Global Radiation Prediction Model

using Multi-variant Time Series Forecasting

Report submitted in fulfillment of the requirements
for the Undergraduate Project(6th Semester)

Third Year IDD

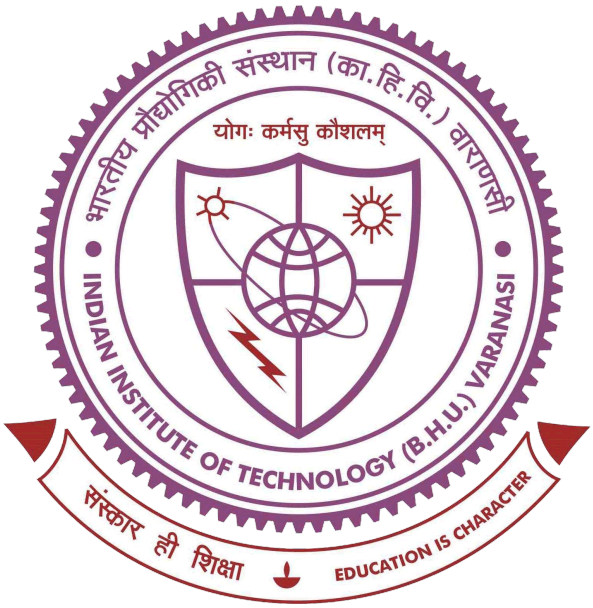
by

Paras Bajpai

20124033

Under the guidance of

Prof. S. K. Pandey



May 2023

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written material from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving the required details in the references.

Place: IIT(BHU) Varanasi
Date: 3rd May 2023

Paras Bajpai
IDD (Part III) Student
Department of Mathematical Sciences
Indian Institute of Technology (BHU)
Varanasi, INDIA 221005

Certificate

*This is to certify that the work contained in this report entitled "**Global radiation prediction model using multi-variant time series forecasting**", being submitted by **Paras Bajpai** (Roll no. **20124033**), carried out in the Department of Mathematical Sciences, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT(BHU) Varanasi
Date: 3rd May 2023

Prof. S. K. Pandey
Professor
Department of Mathematical Sciences
Indian Institute of Technology (BHU)
Varanasi, INDIA 221005

Acknowledgments

I would like to express my sincere gratitude to my supervising Professor, S.K. Pandey, for his constant guidance and support during the whole project work.

Place: IIT(BHU) Varanasi)

Date: 3rd May 2023 **Paras Bajpai**

Abstract

Humankind has been too reliant on fossil fuels for meeting its energy need. This has lead to adverse effects on both the environment and our lives as well. Nonetheless, we have started moving away from these fuels towards greener alternatives such as solar energy.

Through this model, I aim to optimse solar energy production by making predictions about future solar irradiance parameters like the Clearsky Diffused Horizontal Irradiance (DHI), Clearsky Diffused Normal Irradiance (DNI) and the Global Horizontal Irradiance (GHI). All of these parameters can then be used to efficiently locate solar power plants.

To to this, i make use of a dataset of all of these parameters for a period of 10 years. The techniques of multi variate time series forecasting are then applied on the dataset to generate and train a model which makes these predictions. The model makes use of Long Stort Term Memory (LSTM) Neural networks, and does the updation based on the Huber Loss and MAE loss.

Contents

1. Introduction

Over the years, fossil fuels have been humankind's main energy source. Let it be for powering our homes, industries or running automobiles, we have burned away fossil fuels at alarming rates. This has lead to deterioration of the environment and has had very detrimental effects on the various ecosystems surrounding us. Issues due to the ozone layer depletion, global warming, and air pollution have just started surfacing. Not only is this affecting animal and plant life, but has also caused alot of human loss.

Even though a little late, we are waking up to all of these disastrous effects of unclean energy, and are steadily shifting to renewable sources. Solar energy is one the bright spots in the world of renewables. If the earth's solar energy capabilities are used properly, we could easily move away from fossil fuels, and even reverse some of the already done damage to the climate. The huge potential in solar energy is evident from several studies. One of these have revealed that if we use the Sahara's solar capabilities efficiently, we can have enough energy to power up 12 earths!

This project aims to optimise solar energy productions. This is being done by making predictions about future values of the Clearsky Global Horizontal Irradiance (GHI), Clearsky Diffused Horizontal Irradiance (DHI), and the Clearsky Diffused Normal Irradiance (DNI) in a region over several intervals of time. All of these predictions can help us locate solar plants efficiently.

The above task is achieved by using Multi Variant Time series forecasting. The model takes several variables as inputs and based on that makes predictions about the above values. These are made using a particular architecture of RNNs known as LSTMs.

