Global Radiation Prediction Model

UG Project

Paras Bajpai
20124033
Mathematics and Computing
Indian Institute of Technology (BHU), Varanasi

Contents

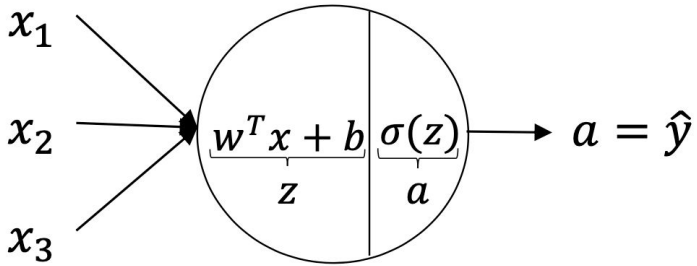
Source:

Motivation and Introduction

- Over the years, the use of fossil fuels has deteriorated climate conditions on Earth. Even though a little late, but we are waking up to the disastrous effects this can have on the ecosystems surrounding us and are turning to renewable energy sources.
- Solar energy is one of the most reliable sources. The amount of energy generated in a region is directly proportional to the solar irradiance received over time.
- The model aims to predict Clearsky DHI, Clearsky DNI and Clearsky GHI values. These predictions can be made effectively using time series forecasting. This in turn will help optimise solar power production.

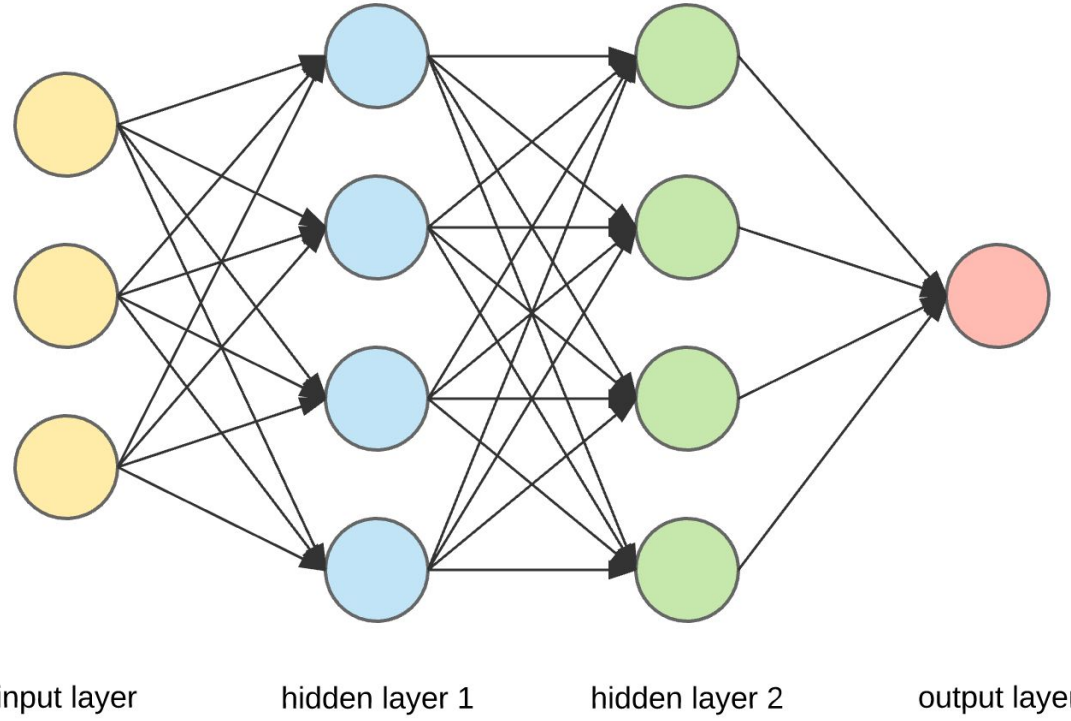
Deep Neural Networks

A neural Network is a collection of interconnected neurons, where each neuron calculates a value based on a certain set of input parameters. These calculations are based on the following equations, which are regularly updated using optimization algorithm using the training dataset.



