INTELLIGENT PV MONITORING AND POWER EFFICIENCY PREDICTION USING ARTIFICIAL NEURAL NETWORK

PREPARED

BY

NASIR-AMEEN NASIR

17/ENG02/045

COMPUTER ENGINEERING PROGRAMME
THE DEPARTMENT OF ELECTRICAL/ELECTRONICS AND COMPUTER ENGINEERING

COLLEGE OF ENGINEERING
AFE BABALOLA UNIVERSITY, ADO-EKITI, EKITI STATE, NIGERIA

JUNE, 2022

INTELLIGENT PV MONITORING AND POWER EFFICIENCY PREDICTION USING ARTIFICIAL NEURAL NETWORK

PREPARED BY

NASIR, NASIR-AMEEN
17/ENG02/045

SUBMITTED TO THE

COMPUTER ENGINEERING PROGRAMME
THE DEPARTMENT OF ELECTRICAL/ELECTRONICS AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING

AFE BABALOLA UNIVERSITY, KM 8.5 AFE BABALOLA WAY, ADO-EKITI, EKITI STATE.

JUNE, 2022

DECLARATION

I, **NASIR-AMEEN Nasir** (17/ENG02/045), hereby declare that this project work titled "INTELLIGENT PV MONITORING AND POWER EFFICIENCY PREDICTION USING ARTIFICIAL NEURAL NETWORK (ANN)", in collaboration with MOHAMMED Raheenat Sunmisola (17/ENG02/043) was carried out under the supervision of Engr. Dr. S. L. Gbadamosi and submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (B.Eng.) A degree in Computer Engineering is my original work and has not been presented for any degree elsewhere, to the best of my knowledge.

06-06-2022

---------- **----------**

NASIR-AMEEN, Nasir Date

CERTIFICATION

This is to certify that this research project titled "INTELLIGENT PV MONITORING AND POWER EFFICIENCY PREDICTION USING ARTIFICIAL NEURAL NETWORK (ANN)" was carried out by NASIR-AMEEN Nasir (17/ENG02/045) in collaboration with MOHAMMED Raheenat Sunmisola (17/ENG02/043) and submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering(B.Eng.) Degree in Computer Engineering.

.....……	06-06-2022 ……
Engr. Dr. S. L. Gbadamosi (Project Supervisor)	Date
…….	06-06-2022 …….
Engr M. A. Ogunlade (Project Coordinator)	Date
.....……	06-06-2022 ……
Engr. Dr. M. O. Onibonoje (Head of Department)	Date
.....……	……
(External Examiner)	Date

DEDICATION

I am dedicating this project work to Almighty Allah for his guidance and mercy, and to my parents who never failed to give me moral and financial support to pursue this degree.

ACKNOWLEDGEMENT

First and foremost, I give immense gratitude to Almighty Allah, for the many blessings and mercies he has granted to me and my parents, making it possible for them to send me to this prestigious institution.

To my parents Mr. Nasir Ameen and Mrs. Taibat Nasir-Ameen , words can not express my appreciation for their immeasurable sacrifices in ensuring I get the best possible education , and for believing in every step of the way.

To my project to my supervisor, Engr. Dr S. L. Gbadamosi , I would like to thank you for your patience, insights and guidance during the whole process of making this project a success. I have benefited greatly from your wealth of knowledge and I truly appreciate the confidence you showed in me.

To my project partner Mohammed Raheenat , thank you for your contributions and due diligence in making this project work a success. To my wonderful friends and coursemates; I thank you all for impacting me on this journey.

To the Provost, staff of the Computer Engineering department College of Engineering, thank you for the provision of an enabling environment for building innovative Engineers.

ABSTRACT

In this paper, we reviewed solar energy as a solution to supply current and future energy needs. The inverter is a major component of these photovoltaic (PV) systems either autonomous or grid connected and It affects the overall performance of the PV system. This paper aims to examine the use of Artificial Neural Network (ANN) in the power output efficiency prediction of a photovoltaic (PV) system. An IoT data logger device was developed to collect intelligently and monitor in real-time the voltage and current flowing through the system. A mathematical equation is used to estimate the generated power. The initial system results show that the data logger device was efficient in transmitting the solar power usage details to the mobile application and the prediction results of a photovoltaic system show that the proposed approach has great accuracy and efficiency for forecasting the power output of the photovoltaic system.

Keywords: Photovoltaic system; Inverter efficiency; Power Monitoring; Microcontrollers; Internet of things (IoT); Artificial neural network;

TABLE OF CONTENTS

DECLARATION ii

CERTIFICATION iii

DEDICATION iv

ACKNOWLEDGEMENT v

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xi

CHAPTER ONE 1

1.0 INTRODUCTION 1

1.1 Background of Study 1

1.2 Problem Statement 3

1.3 Aim and Objectives of Study 3

1.4 Significance of Study 3

1.5 Scope of Work 4

1.6 Organization of Report 4

CHAPTER TWO 5

2.0 THEORETICAL BACKGROUND 5

2.1 Introduction 5

2.2 Internet of Things 5

2.2.1 Internet of Things and PV monitoring systems 6

2.3 Application Software 7

2.3.1 Mobile applications 8

2.3.2 Types of mobile applications 8

2.3.3 Mobile Applications Development 10

2.4 Artificial Neural Networks 11

2.4.1 Elements of neural networks 12

2.5 Literature Review 15

CHAPTER THREE 17
3.0 METHODOLOGY 17
3.1 Introduction 17
3.1.1 Project activities 17
3.2 Mobile Application Development 17
3.3 Data Analysis 19
3.4 Prediction Model Development 20
3.5 Assessing the Performance of the Prediction model 24
CHAPTER FOUR 26
4.0 RESULTS AND DISCUSSION 26
4.1 Introduction 26
4.2 Mobile Application 26
4.3 Connecting the Mobile Application to the Data Logger. 29
4.4 Prediction Model 30
CHAPTER FIVE 32
5.0 CONCLUSION AND RECOMMENDATIONS 32
5.1 Introduction 32
5.2 Conclusion 32
5.3 Limitations of Project 33
5.4 Recommendation 33
REFERENCES 34
APPENDIX A 36
Source Code for Mobile Application 36
Greenhouse Logger Startup 36
Greenhouse Logger Renderer 36
Greenhouse Logger Details View Model 37
Greenhouse Logger History View Model 39
APPENDIX B 39
Python Code For ANN Model 39

Import Libraries 39
Reading the Dataset 40
Cleaning Dataset Before Using It For Model Development 40
Creating The Network 42
Data Normalization 43
Fitting/Training the Network 43
Model Performance Evaluation 44
Data Visualisation 45

LIST OF FIGURES

Figure 1. 1: Diagram of a Hybrid PV system 2

Figure 2.1: The IoT architecture. (a) Three-layer. (b) Middle-ware based. (c) SOA based. (d) Five-layer. 6

Figure 2.2: Artificial Neural Network Architecture 11

Figure 2. 3: Backpropagation model 12

Figure 3. 1:Screenshot illustrating the emulator creation process 18

Figure 3. 2: Screenshot illustrating the debugging process 19

Figure 3.3: a). Flowchart of the ANN Model. b) Architecture of the ANN Model. 21

Figure 3. 4: Screenshot illustrating the code to initiate network training process 24

Figure 4. 1:Screenshots of Greenhouse Data Logger Dashboard 27

Figure 4. 2: Screenshot of Greenhouse Data Logger Data View Page 27

Figure 4. 3: Screenshot of Greenhouse Data Logger Data Export Page 28

Figure 4.4: Mobile Application Interface After Integration with Solar Inverter 29

Figure 4. 5: A regression plot of the actual and predicted values 30

Figure 4.6: Loss Function Trend 31

LIST OF TABLES

Table 3. 1: The components of the three PV systems and their specification 20

Table 3. 2: Input Data for Prediction Model 22

Table 3. 3: Expected Output Data 23

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background of Study

All Nigerians have long been concerned about the erratic nature of power supply. We are increasingly reliant on technologies that require electricity, yet in Nigeria, one of Africa's most populous countries, only 40% of the population is linked to the electrical grid, and those who are are subjected to brownouts and total blackouts 60 percent of the time. Meanwhile, in Nigeria, the sun shines for 43 percent of the daytime hours, averaging 1885 hours per year, indicating that solar as an alternative energy source is quite possible in this region. (Jeffrey et al., 2016)

Solar power systems use photovoltaic energy to generate usable solar power. It comprises of various components, including solar panels that absorb and convert sunlight into power, a solar inverter that converts DC to AC electric current, as well as mounting, cabling, and other electrical accessories for system setup. (Tamilselvi et al., 2019)

PV systems can be classified as stand-alone, grid-connected, or hybrid. The core ideas and features of a solar power system remain the same. Systems are customized to fit specific requirements by modifying the kind and quantity of the basic constituents. One significant feature of the solar power system is that it is modular by nature. A modular system design provides easy expansion as power demands vary.(Learning Electrical Engineering, no date)

An intelligent PV system can utilise both electrical as well as solar energy to charge the system storage battery. This can further be used to generate power in the absence of either or both energy sources. In General, only power is created by solar panels during the daylight, with peak output during noontime. This electricity is variable and not aligned with the usage of the domestic. To overcome this gap between what is generated and what is required during the absence of solar electricity generation. It is vital to store energy for subsequent use and effectively control energy storage and consumption. The last purpose is to monitor and store the data from the PV system via the Internet of Things (IoT) .(Tamilselvi et al., 2019)

Figure 1. 1: Diagram of a Hybrid PV system

1.2 Problem Statement

Consumers spend a lot of money to purchase and install solar power inverters but due to various factors, these installations do not last the life span promised by the installers, there are also cases where the PV systems are not as efficient as they are expected to be. This is mainly due to the lack of an IoT monitoring system in the PV system and the inability to predict the performance of the system.

1.3 Aim and Objectives of Study

1.3.1 Aim

The design and implementation of a smart data logger and to adequately predict the power efficiency of a Photovoltaic system.