2. Neural Networks

Neural Networks are a subset of Machine Learning and are a very important component of Deep Learning Algorithms. Several times, neural networks draw analogy with the human brain, mimicking the way neurons in our brain pass signals to one and another. They are the soul of our Global Radiation Prediction Model.

2.1 What are 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 values are passsed through the network, until a final predicted value results. The calculations in any neuron are based on certain equations which will be elaborated on later.

A neural network usually consists of several neuron layers. What results are called Deep Neural Networks. These networks have been found to be really efficient in encoding dependencies between various variables and the making predictions with great accuracy. The various layers in a neural network can be divided into three parts:

- Input Layer: This layer simply accepts the inputs and passes it onto the next layers for further processing
- Hidden Layers: These layers can be multiple in number. They are the ones which are actually responsible for making the predictions in the model.
- Output Layer: These layer is the final layer of the model and it holds the output of the network

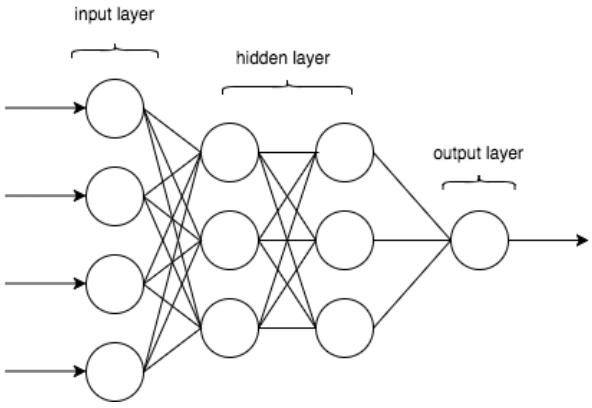


Fig 2.1.1 Different Layers in a neural Network

Each node has certain weights and thresholds assigned to it. If the threshold of a node is met, the connection to the next layer lights up and the data is transmitted further. We see that the neurons rely on training data to learn and improve their accuracy.

2.2 How do neural networks work?

We can think of each neuron as its own linear regression model, with input variables, weights, bias and an output variable. The neuron in general is as follows:

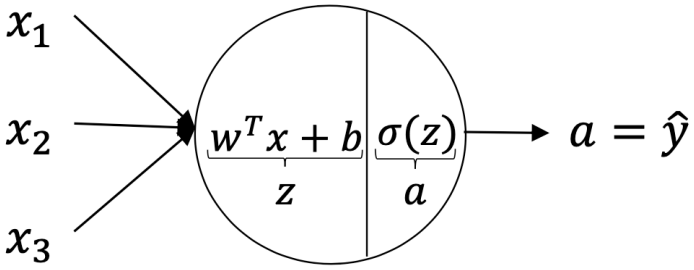


Fig 2.2.1 Equations in a neuron

The weight matrix *W* and the bias *b* are updated using optimisation algorithms, to give the best possible results. This generally works using a forward propagation and a backward propagation step. During the forward propagation step, the output of the neuron is calculated.

Once done we calculate the Loss Function. The loss function can be the Mean Squared Error, the Mean Absolute Error, etc. Upon this the backward propagation step takes, place where in the various parameters are updates based on different optimisation algorithms.

The Loss function used in the model is Huber Loss. It is a loss function which balances the effects of the Mean Squared Error and Mean Absolute Error loss functions. The Function is as follows:

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

This helps us generate a model which performs well in ordinary tests as well as in outliers. Hence we a very well balanced model.

3. Sequence Models

Sequence models are the machine learning models that can process sequences of data and make predictions based on that. Sequential data are datasets where time is the only independent variable, and of the variable depend on it in some way or the other. Sequential data includes text streams, audio clips, video clips, time-series data and etc. Recurrent Neural Networks (RNNs) is a popular algorithm used to train sequence models.

Sequence models have many applications such as:

- Speech recognition
- Sentiment classification
- Video activity recognition
- Time series forecasting

The sequence models can be implemented using a variety of algorithms:

- Recurrent Neural Networks (RNN)
- Gated Recurrent Unit (GRU)
- Long Short Term Memory (LSTM)

3.1 Recurrent Neural Networks (RNN)

Recurrent Neural Network (RNN) is a Deep learning algorithm and it is a type of Artificial Neural Network architecture that is specialized for processing sequential data. RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are mostly used in the field of Natural Language Processing (NLP) and time series forecasting.