Here the sigmoid function can be replaced with different activation functions.

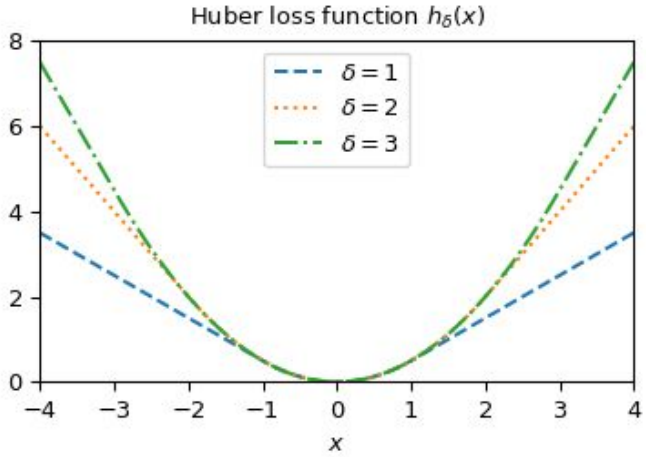
Several layers of these neurons can be stacked on top of one another to create deep neural nets



Huber Loss

It is a loss function which balances the effects of the Mean Squared Error and Mean Absolute Error loss functions.

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \leq \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

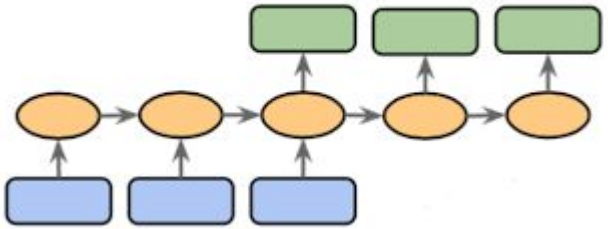


Provides well balanced models which handle both general cases and outliers well.

Sequential Models

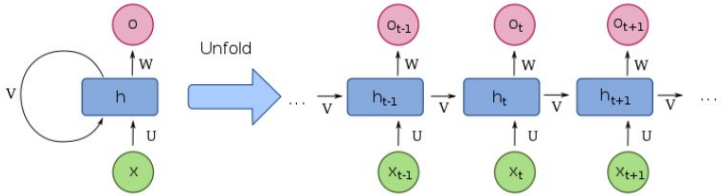
Sequential Models are machine learning models that can process sequences of data. Sequential data includes time series data, audio clips, video clips, text streams, etc. The following architectures are mostly used to implement sequence models:

- RNNs
- GRUs
- LSTMs



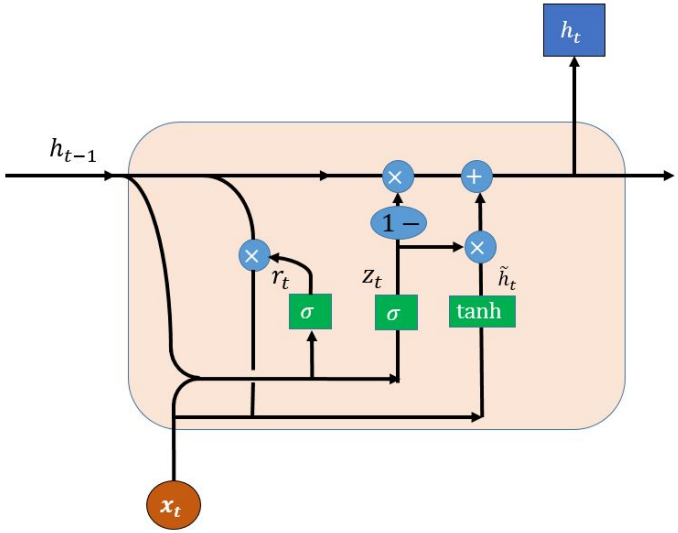
Recurrent Neural Networks

- A RNN calculates its output by factoring in previous output and the current input. This short term memory allows the RNNs to retain past information and hence uncover relationships between data points far away in the sequence.
- RNNs work by doing back propagation through time wherein parameters are updated from the last to the first time step.
- This often results in the vanishing gradient and the exploding gradient problem when trying to remember long term dependencies.



