People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA – Boumerdès

Institute of Electrical and Electronic Engineering
Department of Power & Control

Project Report Presented in Partial Fulfilment of

the Requirements of the Degree of

'MASTER'

In Power Engineering

Title:

Deep Learning Based PV Power Forecasters in Python for Different Time Horizons

Presented By:

- **CHAKHCHOUKH Taha Yassine**
- **TEBBAL Said**

Supervisor:

Prof. **KHELDOUN Aissa**

Registration Number:……../2021

Abstract

The major points worked on throughout this report are: achieving accurate forecaster with less complexity and computational cost, using the minimum available data set for training and reaching the farthest possible span in the future. For the aim of developing forecasters in this work, then RNNs and DL were employed with the use of the python programming language for their modelling. A data set of GHI recordings collected during January 21, 2011, through March 4, 2012 and from December 20, 2012, through January 20, 2014 is used to compare the above DNN based models for three different time spans. Moreover, various evaluation metrics such as MAPE, RMSE, r and R^2 have been used for the assessment of the models to explore their performance when spanning different time horizons such that each one has a specific training samples. The obtained results have showed that the AE LSTM is the most efficient and less sensitive to the number of training samples.

Keywords – PV power forecasting, deep learning, optimization, neural networks, python, LSTM, GHI, prediction.

Dedication

I have a great pleasure to dedicate this modest work to my
beloved **Mother, Grand Mother, Brothers,**
Sisters, Grand Father and all my **family members**
whose affection, love, encouragement and prays
of day and night make me able to get
such success and honor.

Along with all my **friends**, **teachers** from primary school
to my last year of university.

And to all with whom I spent wonderful moments.

Taha Yassine Chakhchoukh

Dedication

I have a great pleasure to dedicate this modest work to my beloved **Parents** and my **family members** whose affection, love, encouragement and prays of day and night make me able to get such success and honor.

Along with all my **friends**, **teachers** from primary school to my last year of university.

And to all with whom I spent wonderful moments.

Said Tebbal

Acknowledgement

In the name of Allah, the Most Gracious and the Most Merciful Alhamdulillah, all praises to Allah for the strengths and His blessing in completing this project.

We would like to express our deepest and sincere gratitude to our project supervisor Pr. KHELDOUN.A It was a great privilege and honor to work and study under your supervision, Thank you very much.

Last but not least, we are infinitely grateful to our family members, particularly our parents for their patience, unwavering support, continuous encouragement, and belief in us throughout our whole life. We would have never made it this far without them beside us every step of the way.

Finally, a special thanks go to all IGEE members.

Institute of Electrical and Electronic Engineering
Boumerdes, July 2021

Taha Yassine Chakhchoukh

Said Tebbal

Contents

List of Tables

List of Figures

List of abbreviations

AE Auto Encoder

API Application Programming Interface

CNN Convolutional Neural Network

ConvNets Convolutional Neural Networks

CSV Comma Seperated Variables

DL Deep Learning

DNN Deep Neural Network

GHI Global Horizontal Irradiance

GRU Gated Recurrent Unit

GW Gega Watt

IDE Integrated Development Environment

KNN K-Nearest Neighbors

LSTM Long Short Term Memory

MAPE Mean Absolute Percentage Error

ML Machine Learning

MLP Multi Layer Perceptron

MW Mega Watt

NN Neural Network

NREL	National Renewable Energy Laboratory
OO	Object Oriented
PV	Photo Voltaic
PVPPs	Photo Voltaic Power Plants
QA	Quality Assessment
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
RMSprop	Root Mean Squared Propagation
RNN	Recurrent Neural Network
Seq2Seq	Sequence to Sequence

Chapter 1

Introduction

In the recent years the world has seen a remarkable increase in load demand compared to the generated power due to several factors: industrialization, modernization and population growth, aside from people's increasing living standard, this has lead to the emergence of the first problem which is load satisfaction for better and stable operation of the grid, however the generated power around the world is mainly produced by thermal power plants which also has raised the problem of global warming due to their huge negative impact on the environment caused by the release of toxic gases. The previous stated problems obliged the scientist to find some other sources of energy that help to satisfy the load and they must be clean sources or environmental friendly alternatives. Along with the different alternatives, the targeted ones were renewable energy sources, among these sources, wind and solar energy are the most acceptable and promising sources due to their potential and availability. However a research study highlighted that the earth receives at an instant 1.8×10^{11} MW from solar radiation, therefore solar gained a higher attention from governments and international organizations, because of its several benefits including environmental and economic advantages.

Solar energy systems or specifically PV systems has achieved world wide acceptance, hence PVPPs have seen an enormous growth over the last few years also some statistics stated that by 2030 the global installation of solar PV power capacity could exceed 1700 GW.

Nevertheless solar energy has some drawbacks due to its intermittent nature, fluctuations, spikes (this is because it is highly dependent on weather parameters) and

its unpredictable output generated power. So in order to integrate PVPPs into the grid and keeping its operation stable, then a reliable and robust forecasters must be built in order to achieve a reliable and economical operation of the overall power system.
A considerable number of experiments has been conducted to develop appropriate forecasting models in forecasting PV power generation with the targets of higher accuracy and minimum complexity with computational cost. These forecasting models are broadly classified into two categories: indirect and direct forecasting models. Other classification for PV power forecasters come in terms of forecast horizon (short, medium and long term forecast horizons), and historical data (persistence method, statistical approaches, ML and DL approaches and hybrid techniques).

1.1 Scope and research questions

The general purpose of this study is to work on forecasters for three different time horizons based on different deep learning methods in order to predict the solar irradiance (here the solar irradiance is specified instead of the output of the PV because the indirect forecasting method was employed for this work). The training and testing of the multiple models were done using the data provided from NREL that were acquired from three different locations. Before going further on this work few particular questions related to the topic will be highlighted in order to be worked on all over this report and deduce their answers at the end of this proposal. Those questions are cited below:

- How DL models behave when the provided training data is large and or shrunk (resampled with different frequencies).
- What is the result of the investigation of DL NNs for PV power forecasting.
- How does the predicted outcome changes when hybridizing different NNs.
- Is the solar irradiance predictable.
- How to achieve tuning the proper hyperparameters to prevent overfitting and falling into local minima.

1.2 Aims and objectives

This study focuses on developing forecasters to help integrating the solar energy sources to the grid for minimizing the risks of global warming and having a reliable operation of the grid that satisfies the load demand.

1.3 Report structure

The overall layout of this report can be decomposed into six key chapters where the first one covers with details different RNNs and CNNs as well as it explains thoroughly NNs hyperparameters in general. The subsequent chapter focuses on clarifying the structure of the data used for the accomplishment of this project in terms of when and where the data was acquired, and how it was measured and recorded. The following section deals with the methods used for this project and at the end it mentions the most notable and recent works related to the topic, thereafter the software tools exploited for realizing the proposal of this work will be introduced. Last but not least the implementation of different models for separate time horizons is well summarized in the last chapter and the conclusion comes afterwards.

Chapter 2

Generalities

Before the invention of AI technology, systems were used to be programmed to perform certain tasks following some specifications, then in the early 1950's scientists conducted experiments trying to find some solutions that imitate human thoughts. 1956 was the first year where the term AI was first appeared, in this year the first conference on AI has been organized by John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon's at Dartmouth College in New Hampshire. One ideal representation of the term "AI" is the simulation of human intelligence on machines. Through the years from the invention of AI, AI has seen several applications especially for resolving complex problems[1].

2.1 Artificial Neural Networks

A neural network is a general mathematical computing paradigm that models the operations of biological neural systems. A neural network is a set of individually interconnected processing units called neurons. Information is passed through these neurons along interconnections. Each neuron connection is associated with two values which are: the input value and its synaptic weight. The output of this neuron will be a function of the net value at the input. artificial neural networks while implemented on computers are not programmed to perform specific tasks, instead they are trained to learn patterns from its input data. Once they are trained new patterns can be represented to them for classification or prediction. Since artificial neural networks are considered as a simplified mathematical model inspired from the biological system so they must be taught or trained. During the learning phase, sample of training data are presented to them to extract patterns at its input so they will be prepared to automatically recognize new input data.

Neural networks can be classified into dynamic and static. In a static neural network the output is calculated directly from the input through feed forward connections. For example, in a basic feed forward neural network, the information flows in a single direction from input to output, such neural networks have no feedback elements. Whereas in a dynamic neural network, in addition to the current input the output depends also on previous inputs, outputs and/or hidden states of the network. Those neural networks are going to be explained furthermore in this chapter, where this chapter is organised as follows, in the first section the simple recurrent neural network is presented, section two talks about LSTM, after that section three presents GRU, then the fourth section describes CNNs, finally the neural network hyperparameters will be the last part of this chapter[3].

2.2 Recurrent Neural Networks (RNNs)

Simple RNNs are one type of dynamic NNs, they are designed for sequential data that are basically just ordered data in which related things follow each other[4]. The main aspect that distinguishes RNNs from standard feed forward MLP NNs is that it has at least one feedback loop. The recurrent or feedback connections add state or memory to the network and allow it to learn broader abstractions from the input sequences, means that in a given layer, each neuron may pass its signal latterly in addition to the forward flow of information towards the next layer, also the output of the network may feedback as an input to the network besides the input vector and so on[5]. RNN can be illustrated in figure 2.1:

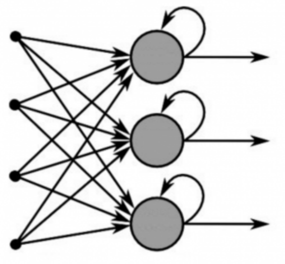


Figure 2.1: Recurrent Neural Network

Based on the use case different RNNs can be used and they are listed below[5]:

- **One-to-Many:** sequence output, for image captioning.
- **Many-to-One:** sequence input, for sentiment classification.
- **Many-to-Many:** sequence in and out, for machine translation.

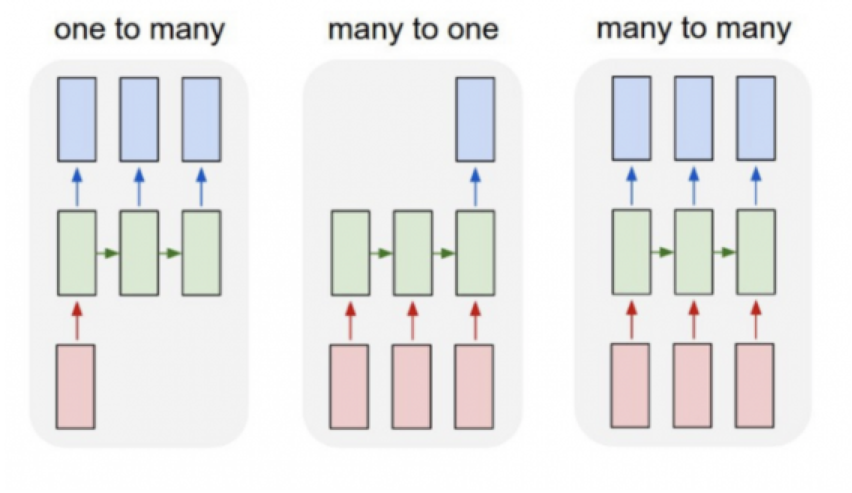


Figure 2.2: Types of RNNs

This type of NN in practice is quite limited due to two major issues that are[4]:

- **Exploding gradients:** They are when the algorithm, without much reason, assigns an unjustified high importance to the weights.
- **Vanishing gradients:** They occur when the values of a gradient are too small and the model stops learning or takes way too long as a result.

The first issue can be easily solved by truncating or squashing the gradients. Whereas the latter issue was solved by introducing the LSTM NN by Sepp Hochreiter and Juergen Schmidhuber[4], which will be detailed in the upcoming section of this chapter.

2.3 Long Short Term Memory(LSTM)

LSTM are an extension of a simple RNN where instead of a neuron it contains a memory block that are connected into layers. A block has components that make it smarter than a classical neuron and a memory for recent sequences[5]. Therefore it is well suited to learn from important experiences that have very long time lags in between[4].

A unit operates upon an input sequence and each gate within a unit uses the sigmoid activation function to control whether they are triggered or not, making the change of state and addition of information flowing through the unit conditional.[5]. Each memory unit contains three types of gates:

- **Forget Gate:** Decides what information should be omitted from the cell in that particular time step.
- **Input Gate:** Decides which values from the input to update the memory state.
- **Output Gate:** Decides what to output based on input and the memory of the cell.

The LSTM structure with its three gates is illustrated in figure 2.3.

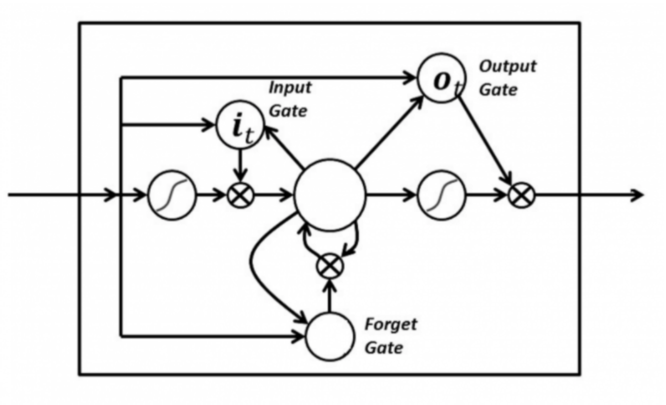


Figure 2.3: LSTM memory block structure

2.3.1 Vanilla LSTM

Vanilla LSTM is the simplest LSTM structure. It is the LSTM architecture defined in the original 1997 LSTM paper and the architecture that will give good results on most small sequence prediction problems. The Vanilla LSTM is composed from[6]:

1. Input Layer.

2. Fully connected LSTM hidden layer.

3. Fully connected output layer.

2.3.2 Stacked LSTM

In contrast to Vanilla LSTM, stacked LSTM has multiple hidden LSTM layers where each layer contains multiple memory cells or blocks[6]. Stacked LSTMs or Deep LSTMs were introduced by Graves, et al. in their application of LSTMs to speech recognition, beating a benchmark on a challenging standard problem[6].

