

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

BY

JASON LIM JING WEI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

JANUARY 2019

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in University
Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

Address:

Date: _____

(Supervisor's signature)

Supervisor's name

Date: _____

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

BY

JASON LIM JING WEI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

JANUARY 2019

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

DECLARATION OF ORIGINALITY

I declare that this report entitled “AN AUTOMATED TIME SERIES DATA
COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING
AND DEPLOYMENT” is my own work except as cited in the references. The report
has not been accepted for any degree and is not being submitted concurrently in
candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisors, Dr Ooi B.Y. who has given me this bright opportunity to engage in a mobile application development project. This can be my starting point in establishing my career in IT field.

Other than that, I want to thank my friends and class acquaintances which is very supportive and helpful in providing useful information and tips the whole time I were doing research about the title. Last not but least, I am very grateful to my parents which support me financially the whole time.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

ABSTRACT

Followed by area of machine learning and deep learning, gesture or pattern recognition then became one of the most active subject matter for deep learning. The problem is that data collection process in gesture recognition research is time consuming when it comes to labelling the data. Thus, a framework which facilitates a time series data collection process through dynamic labelling which is useful for deep learning training and deployment is proposed. The framework can be demonstrated through a simple mobile application with the framework implemented within it. First, the Android mobile application collects the time series data with embedded sensors in smartphone and sends the data off to one of the storage service, S3 which is handled by the AWS cloud. On the EC2 handled by AWS, an instance which acts as a web server with the Flask backend is going to receive a HTTP request from the mobile application with the aid of OKHTTPClient, which is a HTTP client that allows HTTP request transmission. After the web server receives the request, it proceeds to build and train the deep learning model based on the data stored in S3. The deep learning model that is going to be used in the project is LSTM model, which is a variant of RNN model. Finally, the web server is going to return the training output to the cloud storage which can be accessed through the mobile application.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	4
ACKNOWLEDGEMENTS	5
ABSTRACT	6
LIST OF FIGURES	11
LIST OF TABLES	14
CHAPTER 1 INTRODUCTION	16
1.1 Introduction	16
1.2 Project Background	17
HCI and CA.....	17
Automate	18
Hand Action Recognition.....	19
1.3 Motivation and Problem Statement	20
Motivation	20
Problem Statement	20
1.4 Project Objectives	22
1.5 Project Deliverables and Scope	23
1.7 Methodology	25
1.8 Technology Involved in Project	26
Deep Learning	26
RNN and LSTM.....	26
CHAPTER 2 LITERATURE REVIEW	27
CHAPTER 3 SYSTEM DESIGN	31
3.1 Framework Overview	31
3.2 Use Case Diagram	32
3.3 Use Case Description	33
Manage Data Collection Process	33
Get CSV File Locally	34
Upload CSV File Containing Sensor Data to Cloud	35
Manage Label	36
Configure Collection Settings	37
Manage Individual File on Cloud	38
Manage File on Cloud	39
Build Deep Learning Model.....	40

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Configuring Deep Learning Model's Hyperparameters	41
Manage Deep Learning Model Code	42
Manage Android Application's Code	43
Choose Sensor Available on Smartphone to Monitor	44
Check List of Sensors Available on Smartphone	45
Train Deep Learning Model	46
Configuring Model Training's Hyperparameters	47
Convert Deep Learning Model	48
3.4 Activity Diagram	49
3.5 Class Diagram	52
Relationship between Classes	52
CheckSensorActivity	53
CheckSensorAdapter	53
CustomDialogFragment	53
CustomProgressDialog	54
DeviceInfoActivity	54
FitModelActivity	55
IntEditTextPreference	55
loggingTask	55
MainActivity	56
ManageFileActivity	57
NetworkSettingsActivity	58
PermissionAccessHandler	59
PredictGestureActivity	60
PredictMorseCodeActivity	61
SettingsActivity	62
SettingsFragment	62
S3Handler	63
TensorFlowClassifier	63
3.6 Entity Relationship Diagram	64
3.7 Sequence Diagram	65
3.8 Communication Diagram	66
3.9 State Diagram	67
Sensor	67
Database	68