Gated Recurrent Unit

To solve the vanishing gradient problem, GRUs use methods called as the Update Gate and Reset Gate. These gates are basically vectors which allow the network to decide what information should be passed to output.



$$\mathbf{R}_t = \sigma\left(\mathbf{V}_R\mathbf{X}_t + \mathbf{U}_R\mathbf{h}_{t-1} + \beta_R\right)$$

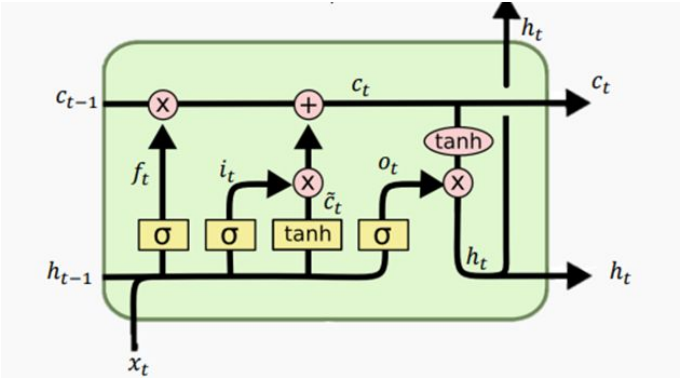
$$\mathbf{Z}_t = \sigma\left(\mathbf{V}_Z\mathbf{X}_t + \mathbf{U}_Z\mathbf{h}_{t-1} + \beta_Z\right)$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{V}\mathbf{X}_t + \mathbf{U}\left[\mathbf{R}_t \odot \mathbf{h}_{t-1}\right] + \beta_1\right)$$

$$\mathbf{h}_t = \mathbf{Z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{h}}_\mathbf{t}$$

Long Short Term Memory

LSTMs is a Recurrent Neural Network architecture that is capable of handling long term dependencies. To do so, LSTMs make use of 3 gates: forget, input and output gate. The key to the LSTM architecture is the cell state, and the network has the capability to decide what to add and remove from the cell state using its gates.



$$\mathbf{f}_t = \sigma\left(\mathbf{V}_f\mathbf{X}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \beta_f\right) \qquad \mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{i}_t = \sigma\left(\mathbf{V}_i\mathbf{X}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \beta_i\right) \qquad \mathbf{o}_t = \sigma\left(\mathbf{V}_o\mathbf{X}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \beta_o\right)$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{V}\mathbf{X}_t + \mathbf{U}\mathbf{h}_{t-1} + \beta_1\right) \qquad \mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)$$

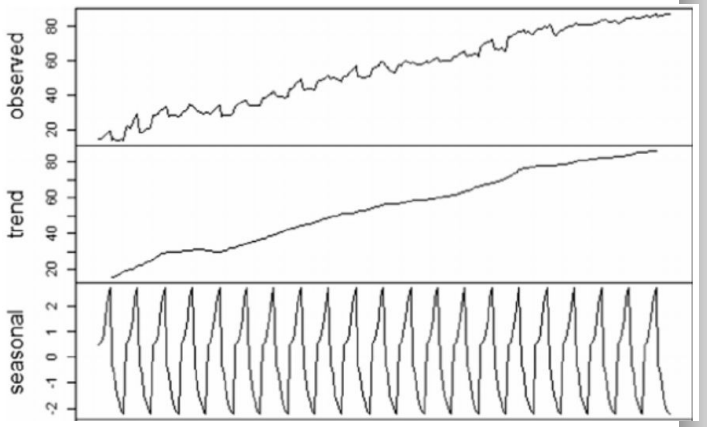
Time Series Forecasting

Simply put a time series is a collection of data points ordered in time. Time series data often has several important characteristics:

Trend: It represents the long term change in the level of the time series.

Seasonality: It refers to seasonal fluctuations in the data. If the correlation graph has a period, then its value will be the seasonality of the time series.

Stationarity: A time series is stationary if its statistical properties like mean and variance do not change over time.