1.3.2 Objectives

- Design and develop an IoT data logging device for the PV systems.
- Integrate the data logger with the Photovoltaic system
- Predict the power efficiency of a Photovoltaic system

1.4 Significance of Study

Designing a smart data logger for PV systems and predicting the power efficiency of PV systems is of great significance as it hopes to aid in diagnosing faults and underperformance of a PV system early on. This would greatly reduce the cost of repairs because consumers would be able to tell when their solar inverter's efficiency drops significantly. This project also serves to widen the horizon of students, especially in the areas of software development, modern-day solar technology and machine learning.

1.5 Scope of Work

This data logger system is proposed to retrieve and log the power harvested from the solar panels, the power stored and power usage; and with this data, we should be able to predict the power output efficiency of the PV system.

1.6 Organization of Report

This report contains five chapters whereby:

- Chapter one contains the background of the study, problem statement, aim and objective, significance of the study, and organization of the report.
- Chapter two discusses the theoretical background and literature review.
- Chapter three discusses the methodology of the project.
- Chapter four discusses the results and observations.
- Chapter five concludes the project, it also includes problems encountered, limitations and recommendations.

CHAPTER TWO

2.0 THEORETICAL BACKGROUND

2.1 Introduction

In Chapter 1, the background and problem, purpose, objectives, research approach and significance of the study were discussed. This chapter provides an overview of the theoretical aspect of IoT and previous research on prediction modelling using Artificial Neural Networks.

2.2 Internet of Things

The Internet of Things (IoT) is a term that refers to physical objects (or groups of such objects) that have sensors, processing power, software, and other technologies that allow them to connect and share data with other devices and systems over the Internet or other communications networks. The Internet of things has been called a misnomer because devices do not need to be connected to the public internet, they only need to be connected to a network and be individually addressable. (*Internet of Things - Wikipedia*, n.d.)

Internet of things (IoT) is playing a crucial role in the daily life of humans by enabling the connectivity of many physical devices through the internet where the devices are intelligently linked together enabling new kinds of communication between things and people, and between things themselves to exchange the data for monitoring and controlling the devices from anywhere around the globe using the internet connection. Additionally, communication between machines or different devices is possible without human intervention using IoT applications. The idea behind the IoT principle is to connect the sensors and devices of a special system on a common network through wired or wireless nodes. In general, IoT based wireless systems are widely chosen to avoid associated risks with wired systems.(Cheddadi et al., 2020)

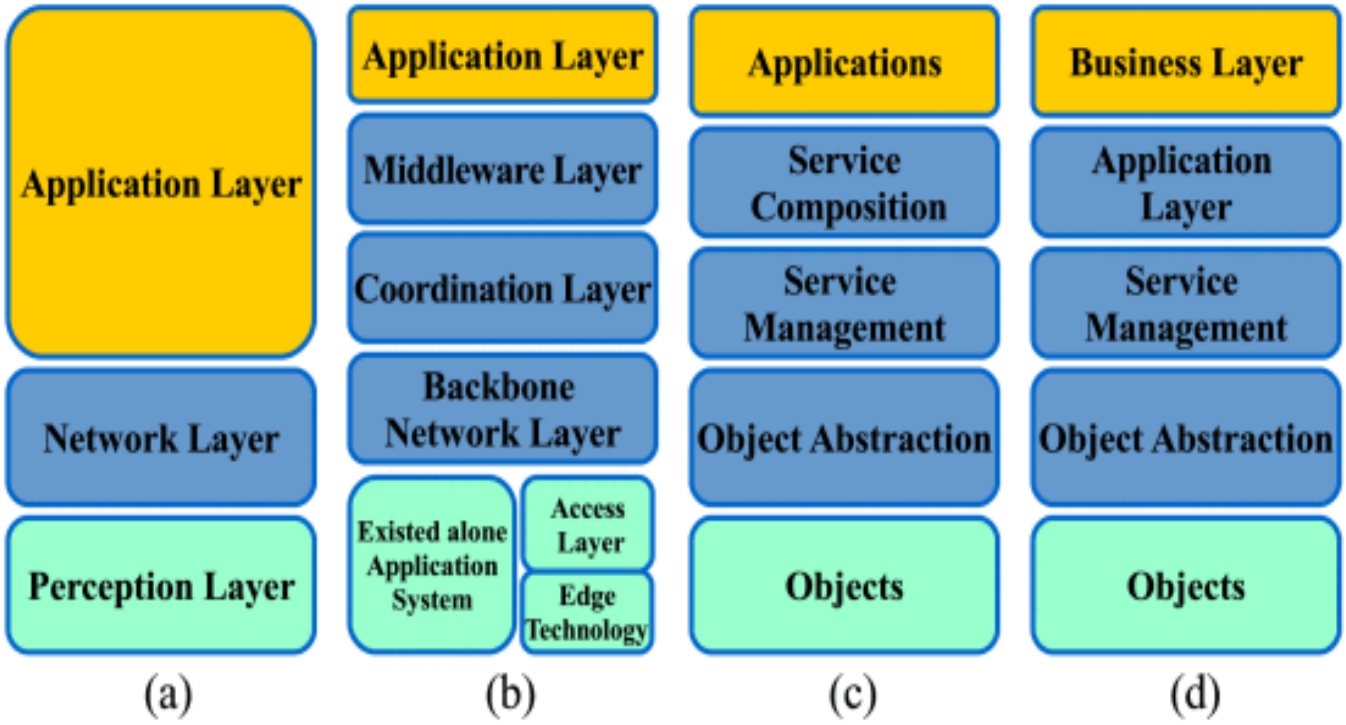

Figure 2.1: The IoT architecture. (a) Three-layer. (b) Middle-ware based. (c) SOA based. (d) Five-layer.

2.2.1 Internet of Things and PV monitoring systems

The connected "smart house" is a good illustration of IoT in action. Internet-enabled thermostats, doorbells, smoke detectors and security alarms create a linked hub where data is shared between physical devices and users can remotely operate the "things" in that hub (i.e., modifying temperature settings, unlocking doors, etc.) via a smartphone app or website. *(What Is The Internet of Things? A Complete Guide To IoT. | Built In, n.d.)*

Smart grids exploit the capability of information and communication technologies (ICT) to improve the sustainability, quality performance and balance of energy production and demand previsions, whereas reducing resource consumption. ICT also help the smart grid to integrate renewable energies.(Cheddadi et al., 2020)

The goal of PV monitoring systems is to deliver continuously clear information about numerous factors, particularly the energy potential, extracted energy, fault detection, historical analysis of the plant, and associated energy loss. Furthermore, the observed data can be used for preventive maintenance, early identification of alerts etc. Many classifications of PV monitoring systems are based on internet technology, data collecting systems employed and monitoring system approaches have been overviewed in depth by the authors..(Madeti & Singh, 2017)

2.3 Application Software

Application software, or simply applications, are often called productivity programs or end-user programs since they enable the user to execute tasks, such as writing documents, spreadsheets, databases and publications, doing online research, sending email, designing graphics etc.. (*What Is Application Software? - Definition, Examples & Types - Video & Lesson Transcript | Study.Com*, n.d.)

Application software is specific to the task it is designed for and can be as simple as a calculator application or as complex as a word processing application. When you begin creating a document, the word processing software has already set the margins, font style and size, and the line spacing for you. But you can change these settings, and you have many more formatting options available. For example, the word processor application makes it easy to add colour, headings, and pictures or delete, copy, move, and change the document's appearance to suit your needs.(*What Is Application Software? - Definition, Examples & Types - Video & Lesson Transcript | Study.Com*, n.d.)

2.3.1 Mobile applications

A mobile application or app is a computer program or software application that is designed to run on mobile devices such as a phone, tablet, or smartwatch. Mobile applications typically stand in contrast to desktop programs which are meant to run on desktop computers, and web applications which run via mobile web browsers rather than directly on the mobile device. (*Mobile App - Wikipedia*, n.d.)

Applications were originally meant for productivity help such as email, calendar, and contact databases, but the public desire for apps spurred quick expansion into other areas such as mobile games, factory automation, GPS and location-based services, order-tracking, and ticket purchasing so that there are now millions of apps available. Many apps demand Internet access. Apps are often downloaded from app stores, which are a type of digital distribution platform. The term "app", short for "software application", has now become immensely popular; in2010, it was listed as "Word of the Year" by the American Dialect Society..(*Mobile App - Wikipedia*, n.d.)

2.3.2 Types of mobile applications

Mobile applications may be classed by several approaches such as to separate the native, web-based, and hybrid apps.

2.3.2.1 Native apps

All apps designed for a particular mobile platform are known as native apps. Therefore, an application intended for iOS devices does not run on Android devices. As a result, most firms develop apps for various platforms.

While developing native apps, professionals incorporate best-in-class user interface elements. This accounts for increased performance, consistency and good user experience.

Users also benefit from wider access to application programming interfaces and make infinite use of all programs from the given device. Further, they also move over from one app to another effortlessly. (*Mobile App - Wikipedia*, n.d.)

2.3.2.2 Web-based apps

Web-based software is implemented with the common web technologies of HTML, CSS, and JavaScript. Internet access is often necessary for appropriate behaviour or being able to use full capabilities compared to offline usage. The majority, if not all, of a user's data, is kept in the cloud. (*Mobile App - Wikipedia*, n.d.)

The performance of these apps is similar to a web application running in a browser, which can be noticeably slower than the equivalent native app. It also may not have the same level of features as the native app. (*Mobile App - Wikipedia*, n.d.)

2.3.2.3 Hybrid apps

The concept of the hybrid app is a mix of native and web-based programs. Apps developed employing Apache Cordova, Flutter, Xamarin, React Native, Sencha Touch, and other platforms fall into this category.

These are made to support web and native technologies across different platforms. Consequently, these apps are simpler and faster to create. It involves the use of a single codebase which works in numerous mobile operating systems. (*Mobile App - Wikipedia*, n.d.)

Despite such advantages, hybrid apps demonstrate worse performance. Often, apps fail to bear the same look and feel in multiple mobile operating systems..(*Mobile App - Wikipedia*, n.d.)

Native apps have greater performance and a more finely-tuned user interface (UI), and normally need to undergo a much tougher development and quality assurance procedure before they are published.

Web apps are employed in HTML5 or CSS and require minimum device memory as they're launched through a browser. The user is redirected on a specified web page, and all information is saved on a server-based database. Web programs require a reliable connection to be used. There are numerous sorts of apps now accessible.