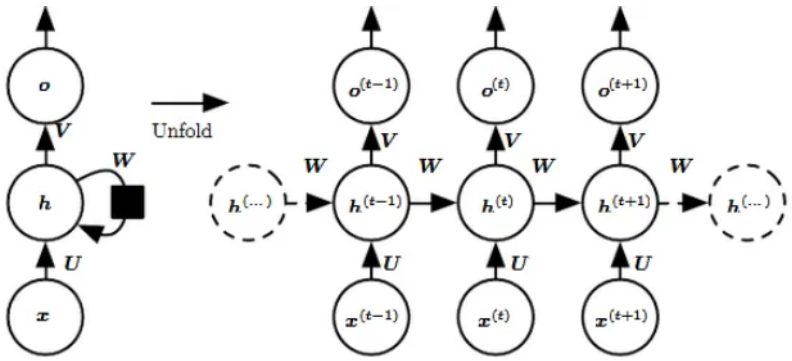


Fig 3.1.1: Unfolding of RNN architecture

The effectiveness of RNNs stems from the fact that they are based upon the principle of recurrence. This means that the output of a particular time step is send as an input for processing the next time step. This gives the RNN the ability to have a memory based on which it is able to identify time dependent relationships.

RNNs update their parameters using Backpropagation through time, which often results in exploding gradient, and vanishing gradient problems while forming long term dependencies. So RNNs are effective while process short term sequences.

3.2 Gated Recurrent Unit (GRU)

GRUs are improved version of standard recurrent neural network, which solve the vanishing gradient problem using methods known as the Update Gate and Reset Gate.

These gates are basically vectors which help us decide which information to remember and what not to remember while processing long tiem sequences.

- Update Gate: It helps the model decide how much of the previous information is to be passed to the current step
- Reset Gate: This vector decides what part of the previous data needs to be forgotten.

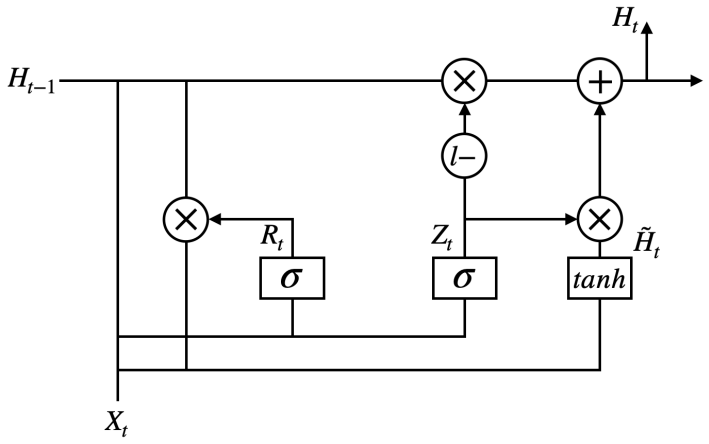


Fig 3.2.1 Basic structure of GRU Module

The above diagram represents a general structure of a GRU module. While the equations at play at the different steps are as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$u_t = \sigma(W_u x_t + U_u h_{t-1} + b_u)$$
$$c_t = tanh(W x_t + U(r_t \cdot h_{t-1}) + b)$$
$$h_t = u_t \cdot h_{t-1} + (1 - u_t) \cdot c_t$$

3.3 Long Short Term Memory (LSTM)

The Long Short Term Memory (LSTM) are another improvement of the RNNs which overcome the vanishing gradient problem. And they are very effective in doing so. LSTMs are often used to find long term dependencies in sequential data like stock prices, natural language, or sensor data.

In order to achieve this, LSTMs maintain a cell state which runs through the entire network. The modules have the power to modify, and to even forget details in the cell state. This is done using methods known as the Forget Gate, Input Gate and the Output Gate.

- Forget Gate: It decides what parts of the cell state are useless and need to be forgotten. This decision is made based on the current input and the previous output as is evident later.
- Input Gate: It decides what part of current input needs to be added to the cell state, that is what updations are to be made.

- Output Gate: It decides what information in the updated cell state is important and needs to be outputed.

Because of this the LSTMs are very good at learning long term dependencies.