2.3.3 The Encoder-Decoder LSTM

The Encoder-Decoder LSTM structure is comprised of two parts: one for reading the input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting the predicted sequence. This architecture is designed specifically for sequence to sequence Seq2Seq prediction problems[6].

2.4 Gated Recurrent Unit (GRU)

GRU can be considered as a simpler version of LSTM, as it contains two gates[7]:

- **Reset gate:** The Reset Gate is responsible for the short-term memory of the network.
- **Update gate:** Similarly to the previous gate, this gate is responsible for long term memory.

GRU in some cases, it has certain advantages over LSTM. GRU uses less memory and is faster than LSTM, however, LSTM is more accurate when using datasets with longer sequences[8].
The GRU was introduced in 2014 by Kyunghyun Cho et al[7].

2.5 Convolutional Neural Networks (ConvNets/CNNs)

Convolutional Neural Networks are a type of neural network that was designed to efficiently handle image data. They have proven effective on challenging computer

vision problems both achieving state-of-the-art results on tasks like image classification and providing a component in hybrid models for entirely new problems such as object localization, image captioning and more[9]. CNNs are considered as a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.
A ConvNet is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.
The ConvNet is built by three types of layers, which are[9]:

1. Convolutional layers.
2. Pooling layers.
3. Fully connected layers.

2.5.1 Convolutional layers

Convolutional layers are comprised of filters and feature maps.

Filters

The filters are essentially the neurons of the layer. They have both weighted inputs and generate an output value like a neuron. The input size is a fixed square called a patch or a receptive field. If the convolutional layer is an input layer, then the input patch will be pixel values. If they deeper in the network architecture, then the convolutional layer will take input from a feature map from the previous layer[9].

Feature maps

The feature map is the output of one filter applied to the previous layer. A given filter is drawn across the entire previous layer, moved one stride at a time. Each position results in an activation of the neuron and the output is collected in the feature map[9].
The distance that filter is moved across the input from the previous layer each activation is referred to as the stride[9].

2.5.2 Pooling layers

The pooling layers down-sample the previous layers feature map. Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map. As such, pooling may be considered as a technique to compress or generalize feature representations and generally reduce the overfitting of the training data by the model[9]. They too have a receptive field, often much smaller than the convolutional layer. Also, the stride or number of inputs that the receptive field is moved for each activation is often equal to the size of the receptive field to avoid any overlap. Its feature maps are created in a very simple manner where it takes only the average or the maximum of the input value, so based on that we can distinguish two types of pooling layers which are[9]:

- Maximum pooling
- Average Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power[10].

2.5.3 Fully connected layer

Fully connected layers are the normal flat feedforward neural network layer. These layers may have a nonlinear activation function or a softmax activation in order to output probabilities of class predictions. Fully connected layers are used at the end of the network after feature extraction and consolidation has been performed by the convolutional and pooling layers. They are used to create final nonlinear combinations of features and for making predictions by the network[9].
As can be seen from the previous description the CNN is being used mainly for the fields of: image processing, object detection, image recognition ..., however the ConvNets ability to automatically extract features from raw input data can be applied to time series forecasting problems, where a sequence of observations can be treated like a one-dimensional image that a CNN model can read and distill into the most salient elements[9].
The illustration of a simple ConvNet is shown figure 2.4.

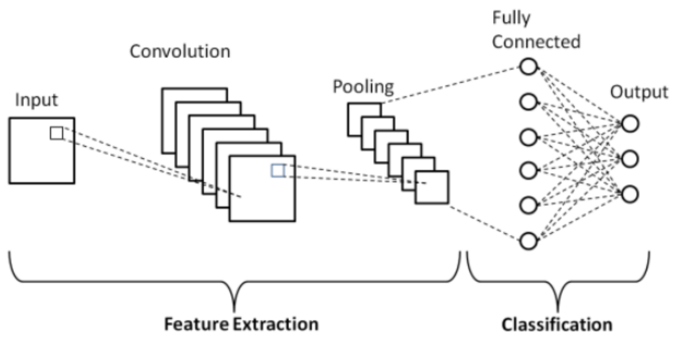


Figure 2.4: Schematic diagram of a basic CNN

2.6 Hyperparameters

Neural Networks (NNs) are the typical algorithms used in Deep Learning analysis. NNs can differ in structures, nevertheless their core elements or building blocks are common, but the things that differentiate each NN from another are its hyperparameters, which determine how the network is built and trained[11]. The hyperparameters are set prior to training. There is also another concept that must be taken into account, which is the NN parameters. Those parameters are the internal coefficients of the model, and they are chosen by the model itself. It means that the algorithm, while learning, optimizes these coefficients (according to a given optimization strategy) and returns an array of parameters which minimize the error[12]. This section will cover the tunable parameters (hyperparameters), the hyperparameters fall mainly into two categories:

- Optimizer hyperparameters
- Model specific hyperparameters

2.6.1 Optimizer hyperparameters

Batch size

When dealing with large datasets it is more convenient to split the overall dataset into sub samples that are given to the network when each parameter update happens, those sub samples are called batches and their size is defined as the batch size[11].

Number of epochs

It is the number of times the whole training data is introduced to the model.

Optimizers

While the training data is being passed to the network for training, there is a metric that indicates how the model performs on the training data, this metric is called the loss. Essentially the minimum the loss is, the better it is for the model, so this process of minimizing any mathematical expression is called optimization. Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. They are used to solve optimization problems by minimizing the function[13].

2.6.2 Model specific hyperparameters

Hidden layers

In general the NNs share two common layers that are: the input and output layer, in between there may be some additional layers that are called hidden layers.

The hidden layers are added until the model arrives at an acceptable performance on the test data[11].

Hidden units

Are the processing units of a NN. Information is passed through these units along interconnections.

Activation functions

It is the transfer function of the hidden units that relates their inputs to their outputs through a linear or non linear transformation.
The hyperparameters in general are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained. A good choice of the hyperparameters can really improve the model performance in generally[14].

The challenge with hyperparameters is that there is no optimal number that works everywhere. The best numbers depend on each task and each dataset.

On this report, the only tuned hyperparameters are the previously described ones, and their optimal values have been reached through the use of trial and error method, which is the most widely adopted by researchers and students, and it is 100 % manual technique.

Chapter 3

Data

The data that has been used for the completion of this project was provided by the NREL (National Renewable Energy Laboratory), which is a national laboratory of the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Operated by the Alliance for Sustainable Energy, LLC. The manual for this data is published by the NREL in their website[15]. The manual describes the performance of the data measured for flat-plate photovoltaic (PV) modules installed in Cocoa, Florida; Eugene, Oregon; and Golden, Colorado. The data include PV module current-voltage curves and associated meteorological data for approximately one-year periods.

3.1 Data description

The data measurement locations were Cocoa, Florida (subtropical climate); Eugene, Oregon (marine west coast climate); and Golden, Colorado (semi-arid climate). The data includes the following periods:

- **Cocoa:** January 21, 2011, through March 4, 2012.
- **Golden:** August 14, 2012, through September 24, 2013.
- **Eugene:** December 20, 2012, through January 20, 2014.

3.1.1 PV technologies

The PV modules tested were for PV technologies available in 2010, when the work effort began. They include:

- Single-crystalline silicon (x-Si) PV modules.
- Multi-crystalline silicon (m-Si) PV modules.
- Cadmium telluride (CdTe) PV modules.
- Copper indium gallium selenide (CIGS) PV modules.
- Amorphous silicon (a-Si) tandem and triple junction PV modules.
- Amorphous silicon/crystalline silicon or heterojunction with intrinsic thin-layer (HIT) PV modules.
- Amorphous silicon/microcrystalline silicon PV modules.

Even though the market share for a-Si tandem and triple PV modules has decreased dramatically since this work began, their large sensitivity to the solar spectrum makes their data useful for validating the robustness of models that account for the effects of variations in the solar spectrum on PV module performance. Because spectral effects directly impact the PV module short-circuit current, the use of I-V curve data is particularly well suited for evaluating models of this type.

3.1.2 File convention

The files contain Comma Separated Variables (CSV). The naming convention uses the deployment location and NREL PV module identifier as the file prefix, with the characters “csv” as the file extension.

3.1.3 File format

The data files consist of rows or lines of data. The data values within a line are separated by commas, which constitutes the CSV format. The CSV format is commonly used, and most software has built-in functions for reading or parsing it. When parsed, the line of data is broken into fields containing the values of the data elements.

File header

The first two lines of data provide information about the PV module and the site location. The first line consists of nine fields containing text describing the header data values that are contained in line 2, the field elements and their description are listed in table A.1 in appendix A.

File data

Line 3 consists of 42 fields and contains text describing the I-V curve and meteorological data contained in line 4 and subsequent lines, similarly the field elements and their corresponding description are tabulated in table A.2 in appendix A.

3.2 Data collection

Measurement equipment was selected to provide low measurement errors, station operations were followed to ensure equipment operated properly, and QA methods were implemented to exclude data not meeting quality thresholds.

3.2.1 Equipment

NREL provided the equipment for measurements at the Cocoa and Eugene sites and the instrument for each weather parameter are specified in table A.3 in appendix A. The equipment was transported to the sites in a shipping container, and then the shipping container was used as an integral part of the test facility. Structure was attached to the shipping container for deploying the PV modules, and the roof was used for locating the solar radiation and other meteorological sensors. Data acquisition equipment was located inside the shipping container in a temperature-controlled environment.

Through this chapter, a comprehensive knowledge of the data used to build the model for this report. Similarly, the following chapter covers the building blocks of the model and the works accomplished in this field.

Chapter 4

Methodology

In this chapter, the workflow of the model is explained with details, where the first section deals with how the data is prepared for modelling, afterwards the models that has been built throughout this report are highlighted in the second section, finally the research findings of previous works that has been done in this field will be pointed out in the last section of this chapter.

4.1 Data preparation

This step of data preparation is a crucial step in real world ML projects and unfortunately some of the ML developers do not pay enough attention to this part. This step is critical because it can have a huge impact on the model's performance, where sometimes ML models do not give an acceptable results on real worlds situations, this is not necessary due to their structure, nevertheless it may be as a consequence of the poorly provided raw data. At the expense of the importance of this step it is a considerably difficult step. The reason is that each dataset is different and highly dependent to the project. Though there are sufficient commonalities across predictive modelling project.

Data preparation is concerned with transforming the raw data that was collected into a form that it suitable for modeling. This part in ML projects generally passes through the following steps[16]:

- **Data cleaning:** Identifying and correcting mistakes or errors in the data.
- **Feature selection:** Identifying those input variables that are most relevant

to the task.

- **Feature engineering:** Deriving new variables from available data.
- **Dimensionality reduction:** Creating compact projections of the data.

The data that has been used to develop this work has only went through data cleaning step, because there is no need for feature engineering or dimensionality reduction since it is already numeric and selected with the least number of input attributes.