- Gaming applications: The equivalent of computer video games, they are among the most popular sorts of apps. They account for one-third of all app downloads and three-fourths of all consumer expenditure.
- Productivity apps: These focus on enhancing business efficiency by easing various chores such as sending emails, tracking job progress, booking hotels, and much more.
- Living and entertainment apps: Increasingly popular, these encompass many facets of personal lifestyle and socialization such as dating, chatting on social media, as well as sharing (and watching movies. Some of the most popularly known apps such as Netflix, Facebook or TikTok belong under this category. (*What Is a Mobile Application? - Definition from Techopedia*, n.d.)

2.3.3 Mobile Applications Development

Developing apps for mobile devices needs addressing the restrictions and features of these devices. Mobile devices run on battery and have less powerful CPUs than personal computers and also have more functions such as location sensing and cameras. Developers also have to consider a broad assortment of screen sizes, hardware specs and configurations due of significant rivalry in mobile software and changes within each of the platforms (although these concerns can be overcome with mobile device identification) (although these issues can be overcome with mobile device detection).(*Mobile App - Wikipedia*, n.d.)

Mobile application development requires the use of specialized integrated development environments. Mobile apps are first tested within the development environment using emulators and later subjected to field testing. Emulators provide an inexpensive way to test applications on mobile phones to which developers may not have physical access. (*Mobile App - Wikipedia*, n.d.)

Mobile user interface (UI) Design is equally vital. Mobile UI addresses restrictions and contexts, screen, input and mobility as outlines for design. The user is often the focus of interaction with their device, and the interface comprises components of both hardware and software. User input allows the users to influence a system, and the device's output allows the system to indicate the impacts of the users' manipulation.

Overall, mobile UI design's purpose is largely towards a comprehensible, user-friendly interface. Mobile UIs, or front-ends, rely on mobile back-ends to allow access to business services. The mobile back-end supports data routing, security, authentication, authorisation, working off-line, and service orchestration.(*Mobile App - Wikipedia*, n.d.)

2.4 Artificial Neural Networks

Artificial Neural Networks, or ANN, is a group of many perceptron/neurons at each layer. ANN is also known as a Feed-Forward Neural network since inputs are handled only in the forward direction. (*"An Analysis of Deep Learning Neural Networks" | by Jitendra R Rathod | Medium*, n.d.)

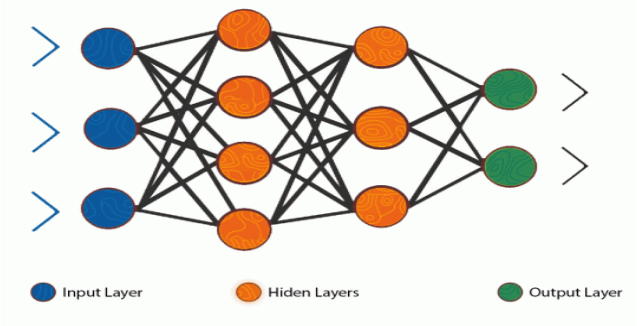

Figure 2.2: Artificial Neural Network Architecture

ANN comprises of the following layers - Input, Hidden and Output. The input layer receives the input, the hidden layer processes the input data, and the output layer produces the result. Essentially, each layer tries to learn particular weights. One recurring difficulty in all these neural networks is the Vanishing and Exploding Gradient. This problem is connected with the backpropagation algorithm. The weights of neural networks are updated by this backpropagation process by identifying the gradients..(*"An Analysis of Deep Learning Neural Networks" | by Jitendra R Rathod | Medium*, n.d.)

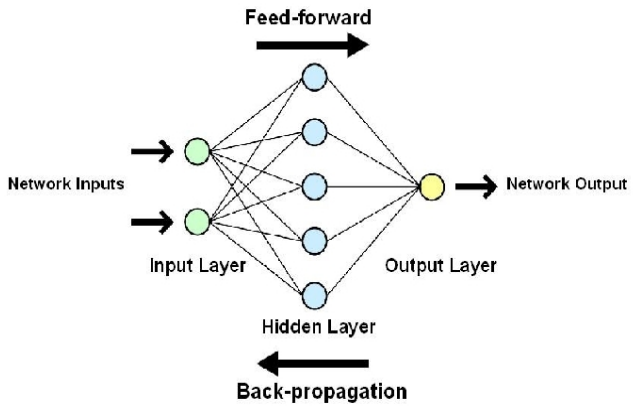

Figure 2. 3: Backpropagation model

2.4.1 Elements of neural networks

A study of several characteristic elements of Neural Networks, whose initialization, optimization and tuning considerably affect the performance of the learning algorithm and model.

Parameters: The model's coefficients are called parameters, and they are chosen by the model itself "It means that the algorithm while learning, optimizes these coefficients (according to a particular optimization strategy) and returns an array of parameters which minimize the error. The only thing you have to do with those parameters is initialize them. (*Neural Networks: Parameters, Hyperparameters and Optimization Strategies | by Valentina Alto | Towards Data Science*, n.d.)

Hyperparameters: These are elements that you need to set differently from the previous ones. Furthermore, the model will not update them according to the optimization strategy.

Some examples of these hyperparameters are :

Learning rate:

$$w_i' = w_i - \gamma \frac{\delta L}{\delta w_i} \tag{2.1}$$

The gamma is our learning rate, and it tells the algorithm how important the impact of the gradient is on the weight. The problem with a small gamma is that the Neural Network will converge (if it converges) very slowly, and we might incur the so-called 'Vanishing Gradient' problem. On the other side, if gamma is very big, the risk is missing the minimum and this scenario is called 'Exploding Gradient'.

Momentum: it is a technique used during the backpropagation phase. It increases the speed of convergence not in terms of learning rate (how much weight is updated each time) but in terms of embedded memory of past re-calibration (the algorithm knows the previous direction of that weight). The suggestion is always to start with low values and then increase them little by little. (*Neural Networks: Parameters, Hyperparameters and Optimization Strategies | by Valentina Alto | Towards Data Science*, n.d.)

Activation function: it is the function through which we pass our weighted sum, to have a significant output, namely as a vector of probability or a 0–1 output. The major activation functions are Sigmoid (for multiclass classification, a variant of this function is used, called SoftMax function: it returns as output a vector of probability whose sum is equal to one), Tanh and RELU.

(*Neural Networks: Parameters, Hyperparameters and Optimization Strategies | by Valentina Alto | Towards Data Science*, n.d.)

Epochs: it represents how many times the algorithm to trains the whole dataset (note that epochs are different from iterations: those latter are the number of batches needed to complete one epoch). The number of epochs depends on the kind of data and task. The best practice is imposing a condition such that epochs stop when the error is close to zero. (*Neural Networks: Parameters, Hyperparameters and Optimization Strategies | by Valentina Alto | Towards Data Science*, n.d.)

Strategies: These are approaches and best practices we might want to have in our algorithm to make it more performing. Among these there are the following:

Data normalization: Normalizing data means converting all of them to the same scale, within the range [0–1]. While data normalization happens before model training, another way you can normalize your data is through the so-called Batch Normalization; it happens directly during model training, specifically after the weighted sum and before the activation function. This was discussed earlier as a method to solve vanishing/exploding gradients.(*Neural Networks: Parameters, Hyperparameters and Optimization Strategies | by Valentina Alto | Towards Data Science*, n.d.)

Optimization algorithm: "Optimization is the process where we train the model iteratively resulting in a maximum and minimum function evaluation. It is one of the most important phenomena in Machine Learning to get better results. Adam optimizer is often the best choice since it allows you to set different hyperparameters and customize your Neural Network.".(Neural Networks: parameters, hyperparameters and optimization strategies | by Valentina Alto | Towards Data Science, no date).

2.5 Literature Review

A large selection of commercial data loggers for climate monitoring is available, however, they are expensive, extremely complicated and cannot be easily maintained. As a result, additional development of data-acquisition systems, it is necessary to gather and process such meteorological data in addition to monitoring the performance of PV systems in operation, under the premise of obtaining the promised parameters using easy-to-handle and low-cost systems.

One of the first systems developed for on-site measurements of PV array characteristics and PV monitoring systems, enabling the collection, analysis and presentation of operational data, was conducted by the authors, where the data acquisition equipment was rather expensive (over 10 percent of the system cost) (over 10 percent of the system cost). Because the prices for data collecting gear have reduced more rapidly than the prices of PV systems, analytical monitoring has been steadily applied to small PV installations.

ANN and its associated models like FFBP (Ahmed and Adam, 2013) have been demonstrated to be popular methods among authors. ANN is a model based on interconnected neurons to create output by processing information found in the input. The relationship between the input and the output is based on the weight of the neurons and the input data for which the neurons have to be trained in the system.

In (Jyosthna et al., 2012) the authors applied FFBP for temperature forecasting and conclude that this model had the proper potential for complex modelling of the relationship between various factors. The author (Alawode, 2013) mentioned that FFBP is the best model of neural network for real-time forecasting due to the least training time and fast response.

Some researchers have used many types of ANN for comparison and analysis of issues related to solar panels. To forecast the power output of PV systems, applications can be grouped into two categories: one is the prediction model based on solar radiation intensity. Firstly, obtain the predicted value of solar radiation based on the solar radiation model, then use the PV output formula to calculate the power output of the PV system (Lorenz et al., 2009). Another way is to estimate the power production of the PV system directly. (Cai et al., 2010). Although the prediction model based on solar radiation intensity has been considered to be an effective method in practical application, it takes a lot of meteorological and geographical data to solve complex differential equations (Yona et al., 2007).

CHAPTER THREE

3.0 METHODOLOGY

3.1 Introduction

In this chapter, the various activities involved in implementing the techniques reviewed in the previous chapter are described.

3.1.1 Project activities

- Mobile Application Development
- Data Analysis: The data required to build the models were obtained from the real-time operation of Solar Inverters in Ado-Ekiti using the Data Logger built in an earlier phase.
- Prediction Model development: An Artificial Neural Network model was designed and implemented to predict the performance of the PV system.
- Assessment of the Performance of the Prediction model: The performance of the model was assessed using several performance metrics such as MAE, MAPE etc.

