MECENG 249 Project 3 Report

Implementation of Machine Learning to Evaluate the Performance of a Solar PV Cell

Submitted By:

Joshua Duarte
Pavan M. Reddy

Date: 11-17-2022

Table of Contents

I. Introduction……..….1

II. Nomenclature….……..2

III. Part One….……..3

IV. Part Two…....……..5

V. WorkDistribution……...……18

VI. Appendices……19

I. Introduction

A simplified yet intelligent model that teaches computers to process the data like a human brain is coined as a neural network. There are three types of neural networks namely, Artificial Neural Networks (ANN) Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN). In this project, the implementation and understanding of a simple artificial neural network is conducted. An Artificial neural network is a model with connected input/output layers and hidden layers with neurons to simulate a human brain. These models are widely used in the energy sector to evaluate systems where physical prototyping and evaluation for various operating conditions are not feasible.

In this project, a study about the performance characteristics of solar photovoltaic cells is conducted. A photovoltaic cell, commonly known as a solar cell is a device that converts sunlight into electricity directly using the principle of the photovoltaic effect, which is basically having a potential voltage difference between the two semiconductor slices when light strikes the cell. These solar cells are generally used in combinations of series, parallel, or both to achieve greater performance efficiency.

In the first part of the project, 72 solar cells are connected in series, each having a surface area of $173cm^2$. With the knowledge of the parameters depending on the output performance of the cells, a neural network model is developed and trained using a set of training data with low intensities. This model is then validated and used to determine the outputs for high incident radiation values.

In the second part, the objective is to determine the best combination for the cells to be organized which fetches the best performance output. For this purpose, a neural network is modeled and trained with a set of data and is validated for its accuracy. Then this model is used to determine the best combination for various initial parameters to give the best output. Further using the best combination the power output for each combination is determined and analyzed.

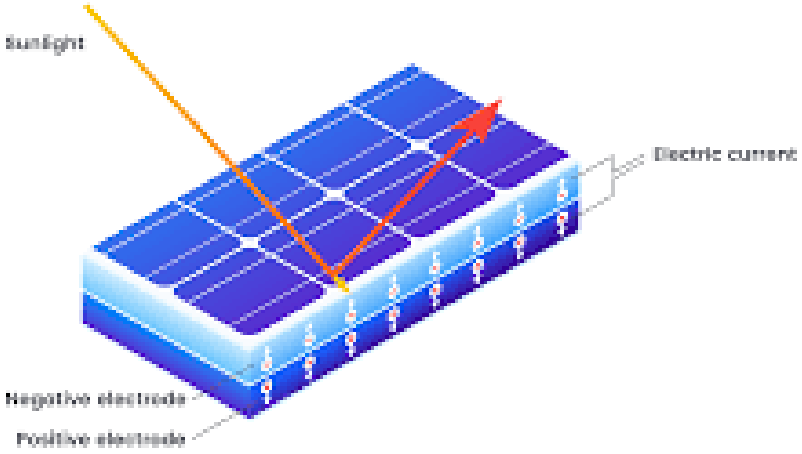

Figure 1: Solar PV cell

II. Nomenclature

I_D	Incident direct normal solar radiation intensity (W/m^2)
T_{air}	Outside air temperatures. (^0C)
R_L	Load Resistance(ohms)
V_L	panel output voltage to load(V)
W	Panel power output (W)
M	Mode stating the combination of the solar cells

III. Part One

This part of the project is focused on understanding the operating parameters and implementation of machine learning models to achieve the best performance. A combination of 72 solar PV cells is connected in series with each having an area of 173cm^2. Ideally, this type of combination is capable of receiving peak incident levels up to 1300W/cm^2. But this model is modified to receive values 50% more than the ideal value. This is achieved by the use of tracking mirrors that reflect additional solar radiation into the panels.

For the purpose of determining the performance of the solar cell in order to achieve higher current and voltage levels, a neural network model is developed and trained using the low intensity/flux operating parameters i.e, with no tracking mirrors, and then used to determine the performance using the high intensity/flux data.

1. Task 1.1

With the understanding of the principal component analysis example code, the same is performed on the low flux data inputs to understand the parameter's relative importance.

a. The low flux operating conditions data file is loaded and standardized. Using the NumPy and math libraries the mean and standard deviation of each parameter namely, T_{air}, I_D , and R_L is calculated. Then to standardize the data each value is subtracted from the respective standard deviation and then divided by the mean.

b. Next, the dependency of each parameter is analyzed. In order to understand this the Eigenvalue of the standardized data is calculated. Using a few simple steps the eigenvalues are determined. We initially find the transpose of the input data matrix and compute the covariance of the same. Then using the linalg library the eigenvalues are calculated. The result helps to understand how each parameter has relative importance on the output performance.

c. Eigenvector obtained:[1.02857143 1.02857143 1.02857143]

Eigenvalue matrix: [[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]]

Visualization: On the basis of the eigenvalues obtained it can be understood that each parameter in the model is equally important. As the matrix is an identity matrix, it justifies the above statement. The data is then represented as a scattered plot.