The data cleaning process done on this project is summarized in the flowchart below:

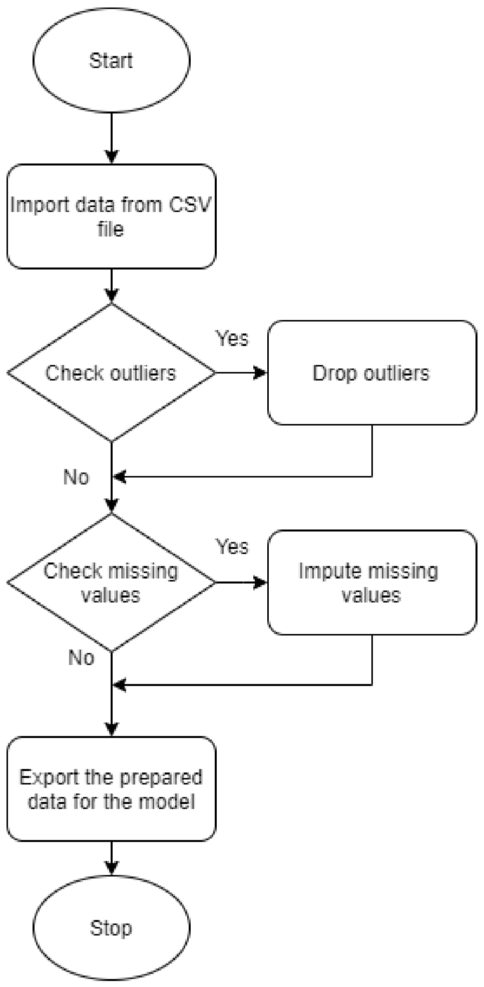


Figure 4.1: Data preparation flowchart

From the above flowchart, the two techniques that have been used while cleaning the data are:

1. **Drop outliers:** The first thing to look at is what is an outlier, an outlier is an observation that is distinct from other observations and it may be due to: measurement error, data corruption or true observation[16]. Below the flowchart graphically explains how the outliers are interpreted:

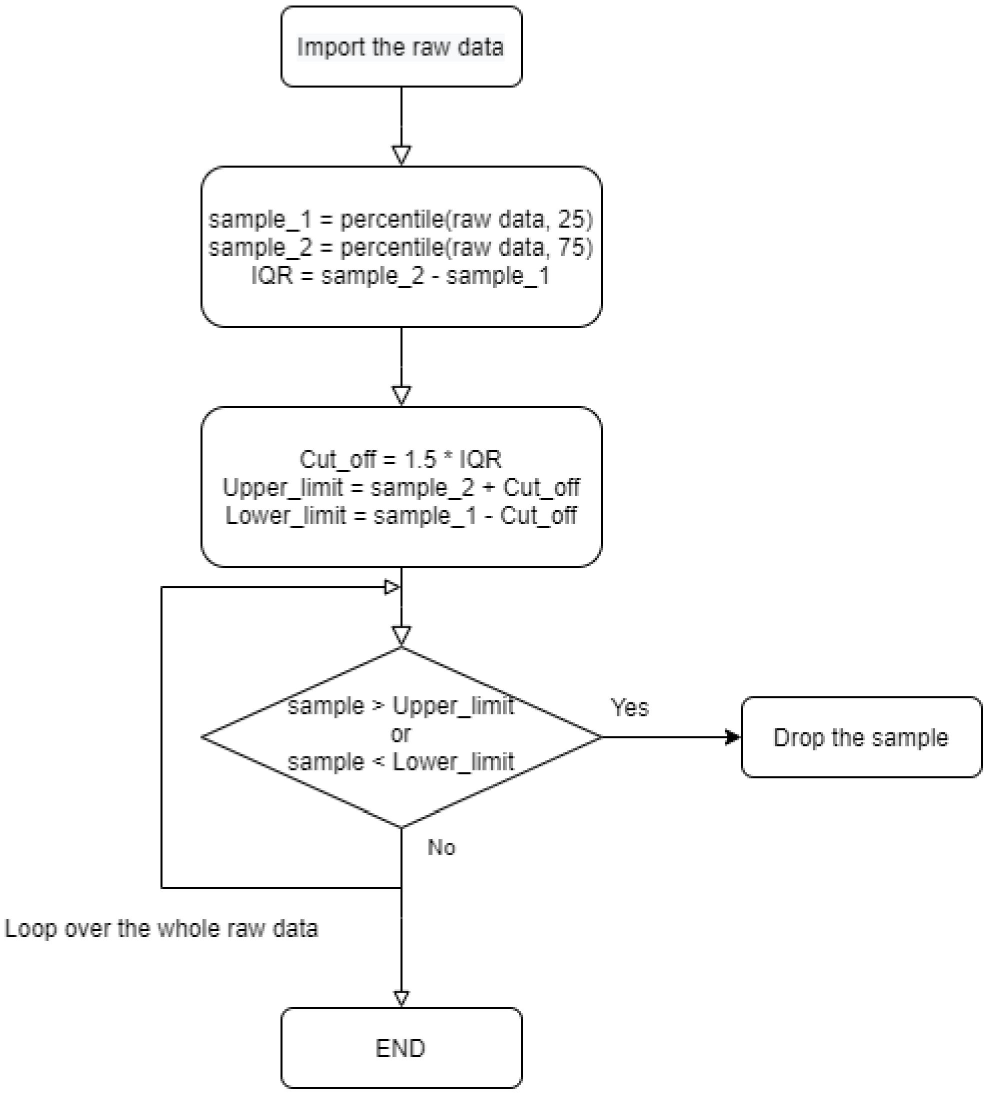

Import the raw data
sample_1 = percentile(raw data, 25)
sample_2 = percentile(raw data, 75)
IQR = sample_2 - sample_1
Cut_off = 1.5 * IQR
Upper_limit = sample_2 + Cut_off
Lower_limit = sample_1 - Cut_off
sample > Upper_limit
or
sample < Lower_limit
Yes
Drop the sample
No
Loop over the whole raw data
END

Figure 4.2: Check and drop outliers flowchart

2. **Impute missing values:** In practice the data often have missing values due to several factors such as: maintenance of measurement equipment, data corruption . . . etc. So missing data must be handled in order to be supported by ML algorithms. Once the missing values are marked or identified then

two options are given in hand to deal with those samples either by removing or imputing them. The data cleaning algorithm built in this project handles missing values by replacing them, for this purpose there exist several methods, however in this project "KNNImputer()" built in function in python has been used. Below the graphical representation summarizes the predescribed part.

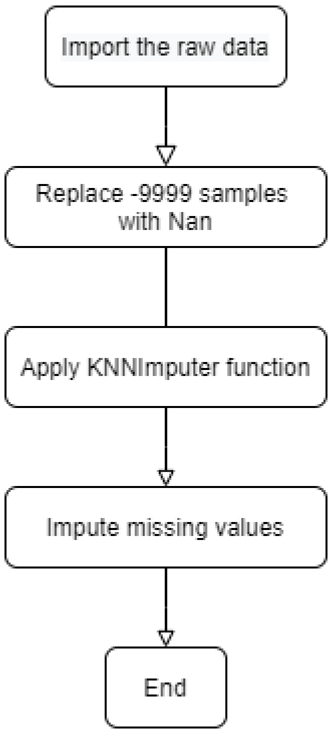


Figure 4.3: Impute missing values flowchart

4.2 Models description and building

4.2.1 Models description

Throughout chapter two, the common DNN has been represented in general in terms of structure, method of operation and their different configurations. So now they are going to be explained on how they have been used to build the models. Firstly, the simplest model that has been implemented is the standalone model such that one of the cited NN above in this literature has been used alone to construct the model and make predictions, to dig deeper into more complex models or in other words constructing hybrid models, then the use of two or more NN has been applied.

Below is a list of the models that has taken place throughout this report (standalone and hybrid):

- RNN
- LSTM (Vanilla, Stacked, Auto-Encoder)
- GRU
- CNN$_{1D}$
- LSTM-RNN
- LSTM-GRU
- GRU-RNN
- LSTM-GRU-RNN
- CNN$_{1D}$-LSTM
- CNN$_{1D}$-LSTM-GRU-RNN

4.2.2 Model building

To explain well how the models were built then, their details will be pointed out from the innermost to the outermost model parameter in the next few lines.

Data preparation

Though this section has the same name as the chapter mentioned earlier, but practically their usage is different, in which the first one deals with data cleaning, in other words transforming the raw data into a cleaner version. However this second part, talks about transforming the latter version of data (clean data) into a format that will be suitable for modelling (training and testing). Transforming the raw data into a relevant format for modelling is the heart of this part.
In general the architecture of DNNs impose to the raw data to have a 3D shape, where the first dimension summarizes the number of samples provided to the model,

whereas the second dimension defines how many time steps in each sample used to make predictions, finally the last one provides how many features employed in the model[9]. Often the shape of the 3D data is summarized using the array notation shown below:
$[samples, timesteps, features]$
The second part of this section talks on how to convert the time series data into a supervised data, meaning that converting the raw data into samples with 3D shape as described previously, where each sample contains an input and an output component. The aim of this conversion is to feed samples of input and output component each to the model, in order for it to capture the patterns of the outputs from the provided inputs, this is done while training the model. On the other hand for the model to make predictions on new unseen data, the second part (output component) is omitted and only the input sequence is taken into consideration, then the forecasted output is compared with the actual output.

Architecture

Up to this point, the data is prepared for all the models stated in the preceding section, and those models have commonalities and differentiation between them. The commonalities between one model and another are:

- The 3D shape of the input.
- The output that must be predicted.

The differentiation between one model from another is the NN being used, either the standalone or the hybrid network.

The time horizon

Till now the input data received by the model and the NN constructing it has gone through a deep explanation. Afterwards, the output shape for the predictions made by the network must be identified in terms of what span of time needed to be forecasted and how many future samples must be predicted. On this report, the point that is taken into consideration is that the output is predicted for only one sample ahead.

The graphical description of the whole process is shown underneath:

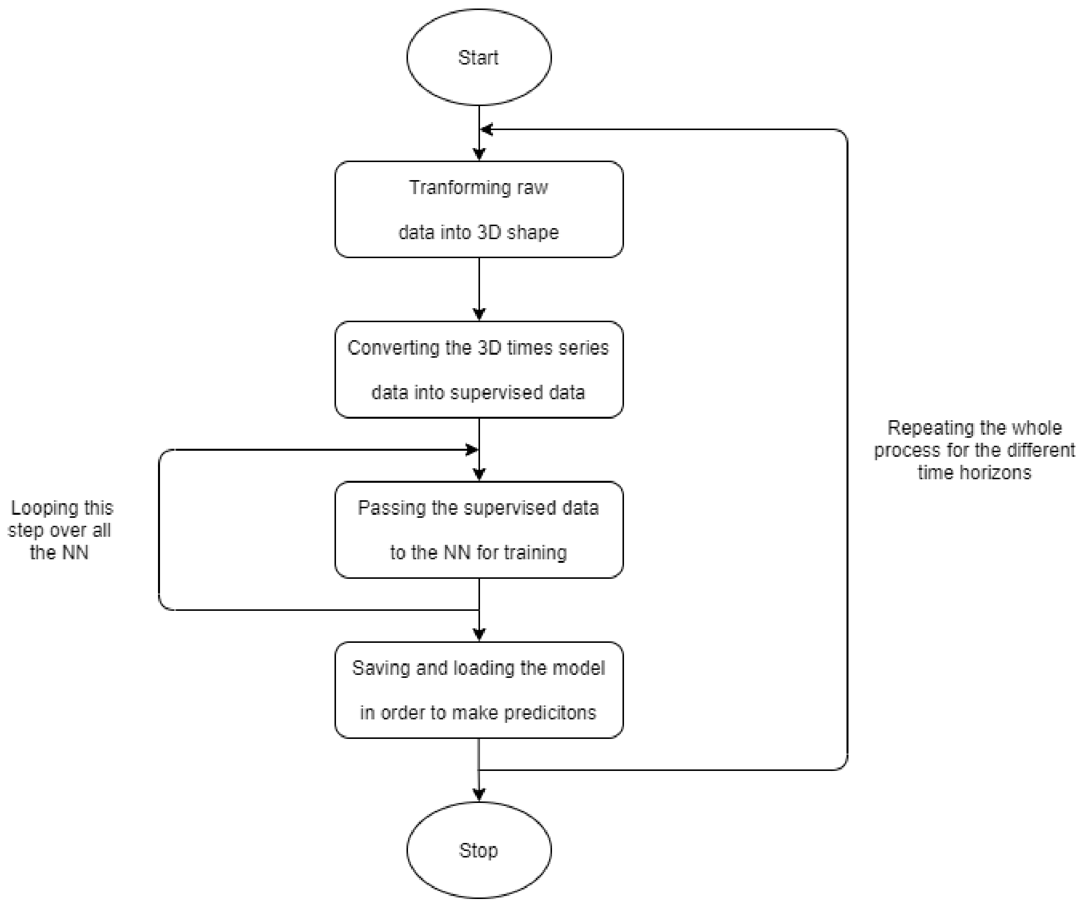

Start
Tranforming raw data into 3D shape
Converting the 3D times series data into supervised data
Passing the supervised data to the NN for training
Saving and loading the model in order to make predicitons
Stop
Looping this step over all the NN
Repeating the whole process for the different time horizons

Figure 4.4: Descriptive flowchart of the model

4.3 Research findings

Wrapping up what we have covered so far about modelling, now some of the major works that has been done similar to this one in terms of input features and neural network used and the output horizon achieved will be cited in table 4.1:

Author	**Method**	**Time horizon**	**Input used**	**Accuracy**
Mahmou et al [17]	LSTM	1 hour ahead	historical powers	RMSE = 82.15
/ [18]	LSTM	Up to 24 h ahead	GHI, weather forecast	RMSE = 76.24%
/ [18]	S2S	Up to 24 h ahead	GHI, weather forecast	$MAE = 30W/m^2\%$
/ [18]	LSTM	1 day ahead	GHI	$MAE = 60.3W/m^2\%$
/ [18]	LSTM, CI	1 hour ahead	GHI	$RMSE = 30\% - 34\%$
S.Boubaker et al [18]	LSTM	1 day ahead	GHI	$MAPE = 6.34\%$
S.Boubaker et al [18]	GRU	1 day ahead	GHI	$MAPE = 6.76\%$
S.Boubaker et al [18]	CNN-LSTM	1 day ahead	GHI	$MAPE = 10.45\%$
S.Boubaker et al [18]	CNN	1 day ahead	GHI	$MAPE = 8.93\%$

Table 4.1: Some of the DNNs forecasting researches

At the arrival to this point, what we have seen so far is the process of building forecasting models from scratch, in the upcoming chapters, the tools used to build them are explained with details and then their implementation and the results obtained from different configurations will discussed further.

Chapter 5

Software toolkit

This chapter provides a general overview about the programming language, libraries and the tools used to the completion of this project.

5.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed[19].

