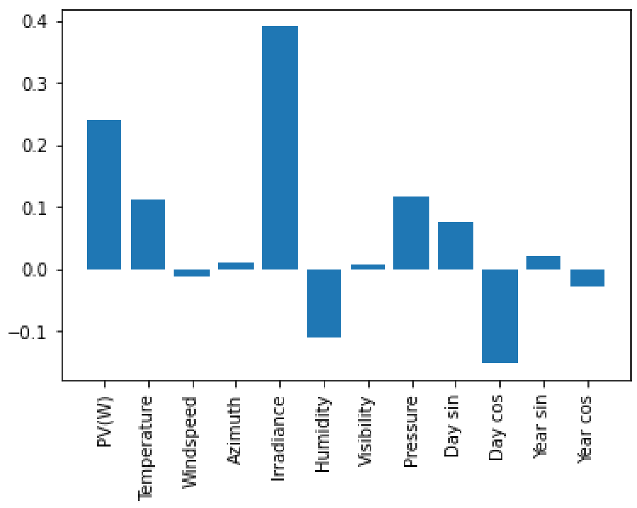


Figure 17: Weights for Multiple Step Ahead Linear Model

11.2.2 *AR LSTM*

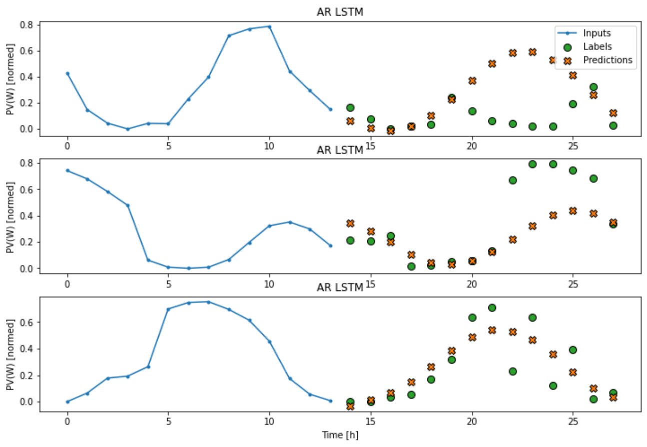


Figure 18: Predicted vs Actual Values for AR LSTM Model

11.2.3 *RNN LSTM*

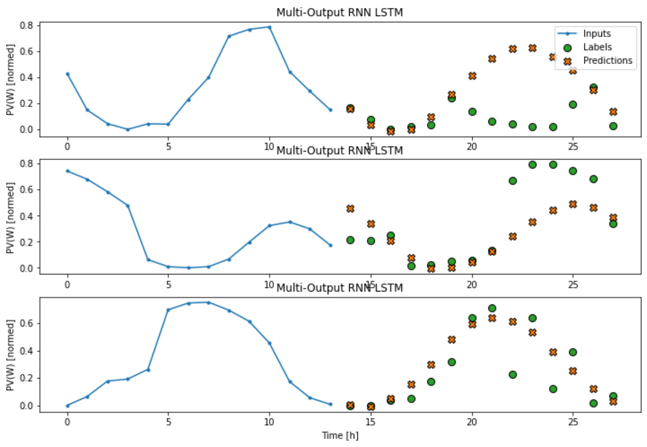


Figure 19: Predicted vs Actual Values for RNN LSTM Model

11.2.4 *CNN*

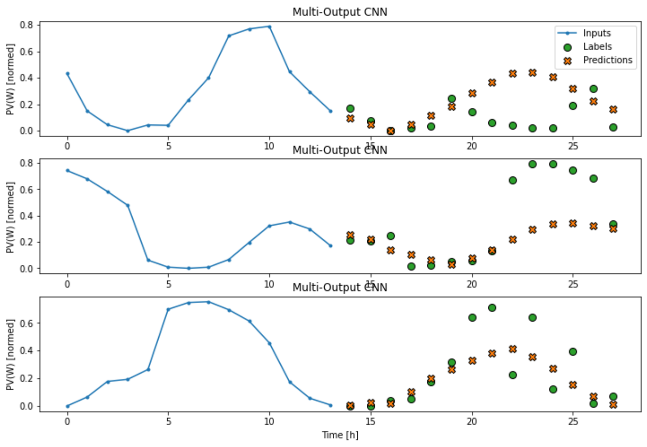


Figure 20: Predicted vs Actual Values for CNN Model

11.2.5 *Dense*

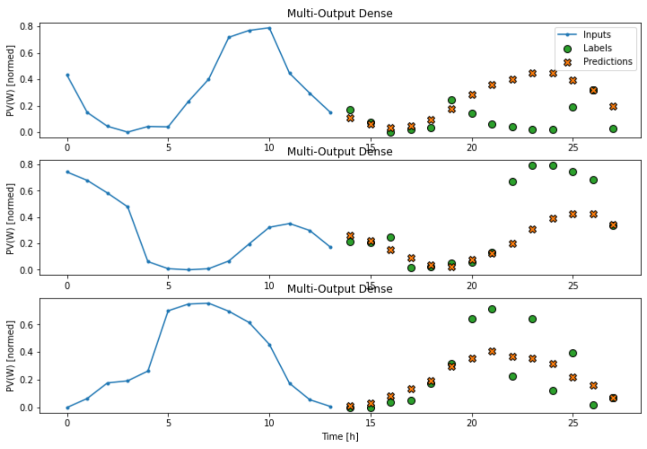


Figure 21: Predicted vs Actual Values for Dense Model

11.2.6 *Linear*

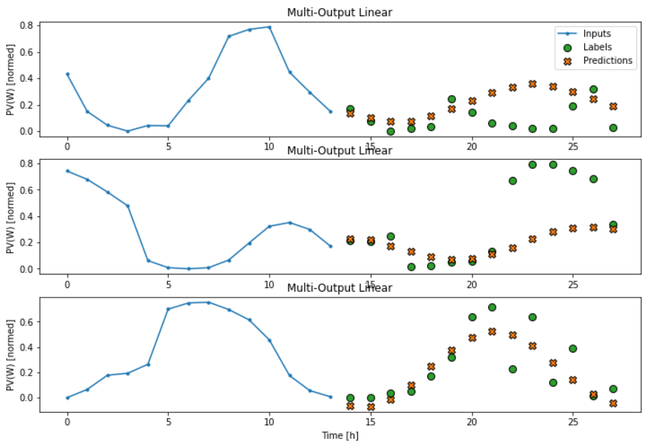


Figure 22: Predicted vs Actual Values for Linear Model

11.2.7 *Comparison*

As can be seen from the figures below [Figure 23] and [Figure 24]. the results for the AR LSTM model are the best, with an error of close to 11 percent for each of the training, test and validation set. The baseline models have a