3.2 Mobile Application Development

To ease the process of retrieving the data stored on the data logger, an android application was developed called the **"GreenHouse Logger".** The mobile application development process was done using Xamarin.Forms. Xamarin.Forms is a feature of Xamarin, the popular mobile development framework that extends the .NET developer platform with tools and libraries for building mobile apps. Xamarin.Forms uses MVVM. MVVM is a design technique employed to divide the user-interface (view), data (model), and application logic (view model).

3.2.1 Deployment and testing

To debug the mobile application we configured Visual Studio to use a virtual machine called an android emulator to run the mobile application.

3.2.1.1 Configuration of android emulator in visual studio code 2019

To configure a new android emulator for testing and the debugging , the following steps were followed. I clicked on the "**Select Device Manager**" in the menu bar of VS code. I was prompted to verify If I would like to create an emulator. I then selected "**Add a Virtual Device**"; This opens the Android Device Manager. Select + **New Device** to start the creation process. The options are automatically populated for a base emulator. If required, change any options and then selected "**Create**".

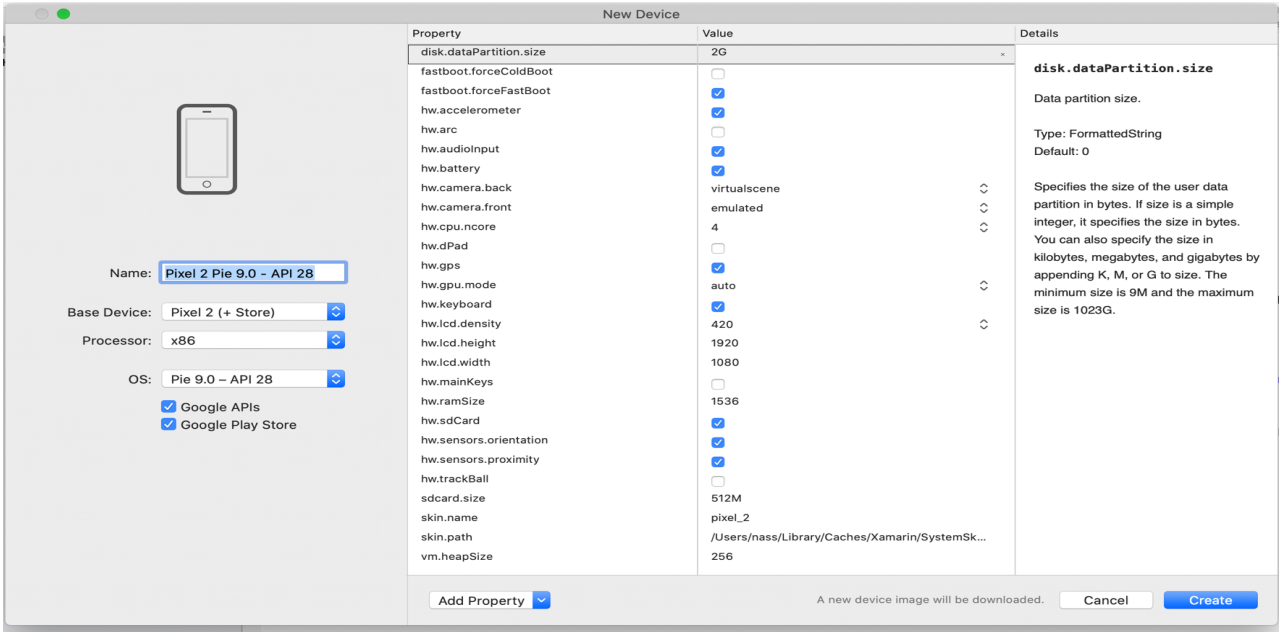

Figure 3. 1:Screenshot illustrating the emulator creation process

3.2.1.2 Deploying and testing the app

Once the emulator has been created, I clicked "**Play**" and the Android emulator launched. From the menu, I selected **Run** > **Start Debugging**. The application was then built, deployed, and ran

on the emulator.

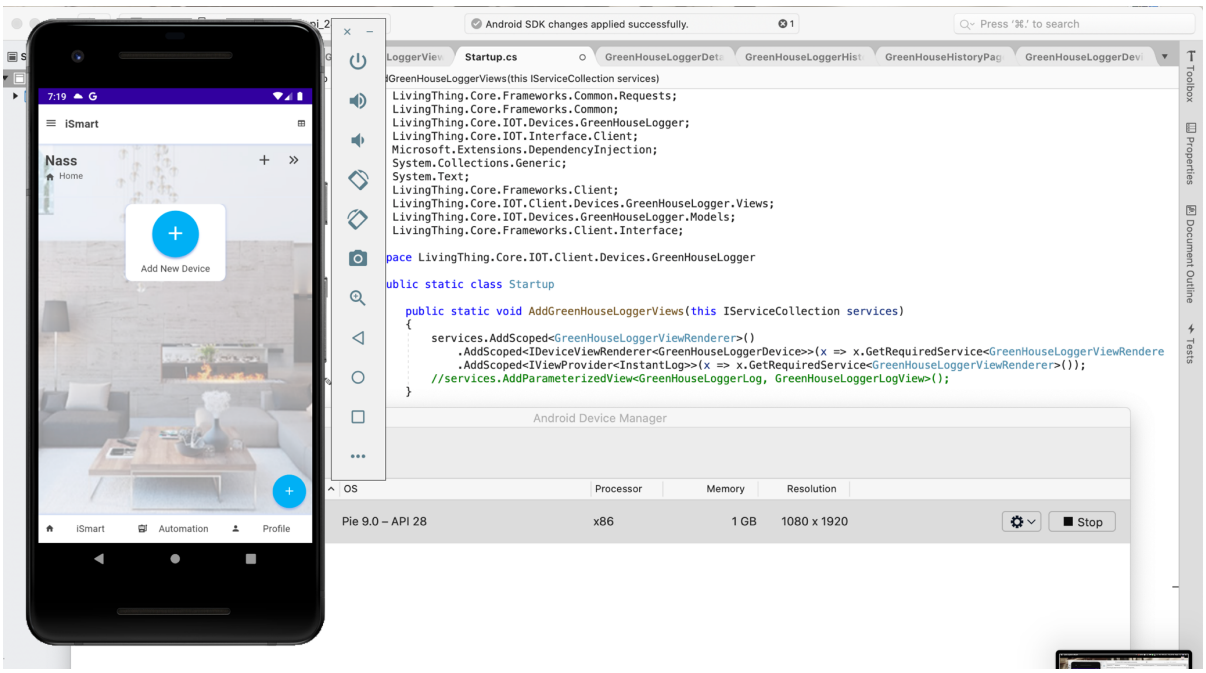

Figure 3. 2: Screenshot illustrating the debugging process

3.3 Data Analysis

The data logger that was designed and built by my project partner after being synchronized with the mobile application, was used to collect data from three homes with Solar Inverter installation in Ado-Ekiti. A total of ***1770*** samples of data were collected from three homes between April 1st and April 20th, 2021. The PV systems in two of those homes had similar specifications to the inverter we proposed to develop, hence only the data from these houses were used for training the model. The components of the three PV systems and their specifications are listed in Table 3.1 below.

Table 3. 1:The components of the three PV systems and their specification

House	Solar Panel Rated Power	Solar Panel Type	Battery Capacity	Inverter Rating	Inverter Voltage	Inverter Type
1.	150 W	Monocrystalline	150 Ah	2.5KVA	12 V	PSW
2.	150 W	Monocrystalline	100 Ah	2.5KVA	12 V	PSW
3.	100 W	Monocrystalline	40 Ah	1.5KVA	12 V	PSW

3.3.1 Data cleaning and pre-processing

Data cleaning is the process of detecting and correcting corrupt or inaccurate records from a recordset, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data. New columns containing the output power, input power, and power efficiency were added, whereas columns containing data not relevant to the model were removed. The Data cleaning process for this project was done using the pandas library in python .

Pre-processing data is also known as normalization, it makes the training process less sensitive to the scale of the features. This results in getting better coefficients after training, the process of making features more suitable for training by rescaling is called **feature scaling**.(*2 Easy Ways to Normalize Data in Python - JournalDev*, n.d.) The normalization for this project was done before the training process using the sci-kit-learn-preprocessing module.

3.4 Prediction Model Development

For this study, an ANN-based technique for prediction the power production efficiency of PV systems was built. We employed a feed-forward backpropagation (FFBP) neural network to

estimate the power output of the PV system directly by reference to the historical data of the PV system.

3.4.1 Creating the network

Creating a network involves defining network parameters , such as the number of neurons for the hidden layer, transfer function, training function, weight and bias learning function, and performance functions. During the training phase, parameters such the number of training epochs, learning rate, and the number of hidden layers could be modified depending on the error performance of the ANN output.