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Dataset File.....	69
Python Code File / Settings File	70
Cloud	71
Web Server.....	72
CHAPTER 4 IMPLEMENTATION	73
4.1 Tools	73
4.2 Requirements	74
4.3 Framework Architecture.....	77
4.4 Resolved Implementation Issues and Challenges	79
4.5 Unresolved Implementation Issues and Challenges	81
CHAPTER 5 TESTING AND EXPERIMENTAL RESULTS	82
5.1 Features to be Tested.....	82
5.2 Features not to be Tested.....	82
5.3 Approach	82
5.4 Environment and Infrastructure.....	83
Hardware	83
Software	83
5.5 Configure Collecting Settings	84
Equivalence Partitioning.....	84
Boundary Value Analysis	86
Test Procedure Setup	87
5.6 Configure Deep Learning Model’s Hyperparameters.....	88
Equivalence Partitioning.....	88
Boundary Value Analysis	88
Test Procedure Setup	89
Testing with Input Shape	90
5.7 Experimental Results	91
Unoptimized vs Optimized Data Collection	91
Manual vs Automated Labelling.....	93
Model Building	94
Model Training	95
5.8 Mobile Application Testing	96
CHAPTER 6 CONCLUSION AND FUTURE WORK	109
6.1 Conclusion	109
6.2 Future Work	110

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

REFERENCES	111
POSTER	114
PLAGIARISM CHECK RESULT	115

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

LIST OF FIGURES

Figure Number	Title	Page
3.2.1	Use Case Diagram	32
3.4.1	Activity Diagram Part 1	49
3.4.2	Activity Diagram Part 2	50
3.4.3	Activity Diagram Part 3	51
3.5.1	Relationship between Classes	52
3.5.2	CheckSensorActivity	53
3.5.3	CheckSensorAdapter	53
3.5.4	CustomDialogFragment	53
3.5.5	CustomProgressDialog	54
3.5.6	DeviceInfoActivity	54
3.5.7	FitModelActivity	55
3.5.8	IntEditTextPreference	55
3.5.9	loggingTask	55
3.5.10	MainActivity	56
3.5.11	ManageFileActivity	57
3.5.12	NetworkSettingsActivity	58
3.5.13	PermissionAccessHandler	59
3.5.14	PredictGestureActivity	60
3.5.15	PredictMorseCodeActivity	61
3.5.16	SettingsActivity	62
3.5.17	SettingsFragment	62
3.5.18	S3Handler	63
3.5.19	TensorFlowClassifier	63
3.6.1	Entity Relationship Diagram	64

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

3.7.1	Sequence Diagram	65
3.8.1	Communication Diagram	66
3.9.1	State Diagram for Sensor	67
3.9.2	State Diagram for Database	68
3.9.3	State Diagram for Dataset File	69
3.9.4	State Diagram for Python Code File/Settings File	70
3.9.5	State Diagram for Cloud	71
3.9.6	State Diagram for Web Server	72
4.2.1	Used SDK Version	74
4.2.2	Android Studio Version	75
4.2.3	Used Android Libraries	76
4.2.5	Python Version	76
4.2.6	Apache Version	76
4.2.7	Mod_wsgi Version	76
4.3.1	Framework Architecture	77
5.7.1	Records Created in Database over 15 Minutes Duration	91
5.7.2	Time Consumed to Write File over 15 Minutes Duration	91
5.7.3	Time Taken to Label Single Record	93
5.7.4	Time Taken to Build Model	94
5.7.5	Time Taken to Tune Training Hyperparameters	95
5.8.1	Main Interface	97
5.8.2	Settings Part 1	98
5.8.3	Settings Part 2	99