very high error, as they are unable to perform auto-regression, and the data changes drastically for multiple hours ahead forecasting and hence the models that do autoregression have a marked improvement. The RNN LSTM model was also very successful, due to the fact that it used the LSTM layer as well, which helped to reduce the unimportant features, but include more important features. For the RNN in particular, it also got rid of the vanishing gradient problem, hence this gave results of close to 12 percent error. The errors of the multiple step forecasting, however, are very significantly higher than that of the single step ahead forecasting, due to the fact that the errors increase incrementally for each timestep the model is forecasting.

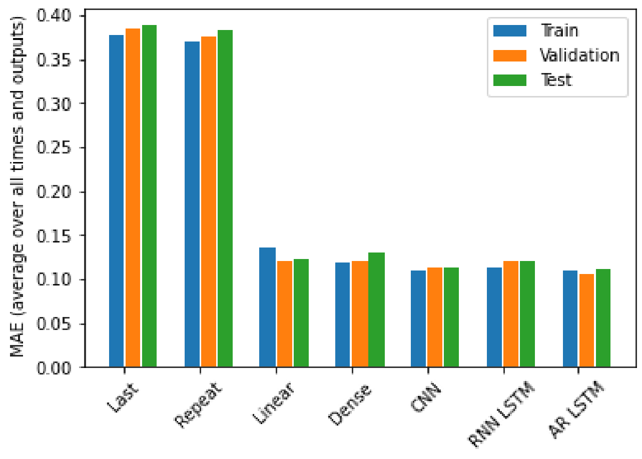
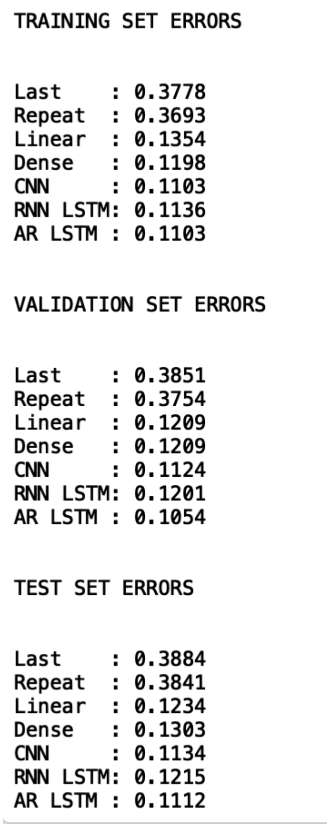


Figure 23: Comparison of MAE for different techniques in training, test and validation set