3.4.2 Flowchart and architecture of the model

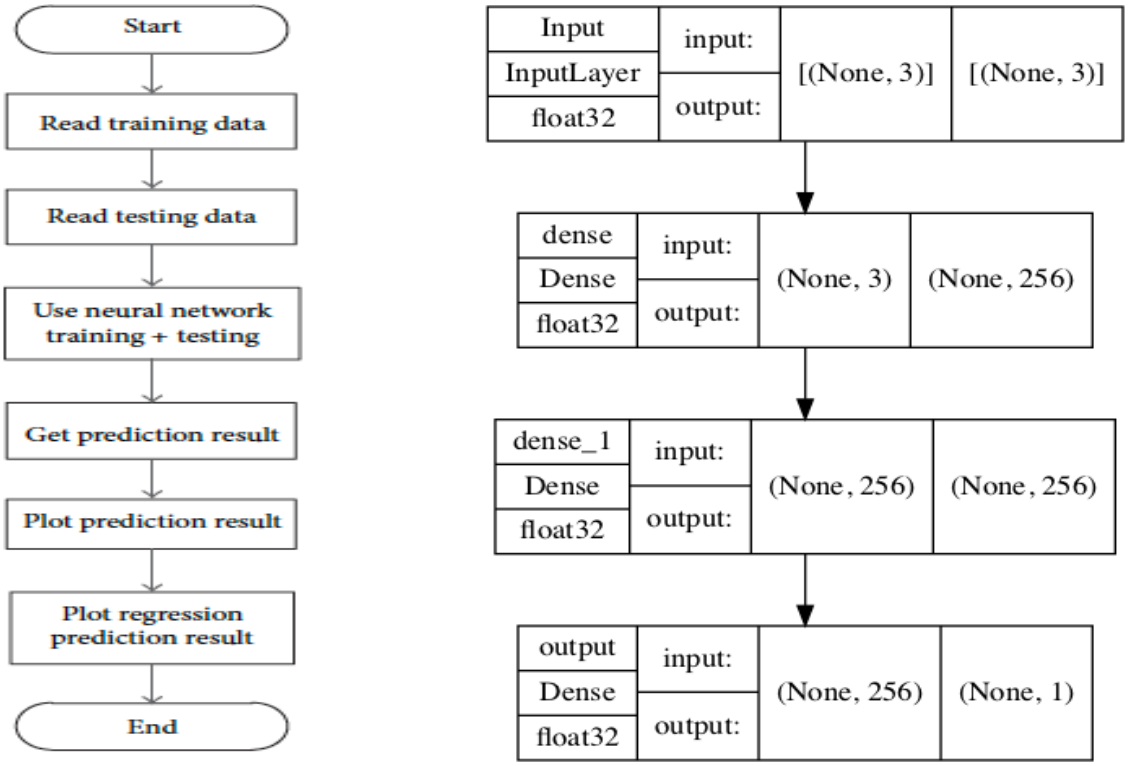

Figure 3.3: a). Flowchart of the ANN Model. b) Architecture of the ANN Model.

3.4.2.1 Input layer

From observations, the power output or production of the PV system is zero from 18:45-to 05:59, hence the data recorded for the study period was trimmed to data samples from 07:30 to 17:30. The parameters selected for the Input Layer are the Solar Power, Battery Power and Load Power. Only the first 15 rows were selected, these rows are displayed in Table 3.2 below.

Table 3. 2:Input Data for Prediction Model

NO	Solar Power	Load Power	Battery Power
1.	13.8539557	39.9821701	27.4842717
2.	14.7519102	40.8178368	27.5015241
3.	15.171035	41.1846886	27.4952736
4.	17.5626571	37.4239883	23.7138299
5.	4.09354767	43.1012188	39.4104065
6.	18.6095287	42.4564195	25.6173049
7.	19.6624259	42.6975097	24.8718902
8.	19.5048193	41.958136	24.2259722
9.	18.8551378	31.9870434	14.8446021
10.	19.0775554	40.3541635	23.0242962

3.4.2.2 Hidden layer

The network has 2 hidden layers with 256 hidden neurons in the feed-forward neural network. A trial-and-error strategy was utilized to identify the proper number of hidden neurons for this model. The first hidden layer, the **dense_1** consists of 256 neurons and the 'relu' activation function. The second hidden layer, **dense_2** consists of 512 neurons and the 'relu' activation function.

3.4.2.3 Output layer

The output layer has just one parameter, which is the power efficiency. Only the first 15 rows of the training data were included in Table 3.3 below. The output layer contains the training data for the parameter we are trying to predict.

Table 3. 3: Expected Output Data

NO	Power Efficiency
1.	92.117036
2.	96.3517894
3.	97.3643506
4.	97.1373593
5.	94.9177752
6.	95.9129313
7.	92.3611008
8.	96.8140107
9.	99.2176453
10.	99.3245579

3.4.3 Training the network

After data normalization, the next step was to train the input data using the Adam optimizer. The Adaptive Moment Estimation (Adam) Adam optimizer involves a combination of two gradient descent methodologies:

- Momentum: This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients.
- Root Mean Square Propagation (RMSP): Root means square prop or RMSprop, it is an adaptive learning algorithm that takes the 'exponential moving average'.(*Intuition of Adam Optimizer - GeeksforGeeks*, n.d.)

Before the training process, the weight and bias values were set into initial values. Then 702 samples of data will be randomly divided by the model's training algorithm into training, testing and validation data. The algorithm selected the data randomly from the data set during the training process, so every instance of the training process will get a varied MSE value that depends on 80% of the input data chosen. The code illustrated in Figure 3.6.3 below was used to Initiate the Network Training Process.

Fitting/Training the Network

In [19]:
```
history = model.fit(
    train_features,
    train_labels,
    epochs=1000,
    validation_data=(test_features,
                     test_labels)
)
```

Figure 3. 4: Screenshot illustrating the code to initiate network training process

3.5 Assessing the Performance of the Prediction model

Then, the next stage was analyzing the performance of the constructed ANN Model. To analyze the ANN model employed in this research, we took use of numerous error statistics. The mean squared error (MSE) informs you how near a regression line is to a set of points. MSE is determined by the sum of the square of prediction error which is real output minus predicted output and then divide by the number of data points. It offers you an absolute number on how much your expected outcomes depart from the actual amount. You cannot understand numerous insights from one single result but it gives you an actual number to compare against other model results and help you select the best regression model. In the context of machine learning, mean absolute error (MAE), refers to the extent of difference between the prediction of observation and the true value of that observation.(*3 Best Metrics to Evaluate Regression Model? | by Songhao Wu | Towards Data Science*, n.d.)

R Square is derived by the sum of the squared prediction error divided by the square's entire sum, which substitutes the calculated forecast with the mean. R Square value is between 0 to 1 and a greater value implies a better fit between prediction and actual value. R Square is better suited to communicate the model to other people because you can explain the number as a proportion of the output variability. (*3 Best Metrics to Evaluate Regression Model? | by Songhao Wu | Towards Data Science*, n.d.)

All the errors formula expressed in a percentage as defined as follow:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.1}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.2}$$

$$R^2 = 1 - \frac{SS_{Regression}}{SS_{Total}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{3.3}$$

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}} \tag{3.4}$$

CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the results of the study are presented and discussed with reference to the aim and objectives of the project, which was the design and implementation of a smart data logger and prediction of the power efficiency of a PV system. With regards to my contribution to this group project, the objectives we would be discussing are

- The design and development of the mobile application.
- Prediction model development.

These sections were described in the previous chapter that presented the methodology used.

4.2 Mobile Application

In summary, the proposed IoT application can continuously supervise and maintain the PV system in a safe state and high working performance. The crucial data, such as related PV panel and battery system voltages, current, state of charge of batteries, etc., are logged and transmitted from the data logger synchronised with the mobile application which is connected through WiFi.

The data may be presented on the main dashboard as the device's state as shown in figure 4.1, showing several parameters or it may be viewed as charts or as a table as shown in the figure 4.2 below. This allows the user to easily choose what kind of data , which period and the format he/she wishes to see. The screenshot in figure 4.1 also shows the commands that can be executed on the data logger device through the mobile application.

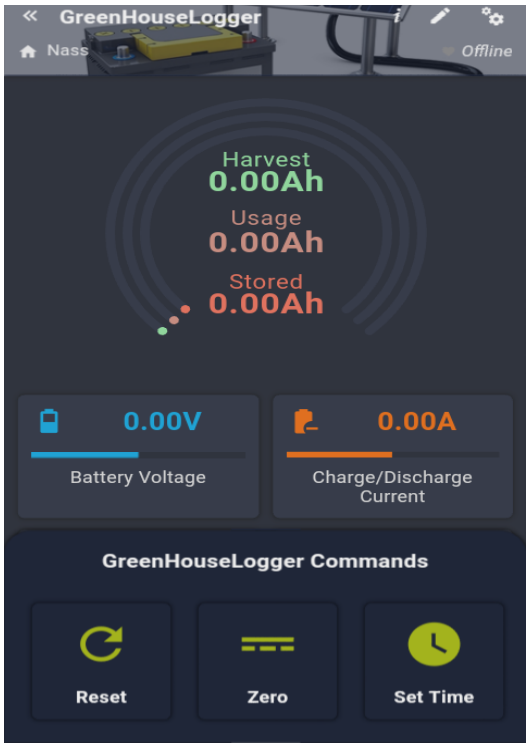

Figure 4. 1: Screenshot of Greenhouse Data Logger Dashboard

Figure 4. 2: Screenshot of Greenhouse Data Logger Data View Page

As shown in figures 4.1 to 4.3, the mobile application is able to display different views and information of the data logger device. Observing the screenshot in Figure 4.3 below, it illustrates the data export view, from which the data that has been collected and logged can be exported as a CSV file. A CSV (comma-separated values) file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

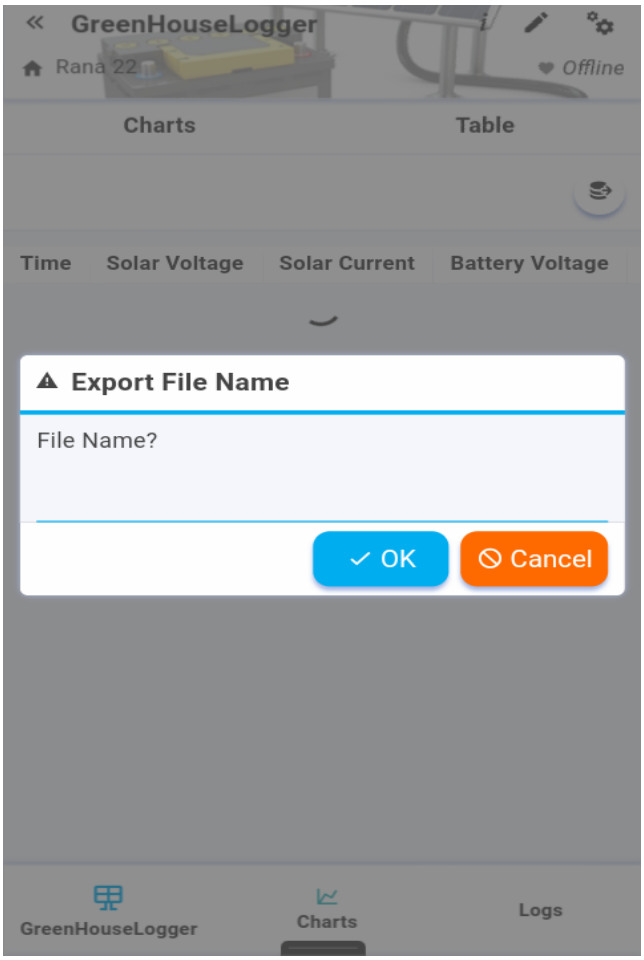

Figure 4. 3: Screenshot of Greenhouse Data Logger Data Export Page

4.3 Connecting the Mobile Application to the Data Logger.

After installing the application on the mobile phone, the phone is connected to the data logger's WiFi. This connection enables the mobile application to display a visualized format of the data recorded by the data logger hardware without a wired connection. The results show a mobile application was successfully developed to perform the desired function, figure 4.4 below illustrates the dashboard view of the data logger device. The dashboard view shows information about the PV system's Total Harvested battery power , the amount of which has been used and that being stored. It also gives the voltage , current and charging/discharge current of the solar battery , solar module and inverter respectively.