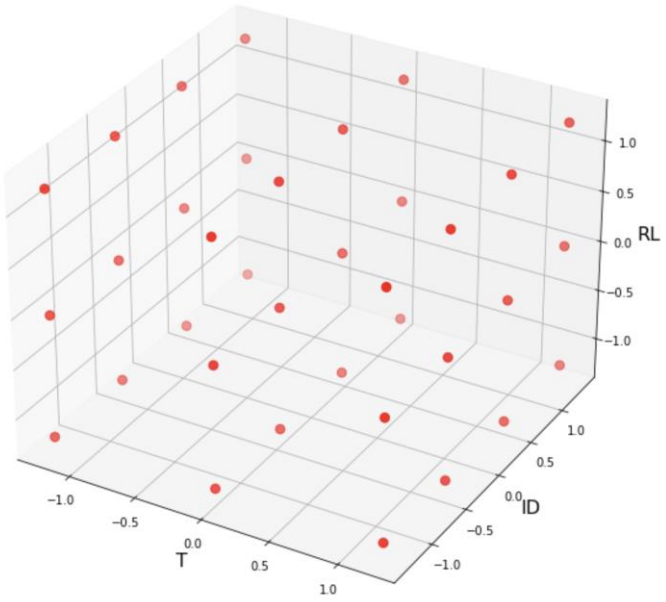

Figure 2: Scattered plot of the input standardized data

2. Task 1.2

This task is oriented towards building and training the neural network to evaluate the performance characteristics of a solar PV cell.

a. Once the relative importance of each parameter is known the input data is used to model a neural network and train it to predict outputs for varying operating conditions. Firstly the input data is normalized, for which the median value of each parameter is calculated and divided respectively.
b. Before inputting the data into the model, the data is divided into two sets, a training set with 2/3rds of the data and a second validation set with 1/3rd of the data. This is done using the scikit-learn, train_test_split function.
c. Next, a Keras sequential neural network with an input layer of 6 neurons, 3 hidden layers with 8, 16, and 8 neurons respectively, and an output layer with 2 neurons is modeled. The elu function is used as the activation function in all layers.
d. Using this model and model.fit function the network is trained to achieve a loss of less than 0.025. To achieve this, the learning rate is reduced in small decremental steps and the patience is increased with the epochs, in the least epochs possible.
   Firstly the model is run with the preset parameter values to check the loss. Next, the learning rate is reduced to 0.015 from the initial 0.020 to check the level of convergence. It's observed there was a significant convergence achieved. Finally, the patience was increased along with the number of epochs to 300 and 1000 respectively. It was noticed that a loss of 0.017054451629519463 was achieved with 1467 epochs in total. The neural network model name is set to "model".

e. Once the model is trained it is used to predict the output values. These are then compared to the training data set by determining the mean absolute error. For calculating the error the built-in mean absolute error function under the sklearn.metrics library is used. A mean absolute error of 0.037755030512470124 W was obtained. Then the predicted vs actual power output data is plotted over a logarithmic plot as shown below.

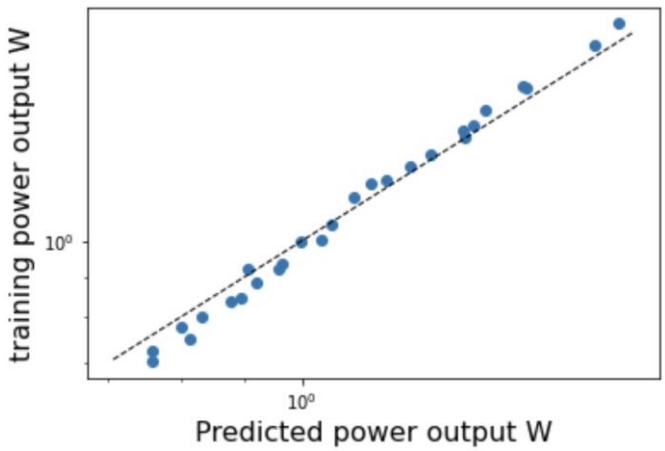

Figure 3: Log-Log plot of predicted vs training

f. For comparing the validation data similar steps are followed and the mean absolute error and the graph are determined. The mean absolute error obtained is 0.07268986425808396 W.

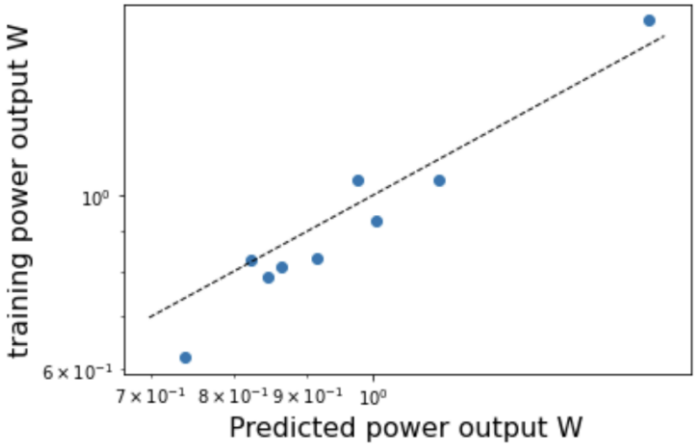

Figure 4: Log-Log plot of predicted vs validation

g. Now coming to the main objective of the model as explained in the introduction for this task is to use this trained model to predict output for the high flux data. So the high flux data is input as the initial input data set and then normalized by dividing each parameter with the respective median value before loading it into the model. Once normalized, this data is used to predict the output data values. The output data set returned is then

compared to the actual data available by calculating the mean absolute error and plotting the logarithmic plot.

The mean absolute error obtained is 0.07705652303362559 W.