11.2.8 *Results Table*



```
TRAINING SET ERRORS

Last    : 0.3778
Repeat  : 0.3693
Linear  : 0.1354
Dense   : 0.1198
CNN     : 0.1103
RNN LSTM: 0.1136
AR LSTM : 0.1103

VALIDATION SET ERRORS

Last    : 0.3851
Repeat  : 0.3754
Linear  : 0.1209
Dense   : 0.1209
CNN     : 0.1124
RNN LSTM: 0.1201
AR LSTM : 0.1054

TEST SET ERRORS

Last    : 0.3884
Repeat  : 0.3841
Linear  : 0.1234
Dense   : 0.1303
CNN     : 0.1134
RNN LSTM: 0.1215
AR LSTM : 0.1112
```

Figure 24: Different techniques - MAE for multiple time step forecasting

12 Conclusion and Scope for Future Work

From the results above, it can be concluded that the best technique for solar PV forecasting of one timestep ahead is the CNN LSTM method, and best technique for solar PV forecasting of multiple timesteps ahead is the AR LSTM method. For both techniques, the common layer in the model is the LSTM layer, which helps to remove unimportant features and gives important features higher weights. For the single timestep ahead forecasting, the convolutional layer helps to split the original features into multiple different features, which is fed into the LSTM layer, to find the most important features. For the multiple timestep ahead forecasting, the autoregression of the AR LSTM model helps to better predict the weather data for multiple timesteps ahead, and this in turn makes it easier to predict the solar PV output data.

To give the model a better sense of seasonality, the dataset for training, test and validation set could be split differently. Instead of choosing the first 75 percent to be training data, the first 75 percent of data for each month could instead be chosen to be training data, the next 15 percent of each month to be validation data and the last 10 percent of each month to be test data. In this manner, the algorithm would be able to distinguish between months and give the seasonality for different months, which would make the model better.

Another method which could be used for future improvements is shown in [Wang et al. 2017]. This method involves the clustering of days of data into certain categories, and then performing pairwise predictions on this data using k-nearest neighbors and neural networks. This is useful if weather does not change drastically throughout a day, as it classifies that a day as sunny or rainy and then calculates the PV output given this classification.

Acknowledgements

I thank Professor Zhongming Lu (Hong Kong University of Science and Technology) for guiding me through this project and providing helpful insights, as well as helping me access weather features data from the HKUST Supersite.

I thank Li Mengru (Hong Kong University of Science and Technology) for helping to provide me with data for the PV output.

I thank Professor Wei Shyy (President of Hong Kong University of Science and Technology) for giving me an opportunity to work with one of the esteemed professors in the university.

I thank Professor Hong K. Lo, JP (Chair Professor of Department of Civil and Environmental Engineering at HKUST) for introducing me to Professor Zhongming Lu.

References

Akhter, M. N., S. Mekhilef, H. Mokhlis, R. Ali, M. Usama, M. A. Muhammad, and A. S. M. Khairuddin (2022). “A hybrid deep learning method for an hour ahead power output forecasting of three different photovoltaic systems”. *Applied Energy* 307, page 118185.

Alam, A. M., I. A. Razee, M. Zunaed, et al. (2021). “Solar PV Power Forecasting Using Traditional Methods and Machine Learning Techniques”. In: *2021 IEEE Kansas Power and Energy Conference (KPEC)*. IEEE, pages 1–5.

Elfeky, M. G., W. G. Aref, and A. K. Elmagarmid (2005). “Periodicity detection in time series databases”. *IEEE Transactions on Knowledge and Data Engineering* 17(7), pages 875–887.

Feng, C., J. Zhang, W. Zhang, and B.-M. Hodge (2022). “Convolutional neural networks for intra-hour solar forecasting based on sky image sequences”. *Applied Energy* 310, page 118438.

Gensler, A., J. Henze, B. Sick, and N. Raabe (2016). “Deep Learning for solar power forecasting—An approach using AutoEncoder and LSTM Neural Networks”. In: *2016 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, pages 002858–002865.

Isaksson, E. and M. Karpe Conde (2018). *Solar power forecasting with machine learning techniques*.

Luo, X., D. Zhang, and X. Zhu (2021). “Deep learning based forecasting of photovoltaic power generation by incorporating domain knowledge”. *Energy* 225, page 120240.

Pedregal, D. J. and J. R. Trapero (2021). “Adjusted combination of moving averages: A forecasting system for medium-term solar irradiance”. *Applied Energy* 298, page 117155.

Theocharides, S., G. Makrides, A. Livera, M. Theristis, P. Kaimakis, and G. E. Georghiou (2020). “Day-ahead photovoltaic power production forecasting methodology based on machine learning and statistical post-processing”. *Applied Energy* 268, page 115023.

Wang, Z., I. Koprinska, and M. Rana (2017). “Solar power forecasting using pattern sequences”. In: *International Conference on Artificial Neural Networks*. Springer, pages 486–494.

Zheng, J., H. Zhang, Y. Dai, B. Wang, T. Zheng, Q. Liao, Y. Liang, F. Zhang, and X. Song (2020). “Time series prediction for output of multi-region solar power plants”. *Applied Energy* 257, page 114001.