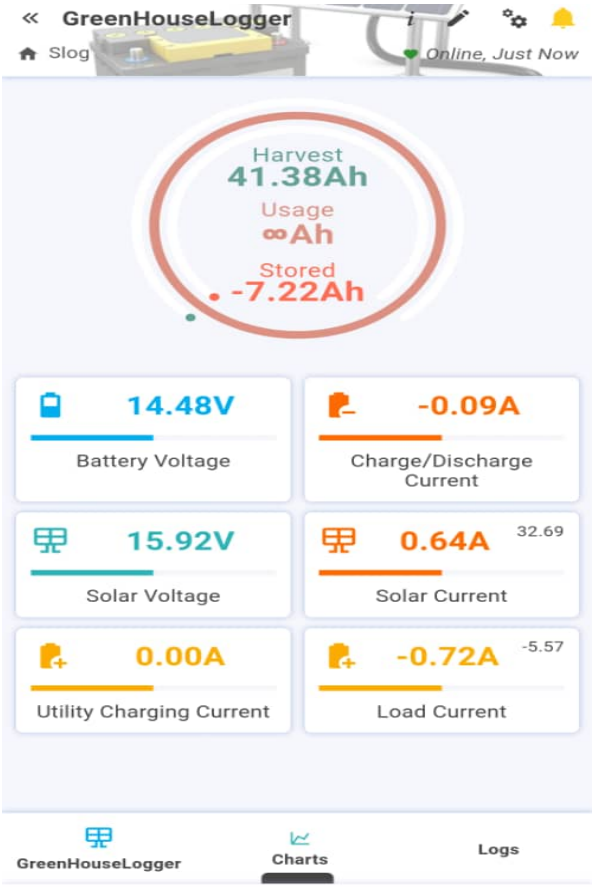

Figure 4.4: Mobile Application Interface After Integration with Solar Inverter

4.4 Prediction Model

When assessing the performance of the model, we produced a regression plot of the actual and predicted values, with the R-squared score of the model as illustrated in Figure 4.5. We observed that, the data points are concentrated on a straight line. While the best fit for testing was approaching 1 means a positive linear relationship . It indicates that our model is performing well. From my research on prediction analysis, stronger relationships produce a tighter clustering of data points.

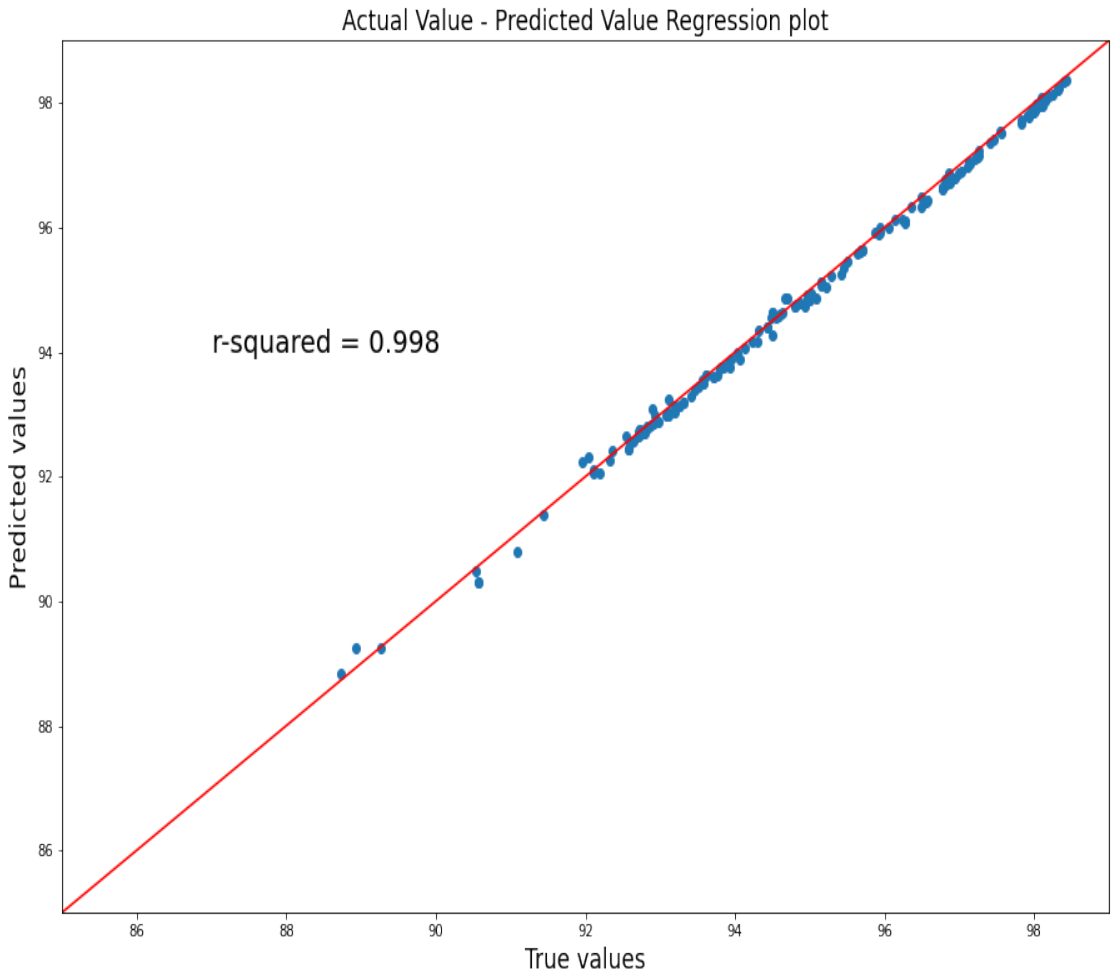

Figure 4. 5: A regression plot of the actual and predicted values

The graph in Figure 4.6 illustrates the training and validation loss function trend. From the graph, we notice that the model's loss stabilizes after the first 30 epochs.This is also another indication that the model's performance was very good.

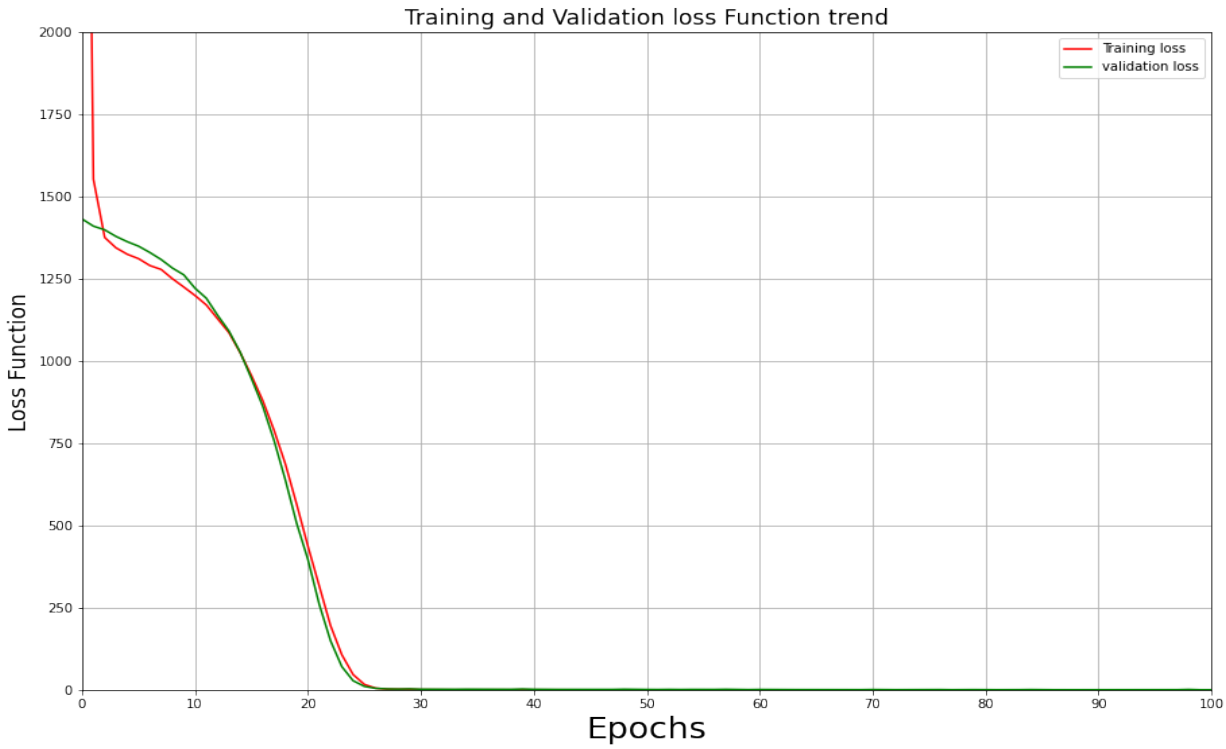

Figure 4.6: Loss Function Trend

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

The closing remarks on the study are presented in this chapter, giving an overview and conclusion of the research work. My input into this study is examined in terms of both its contribution to the field ,achievements and its limitations. Recommendations are made with regards to aspects that could be explored in future research, especially after considering the limitation of the study.

5.2 Conclusion

This project explored the tools and techniques used in the monitoring systems of existing PV systems. A prototype was developed by my project partner to demonstrate a system for monitoring PV systems to provide real-time data to the consumer. The system design features an out-of-the-box hardware setup and easy IoT synchronization, this flexibility makes the system more suitable for applications in remote areas of developing countries.