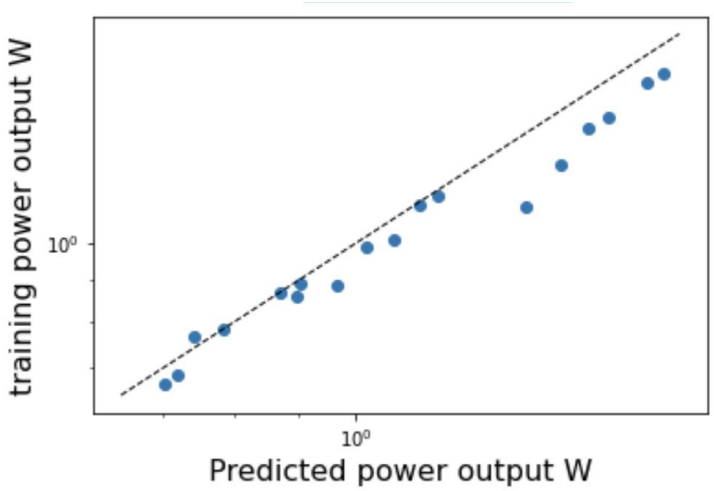

Figure 5: Log-Log plot of predicted vs validation for the high flux data

h. In obtaining a trained neural network, the neural network was used to determine the variation of W for a set of operating conditions. At a fixed T_{air} value equal to 20^0C, 4 ohms < R_L < 8 ohms and 500 < I_D < 1800W/cm^2, a surface plot is created for W as a function of R_L and I_D.

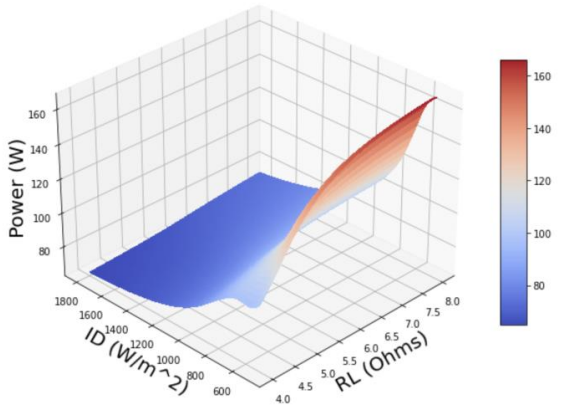

Figure 6: Surface plot of power against radiation intensity and load resistance

3. Task 1.3

The model obtained in task 1.2 is modified with an added second hidden layer with 12 neurons making it a total of 4 hidden layers with 8, 12, 16, and 8 neurons respectively. The model is then trained with the same data set as done earlier to achieve a low loss.

The loss obtained after training the data is 0.01639699749648571 in 1961 epochs with a learning rate set to 0.010. After this, the model is used to predict

the power output and compared to the training dataset. The mean absolute error of this comparison is obtained as 0.05823472226575107 W. Finally, a log-log plot of the predicted vs training data is plotted.

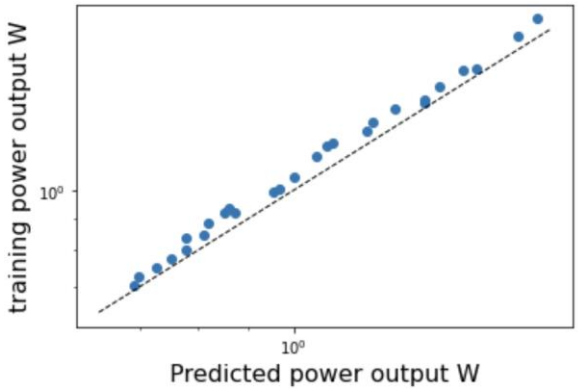

Figure 7: Log-Log plot of predicted vs training

The same is repeated with the validation data set. The mean absolute error was obtained to be 0.05861105916502475 W. The logarithmic plot for the same is shown below.

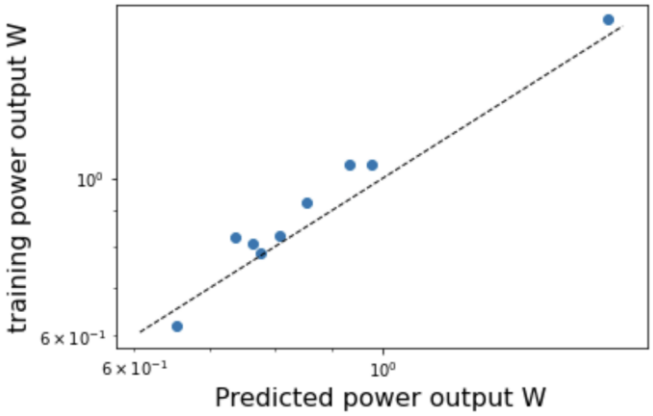

Figure 8: Log-Log plot of predicted vs validation

A surface plot is also plotted as done with the earlier model to analyze the variation of power with the radiation intensity and load resistance when the temperature of the air is kept constant.

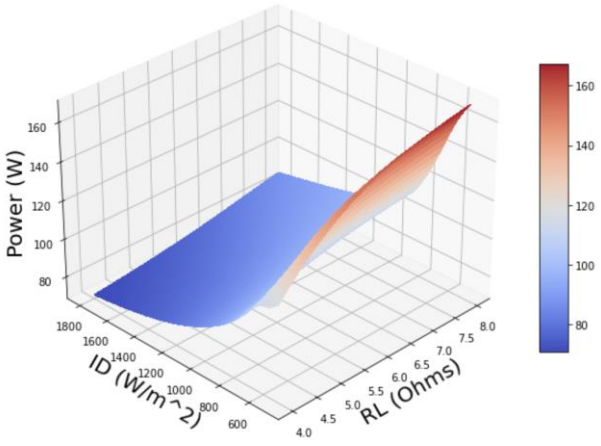

Figure 9: Surface plot of power against radiation intensity and load resistance for the modified model

i. The model was then compared to see if it better matches the data or shows signs of overfitting.