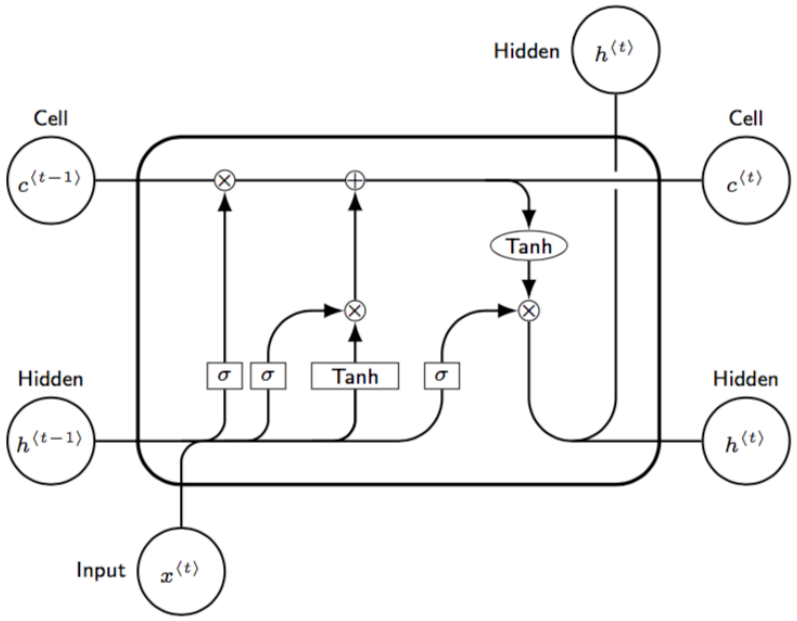


Fig 3.3.1 Structure of LSTM module

The equations involved are as follows:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \sigma_h(c_t)
\end{aligned}
$$

4. Time Series Forecasting

4.1 Time Series

A time series is simply a series of data points ordered in time. In a time series, time is often the independent variable and the goal is usually to make a forecast for the future. Time series analysis usually depends on the various characteristics of the time series.

4.2 Different characteristics of time series

- **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, variance and covariance do not change over time.

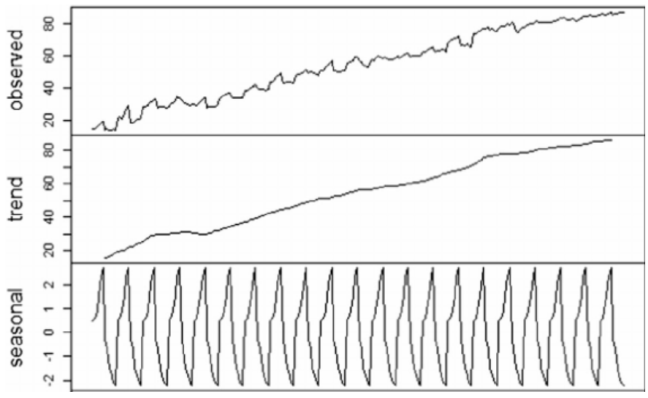


Fig 4.2.1 Examples of trend and seasonality in a given data

4.3 Modeling a Time Series

Modelling a time series refers to learning various parameters in order to make predictions. To achieve this several learning algorithms are applied which are as follows:

- **Moving Average**

  The moving average model is one of the very basic approaches to time series modelling. This model simply states that the next observation is simply the mean of the observations over a certain sliding window. We see that on increasing the window size the prediction curve becomes smoother.

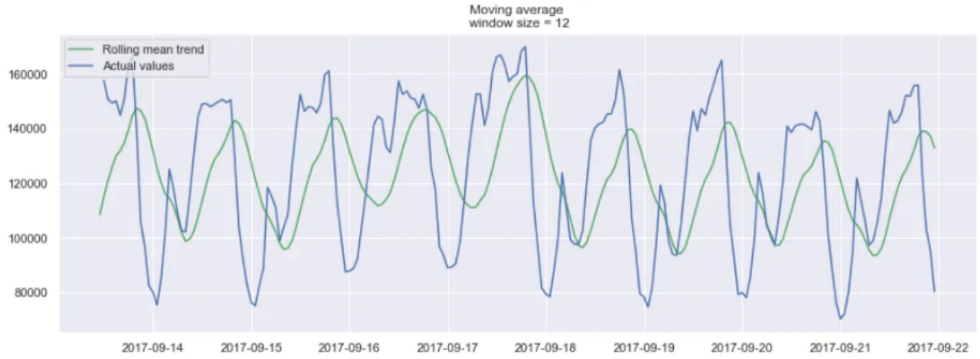


Fig 4.3.1 Moving Average applied to a time series with window length of 12 days.

- **Exponential smoothing**

  Exponential smoothing uses a similar logic to moving average, but here a decreasing weight is assigned to each observation, based on how old the observation gets.

  Mathematically, exponential smoothing is expressed as:

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

  Here, alpha is a smoothing factor that takes values between 0 and 1. It decides hwo fast the weights decrease. Lower the alpha smoother the model, because then the model tends to act like a moving average model, since old data is not forgotten.