5.2 Pycharm / Colab

5.2.1 Pycharm

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web and data science development[20]. In addition the complete package of the IDE includes: Intelligent Python Assistance, Web Development Frameworks, Scientific Tools, Cross-technology Development, Remote Development Capabilities and Built-in Developer Tools[21].

5.2.2 Colab

Colaboratory, or "Colab" for short, allows to write and execute Python in your browser, with: zero configuration needed, free access to GPUs and easy sharing. Colab notebooks are Jupyter notebooks that are hosted by Colab. Colab notebooks allow to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When creating Colab notebooks, they are stored in the Google Drive user's account. They can be easily shared with co-workers or friends, allowing them to comment on the notebooks or even edit them[22].

5.3 TensorFlow / Keras

5.3.1 Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It is a comprehensive and flexible ecosystem of tools, libraries and other resources that provide workflows with high-level APIs. The framework offers various levels of concepts need to build and deploy machine learning models based on the choice of the user. Some of the salient features are described below[23]:

- **Easy Model Building:** It offers multiple levels of abstraction to build and train models.
- **Robust ML Production Anywhere:** It provides flexibility and ease to train and deploy models, no matter what language or platform is used.
- **Easy to use:** Well documented so easy to understand.

5.3.2 Keras

Keras is an open source neural network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. It is a useful library to construct any deep learning algorithm[24]. The main advantages of Keras are described below[23]:

- **User-Friendly:** It has a simple, consistent interface optimized for common use cases which provides clear and actionable feedback for user errors.
- **Modular and Composable:** Its models are made by connecting configurable building blocks together, with few restrictions.
- **Easy To Extend:** With the help of Keras, custom building blocks can easily be written for new ideas and researches.

5.4 CSV

Comma Separated Variables or CSV for short it is a python module that gives the ability to the user to parse in CSV files. A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter. Each line of the CSV file is considered as a row and each field as a column. The CSV format has no standard, but they are similar enough that the CSV module will be able to read the vast majority of CSV files. CSV files can also be written the CSV module[25].

5.5 Numpy

NumPy is a module for Python. The name is an acronym for “Numeric Python” or “Numerical Python”. It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed. Furthermore, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices[26]. NumPy was created in 2005 by Travis Oliphant. It is an open source project that can be used it freely, its source code is located at the github repository “https://github.com/numpy/numpy”[27].

5.6 Matplotlib

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension is NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications. A Python matplotlib script is structured so that a few lines of code are all that is required in most instances to generate a visual data plot. The matplotlib scripting layer overlays two APIs[28]:

- The pyplot API is a hierarchy of Python code objects topped by matplotlib.pyplot
- An OO API collection of objects that can be assembled with greater flexibility than pyplot. This API provides direct access to Matplotlib's backend layers.

It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram . . . [29].

5.7 Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language[30]. Its class DataFrame is an excellent method for representing tabular data, assisting in data preprocessing, modification or slicing[31].

5.8 Scikit Learn

Scikit-learn (Sklearn) is a useful library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy, Pandas and Matplotlib[32]. Sklearn provides the following functionalities[33]:

- **Regression:** Predicting a continuous-valued attribute associated with an object.
- **Classification:** Identifying which category an object belongs to.
- **Clustering:** Grouping or gathering similar objects into different sets automatically.
- **Model selection:** Comparing, validating and choosing parameters and models.
- **Pre-processing:** Feature extraction and normalization.
- **Dimensionality reduction:** Reducing the size of the input dataset to a smaller one such that it contains only the most relevant features.

5.9 Scipy

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python, It provides more utility functions for optimization, statistics and signal processing. Like NumPy, SciPy is an open source library that can be used freely. SciPy was created by NumPy's creator Travis Olliphant[34].

After acquiring an intuitive overview about the toolkits used in this project, thereafter the implementation of this study search using those tools will be covered further in the forthcoming chapter.

Chapter 6

Implementation and Evaluation

6.1 General workflow of the forecasters

Summing up what has been covered up to now, the model's building elements has gone through an extensive theoretical explanation, however from now on the practical aspect of this implementation will be clarified in the following sections. This section summarizes the whole implementation process and the interaction between each building block. In general the graphical representations show a better understanding compared to the written one. So the flowchart on figure 6.1 summarizes well how the whole process has been done.

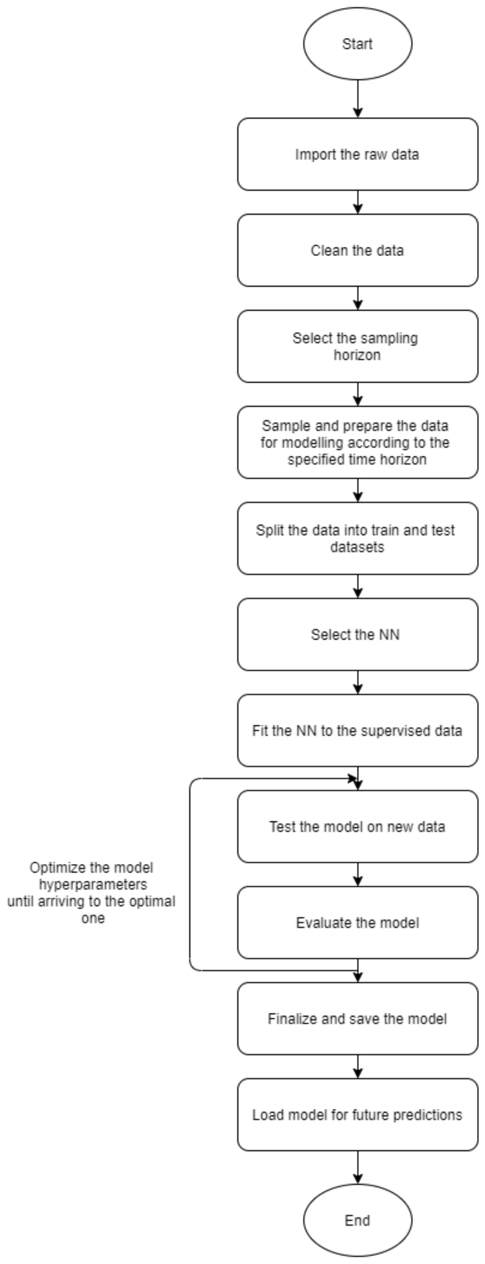

Start
Import the raw data
Clean the data
Select the sampling horizon
Sample and prepare the data for modelling according to the specified time horizon
Split the data into train and test datasets
Select the NN
Fit the NN to the supervised data
Test the model on new data
Evaluate the model
Optimize the model hyperparameters until arriving to the optimal one
Finalize and save the model
Load model for future predictions
End

Figure 6.1: General flowchart of the model

6.2 Evaluation metrics

For every model built in general it must go through an evaluation process, where this process is done by the use of some metrics. Below the only four metrics that has been used to evaluate the forecasters built all over this report are:

1. MAPE (Mean Absolute Percentage Error):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|G_{actual} - G_{forecasted}|}{|G_{actual}|} \tag{6.1}$$

2. RMSE (Root Mean Squared Error):

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(G_{forecasted} - G_{actual})^2} \tag{6.2}$$

3. r (Pearson's correlation coefficient):

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}} \tag{6.3}$$

   Where:

   - **X**: The actual value.
   - $\bar{X}$: The mean value of the variable X.
   - **Y**: The predicted value.
   - $\bar{Y}$: The mean value of the predicted variable Y.
   - **N**: Length of time series data.
   - **i**: Series number.

4. R^2 (coefficient of determination):

$$R^2 = 1 - \frac{RSS}{TSS} \tag{6.4}$$

   Where:

   - **RSS:** Sums of squares of residuals.

- **TSS:** Total sum of squares.

6.3 Results

This section of this chapter will summarize the results of all the models that have been implemented in this project. For a better distribution of the results, this part will be divided into three main subsections where each subsection will take into consideration a specific time horizon (five minutes ahead, one hour ahead, one day ahead).
However the common aspects that each specific time horizon model has are pointed down:

- Lagging time steps: 1 (minutes and hours), 7 (days)
- Future samples: 1.
- Epochs: 3.
- Batch size: 8.
- Optimizer: RMSprop (Root Mean Squared propagation).
- Activation function: ReLU (Rectified Linear Unit).

The above listed hyperparameters have been identified through the use of trial and error method. However in some cases, where the model does not converge to its optimal version, then a slight change in its number of epochs was applied and the last optimal reached value was eight for some and fifty for others.
On the other, all the models have been trained and tested using the same data sets, where:

- **Training data sets:**
  1. Eugene_aSiMicro03036.
  2. Cocoa_aSiMicro03036.
- **Test data set:**
  1. Eugene_xSi12922.

At this point three things must be highlighted before going further in the discussion, the first thing is that the test data set has not been used fully instead, only 2500 samples were used for the minute and hour sampling rates, however just 359 samples has been employed for the day sampling rate due to the lack of data, here the latter mentioned term "sampling rate" signifies that the mean value is taken for the specified horizon. The second thing is the meteorological parameter used for both the training and testing was the GHI, the last thing is that the common structure used for each model at each time horizon which will be well summarized in the table 6.1.

Neural Network	**Structure**
Vanilla LSTM	50:1
RNN	50:1
GRU	50:1
LSTM-RNN	50:50:1
LSTM-GRU	50:50:1
GRU-RNN	50:50:1
LSTM-GRU-RNN	50:50:50:1
CNN	100:100:1
Stacked LSTM	50:50:1
AE LSTM	50:50:100:1
CNN-LSTM	64:64:1:200:100:1
CNN-LSTM-GRU-RNN	64:64:1:200:100:200:100:200:100:1

Table 6.1: Structure of each NN in the project

6.3.1 Five minutes ahead forecasting

The results summary and the plots of the five minutes ahead forecasters are presented in table 6.2 and figures from 6.2 to 6.13 respectively.

Metric / NN	**MAPE %**	**RMSE** (W/m^2)	**Epoch time (sec)**	**r**	R^2
Vanilla LSTM	16.55	46.91	17.33	0.89	0.78
RNN	16.61	46.63	15.33	0.89	0.78
GRU	16.55	46.73	18.33	0.89	0.78
LSTM-RNN	16.55	46.94	22.67	0.89	0.78
LSTM-GRU	16.57	47.03	26.33	0.89	0.78
GRU-RNN	16.55	46.93	24	0.89	0.78
LSTM-GRU-RNN	16.55	46.94	33.67	0.89	0.78
CNN	16.62	46.64	13.33	0.89	0.79
Stacked LSTM	16.56	46.9	27.33	0.89	0.78
AE LSTM	16.67	47.02	30.67	0.89	0.78
CNN-LSTM	16.56	46.96	52.33	0.89	0.78
CNN-LSTM-GRU-RNN	16.61	46.86	98.33	0.89	0.78