- With the help of the derived mean absolute errors of the training dataset and validation set for both models, comparing them with each other, it can be observed that for the second model the error is lower implying it is better. Hence, the predicted data matches closely with the actual data on the second model.
- Observing the error for the training set and validation set, there is overfitting of both the models as the error is low for the train set but high for the validation set, showing the model predictions are not accurate. But there is a clear improvement in the accuracy with the second model as the mean absolute errors obtained are very close to each other.

IV. Part Two

Part one aimed to provide us with a greater understanding in developing and evaluating an artificial neural network in assessing the power output of PV panels. With this greater understanding, part two asks us to develop a separate neural network model that can predict the most effective mode with respect to specific operating conditions. A mode in this regard is the manner in which a PV system that is composed of four solar panels is structured. The three modes with the four solar panels are wired in parallel (mode 0), 2x2 in series/parallel (mode 1), or having each of the panels wired in series (mode 2). These are visualized in figure.10 which also provides the V-I characteristics of each respective mode.

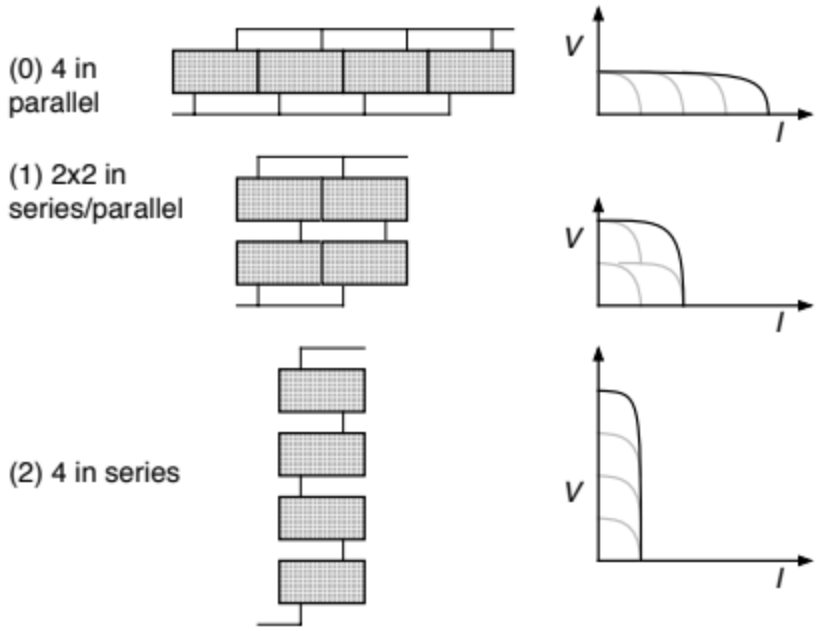

Figure 10: Four PV panel System in different modes

1. **Task 2.1**

   This task develops a neural network model that predicts the power output of a PV solar panel for a set of specified operating conditions. Unlike the previous tasks, this model incorporates data that is tied to the specific modes that were previously described.

a) This is done by beginning with a new skeleton code that was adjusted to incorporate the new data set. Prior to introducing this new data set, the parameters are normalized by dividing the parameters by their respective median values.

b) The new set of data was then randomly split between a training set and a validation set with ¾ of the data set being used for training. This is done using the scikit-learn, train_test_split function.

c) The normalized training data set is input within a skeleton script of a neural network. A sequential neural network named “modelpower” is created. Based on the experience and class lectures, the number of inputs, the number of hidden layers, and the number of neurons within each layer are selected. The challenge is to make it complex enough to accurately fit the data, but not make it too complex where the model is overfitted to the data or requires too many iterations to reach convergence. Using this basic architecture the modelpower network model is created with an input layer of 12 neurons(three times the number of input parameters), and an input_shape of 4. There are three hidden layers created with 32, 16, and 8 neurons respectively. All the layers are assigned the relu activation function. Finally, an output layer with 2 neurons is created.

d) By training the model with the training set and adjusting the learning rate, the model reached a mean absolute error loss of 0.0196899589565 W, which was well below the 0.020 goal. To achieve this the model was run three times, where with each iteration the learning rate was reduced with increasing the number of epochs. The final learning rate used is 0.0067 and the model was trained for 3150 epochs in total.

e) With the trained model, the predicted values were compared to the data set and can be visualized below in the logarithmic plot. With the predicted and trained value data, the mean absolute error was calculated to be 0.03017523549085747 W which proves that the model trained is accurate.

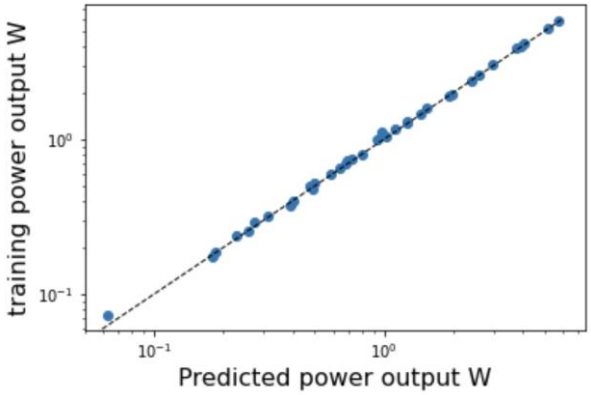

Figure 11: Log-Log plot of predicted vs training

f) The trained model was then verified for accuracy and potential overfitting through the use of the validation data that comprised ¼ of the original provided data set. In doing so, a separate log-log plot was created to compare the predicted values which can be seen below in the logarithmic plot. The mean absolute error was also calculated to be 0.57812509994 W which shows the data is overfit.