Modelling a Time Series

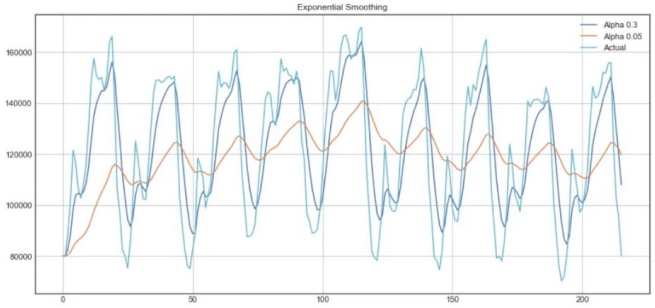
Moving Average

This model simply states that the next observation is the mean of all past observations. We usually define a window over which we apply the moving average model.

Exponential Smoothing

Uses a similar logic to moving averages, but here we assign a decreasing weight to each observation.

$$y = \alpha x_t + (1 - \alpha) y_{t-1}, t > 0$$



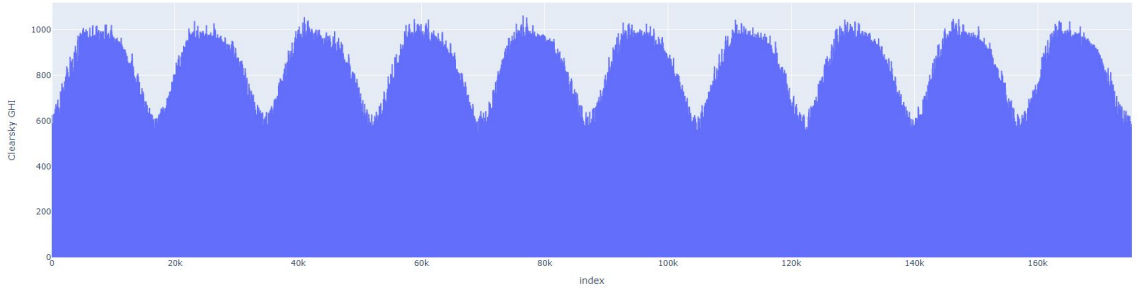
Time Series Forecasting using LSTM

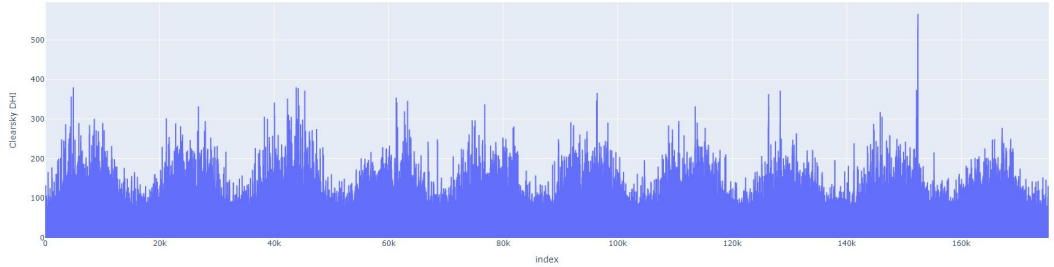
- The model works using a Deep Neural network consisting of LSTM layers along with dense layers.
- A window of 400 days has been created to make predictions regarding future values.
- Further the entire training dataset has been divided into batches of size 512.
- The multiple epochs the model makes ensures proper tuning of each of the parameters.