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

5.8.4	Dynamic Labelling	100
5.8.5	After Clicking Stop Collect Button	101
5.8.6	Manage File in Cloud Storage	102
5.8.7	Accessing Device Info	103
5.8.8	Tune Model Building and Training Hyperparameters	104
5.8.9	Fit Model and Convert Model Function	105
5.8.10	Training Results and Converted Model	106
5.8.11	Predict Hand Gesture	107
5.8.12	Predict Morse Code	108

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

LIST OF TABLES

Table Number	Title	Page
2.1	Hand Gesture Study Comparison	28
2.2	Mobile Applications' Comparison Part 1	29
2.3	Mobile Applications' Comparison Part 2	30
4.2.4	Android API/SQLite Version	75
5.4.1	Hardware Environment	83
5.4.2	Software Environment	83
5.5.1	Equivalence Partitioning for Configuring Collecting Settings Part 1	84
5.5.2	Equivalence Partitioning for Configuring Collecting Settings Part 2	85
5.5.3	Boundary Value Analysis for Configuring Collecting Settings	86
5.6.1	Equivalence Partitioning for Configuring Deep Learning Model's Hyperparameters Part 1	88
5.6.2	Equivalence Partitioning for Configuring Deep Learning Model's Hyperparameters Part 1	88
5.6.3	Boundary Value Analysis for Configuring Deep Learning Model's Hyperparameters	88
5.6.4	List of Test Cases for Input Shape	90

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AWS	Amazon Web Service
CA	Context Aware
CNN	Convolutional Neural Network
CPU	Computer Processing Unit
CW	Continuous Wave
DTW	Dynamic Time Warping
EC2	Elastic Compute Cloud
EMG	Electromyography
HCI	Human Computer Interaction
LSTM	Long Short-Term Memory
RGB-D	Red Green Blue-Depth
RNN	Recurrent Neural Network
S3	Simple Storage Service
WIMP	Windows, Icons, Menus, Pointers
Mod-WSGI	Apache HTTP server module

CHAPTER 1 INTRODUCTION

1.1 Introduction

Typically, data collection and processing are the first step for preliminary work throughout any deep learning research especially gesture recognition, then followed by building as well as training a deep learning model in order to do prediction. In this paper, an automated time series data collection framework is introduced to simplify and facilitate the data collection process as well as the process of building and training the deep learning model. The framework is proposed through the use of multiple sensors embedded in smartphone devices. The framework proposed in the paper also emphasizes on the dynamic labelling of the time series data which is collected from the sensors as well as optimising the process of building and training the deep learning model.

1.2 Project Background

HCI and CA

Human have five senses or perception, which typically refers to sight, sound, taste, touch and smell. Those senses usually interact with the surrounding environment including some technology gadgets. This is called HCI. Currently, HCI is described through most of the digital gadgets that human interacts with, such as the text entry devices, positioning devices, display devices and sensors. Besides that, HCI included user interface of an application as well as WIMP interface. For example, in Windows operating system of our computer, there is command prompt, which enables us to enter command sequentially to instruct computer about a series of action we want it to perform. The usage of HCI is involved in many applications, in particularly hand gesture recognition (Cai, L. et al., 2017) and eye tracking (Zhang et al., 2017). Also, it is used for people with disabilities as well as deducing human emotions (Bergman & Johnson, 1997; Hibbeln et al., 2017).