From the results , my main achievement with regards to my contribution to the project was the successful deployment of the mobile application and the accuracy of the developed predictive model of this project. Accurate power planning requires power prediction for PV modules. In this study, the power efficiency of solar panels was estimated using mathematical formulas and then the FFBP network was trained with the estimated historical data (as output data). The performance of the ANN model was evaluated using MSE, MAE, R-squared score and Pearson correlation coefficient which presented the accuracy and efficiency of the system itself. The trained model can be said to have shown excellent performance.

5.3 Limitations of Project

This project has some significant success claims and some recognized limitations such as the issues of the mobile application is only available for Android phones; this limitation exists because iOS devices do not allow the installation of applications that are not available on the app store.

5.4 Recommendation

Observations from this project shed light on how key parameters could affect the efficiency of the forecasts made by a prediction model, it is to be taken into consideration the number of data samples used in training the model and the weather conditions in which the samples were taken. In that case, it is recommended that the data logger system be modified to capture solar irradiance and climate data. With this supplementary data, the model can be trained to accurately predict the power efficiency of the PV system on days with different weather conditions. We also suggest that tests for inverter efficiency should be conducted with properly sized arrays to ensure accurate computations.

REFERENCES

2 Easy Ways to Normalize data in Python - JournalDev. (n.d.). Retrieved March 6, 2022, from https://www.journaldev.com/45109/normalize-data-in-python

3 Best metrics to evaluate Regression Model? | by Songhao Wu | Towards Data Science. (n.d.). Retrieved May 7, 2022, from https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b

Alawode, K. (2013). *A Comparison of Neural Network Models for Load Forecasting in Nigerian Power System Group Consensus Algorithms for Multi-agent Systems View project NSGA-III Reference Based approach for solving Multi-Objective Economic/Emission and Transmission Loss optimization. View project*. https://www.researchgate.net/publication/273575221

"An Analysis of Deep Learning Neural Networks" | by Jitendra R Rathod | Medium. (n.d.). Retrieved April 21, 2022, from https://medium.com/@jiterockzz/an-analysis-of-deep-learning-neural-networks-e9e3e038426a

Cai, T., Duan, S., & Chen, C. (2010). Forecasting power output for grid-connected photovoltaic power system without using solar radiation measurement. *2nd International Symposium on Power Electronics for Distributed Generation Systems, PEDG 2010*, 773–777. https://doi.org/10.1109/PEDG.2010.5545754

Cheddadi, Y., Cheddadi, H., Cheddadi, F., Errahimi, F., & Es-sbai, N. (2020). Design and implementation of an intelligent low-cost IoT solution for energy monitoring of photovoltaic stations. *SN Applied Sciences*, *2*(7). https://doi.org/10.1007/s42452-020-2997-4

Internet of things - Wikipedia. (n.d.). Retrieved March 14, 2022, from https://en.wikipedia.org/wiki/Internet_of_things

Intuition of Adam Optimizer - GeeksforGeeks. (n.d.). Retrieved March 14, 2022, from https://www.geeksforgeeks.org/intuition-of-adam-optimizer/

Jeffrey, A., Igogori, E., & Obidike, I. v. (2016). THE IMPLEMENTATION OF A 5KVA PURE SINE WAVE INVERTER. *THE IMPLEMENTATION OF A 5KVA PURE SINE WAVE INVERTER*. https://www.academia.edu/38910204/THE_IMPLEMENTATION_OF_A_5KVA_PURE_SINE_WAVE_INVERTER

Jyosthna, C., #1, D., Syam, B., Reddy, P., Vagdhan, K., #3, K., Musala, B., #4, R., Raja, N., & #5, N. (2012). ANN Approach for Weather Prediction using Back Propagation. *International Journal of Engineering Trends and Technology*. http://www.internationaljournalssrg.org

Learning Electrical Engineering. (n.d.). Retrieved November 26, 2021, from https://www.electricalengineeringtoolbox.com/2019/11/types-of-solar-pv-power-supply-systems.html/

Lorenz, E., Hurka, J., Heinemann, D., & Beyer, H. G. (2009). Irradiance Forecasting for the Power Prediction of Grid-Connected Photovoltaic Systems. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *2*(1), 2–10. https://doi.org/10.1109/JSTARS.2009.2020300

Madeti, S. R., & Singh, S. N. (2017). Monitoring system for photovoltaic plants: A review. *Renewable and Sustainable Energy Reviews*, *67*, 1180–1207. https://doi.org/10.1016/J.RSER.2016.09.088

Mobile app - Wikipedia. (n.d.). Retrieved March 19, 2022, from https://en.wikipedia.org/wiki/Mobile_app

Neural Networks: parameters, hyperparameters and optimization strategies | by Valentina Alto | Towards Data Science. (n.d.). Retrieved May 5, 2022, from https://towardsdatascience.com/neural-networks-parameters-hyperparameters-and-optimization-strategies-3f0842fac0a5

Tamilselvi, K., Jananandhini, E., & Vijayakumar, N. (2019). Design and Implementation of IOTEnabledSmart Solar Power Monitoring System. In *ijartet.com International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)* (Vol. 6). www.ijartet.com

What is a Mobile Application? - Definition from Techopedia. (n.d.). Retrieved March 19, 2022, from https://www.techopedia.com/definition/2953/mobile-application-mobile-app

What is Application Software? - Definition, Examples & Types - Video & Lesson Transcript | Study.com. (n.d.). Retrieved May 19, 2022, from https://study.com/academy/lesson/what-is-application-software-definition-examples-types.html

What Is The Internet of Things? A Complete Guide To IoT. | Built In. (n.d.). Retrieved March 4, 2022, from https://builtin.com/internet-things

Yona, A., Senjyu, T., Saber, A. Y., Funabashi, T., Sekine, H., & Kim, C. H. (2007). Application of neural network to one-day-ahead 24 hours generating power forecasting for photovoltaic system. *2007 International Conference on Intelligent Systems Applications to Power Systems, ISAP*. https://doi.org/10.1109/ISAP.2007.4441657

APPENDIX A

Source Code for Mobile Application

Greenhouse Logger Startup

```
using LivingThing.Core.Frameworks.Common.Requests;
using LivingThing.Core.Frameworks.Common;
using LivingThing.Core.IOT.Devices.GreenHouseLogger;
using LivingThing.Core.IOT.Interface.Client;
using Microsoft.Extensions.DependencyInjection;
using System.Collections.Generic;
using System.Text;
using LivingThing.Core.Frameworks.Client;
using LivingThing.Core.IOT.Client.Devices.GreenHouseLogger.Views;
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Models;
using LivingThing.Core.Frameworks.Client.Interface;

namespace LivingThing.Core.IOT.Client.Devices.GreenHouseLogger
{
   public static class Startup
   {
      public static void AddGreenHouseLoggerViews(this IServiceCollection services)
      {
         services.AddScoped<GreenHouseLoggerViewRenderer>()
            .AddScoped<IDeviceViewRenderer<GreenHouseLoggerDevice>>(x =>
x.GetRequiredService<GreenHouseLoggerViewRenderer>())
            .AddScoped<IViewProvider<InstantLog>>(x =>
x.GetRequiredService<GreenHouseLoggerViewRenderer>());
         //services.AddParameterizedView<GreenHouseLoggerLog,
GreenHouseLoggerLogView>();
      }
   }
}
```

Greenhouse Logger Renderer

```
using LivingThing.Core.Frameworks.Client;
using LivingThing.Core.Frameworks.Client.Interface;
using LivingThing.Core.IOT.Devices.GreenHouseLogger;
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Models;
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Settings;
using LivingThing.Core.IOT.Interface.Client;
using LivingThing.Core.IOT.Client.Devices.GreenHouseLogger.ViewModels;
using LivingThing.Core.IOT.Client.Devices.GreenHouseLogger.Views;
using LivingThing.Core.IOT.Client.Devices.ViewModels;
```

```
using LivingThing.Core.IOT.Client.ViewModels;
using Microsoft.AspNetCore.Components;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;

namespace LivingThing.Core.IOT.Client.Devices.GreenHouseLogger
{
   public class
GreenHouseLoggerViewRenderer:DeviceViewRenderer<GreenHouseLoggerDevice,
GreenHouseLoggerSettings, GreenHouseLoggerHardwareSettings,
GreenHouseLoggerReceivedDataModel, GreenHouseLoggerControlModel>,
IViewProvider<InstantLog>
   {
      public GreenHouseLoggerViewRenderer(AppMainPageViewModel root) : base(root)
      {
      }

      public override RenderFragment GetView(IUIService ui, GreenHouseLoggerDevice device,
DeviceViewType type, Func<int, Task> clicked = null, Func<Task> pressed = null)
      {
         if (type == DeviceViewType.Details)
         {
            return FragmentView.From<GreenHouseLoggerDeviceDetails,
GreenHouseLoggerDetailsViewModel>(new GreenHouseLoggerDetailsViewModel(Root,
device.OwnerScope, device), true);
         }
         return base.GetView(ui, device, type, clicked, pressed);
      }

      public RenderFragment GetView(ContainerContext<InstantLog> value)
      {
         return FragmentView.From<GreenHouseInstantLog>(new Dictionary<String, object>()
         {
            [nameof(GreenHouseInstantLog.ViewModel)] = value.Model
         });
      }
   }
}
```