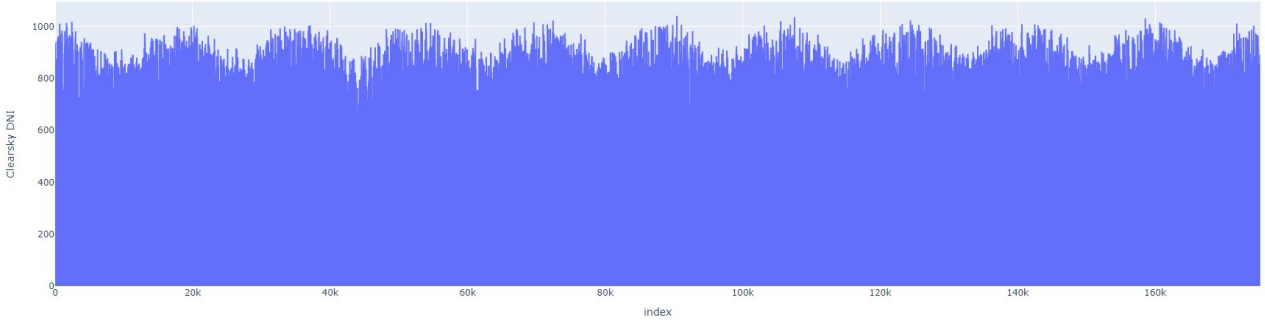
Exploratory Data Analysis

	Year	Month	Day	Hour	Minute	Clearsky DHI	Clearsky DNI	Clearsky GHI	Cloud Type	Dew Point	Temperature	Pressure	Relative Humidity	Solar Zenith Angle	Precipitable Water	Wind Direction	Wind Speed	Fill Flag
0	2009	1	1	0	0	0	0	0	0	0.0	5.0	1010	75.34	106.15	0.499	346.1	3.1	0
1	2009	1	1	0	30	0	0	0	0	1.0	5.0	1010	80.81	112.28	0.490	346.1	3.1	0
2	2009	1	1	1	0	0	0	0	4	0.0	5.0	1010	78.27	118.50	0.482	347.9	3.2	0
3	2009	1	1	1	30	0	0	0	4	0.0	4.0	1010	78.27	124.78	0.478	347.9	3.1	0
4	2009	1	1	2	0	0	0	0	4	0.0	4.0	1010	76.45	131.12	0.475	350.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
175291	2018	12	31	21	30	51	555	168	4	19.4	20.8	1008	91.77	77.86	3.700	204.0	3.5	100
175292	2018	12	31	22	0	37	388	84	4	19.1	20.1	1008	93.88	83.03	3.800	209.0	3.2	100
175293	2018	12	31	22	30	15	115	18	7	19.1	19.6	1008	96.83	88.32	3.800	208.0	2.6	57
175294	2018	12	31	23	0	0	0	0	7	18.7	19.2	1009	96.84	94.34	3.700	206.0	2.1	0
175295	2018	12	31	23	30	0	0	0	7	18.7	19.2	1009	96.84	100.22	3.700	206.0	2.1	0

175296 rows × 18 columns


Clearsky GHI
1000
800
600
400
200
0
0
20k
40k
60k
80k
100k
120k
140k
160k
index


Clearsky DHI
500
400
300
200
100
0
0
20k
40k
60k
80k
100k
120k
140k
160k
index

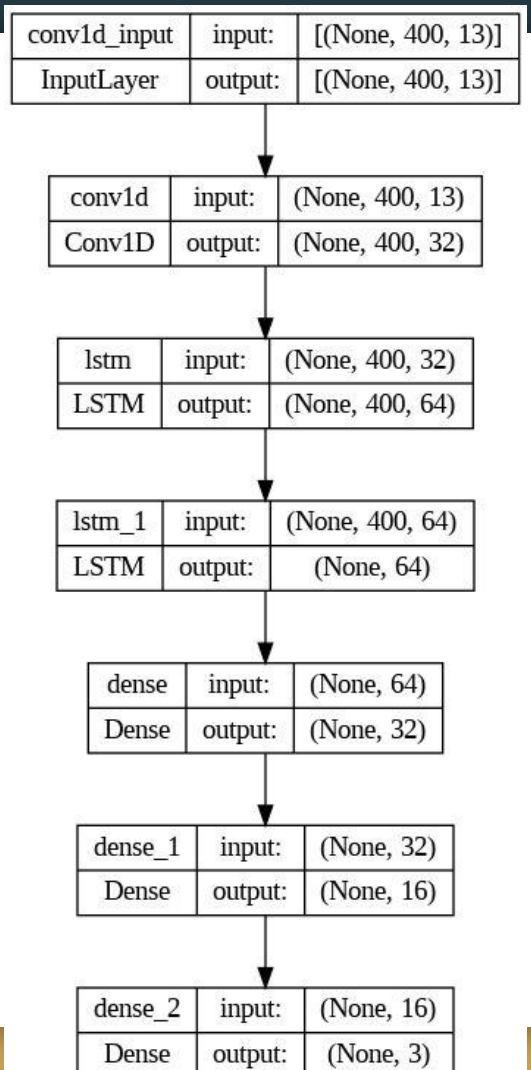

Clearsky DNI
1000
800
600
400
200
0
0
20k
40k
60k
80k
100k
120k
140k
160k
index

Creating Training Data

```
win_length = 400     # number of days used for forecasting
batch_size = 512     # number of samples processed before the model is updated
num_features = 13    # important features
```