In Schmidt's study (2000), HCI is classified into two categories: implicit and explicit. Implicit HCI is referring to the avoidable interaction between an individual's act and the system but then that the system has the capability to understand the action done by the individual. On the other hand, explicit HCI is referring to the system being instructed by the human, mostly with the usage of graphical user interface or a pattern executed by the individual. Another paper from the same author (2003) illustrates the modelling of a simple implicit HCI. The paper further discusses that the major priority within the scope of implicit HCI is the context, denoting the real setting where the task is initiated. It is said that the application with the characteristic of implicit HCI has the ability to affect and alter the environment through interacting with the context. The implicit HCI discussed above is very similar to the CA, which is primarily our focus for the topic. The latest definition of context and CA were found in Abowd and his colleagues' paper (1999). In the paper, context is defined as 'any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves'. CA can be used to describe a system 'if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task'. Currently, CA is

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

implemented in many mobile applications such as Facebook and YouTube. Other than that, CA is utilized in many important ways such as learning system for incapacitated learner, healthcare monitoring, city routes recommendation, mobile web browsing and decision making (Salah et al., 2017; Khozouie, Fotouhi-Ghazvini & Minaei-Bidgoli, 2017; Casino, et al., 2017; Wang et al., 2016; Moore & Van Pham, 2017). Below is the overview of a mobile application which utilizes the concept of CA as well as smartphone sensor to achieve its owner's task once certain condition is met.

Automate

Automate is a mobile application that is available through Google Playstore (LlamaLab, 2018). When Automate app is first downloaded from Google Playstore, it consists of several permission modules though they can be downloaded externally from Google Playstore too. Those modules are required in order to execute the task that user requests. Several tasks are performed sequentially in a series of flowchart as designed by the user himself. Some of the tasks require root access, or in the other words, super user privilege to the Android smartphone. It means that the user must 'root' their phone before implementing those tasks into their flowchart.

The perks of having this mobile application to automate your task is that the mobile application is able to have infinite possibilities since there are tons of flowchart consisting of multiple different tasks designed by the community. Flowcharts are designed specifically for several areas which are categorized based on which occupation the user has. It ranges from business, education to transportation and weather which is very helpful to the user, since it clearly informs the user. The user requirements are considered systematically and planned properly. Other than that, the user has the ability to design as well as customize his own flowchart to automate the preferred task. The user can choose a series of task to be performed by the Android smartphone depending on what action types that the user desired. For instance, the user can detect whether if they want their phone to turn on their wireless network when they are at home. The user also can tell their phone to turn on power saver mode or lower the brightness when the smartphone's battery falls below certain limit. Furthermore, if the Automate app can gain root access to the smartphone, the Automate app can achieve most of the work that the user wants them to do including

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

acquiring the information about CPU speed and adjusting the speed based on the user's preferences or even turn on the standby mode aka Doze to save battery power of the Android device.

Hand Action Recognition

Non-verbal communication is the second important method used to communicate with other people that we met in our life other than the verbal communication. Hand gesture is the first one that comes to mind. When we talk about the other use of hand gesture, people use hand to interact with their smartphone, in particularly the user interface of their phone. For instance, there are some examples of a hand behaviour such as scrolling down the web page of a web browser, facing down the smartphone screen to enable vibrate mode of the smartphone and double tapping the screen to turn off the display. Electromyography (EMG) is the common terms in the subject of hand gesture recognition of the papers that had been researched in the literature review. It is well-defined as 'measurement of the electrical activity produced by the muscles of the human body' (Benalcázar et al., 2017; Jaramillo & Benalcázar, 2017).

1.3 Motivation and Problem Statement

Motivation

Nowadays, the current generation phone is known as smartphone. It has a friendly user interface that interacts with smartphone owner at ease. In daily life, people are known to interact with the smartphone more often. Smartphone users need to perform a hand gesture which allows the smartphone to receive the notification in order to execute the task. In fact, the smartphone is actually not 'smart' enough. It is unable to 'perceive' the surrounding changes, let alone respond to those changes. Those problems exist for every smartphone owner especially for a businessman or salaryman. For businessman, they usually hold a lot of meeting and sometimes they forget to put their phone in vibrate or mute mode. Unfortunately, the phone is unable to automatically check the environment the owner is in and automatically executes the task. The Automate app is an example for that. The application turns on the vibration mode when the owner enters certain area, that is the owner's company. Corresponding to that, the application of hand activity recognition on a mobile application can help businessman, tending to their personal needs in helping them to turn on the vibrate mode automatically after the smartphone receives a specific pattern of hand gesture.