Table 6.2: Results for five minutes ahead forecasting

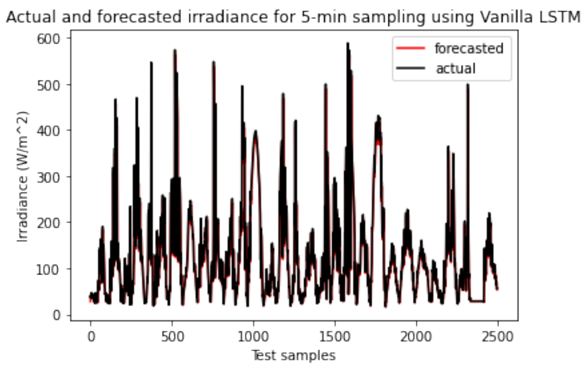


Figure 6.2: Output test for Vanilla LSTM

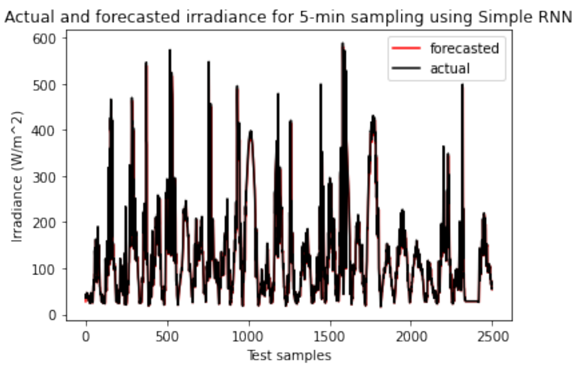


Figure 6.3: Output test for Simple RNN

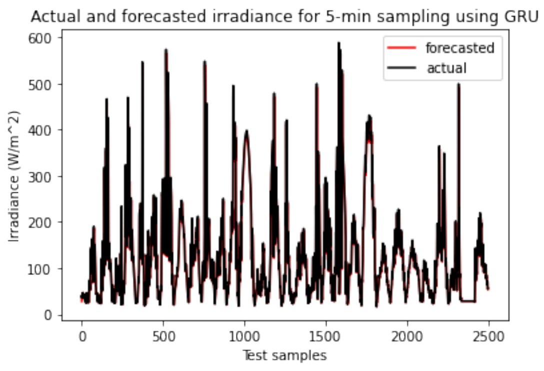


Figure 6.4: Output test for GRU

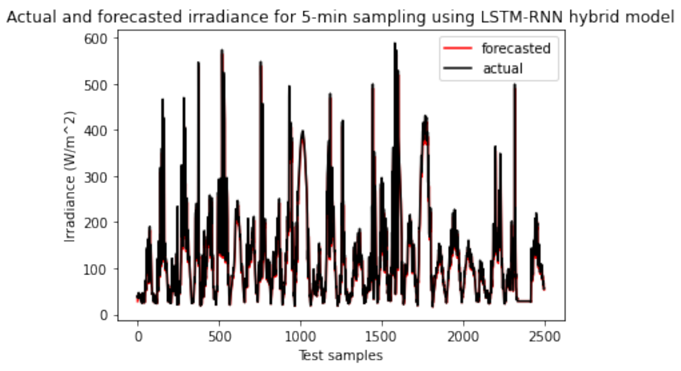


Figure 6.5: Output test for LSTM RNN

*

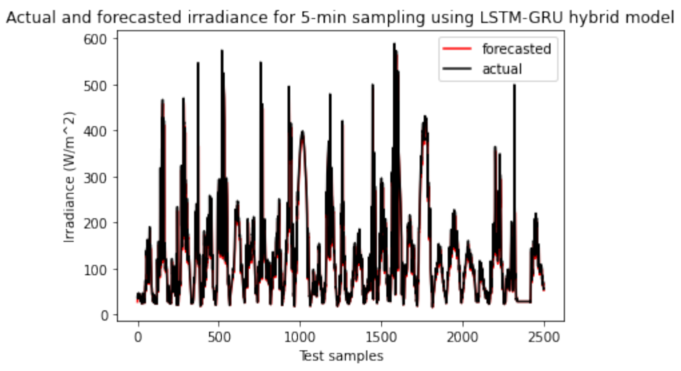


Figure 6.6: Output test for LSTM GRU

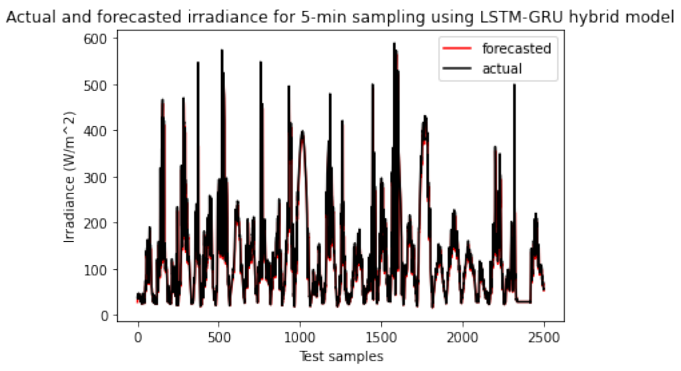


Figure 6.7: Output test for GRU RNN

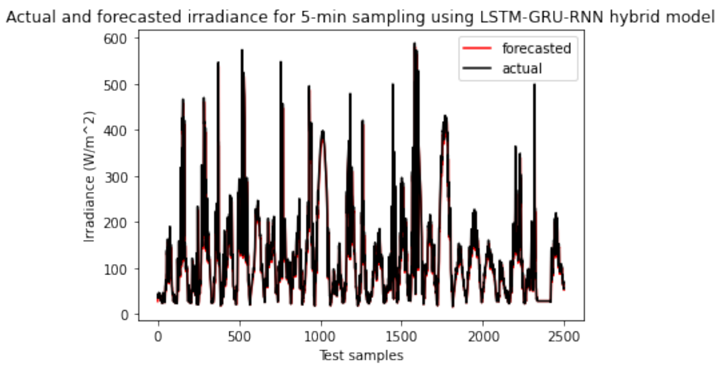


Figure 6.8: Output test for LSTM GRU RNN

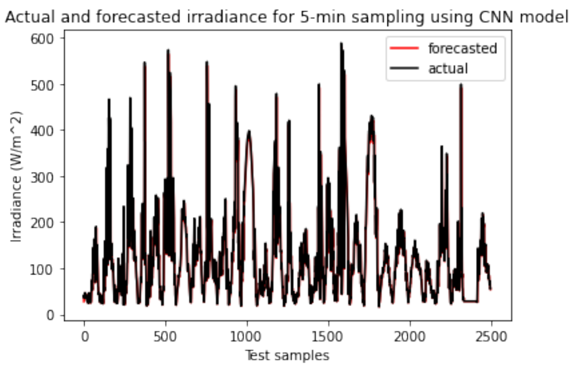


Figure 6.9: Output test for CNN

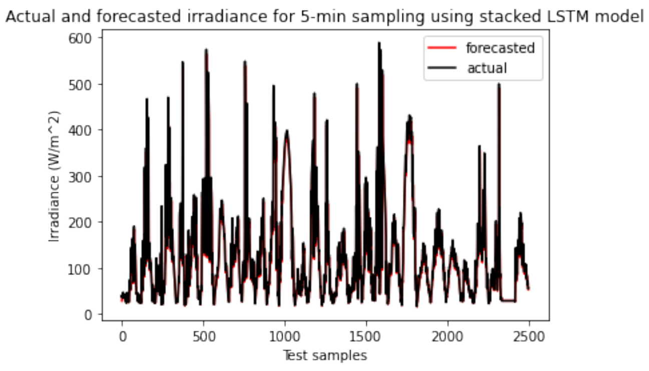


Figure 6.10: Output test for Stacked LSTM

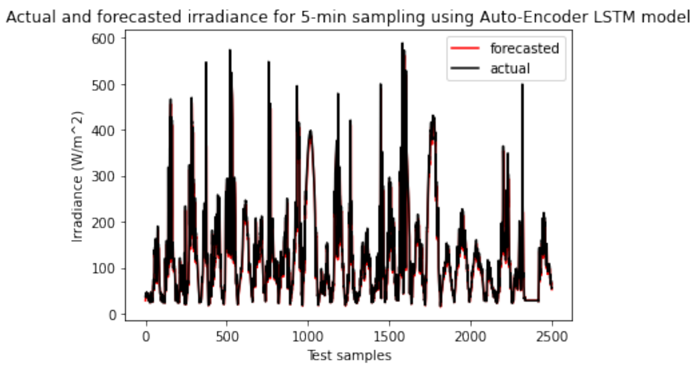


Figure 6.11: Output test for AE LSTM

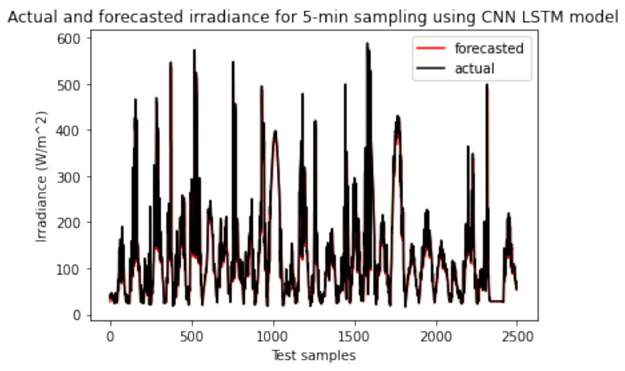


Figure 6.12: Output test for CNN LSTM

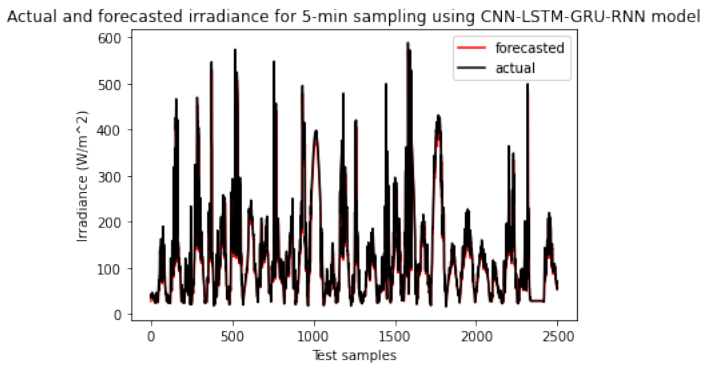


Figure 6.13: Output test for CNN LSTM GRU RNN

6.3.2 One hour ahead forecasting

The results summary and the plots of the one hour ahead forecasters are presented in table 6.3 and figures from 6.14 to 6.25 respectively.

Metric / NN	**MAPE %**	**RMSE** (W/m^2)	**Epoch time (sec)**	**r**	R^2
Vanilla LSTM	45.31	187.8	2.125	0.89	0.58
RNN	45.22	175.93	1.875	0.89	0.63
GRU	50.3	241.92	2.125	0.89	0.3
LSTM-RNN	44.46	173.7	2.625	0.89	0.64
LSTM-GRU	44.23	168.52	3.25	0.89	0.66
GRU-RNN	44.44	170.4	2.375	0.89	0.65
LSTM-GRU-RNN	43.6	178.28	3.75	0.89	0.62
CNN	45.01	166.75	2.125	0.89	0.67
Stacked LSTM	44.7	179.02	3.25	0.89	0.61
AE LSTM	41.36	162.49	3.25	0.89	0.69
CNN-LSTM	44.59	165.34	5.33	0.89	0.67
CNN-LSTM-GRU-RNN	43.19	158.41	11.33	0.89	0.7