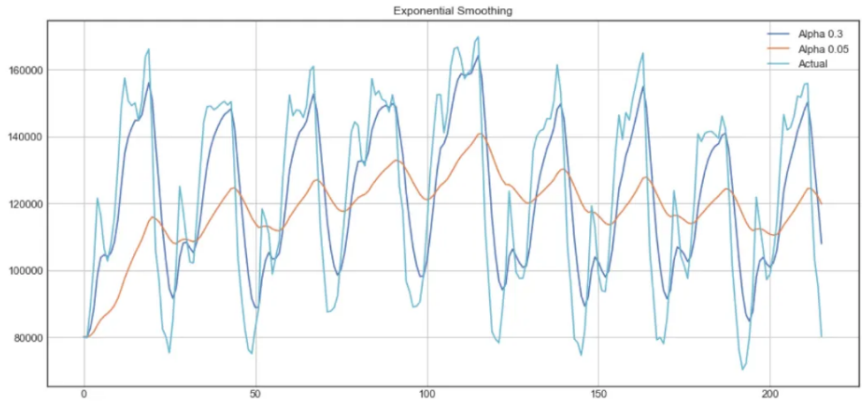


Fig 4.3.2 Exponential smoothing Applied with different Alphas

- Other training methods like double exponential smoothing, triple exponential smoothing, Seasonal autoregressive integraded moving average model (SARIMA), etc can also be applied.

4.4 Time Series Forcasting using LSTM

This model consists of several layers of LSTM and Dense neural networks. A dense layer is a layer that is deeply connected with its preceding layer which means the neurons of the layer are connected to every neuron of its preceding layer. The LSTM layer provides the main prediction power using its long term memory. A window size of 400 days has been used to make future predictions. Further, the training dataset has been divided into batches of size 512.The multiple epochs the model makes ensures proper tuning of each of the parameters. The batch size represents the number of samples processed before the model is updated. Whereas the number of epochs is the number of complete passes through the training set.

5. Global Radiation Prediction Model

Using the model we want to make predictions regardign values of various irradiance parameters over a period of 1 year of time.

5.1 Exploratory data analysis

We have a training dataset of consisting of the values of various Irradiance parameters at the interval of every 30 minutes for a period of 10 years. The various data parameters are as follows:

	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

Fig 5.1.1 The figure shows the various data parameters in the training dataset

In this data, the last columns representing the last 13 features, are those which are of importance. Out of these 13 features, the first three are the ones that we want to predict.

- Clearsky DHI
  Diffused Horizontal Irradiance represents the irradiance which does not arrive directly from the sun, but is scattered by clouds and the various particles in the sky, and comes from all directions.

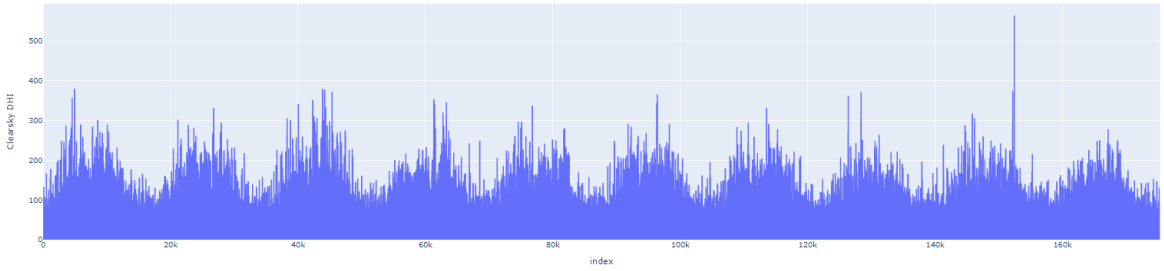


Fig 5.1.1 Represents DHI values over a period of 10 years

- Clearsky DNI

  Diffused Normal Irradiance represents the amount of irradiance which is coming perpendicular to the surface.This is the energy we receive in a straight line from the direction of the sun in its current position.

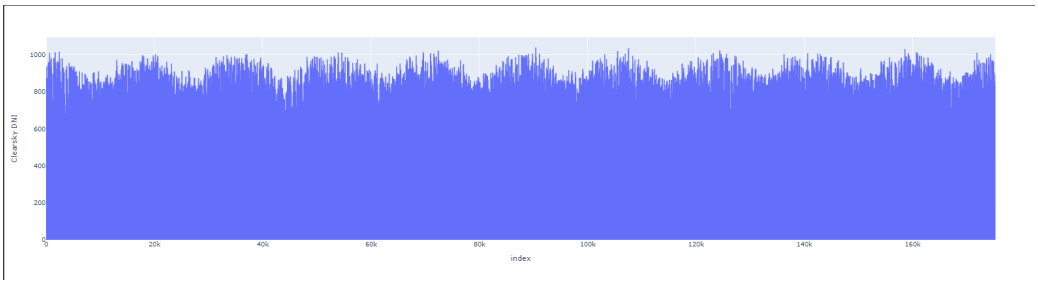


Fig 5.1.2 Represents DNI values over a period of 10 years

- Clearsky GHI

  Global Horizontal Irradiance is the total amount of shortwave radiation recieved from above which is horizontal to the ground. This is the most important factor in the calculation of the solar energy we receive from the sun.