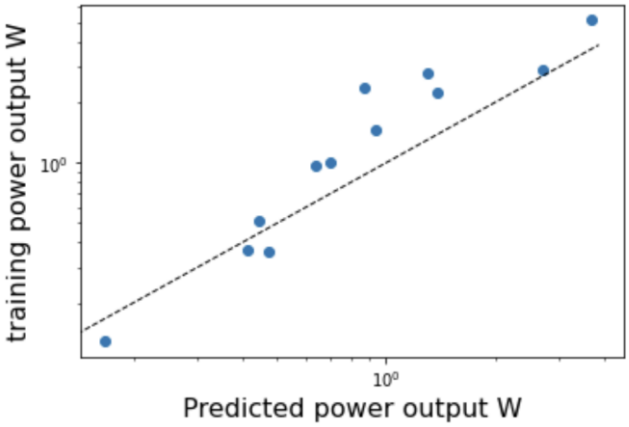

Figure 12: Log-Log plot of predicted vs validation

Task 2.2

Now that a neural network model is modeled and trained to predict the output power, the analysis of which mode to be used to maximize the output for a set of operating input parameters is to be performed. For this, another neural network model is modeled and trained with a set of input data sets and a set of output labels.

a) Similar to all tasks, the input data set is first normalized using the median values calculated for the respective parameters. Here, the median values are stored in variables Tamed, IDmed, and RLmed to ease their use later in the task. A note is that the output data is not normalized as they are just the mode numbers.

b) Next, the ydata provided in the code is an array of 0,1, and 2 values which represent the type of mode. Now for the purpose of training the model, they are converted and replaced with a one-hot encoding array as shown below.

[0.] with [1, 0, 0]
[1.] with [0, 1, 0]
[2.] with [0, 0, 1]

Then this one-hot encoded data is stored in an array named ydataCatOHEarray for further use.

c) Before inputting the data into the model, the data is divided into two sets, a training set with 3/4ths of the data and a second validation set with 1/4th of the data. This is done using the scikit-learn, train_test_split function. Once the split is done a line of code (num_classes = 3) is used to set the number of classes to 3.

d) Once the initial setup of the data is done, the neural network model is built. Here, a model with one input layer of 16 neurons, two hidden dense layers with 32 and 16 neurons respectively, and an output layer with the num_classes is used i.e, set to 3 which is done in the earlier step. The neural network model was defined using a softmax activation function. The name of the model is set to “modelmode”.

e) In the next cell, the model compiler statement is completed as given. Here the adam optimizer is used instead of the usual RMSpropoptimizer used in all previous models. This optimizer automatically adapts the learning rate to achieve convergence in the least number of epochs.

f) The model.fit statement is then completed with the input and output data set. The variable historydata is then replaced with the model_train variable. Then the model is trained to achieve a very low loss. First, cell 4 is run once to check the loss achieved. Then the number of epochs changed to 1000 and with the patience at 100, the cell is rerun. There was a significant reduction in the loss. Hence for a final run, the number of epochs was increased to 1500 and patience set to 200, and then the cells 3 and 4 were run together. A loss of 0.0046219672076404095 W was achieved.

g) Using the skeleton code, the training versus validation set is done and then the loss and the accuracy of the trained model are determined. For each input data set the M_{maxint} is determined and printed. In doing so, the input array is loaded

individually into the test variable using a for loop. Then the test variable is pushed into the trained model “modelmode”, to predict the mode which gives the maximum performance.

h) Using the trained model the loss and accuracy are calculated and tabulated.

```
training versus validation comparisons
30/30 [==============================] - 0s 41us/step
train loss: 0.00013716959801968187
train accuracy: 1.0
10/10 [==============================] - 0s 90us/step
validation loss: 0.0002462394186295569
validation accuracy: 1.0
```

i) Now the loss value of the training and validation are analyzed. It’s seen the loss of the training set is less than the validation set showing that there is an overfitting of the model. The accuracy of both sets being 1.00 says the model can predict data for any test data accurately.

j) Now as there was overfitting observed in order to reduce it, dropout layers are added after the hidden layers in cell 5. Here a dropout probability of 0.25 is set. Then the whole model is run and trained to achieve a loss of less than 0.020. For achieving the convergence similar to the previous training method the number of epochs and patience is increased gradually.; Here the dropout probability is modified by a bit if there is not much of a difference seen in the fitting of the model.

```
training versus validation comparisons
30/30 [==============================] - 0s 38us/step
train loss: 0.021406684070825577
train accuracy: 1.0
10/10 [==============================] - 0s 100us/step
validation loss: 0.03500159829854965
validation accuracy: 1.0
```

Model1: model2:

x=valid loss/train loss~1.81 x=valid loss/train loss~1.62

Using the data obtained we see that the ratio of the validation to train loss is less after the addition of the dropout layers. For the initial model, the loss was nearly double the training loss for the validation set stating a clear overfit. But it was reduced as the loss difference was reduced between the sets. Hence the addition of the dropout layers does help in reducing the overfitting of the model to an extent.

k) Using the modelmode from task 2.2 the best mode is predicted and tabulated in the table below. Then the modelpower is used to predict the power for each input data with a specified mode. These values are then tabulated below.