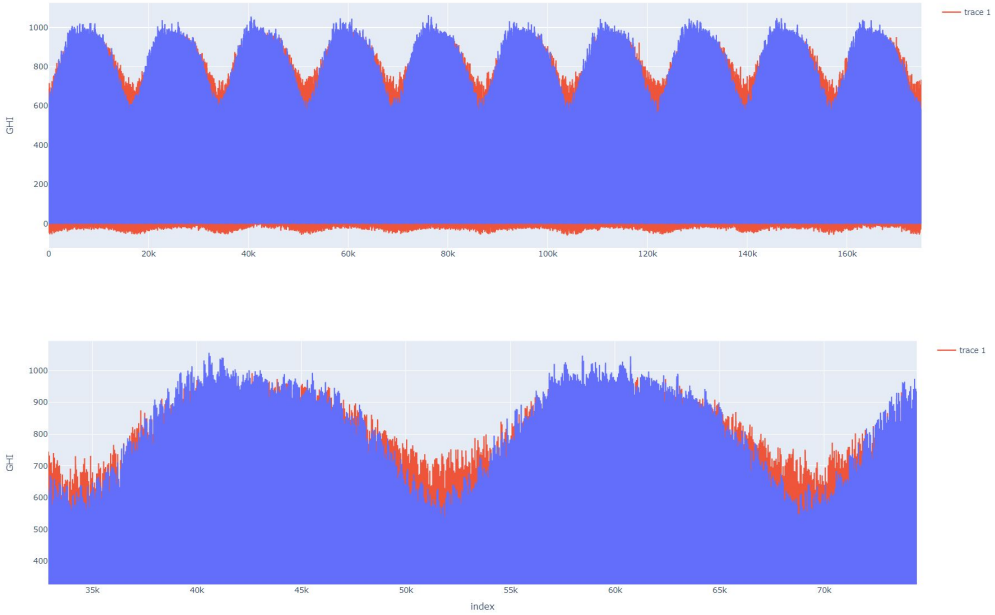
```
# normalizing the data
scaler = MinMaxScaler()
data = scaler.fit_transform(data)
```

```
# the final vector (dataset) passed to train the model
train_generator = TimeseriesGenerator(features, target, length = win_length, sampling_rate = 1, batch_size = batch_size)
```