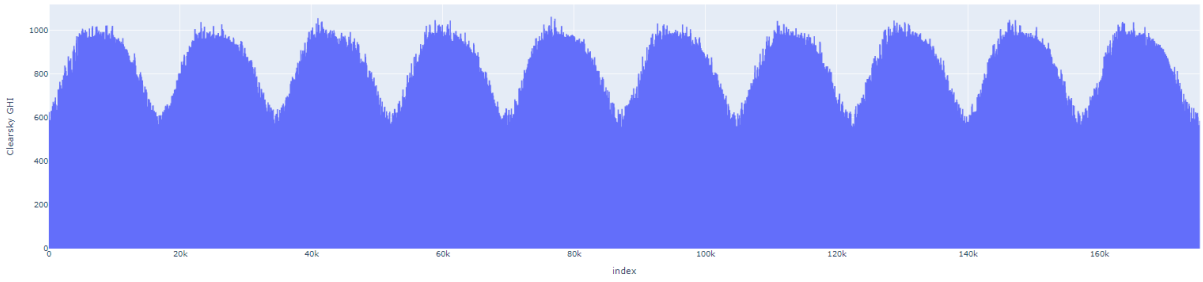


Fig. 5.1.3 Represents GHI values over a period of 10 years

5.2 Creating training data

Creating training data involves normalising the data, and converting into a vector that can be passed into the our model to train it.

- Initialising the parameters like window length and batch size

```
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 using MinMaxScaler() function

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

- Using TimeSeriesGeneraor() function to create the vector to be passed

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

5.3 Model Architecture and Training

The model consists of the following layers:

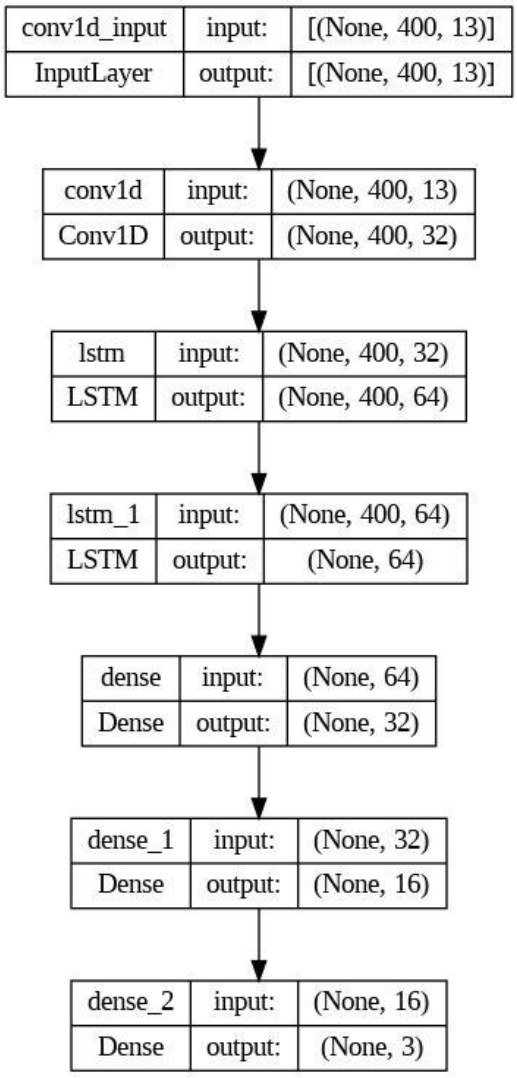


Fig 5.3.1 Layers in the Model

Having prepoccessed the dataset and defined the architecture of the model, we now train the model.

```
history = model.fit(train_generator, epochs = 100, shuffle = False)
# model training
```

Fig 5.3.2 Training the Model

We see that in the initial epochs the Huber Loss and MAE Loss are quite high.

```
Epoch 1/100
342/342 [==============================] - 38s 63ms/step - loss: 0.0442 - mae: 0.2308
Epoch 2/100
342/342 [==============================] - 22s 64ms/step - loss: 0.0352 - mae: 0.2233
Epoch 3/100
342/342 [==============================] - 22s 63ms/step - loss: 0.0290 - mae: 0.2021
Epoch 4/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0207 - mae: 0.1673
Epoch 5/100
342/342 [==============================] - 22s 66ms/step - loss: 0.0121 - mae: 0.1196
Epoch 6/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0076 - mae: 0.0897
Epoch 7/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0060 - mae: 0.0777
Epoch 8/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0053 - mae: 0.0718
Epoch 9/100
342/342 [==============================] - 23s 66ms/step - loss: 0.0048 - mae: 0.0678
Epoch 10/100
342/342 [==============================] - 22s 65ms/step - loss: 0.0045 - mae: 0.0649
```

Fig 5.3.3 Inital Epochs

The Huber Loss and the MAE loss both reduce as we go through various iterations training the model.