T_{air} (deg, C)	I_D (W/m^2)	R_L (Ohms)	$M_{max,\ int}$ Predicted by Task 2.2 model$_I$	$\dot{W}(W)$Predicted by Task 2.1 model for M = 0	$\dot{W}(W)$ Predicted by Task 2.1 model for M = 1	$\dot{W}(W)$ Predicted by Task 2.1 model for M = 2
10.0	200	50.	1	0.5408702	1.655226	0.40792674
20.0	200	130.	2	0.17959945	0.5016117	0.68948674
10.0	500	40.	1	0.584461	3.736253	2.667686
20.0	500	80.	2	0.250112	1.0798082	1.9486499
20.0	700	30.	1	0.57322764	3.8921432	2.2643924
20.0	700	55.	2	0.43575126	1.4929088	3.659889
10.0	1000	12.	1	1.6326834	5.246042	0.7956688
20.0	1000	25.	2	0.65983796	3.5259593	1.2904174
20.0	1000	39.	2	0.57386976	2.5300159	3.9876418

Table 1: Task 2.1 and Task 2.2 Model Prediction Output

In comparing the mode that is predicted to produce the greatest power output from the task 2.2 model, the task 2.1 model was able to accurately predict the highest value mode for 9 of the 10 scenarios. With this in mind, it would be safe to say that the model would be accurate enough to be used to accurately switch setting the optimal performance in the multi-mode 4 PV panel system.

V. Work Distribution

Throughout the project timeline, the work was individually done and then compared as most tasks were to be done by every member. Using GitHub, and living close in proximity, the team was able to easily collaborate. The report was worked on collaboratively.