Table 6.3: Results for one hour ahead forecasting

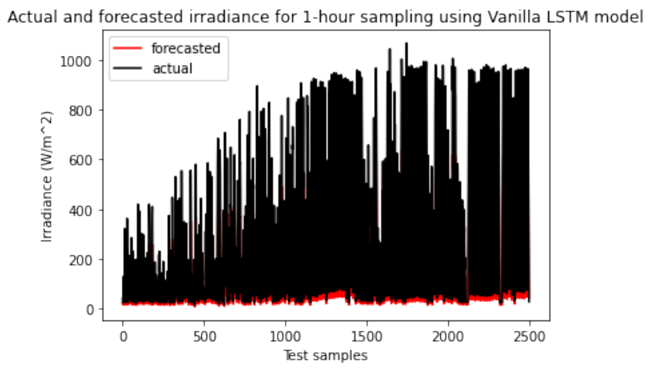


Figure 6.14: Output test for Vanilla LSTM

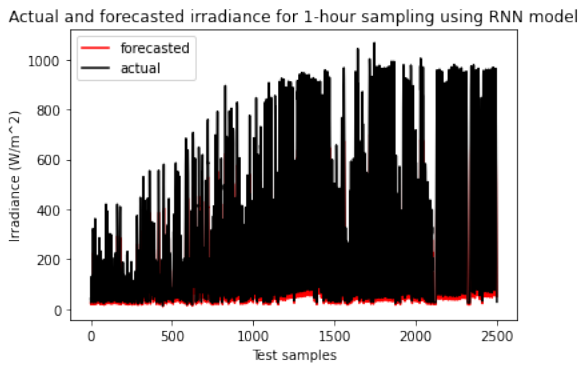


Figure 6.15: Output test for Simple RNN

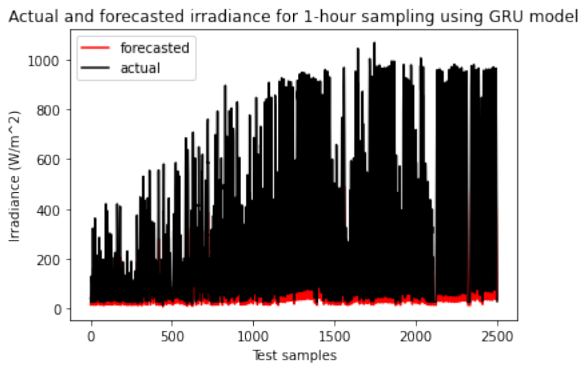


Figure 6.16: Output test for GRU

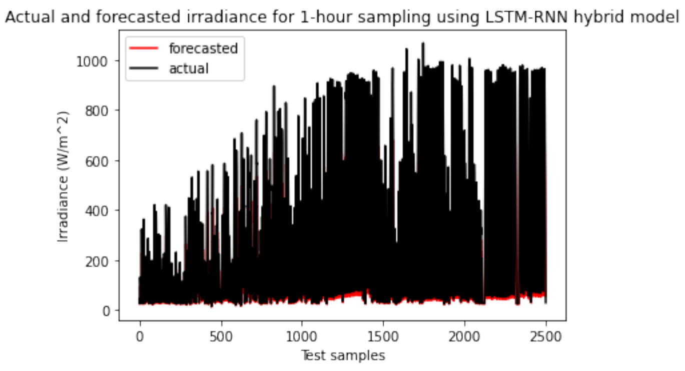


Figure 6.17: Output test for LSTM RNN

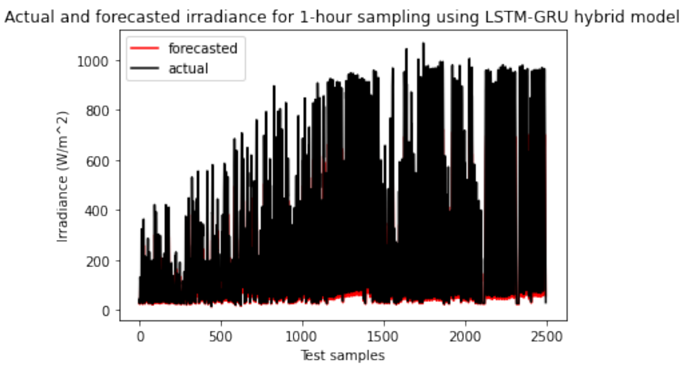


Figure 6.18: Output test for LSTM GRU

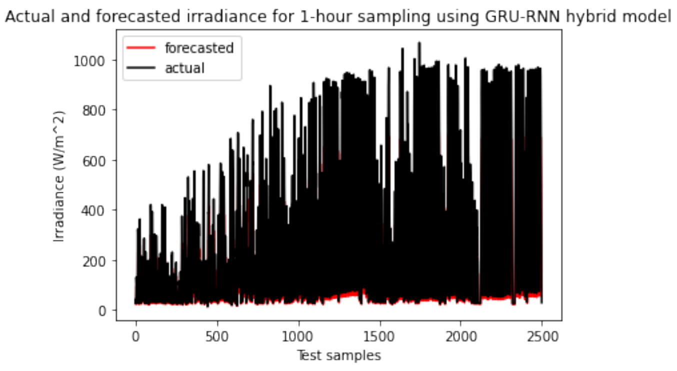


Figure 6.19: Output test for GRU RNN

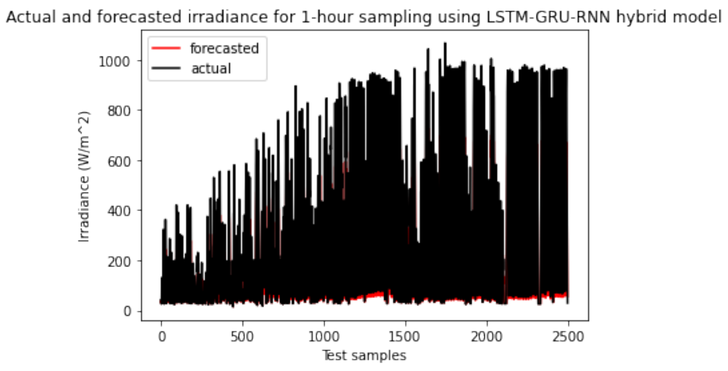


Figure 6.20: Output test for LSTM GRU RNN

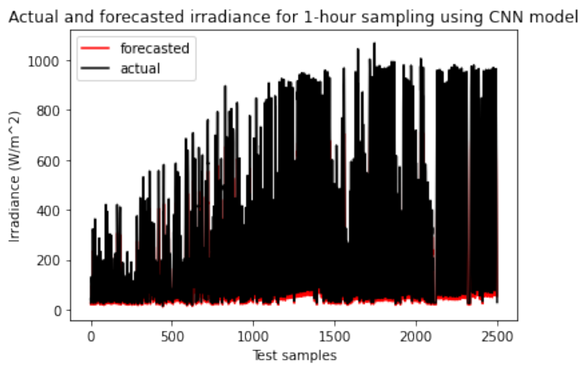


Figure 6.21: Output test for CNN

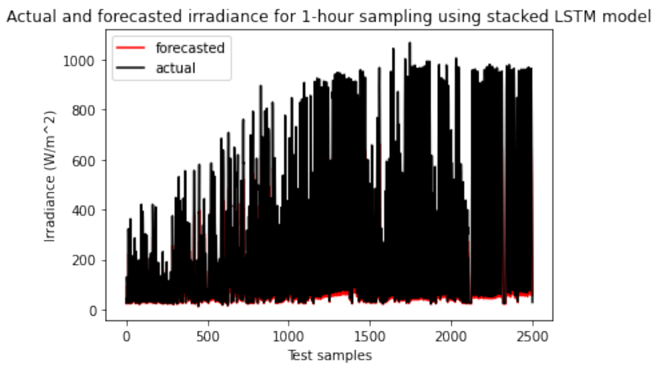


Figure 6.22: Output test for Stacked LSTM

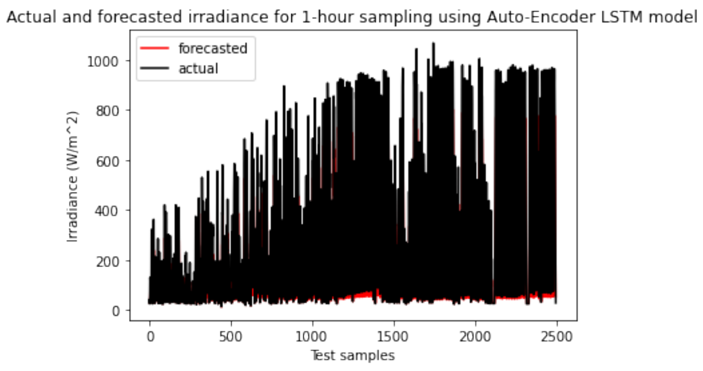


Figure 6.23: Output test for AE LSTM

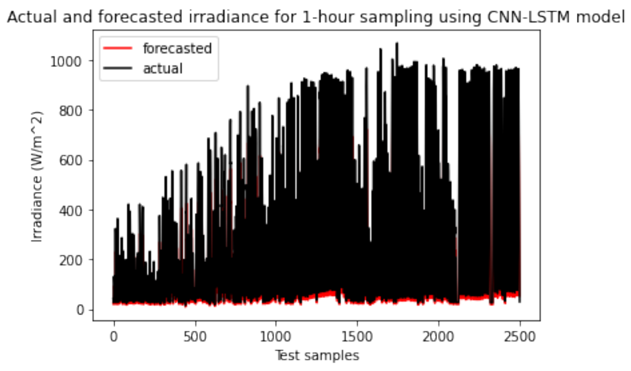


Figure 6.24: Output test for CNN LSTM

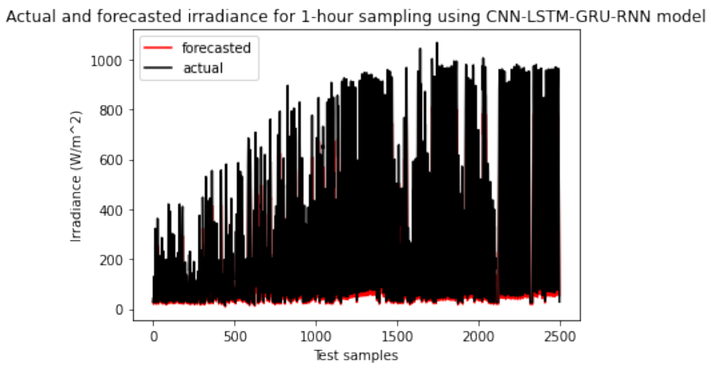


Figure 6.25: Output test for CNN LSTM GRU RNN

6.3.3 One day ahead forecasting

The results summary and the plots of the one day ahead forecasters are presented table 6.4 and figures from 6.26 to 6.37 respectively.

Metric / NN	**MAPE %**	**RMSE** (W/m^2)	**Epoch time (sec)**	**r**	R^2
Vanilla LSTM	43.81	138.29	0.56	0.71	0.38
RNN	44.39	132.33	0.31	0.75	0.43
GRU	44.09	129.99	0.43	0.74	0.45
LSTM-RNN	44.92	133.87	1.6	0.73	0.42
LSTM-GRU	43.55	127.99	1.06	0.75	0.47
GRU-RNN	43.83	128.47	1.02	0.76	0.47
LSTM-GRU-RNN	44.92	132.98	1.06	0.73	0.43
CNN	43.85	131.78	0.25	0.76	0.44
Stacked LSTM	45.09	134.88	1.04	0.72	0.41
AE LSTM	44.71	127.37	0.26	0.74	0.48
CNN-LSTM	44.47	129.29	1.25	0.75	0.46
CNN-LSTM-GRU-RNN	44.63	132.02	1.5	0.75	0.44