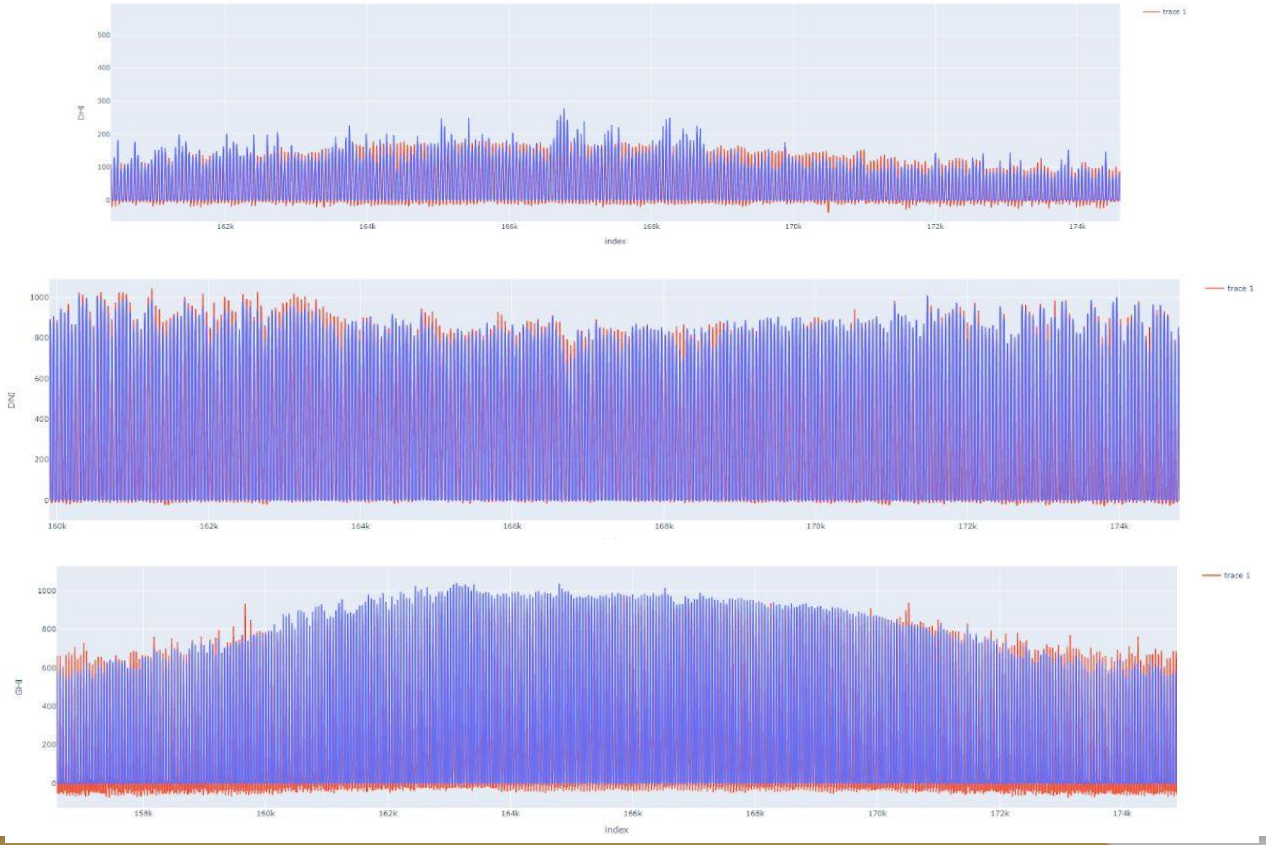
Creating the Model

Results on Training Data




```
Epoch 90/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0305
Epoch 91/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0304
Epoch 92/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0303
Epoch 93/100
342/342 [==============================] - 23s 67ms/step - loss: 0.0011 - mae: 0.0302
Epoch 94/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0300
Epoch 95/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0299
Epoch 96/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0298
Epoch 97/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0297
Epoch 98/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0011 - mae: 0.0296
Epoch 99/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0011 - mae: 0.0294
Epoch 100/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0010 - mae: 0.0292
```

Results on Test Data



Conclusion

- To build this project, I studied the concepts of deep neural networks. This included building sequential models, RNNs, LSTMs, and then applying these to do time series forecasting.
- The model was built using the tensorflow library, and was trained on the training dataset to achieve a low Huber loss of 0.001, and a MAE loss of 0.0292.
- The model predicted the GHI, DHI, and DNI values for the next year quite accurately with Huber Loss of 0.0013 and MAE loss of 0.0312.

Further Improvements

- The model can be improved by using other statistical models like Autoregressive Integrated Moving Average (ARIMA) model or the Seasonal Autoregressive Integrated Moving Average (SARIMA) model.
- Further we can improve our data by removing stationarity, which would make the model work better on a general data.
- The model seems to overfit the data a little, this can be reduced by some techniques like Dropout regularisation, early stopping etc.

References

The following resources were used while studying deep neural networks and then building the model.

- Deep Learning Specialization by *Andrew Ng at coursera.org*
- *Understanding LSTM Networks by colah's blog*
- *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling by Junyoung Chung*
- *The Complete Guide to Time Series Analysis and Forecasting by Rian Dolphin*
- *Time Series Analysis and Forecasting by Marco Peixeiro*
- *MachineHack for providing the dataset*