VI. Appendices

1. Figures

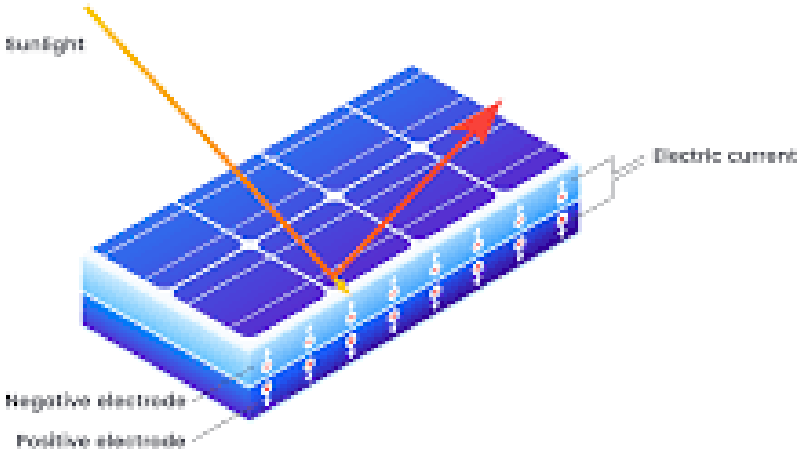

Figure 1: Solar PV cell

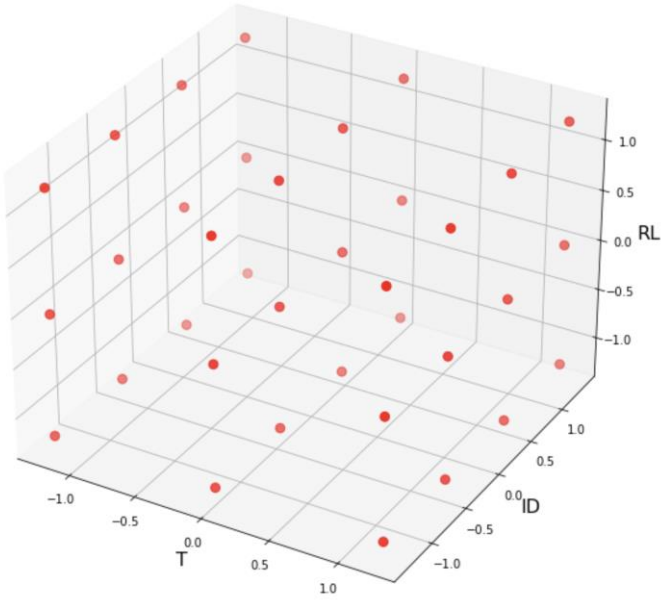

Figure 2: Scattered plot of the input standardized data

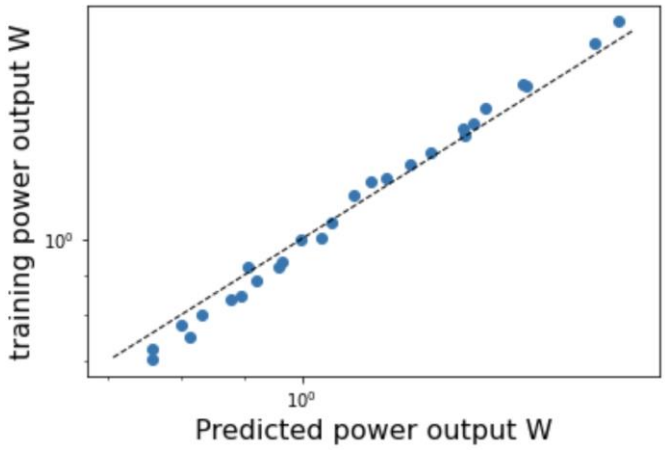

Figure 3: Log-Log plot of predicted vs training

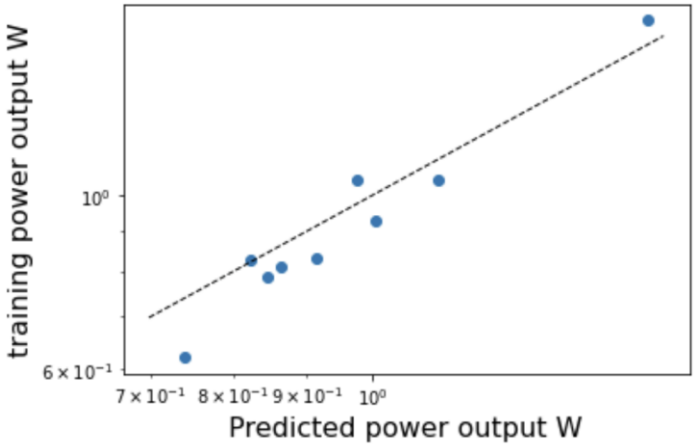

Figure 4: Log-Log plot of predicted vs validation

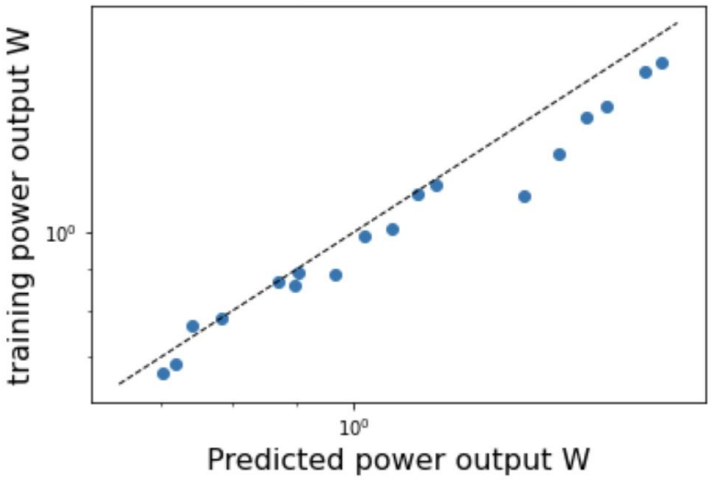

Figure 5: Log-Log plot of predicted vs validation for the high flux data

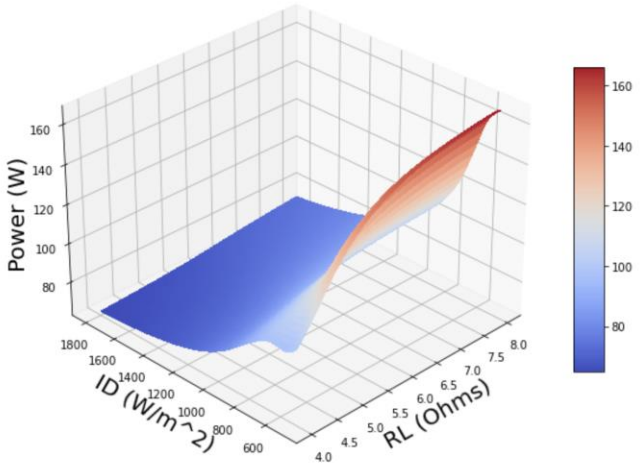

Figure 6: Surface plot of power against radiation intensity and load resistance

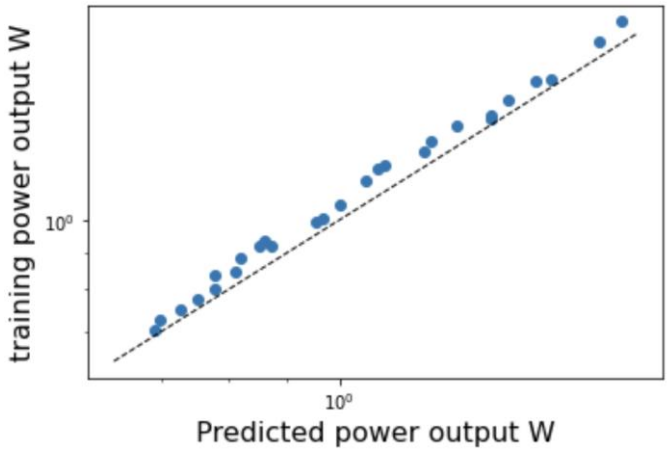

Figure 7: Log-Log plot of predicted vs training

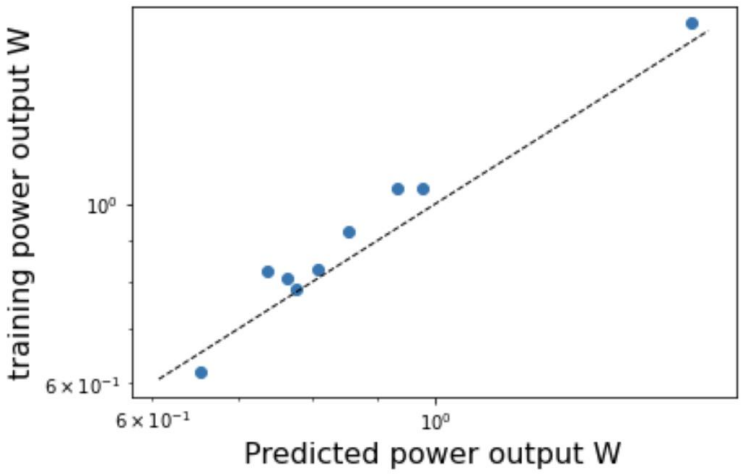

Figure 8: Log-Log plot of predicted vs validation

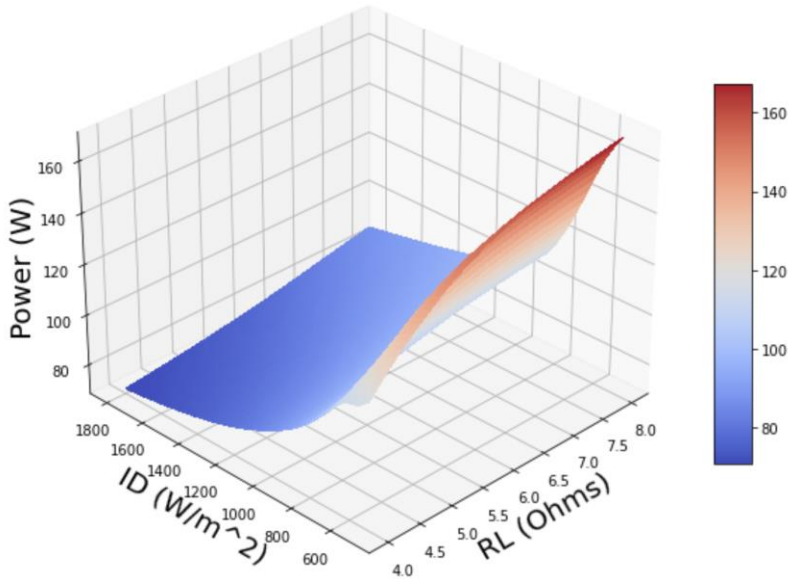

Figure 9: Surface plot of power against radiation intensity and load resistance for the modified model

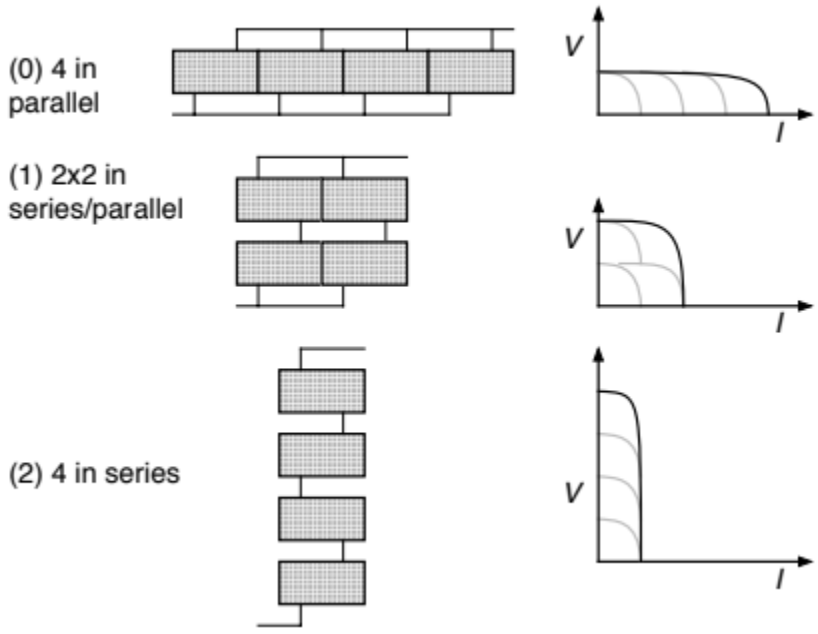