Table 6.4: Results for one hour ahead forecasting

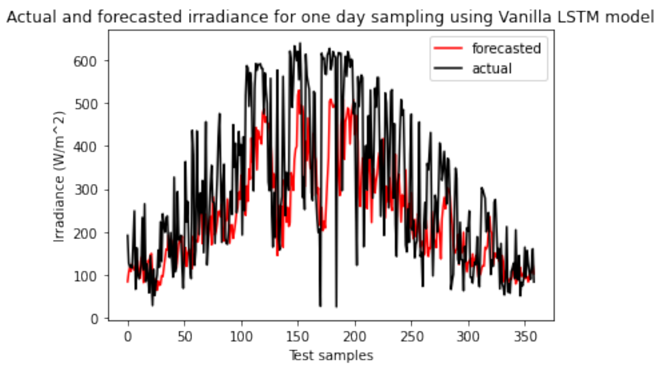


Figure 6.26: Output test for Vanilla LSTM

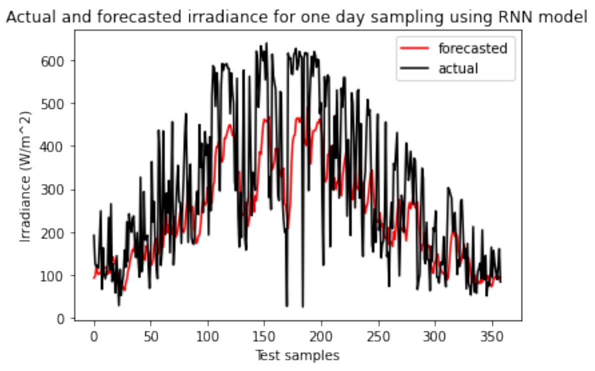


Figure 6.27: Output test for Simple RNN

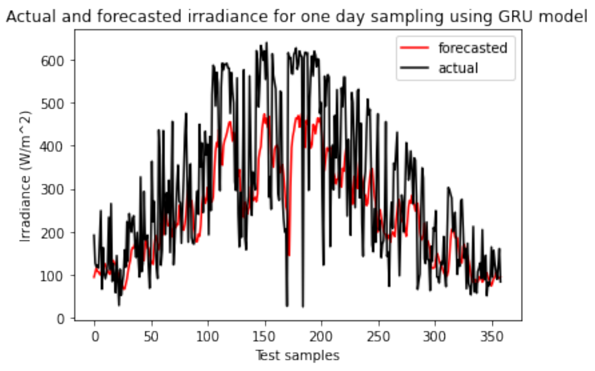


Figure 6.28: Output test for GRU

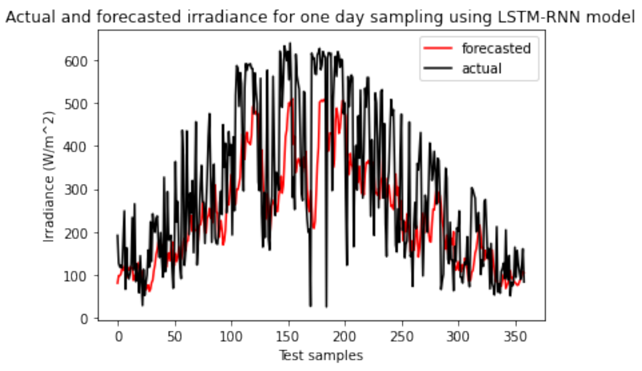


Figure 6.29: Output test for LSTM RNN

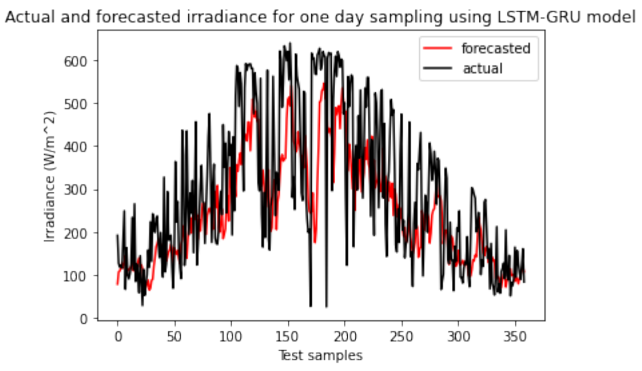


Figure 6.30: Output test for LSTM GRU

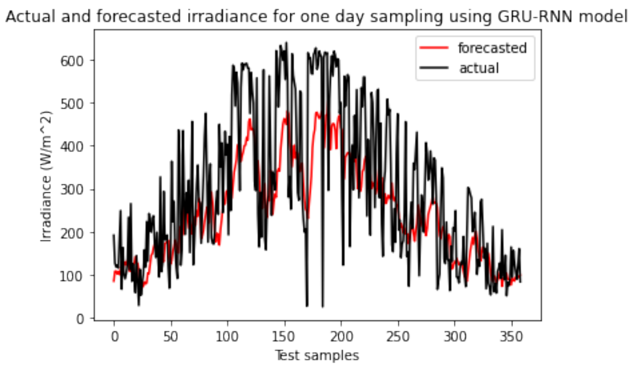


Figure 6.31: Output test for GRU RNN

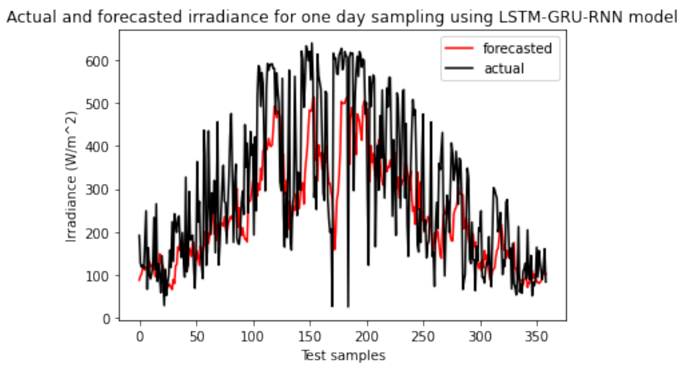


Figure 6.32: Output test for LSTM GRU RNN

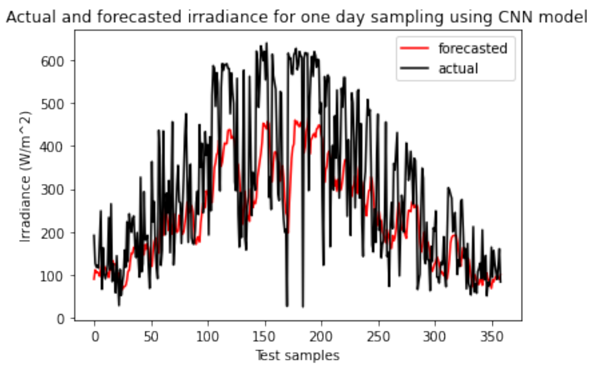


Figure 6.33: Output test for CNN

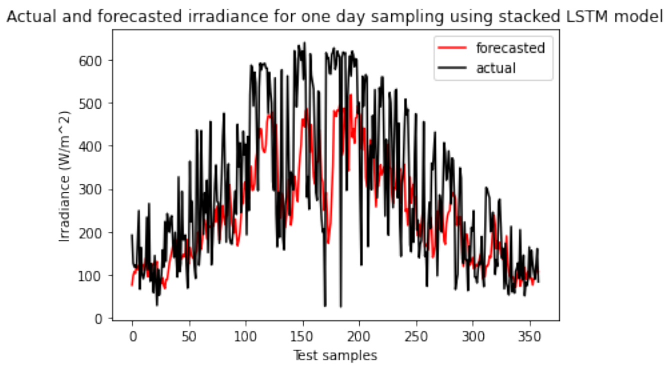


Figure 6.34: Output test for Stacked LSTM

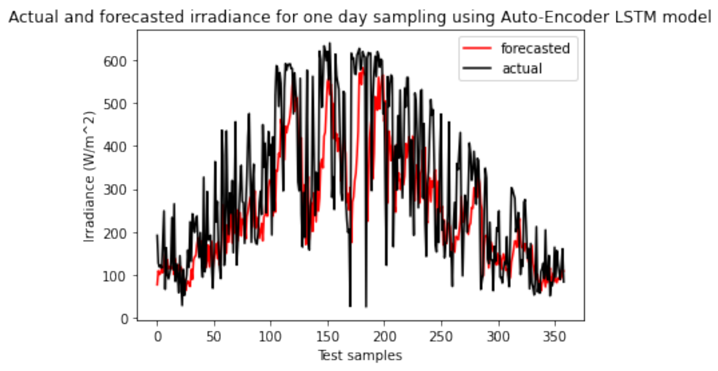


Figure 6.35: Output test for AE LSTM

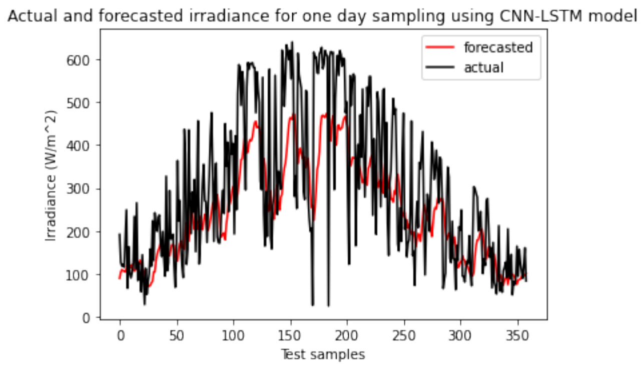


Figure 6.36: Output test for CNN LSTM

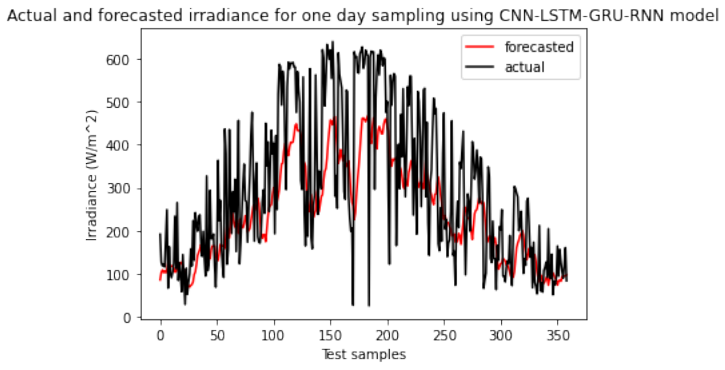


Figure 6.37: Output test for CNN LSTM GRU RNN

6.4 Discussion and analysis

For a better insight, the detailed discussion at the beginning will take each time horizon independently, then the global discussion will differentiate the performance of each model at every time horizon and also it will point out the influence of the deep learning algorithms on the size of the training data set.

6.4.1 Five minutes ahead forecasting

First thing first, the remarkable points on the results are listed down below:

- The correlation coefficient and the coefficient of determination were constant and high no matter what the model is.
- The MAPE and RMSE were almost the same for each and every model, and they were acceptable varying from the range of 16.55 % to 16.67 % and 46.63 (W/m^2) to 47.02 (W/m^2) for MAPE and RMSE respectively.
- The training time differs significantly according to the nature of the model.
- The output graphs follow the same pattern of the actual data, however at the sharp changes all the models could not follow well the shape of the actual data.

The interpretation of the above results yielded to the choose of the simple RNN as the outstanding model for five minutes ahead forecasting, also the notable remark is that no matter how complex is the model, there is no surprising evolution on the results and only the computational cost and the complexity is added, the latter point can be explained but without rushing events it will be detailed in the posterior discussions.

6.4.2 One hour ahead forecasting

As before the major remarks are mentioned beneath:

- The correlation coefficient is constant for all the models.
- The coefficient of determination fluctuates from a model to another between 0.3 to 0.7.

- The MAPE compared to the previous time horizon is higher, but the for entire set of models they are more or less close to each other and it ranges from 41.36 % to 50.3 %.
- The RMSE has the same remark as the latter pointed metric compared to the previous time horizon, but the range is wider starting from 158.41 (W/m^2) to 241.92 (W/m^2).
- Similarly to the preceding discussion the nature of the model significantly impacts the epoch time or the training time.
- The output graphs chase the graph of the actual data.

First the analysis of the up above observations signifies that:

1. There is a strong relationship between the relative movements for the actual and the forecasted data, which is represented by the correlation coefficient "r" and it is constant for all the forecasters.
2. The fitness of the predicted output to the real data is acceptable for almost all the forecasters and it is measured in terms of the coefficient of determination "R^2".

Through the understanding of the above observation, the two skilful models that were selected for being the best ones at the current time horizon are:

1. AE LSTM.
2. CNN-LSTM-GRU-RNN.

The selection between the last two models does not significantly appear in terms of neither MAPE, RMSE, r nor R^2, however it really shines when it comes to the computational time where 3.25 (sec) and 11.33 (sec) per training epoch for AE LSTM and CNN-LSTM-GRU-RNN respectively.

6.4.3 One day ahead forecasting

Continuing in the same manner the significant remarks are summarized next:

- Mean value of the determination coefficient.

- Satisfying correlation coefficient.
- No significant difference in the training time no matter how complex or simple the model is.
- The MAPE and RMSE do not have huge variation from one model to another and their scores range from 43.81 % to 45.09 % and from 127.37 (W/m^2) to 138.29 (W/m^2) for MAPE and RMSE respectively.
- The output graphs seem to have the same pattern as the real world data with a slight deviation.

From the obtained results and by going through the process of comparing the different evaluation metrics, then AE LSTM seems to outperform the other realized models.

Chapter 7

Conclusion

Finally, the last chapter has been reached by going through several experiments and tests and various outcomes had been yielded, those outcomes are summed up in the next few lines.

The common thing for the three different time horizons is that, the more complex the forecaster is the more is the computational time and training time with no considerable improvement in the performance (though sometimes it worsen the performance), this point practically is undesired, because the aim for real world projects is to achieve better results with less computational and execution time and cost, this point has been highlighted earlier in the discussion and analysis section, talking as a deep learning developer when it is desired to develop a powerful forecaster this is not done directly by randomly increasing the size of the model, however several considerations must be taken into account which are:

- The size of the training data set.
- The quality and predictability of the data.
- Optimizing the model to its optimal hyperparameters such that the loss during the training phase falls to its global minima and taking into consideration the test error, where it must be bounded (this is called preventing overfitting in ML).

This could be a satisfying answer for why complex model does not necessary mean a better performance. Also, recurrent neural networks (simple RNN, GRU, LSTM) showed almost similar performance. Though some used NNs in this project showed an excellent performance when being applied for different tasks like CNN for image processing, object detection, image classification . . . etc or LSTM for dealing with problems that have long term dependencies, instead they have showed a humble performance when applied to this task, this may be a result of:

1. Lack of the training data.
2. Some NNs have some criteria that do not capture well the details for time series data sets with insufficient training samples.
3. Tuning the inappropriate model hyperparameters with no prior knowledge of the data set.

As a suggestion for the latter issue the selection of the model's hyperparameters must be done in an advanced manner unlike the method used here (trial and error), this will be kept as a further work along with others which will be stated in the succeeding part.

Further work

The list of works left as a further development for the proposed architectures are pointed below:

- Applying the models with larger data sets for training and testing.
- Trying to see the improvement of the model with additional attributes.
- Extending the models' prediction horizon to the furthermost possible point in the future.
- Simulating the power from the forecasted irradiance.
- Predicting the power directly using the proposed models and comparing with the simulated one obtained from the forecasted irradiance.
- Employing different preprocessing techniques for the data.
- Tuning the proposed models' hyperparameters using more advanced research techniques such as grid search.
- Try modifying the configuration of keras deep neural networks in terms of the training algorithm from backpropagation through time into Levenberg Marquardt algorithm in a similar or a different framework.

Bibliography

[1] Mellit A, Kalogirou SA. Artificial intelligence techniques for photovoltaic applications: a review. Progress Energy Combust Sci 2008;34:574–632.

[2] Dalton G, Lockington D, Baldock T. Feasibility analysis of renewable energy supply options for a grid-connected large hotel. Renew Energy 2009;34:955–64.

[3] Deep Time Series Forecasting with Python An Intuitive Introduction to Deep Learning for Applied Time Series Modeling by N D Lewis.

[4] Builtin.com, A GUIDE TO RNN: UNDERSTANDING RECURRENT NEURAL NETWORKS AND LSTM.
[*Online*]. Available:
https://builtin.com/data-science/recurrent-neural-networks-and-lstm
Accessed: [05-06-2021]

[5] Jason Brownlee, Deep learning with python, Machine Learning Mastery, 2016.

[6] Jason Brownlee, Long Short-Term Memory Networks With Python, Machine Learning Mastery, 2017.

[7] Kyunghyun Cho et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.

[8] Blog.marketmuse.com, Gated Recurrent Unit (GRU).
[*Online*]. Available:
https://blog.marketmuse.com/gated-recurrent-unit-gru-definition/
Accessed: [07-06-2021]

[9] Jason Brownlee, Deep learning for Time series. Machine Learning Mastery, 2019.

[10] Towardsdatascience.com, A Comprehensive Guide to Convolutional Neural Networks.
[*Online*]. Available:
https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
Accessed: [08-06-2021]

[11] Towardsdatascience.com, What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?
[*Online*]. Available:
https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a
Accessed: [08-06-2021]

[12] Towardsdatascience.com, Neural Networks: parameters, hyperparameters and optimization strategies.
[*Online*]. Available:
https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5
Accessed: [08-06-2021]

[13] Kdnuggets.com, Optimization Algorithms in Neural Networks.
[*Online*]. Available:
https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html
Accessed: [09-06-2021]

[14] Towardsdatascience.com, Understanding Hyperparameters and its Optimisation techniques.
[*Online*]. Available:
https://towardsdatascience.com/understanding-hyperparameters-and-its-optimisation-techniques-f0debba07568
Accessed: [09-06-2021]

[15] User's Manual for Data for Validating Models for PV Module Performance. *https://www.nrel.gov/publications*

[16] Jason Brownlee, Data Preparation for Machine Learning. Machine Learning Mastery, 2020.

[17] Abdel-Nasser, M.; Mahmoud, K. Accurate photovoltaic power forecasting models using deep LSTM-RNN. Neural Comput. Appl. 2017, 31, 2727–2740.

[18] S. Boubaker et al, Deep Neural Networks for Predicting Solar Radiation at Hail Region, Saudi Arabia. IEEE Access. 2021.