```
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
```

Fig 5.3.4 Final stages of the epochs

5.4 Results on training data

The results of the model are as follows:

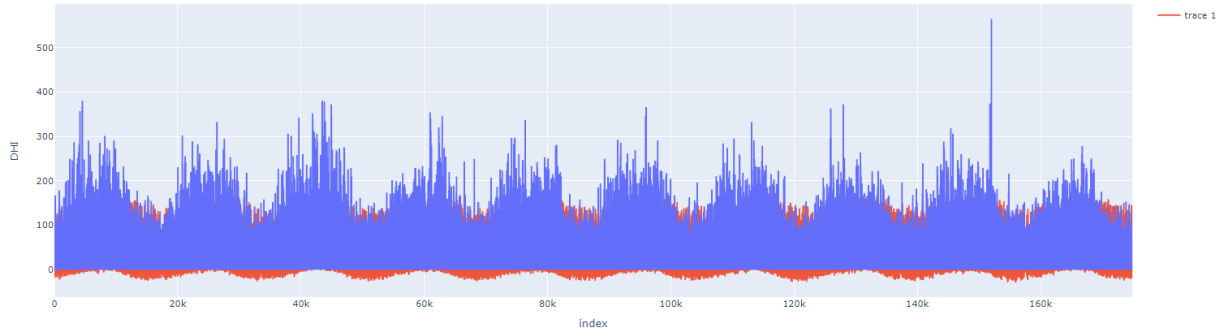


Fig 5.4.1 Predicted DHI values over a period of 10 years. Blue region are actual values, while the red are the predicted ones

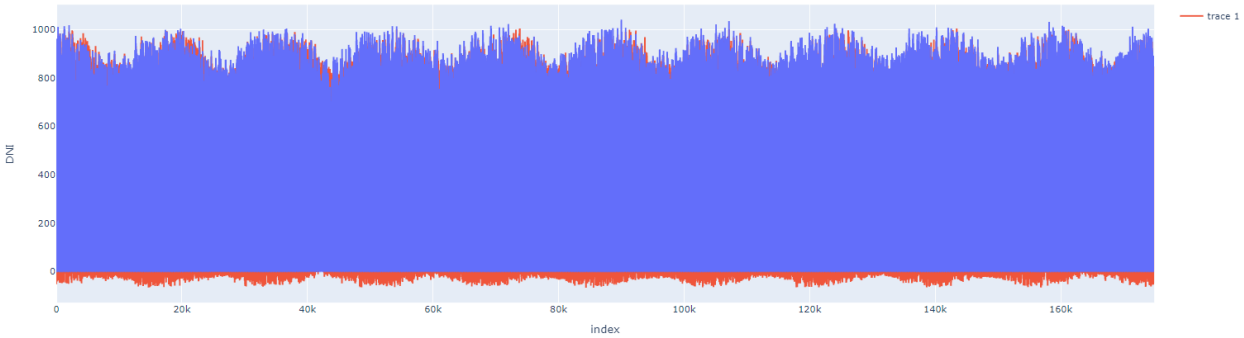


Fig 5.4.2 Predicted DNI values over a perios of 10 years.Blue region are actual values, while the red are the predicted ones

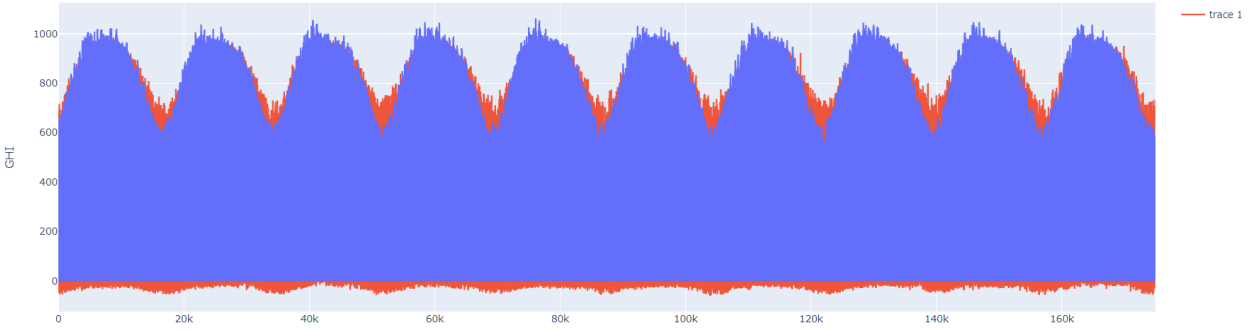


Fig 5.4.3 Predicted GHI values over a period of 10 years. Blue region are actual values, while the red are the predicted ones

5.5 Results on Test Data

The results on the test data are as follows:

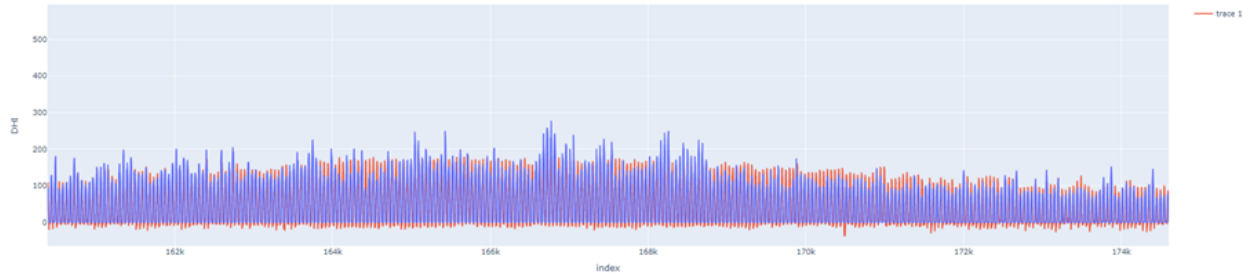


Fig 5.5.1 Predicted DHI values on the test set. Blue region are actual values, while the red are the predicted ones

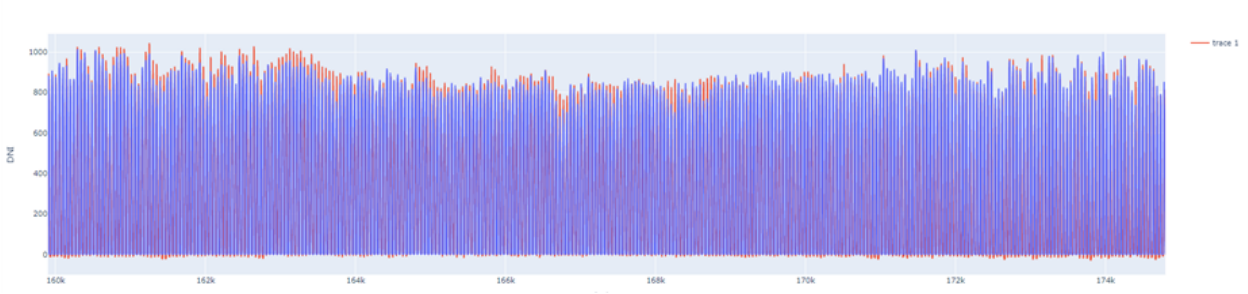


Fig 5.5.2 Predicted DNI values on the test set. Blue region are actual values, while the red are the predicted ones

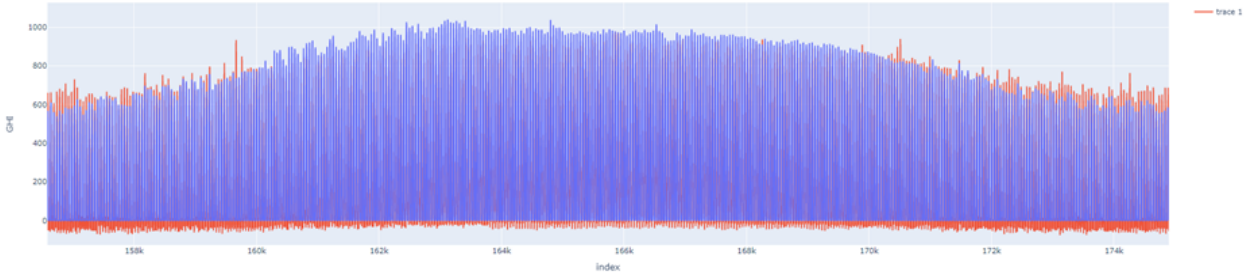


Fig 5.5.3 Predicted GHI values on the test set. Blue region are actual values, while the red are the predicted ones

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. In implementing the above models, the Tensorflow library was used. Then time series forecasting was applied on the same to get the desired results.

By training the model on the train data set, the model was able to achieve a low Huber Loss of 0.001 and MAE loss of 0.0292.

The model was able to make the predictions accurately. On the test data set, the Huber Loss had a value of 0.0013 and the MAE loss a value of 0.0312.

Scope for Improvement

The model can be improved by applying techniques like Regularisation, Early Stopping, using different optimisation techniques and different modelling algorithms.

By trying different Regularisation techniques like L2, Dropout, we can check whether reducing overfitting leads to better performance on the test dataset. Also by using different optimization techniques, and normalizing differently can help the model make better predictions.

Further using modelling algorithms like Auto Regressive Integrated Moving Average (ARIMA) model or the Seasonal Auto Regressive Integrated Moving Average (SARIMA) model can help improve the model.

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*