Figure 10: Four PV panel system in different modes

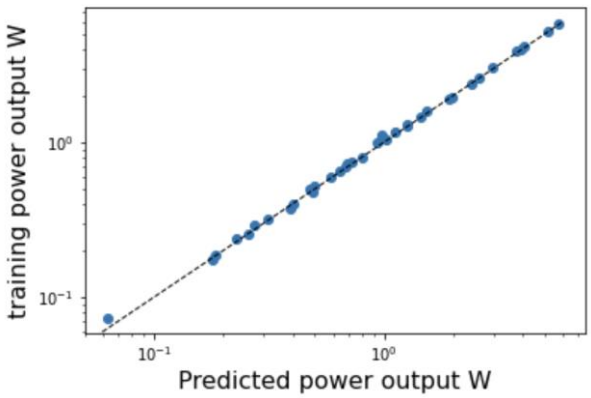

Figure 11: Log-Log plot of predicted vs training

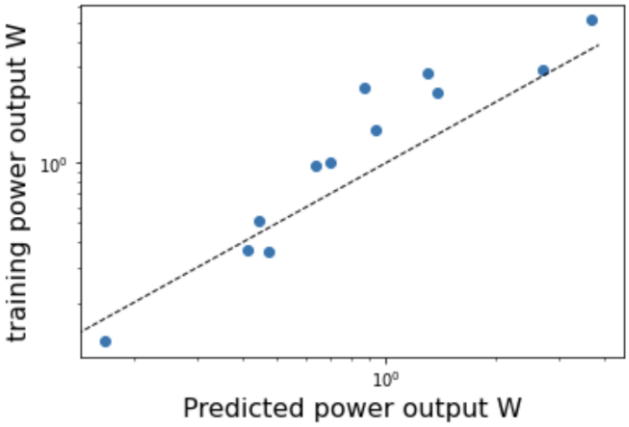

Figure 12: Log-Log plot of predicted vs validation

2. Tables

T_{air} (deg, C)	I_D (W/m^2)	R_L (Ohms)	$M_{max,\ int}$ Predicted by Task 2.2 $model_I$	$\dot{W}(W)$ Predicted by Task 2.1 model for M = 0	$\dot{W}(W)$ Predicted by Task 2.1 model for M = 1	$\dot{W}(W)$ Predicted by Task 2.1 model for M = 2
10.0	200	50.	1	0.5408702	1.655226	0.40792674
20.0	200	130.	2	0.17959945	0.5016117	0.68948674
10.0	500	40.	1	0.584461	3.736253	2.667686
20.0	500	80.	2	0.250112	1.0798082	1.9486499
20.0	700	30.	1	0.57322764	3.8921432	2.2643924
20.0	700	55.	2	0.43575126	1.4929088	3.659889

10.0	1000	12.	1	1.6326834	5.246042	0.7956688
20.0	1000	25.	2	0.65983796	3.5259593	1.2904174
20.0	1000	39.	2	0.57386976	2.5300159	3.9876418

Table 1: Task 2.1 and Task 2.2 Model Prediction Output

3. Code