[19] Python.org, What is Python? Executive Summary.
[*Online*]. Available:
https://www.python.org/doc/essays/blurb/
Accessed: [03-06-2021]

[20] Jetbrains.org, Get started.
[*Online*]. Available:
https://www.jetbrains.com/help/pycharm/quick-start-guide.html
Accessed: [03-06-2021]

[21] Jetbrains.org, ALL THE PYTHON TOOLS IN ONE PLACE.
[*Online*]. Available:
https://www.jetbrains.com/pycharm/
Accessed: [03-06-2021]

[22] Colab.research.google.com, What is Colaboratory?
[*Online*]. Available:
https://colab.research.google.com/notebooks/intro.ipynb
Accessed: [03-06-2021]

[23] Analyticsindiamag.com, TensorFlow vs Keras: Which One Should You Choose.
[*Online*]. Available:
https://analyticsindiamag.com/tensorflow-vs-keras-which-one-should-you-choose/
Accessed: [04-06-2021]

[24] Guru99.com, Keras vs Tensorflow: Must Know Differences!
[*Online*]. Available:
https://www.guru99.com/tensorflow-vs-keras.html#5
Accessed: [04-06-2021]

[25] Dzone.com, Python CSV Files: Reading and Writing.
[*Online*]. Available:
https://dzone.com/articles/python-101-reading-and-writing
Accessed: [03-06-2021]

[26] Python-course.eu, Numpy Tutorial.
[*Online*]. Available:
https://www.python-course.eu/numpy.php
Accessed: [03-06-2021]

[27] W3schools.com, NumPy Introduction.
[*Online*]. Available:
https://www.w3schools.com/python/numpy/numpy_intro.asp
Accessed: [03-06-2021]

[28] Activestate.com, What Is Matplotlib In Python?
[*Online*]. Available:
https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/
Accessed: [03-06-2021]

[29] Geeksforgeeks.org, Python — Introduction to Matplotlib.
[*Online*]. Available:
https://www.geeksforgeeks.org/python-introduction-matplotlib/
Accessed: [03-06-2021]

[30] Pandas.pydata.org, pandas.
[*Online*]. Available:
https://pandas.pydata.org/
Accessed: [03-06-2021]

[31] Otik.zcu.cz, Time Series Forecasting using Deep Neural Networks.
[*Online*]. Available:
https://otik.zcu.cz/bitstream/11025/39190/1/Diploma_Thesis_Lada_Zadranska.pdf
Accessed: [03-06-2021]

[32] Tutorialspoint.com, Scikit Learn Tutorial.
[*Online*]. Available:
https://www.tutorialspoint.com/scikit_learn
Accessed: [04-06-2021]

[33] Codecademy.com, What is Scikit-Learn?
[*Online*]. Available:
https://www.codecademy.com/articles/scikit-learn
Accessed: [04-06-2021]

[34] W3schools.com, Scipy Introduction.
[*Online*]. Available:
https://www.w3schools.com/python/scipy/scipy_intro.php
Accessed: [04-06-2021]

Appendix A

A.1

Field	Element	Description
1	PV Module Identifier	Unique alphanumeric module identifier assigned by NREL
2	City	City where measurement site located
3	State	State where measurement site located
4	Time zone	Eastern = -5, Western = -7, Pacific = -8
5	Latitude	Latitude in decimal degrees, N+
6	Longitude	Longitude in decimal degrees, W-
7	Elevation	Elevation in meters above sea level
8	PV module tilt	PV module tilt angle from horizontal in degrees
9	PV module azimuth	PV module azimuth angle from north in degrees (N=0, E=90, S=180, W=270)

Table A.1: File header elements and definitions

A.2

Field	Element	Description
1	Date and time	Local standard time for the site, formatted as yyyy-mm-ddThh:mm:ss
2	Plane of array (POA) irradiance	Amount of solar irradiance in watts per square meter received on the PV module surface at the time indicated, measured with CMP 22 pyranometer.

3	POA irradiance uncertainty	Uncertainty in percent based on random and bias error estimates.
4	PV Module Back-Surface Temperature	PV module back-surface temperature in degrees Celsius at the time indicated, measured behind center of cell near center of PV module.
5	PV Module Back-Surface Temperature Uncertainty	Uncertainty in degrees Celsius based on random and bias error estimates.
6	PV Module I_{sc}	Short-circuit current of PV module in amperes at the time indicated.
7	PV Module I_{sc} Uncertainty	Uncertainty in percent based on random and bias error estimates.
8	PV Module P_m	Maximum power of PV module in watts at the time indicated.
9	PV Module P_m Uncertainty	Uncertainty in percent based on random and bias error estimates.
10	PV Module I_{mp}	Current of PV module in amperes when operating at maximum power at the time indicated.
11	PV Module I_{mp} Uncertainty	Uncertainty in percent based on random and bias error estimates.
12	PV Module V_{mp}	Voltage of PV module in volts when operating at maximum power at the time indicated.
13	PV Module V_{mp} Uncertainty	Uncertainty in percent based on random and bias error estimates.
14	PV Module V_{oc}	Open-circuit voltage of PV module in volts at the time indicated.
15	PV Module V_{oc} Uncertainty	Uncertainty in percent based on random and bias error estimates.

16	PV Module FF	Fill-factor of PV module in percent at the time indicated.
17	PV Module FF Uncertainty	Uncertainty in percent (relative) based on random and bias error estimates.
18	Delta CMP 22 POA	Change in POA irradiance measured with CMP 22 pyranometer from the time indicated to the end of the I-V curve measurement (1 second elapsed time).
19	Delta LI-COR POA	Change in POA irradiance measured with LI-COR pyranometer from the time indicated to the end of the I-V curve measurement (1 second elapsed time).
20	MT5 Cabinet Temperature	Air temperature within cabinet containing the MT5 multi-tracer in degrees Celsius at the time indicated.
21	Dry Bulb Temperature	Dry bulb temperature at the site in degrees Celsius at the time indicated for Golden, nearest 5-second average to the time indicated for Cocoa and Eugene.
22	Dry Bulb Temperature Uncertainty	Uncertainty in degrees Celsius based on random and bias error estimates.
23	Relative humidity	Relative humidity at the site in percent, nearest 5-second average to the time indicated.
24	Relative humidity Uncertainty	Uncertainty in percent (relative) based on random and bias error estimates.
25	Atmospheric pressure	Atmospheric pressure at the site in millibars, nearest 5-second average to the time indicated.

26	Atmospheric pressure Uncertainty	Uncertainty in percent based on random and bias error estimates.
27	Precipitation	Accumulated daily total precipitation in millimeters at the time indicated.
28	Direct Normal Irradiance	Amount of solar irradiance in watts per square meter received within a 5.7° field-of-view centered on the sun, nearest 5-second average to the time indicated.
29	Direct Normal Irradiance Uncertainty	Uncertainty in percent based on random and bias error estimates.
30	Direct Normal Irradiance Uncertainty	Standard deviation in watts per square meter of the 1-second samples in the 5-second average for the direct normal irradiance.
31	Global Horizontal Irradiance	Total amount of direct and diffuse solar irradiance in watts per square meter received on a horizontal surface, nearest 5-second average to the time indicated.
32	Global Horizontal Irradiance Uncertainty	Uncertainty in percent based on random and bias error estimates.
33	Global Horizontal Irradiance Standard Deviation	Standard deviation in watts per square meter of the 1-second samples in the 5-second average for the global horizontal irradiance.
34	Diffuse Horizontal Irradiance	Amount of solar irradiance in watts per square meter received from the sky (excluding the solar disk) on a horizontal surface, nearest 5-second average to the time indicated.

35	Diffuse Horizontal Irradiance Uncertainty	Uncertainty in percent based on random and bias error estimates.
36	Diffuse Horizontal Irradiance Standard Deviation	Standard deviation in watts per square meter of the 1-second samples in the 5-second average for the diffuse horizontal irradiance.
37	Solar QA Residual	Residual of solar irradiance elements in watts per square meter determined by adding the diffuse horizontal irradiance to the product of the direct normal irradiance and the cosine of the zenith angle, and then subtracting the global horizontal irradiance. If in perfect agreement, the result is zero.
38	PV Module Soiling Derate	Normalized metric comparing daily performance of a PV module to an identical PV module that is cleaned during daily maintenance. Examples: 1.000 = no soiling loss, 0.980 = 2% soiling loss.
39	Daily Maintenance Start Time	Local standard time in HH:MM format when daily maintenance activities began. 99:99 = no daily maintenance.
40	Daily Maintenance End Time	Local standard time in HH:MM format when daily maintenance activities completed. 99:99 = no daily maintenance.
41	Precipitation Prior to Daily Maintenance	Accumulated daily total precipitation in millimeters prior to completion of the daily maintenance. If no daily maintenance, equals -9999.

42	I-V Curve Data Pairs	Integer N with value equal to the number of current-voltage pairs in the I-V curve. Varies by I-V curve.
43 to 43 + N - 1	I-V Curve I Values	N number of current values of the I-V curve, one per field.
43 + N to 43 + 2N - 1	I-V Curve I Values	N number of voltage values of the I-V curve, one per field, in same order as the I-V curve current values.

Table A.2: File Data Elements and Definitions

A.3

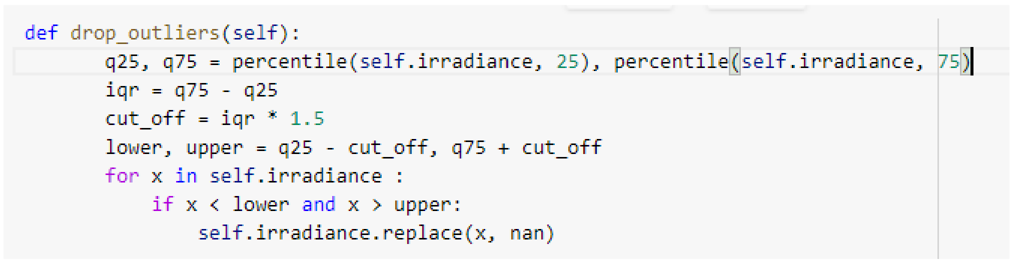
Item	**Parameter**	**Instrument**
1	Wind Speed/ Wind Direction/ Precipitation/ Temperature/ Relative Humidity/ Barometric Pressure	Vaisala WXT520 Weather Sensor
2	Direct Normal Irradiance	Kipp and Zonen CHP1 pyrheliometer
3	Global Horizontal Irradiance	Kipp and Zonen CHP1 pyrheliometer
4	Diffuse Horizontal Irradiance	Kipp and Zonen CHP1 pyrheliometer
5	Plane-of-Array Irradiance	Kipp and Zonen CHP1 pyrheliometer
6	Plane-of-Array Irradiance	LI-COR pyranometer
7	Solar Tracker	Kipp and Zonen Model SOLYS 2
8	Data Logger	Campbell Scientific, Inc. Model CR1000

9	Data Logger Communications	RAVEN XE-EVDO (Verizon network)
10	PV Module I-V Curve	Daystar MT5 Multi-Tracer
11	PV Module Back-Surface Temperature	Omega Model CO1-T Style I Thermocouple

Table A.3: List of Sensors and Data Acquisition Equipment

Appendix B

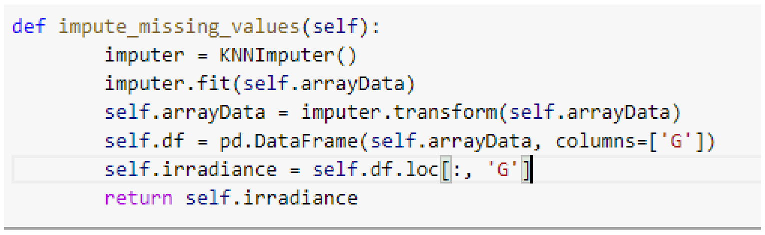
B.1



```
def drop_outliers(self):
        q25, q75 = percentile(self.irradiance, 25), percentile(self.irradiance, 75)
        iqr = q75 - q25
        cut_off = iqr * 1.5
        lower, upper = q25 - cut_off, q75 + cut_off
        for x in self.irradiance :
            if x < lower and x > upper:
                self.irradiance.replace(x, nan)
```

Figure B.1: Python script for replacing outliers by NaN

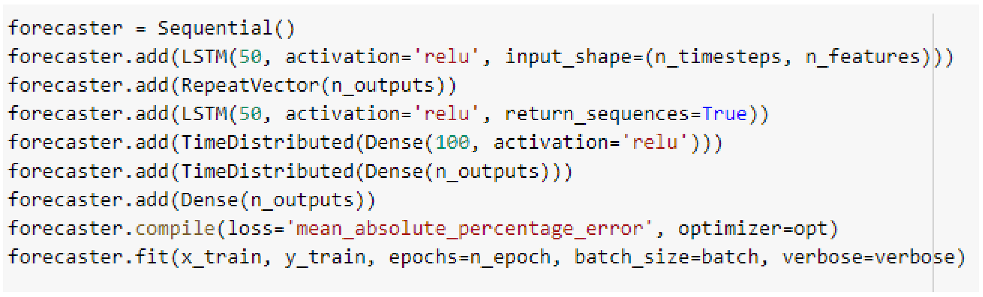
B.2



```
def impute_missing_values(self):
        imputer = KNNImputer()
        imputer.fit(self.arrayData)
        self.arrayData = imputer.transform(self.arrayData)
        self.df = pd.DataFrame(self.arrayData, columns=['G'])
        self.irradiance = self.df.loc[:, 'G']
        return self.irradiance
```

Figure B.2: Python script for imputing NaN values

B.3



```
forecaster = Sequential()
forecaster.add(LSTM(50, activation='relu', input_shape=(n_timesteps, n_features)))
forecaster.add(RepeatVector(n_outputs))
forecaster.add(LSTM(50, activation='relu', return_sequences=True))
forecaster.add(TimeDistributed(Dense(100, activation='relu')))
forecaster.add(TimeDistributed(Dense(n_outputs)))
forecaster.add(Dense(n_outputs))
forecaster.compile(loss='mean_absolute_percentage_error', optimizer=opt)
forecaster.fit(x_train, y_train, epochs=n_epoch, batch_size=batch, verbose=verbose)
```

Figure B.3: Python script for creating an AE LSTM model