Greenhouse Logger Details View Model

```
using LivingThing.Core.Frameworks.Client.Interface;
using LivingThing.Core.Frameworks.Common.Data;
using LivingThing.Core.Frameworks.Common.String;
using LivingThing.Core.IOT.Devices.GreenHouseLogger;
```

```
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Models;
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Settings;
using LivingThing.Core.IOT.Common.Interface;
using LivingThing.Core.IOT.Client.Devices;
using LivingThing.Core.IOT.Client.Devices.ViewModels;
using System;
using System.Collections.Generic;
using System.Text;
using LivingThing.Core.IOT.Client.ViewModels;

namespace LivingThing.Core.IOT.Client.Devices.GreenHouseLogger.ViewModels
{  public class GreenHouseLoggerDetailsViewModel :
DeviceDetailsViewModel<GreenHouseLoggerDevice, GreenHouseLoggerSettings,
GreenHouseLoggerHardwareSettings, GreenHouseLoggerReceivedDataModel,
GreenHouseLoggerControlModel>, IChartable<GreenHouseLoggerReceivedDataModel>
   {
      public GreenHouseLoggerDetailsViewModel(AppMainPageViewModel root,
IIOTDeviceScope scope, GreenHouseLoggerDevice device) : base(root, scope, device)
      {
      }

      public IEnumerable<IDataPointDescriptor> Charts => new DataPointDescriptor[]
      {
         new DataPointDescriptor
         {
            Name = nameof(Device.Status.Status.SolarVoltage),
            Label = nameof(Device.Status.Status.SolarVoltage).CamelCaseToTitle()
         },
         new DataPointDescriptor
         {
            Name = nameof(Device.Status.Status.SolarCurrent),
            Label = nameof(Device.Status.Status.SolarCurrent).CamelCaseToTitle()
         },
         new DataPointDescriptor
         {
            Name = nameof(Device.Status.Status.BatteryVoltage),
            Label = nameof(Device.Status.Status.BatteryVoltage).CamelCaseToTitle()
         },
         new DataPointDescriptor
         {
            Name = nameof(Device.Status.Status.InverterChargeCurrent),
            Label = nameof(Device.Status.Status.InverterChargeCurrent).CamelCaseToTitle()
         },
         new DataPointDescriptor
         {
            Name = nameof(Device.Status.Status.UsageCurrent),
```

```
            Label = nameof(Device.Status.Status.UsageCurrent).CamelCaseToTitle()
        }
    };
  }
}
```

Greenhouse Logger History View Model

```
using LivingThing.Core.Frameworks.Client.Interface;
using LivingThing.Core.IOT.Devices.GreenHouseLogger;
using LivingThing.Core.IOT.Devices.GreenHouseLogger.Models;
using LivingThing.Core.IOT.Common.Interface;
using LivingThing.Core.IOT.Client.ViewModels;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Text;
using System.Threading.Tasks;

namespace LivingThing.Core.IOT.Client.Devices.GreenHouseLogger.ViewModels
{
  public class
GreenHouseLoggerHistoryViewModel:DeviceViewModel<GreenHouseLoggerDevice>
  {
    public GreenHouseLoggerHistoryViewModel(AppMainPageViewModel root,
IIOTDeviceScope scope, GreenHouseLoggerDevice device) : base(root, scope, device)  {
    }
  }
}
```

APPENDIX B

Python Code For ANN Model

Import Libraries

```
In [1]:
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from datetime import datetime
```

Reading the Dataset

In [2]:

```
#code to load the dataset
dataset = pd.read_csv("cleaneddata.csv")
df = dataset
```

In [3]:

```
#displays first 5 rows of the dataset
dataset.head()
```

Cleaning Dataset Before Using It For Model Development

In [4]:

```
# To get the list of columns and their data types
df.dtypes
```

In [5]:

```
# Describes the dataset based on statistical indices
df.describe()
```

In [6]:

```
#Renaming the Columns to include an underscore instead of space


df.rename(columns={' Battery Voltage':'Battery_Voltage',' Solar Voltage':'Solar_Voltage',' Solar Current':'Solar_Current', 'Date Created':'Date_Created',' Usage Current':'Usage_Current','Unnamed: 11':'Battery_Current','Unnamed: 12': 'Solar_Power' ,' Date Extended':'Date_Extended' }, inplace=True)
df.columns
```

In [7]:

```
# Creating new columns for caclulated parameters
df['Battery_Current']=  df['Solar_Current'] + (df['Usage_Current']/2)
df['Solar_Power'] = df['Solar_Voltage'] * df['Solar_Current']
df['load_Power'] = df['Battery_Voltage'] * df['Usage_Current']/2 * -1
```

```
df['Battery_Power'] = df['Battery_Voltage'] * df['Battery_Current'] * -1
df['Total_Dc_Power'] = df['Solar_Power'] + df['Battery_Power']
df['Efficiency'] = (df['load_Power']/df['Total_Dc_Power']) * 100
```

In [9]:

```
# Converting the Date Created column from an Object data type to DateTime
df['Date_Created'] = pd.to_datetime(df['Date_Created'])
df['Date_Created']
df.head()
```

In [10]:

```
#Select rows within a specific time in this case between 7:30am and 17:45
df[(df['Date_Created'] > '07:30:00') & (df['Date_Created'] < '17:40:00')]
df.tail()
```

In [11]:

```
# Change order of columns

df=df[['Date_Created',' Inverter Charge Current', 'Usage_Current', ' Solar Harvest',' Battery Charge', ' Inverter Charge', ' Usage Charge', 'Date_Extended','Solar_Voltage', 'Solar_Current', 'Battery_Voltage','Battery_Current', 'Total_Dc_Power','Solar_Power','load_Power','Battery_Power', 'Efficiency']]
df.head()
```

In [12]:

```
# trim the dataset to contain only the parameters that are going in the input layer of the model
model_data = df.iloc[:,13:17]
# exports the content of train to a csv file
model_data.to_csv(r'modeldata.csv', index=False)
model_data
```

Creating The Network

In [13]:

```
from sklearn import preprocessing

from sklearn.model_selection import train_test_split

import seaborn as sns

import tensorflow as tf
```

In [14]:

```
from tensorflow.keras.layers import Dense, Input

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam
```

In [15]:

```
input_data = Input(shape=(3), name='Input')

dense1 = Dense(256, activation=tf.nn.relu)(input_data)

dense2 = Dense(256, activation=tf.nn.relu)(dense1)

output = Dense(1, name='output')(dense2)
model = Model(input_data, output)

model.compile(optimizer=Adam(learning_rate=0.001), loss='mse', metrics=['mae', 'mse', 'mape'] )

model.summary()
```

```
Model: "model"
```

In [16]:

```
tf.keras.utils.plot_model(

    model, to_file='model.png', show_shapes=True, show_dtype=True,

    show_layer_names=True, rankdir='TB', expand_nested=False, dpi=96

)
```

Data Normalization

In [17]:

```
def separate(df):
    return df[['Efficiency']].to_numpy(), df.drop(df[['Efficiency']], axis=1).to_numpy()
```

In [18]:

```
min_max_scaler = preprocessing.MinMaxScaler()
df_normed = pd.DataFrame(min_max_scaler.fit_transform(model_data))
df_normed.columns = list(model_data.columns)
train_set, test_set = train_test_split(model_data)
train_labels, train_features = separate(train_set)
test_labels, test_features = separate(test_set)
```

Fitting/Training the Network

In [19]:

```
history = model.fit(
    train_features,
    train_labels,
    epochs=1000,
    validation_data=(test_features,
                     test_labels)
)
```

Model Performance Evaluation

In [20]:

```
import numpy as np
from sklearn.metrics import r2_score
from sklearn.metrics import average_precision_score
from scipy.stats import pearsonr
for i in range(np.shape(train_labels)[1]):
 metrics= {
    'mae-train': np.mean(np.abs(train_labels[:, i] - model.predict(train_features)[:, i])),
    'mse-train': np.mean(np.square(train_labels[:, i] - model.predict(train_features)[:, i])),
    'r2-train': r2_score(train_labels[:, i], model.predict(train_features)[:, i]),
    'pearson-train': pearsonr(train_labels[:, i], model.predict(train_features)[:, i])[0],
    'mae-test': np.mean(np.abs(test_labels[:, i] - model.predict(test_features)[:, i])),
    'mse-test': np.mean(np.square(test_labels[:, i] - model.predict(test_features)[:, i])),
    'r2-test': r2_score(test_labels[:, i] ,model.predict(test_features)[:, i]),
    'pearson-test': pearsonr(test_labels[:, i], model.predict(test_features)[:, i])[0]
 }
 blue = lambda x: '\033[94m' + x + '\033[0m'
 yellow = lambda x: '\033[93m' + x + '\033[0m'
 for key in metrics:
    if 'train' in key:
        print(f'Y{i} - {blue(key)} - {str(metrics[key])[:7]}')
    else:
print(f'Y{i} - {yellow(key)} - {str(metrics[key])[:7]}')
```

Data Visualisation

In [22]:

```
import NumPy as np
import matplotlib.pyplot as plt
```

In [23]:

```
import matplotlib.ticker as ticker
loss = history.history['loss']
val_loss = history.history['val_loss']
fig, ax = plt.subplots(figsize=(15, 10))
plt.plot(loss ,'r', label='Training loss')
plt.plot(val_loss , 'g', label='validation loss')
plt.ylim([0, 2000])
plt.grid('both')
plt.xlabel('Epochs',fontsize=25)
plt.ylabel('Loss Function',fontsize=17)
plt.title('Training and Validation loss Function trend',fontsize=17)
plt.legend()
tick_spacing = 10
ax.xaxis.set_major_locator(ticker.MultipleLocator(tick_spacing))
plt.xlim([0, 100])
plt.show()
fig.savefig("Graphs/lossfunction.png")
```

In [24]:

```
fig,ax = plt.subplots(figsize=(15,10))

true_value = (test_labels)

predicted_value = model.predict(test_features)

plt.plot(true_value)

plt.plot(predicted_value)

plt.legend(labels=['true_value','predicted_value'],loc='lower right',fontsize=15)

plt.title('Actual Value - Predicted Value Comparison',fontsize=17)

fig.tight_layout(pad=3.0)

fig.savefig("Graphs/predictioncomparison.png")
```

In [25]:

```
y_pred = model.predict(test_features).flatten()

fig,ax = plt.subplots(figsize=(15,10))

ax.text(87,94, "r-squared = {:.3f}".format(r2_score(test_labels ,y_pred)), fontsize=20)

plt.scatter(test_labels ,y_pred)

plt.xlabel('True values')

plt.ylabel('Predicted values')

plt.title('Actual Value - Predicted Value Regression plot' ,fontsize=17)

plt.xlim([85, 99])

plt.ylim([85, 99])

plt.plot([85, 99], [85, 99],'red')

plt.show()

fig.savefig("Graphs/scatter.png")
```