Problem Statement

In order to carry out a hand gesture study, data collection is the most important beginning step for any researches, especially in the field of deep learning. Other researchers proceeded to carry multiple study related to hand gesture recognition with several methods and sensors (Arenas, Murillo & Moreno, 2017; Benalcázar et al., 2017; Jaramillo & Benalcázar, 2017; Sakamoto et al., 2017; Cai et al., 2017). But yet so far, there are no any optimised or automated data collection process as seen in multiple study. Though a much better sensor was introduced from one study to another, it stills yield extremely big research costs in terms of time. Furthermore, the research cost increases significantly depending on the scale of the research that taken place.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

On the other hand, while carrying out data collecting process in their researchers, the data collection and processing task especially in data labelling may either prone to human error or very time-consuming if the research requires significant amount of data. This causes the experimental result may have deemed unreliable due to those mistake. Also, multiple studies that had been carried during these few years were very specific and detailed in only a few types of hand gesture. This severely limited the scope of the research. In fact, through an example of mobile application developed in this project, the data collection process can be further facilitated and becomes less time-consuming even though there are needs to prepare a large amount of dataset and to perform data labelling for those dataset at the same time. To this, an opportunity to produce a mobile application framework with the aid of mobile embedded sensors arise in order to inform the people as well as the researchers about the practical usefulness of the smartphone sensor. In this case, the smartphone can contribute to the community when their embedded sensors are utilized well enough for those smartphone owners or researchers' study.

1.4 Project Objectives

The project has one main objective, that the project aims to provide a viable platform such as the mobile application itself that could facilitate the time series data collection process for the deep learning model building and training. The data collection process can be facilitated by allowing automated data labelling. With automated data labelling, the data labelling process can be sped up since it is not done by human but done by machine. Other than that, dynamic labelling might be able to solve the limitations in terms of the number of labels unlike there are only a few specific ones in multiple study during these past few years. Also, for the data labelling process, they might prone to human error as a lot of the data points may need to be labelled. The application developed aims to resolve the human labelling error. On the other hands, by allowing parallel data collection and model training, the time-consuming deep learning process can be shortened even further. This is because the time taken for model training is directly proportional to the model complexity as well as the training data. Thus, parallelizing both of the processes may reduce the time even further. To facilitate the deep learning processes even more, the project aims to automate the rest of the deep learning process, especially model building and training by building an interface whereby the parameters of the deep learning model as well as the training hyperparameter can be freely configured.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

1.5 Project Deliverables and Scope

An Android mobile application with automated data collection framework will be submitted as final deliverable at the end of this project. This Android application consists of a simple data collecting procedure which uploads the raw data file to cloud for data processing as well as deep learning. First, the Android application will collect the raw data about the smartphone sensors which then the application will compile it into a file in order to be uploaded to the cloud. The application will collect user input for model configuration and training hyperparameters. At the cloud, the deep learning instance will obtain the dataset for data preprocessing. Also, the deep learning instance consists of a preloaded python script which takes the data into building the neural network model. Lastly, the deep learning instance will compute the training output and store it on the Amazon cloud storage.

1.6 Contribution

The project can solve the primary concern of the communities which has the interest in learning about deep learning. Occasionally, deep learning communities need to go through a series of time-consuming procedure in building a deep learning model for their intended research. With the framework implemented, the project can reduce the burden of the communities in such a way that most of the data collection as well as model training process is automated with the help of an Android application. Furthermore, the project solves the necessity of having a computer or laptop for deep learning training by allocating the task to the cloud instance through an Android application. Now, the communities are able to use a simple Android application in order to perform their data collecting as well as model training process and lastly converting the model for actual deployment and practices.

1.7 Methodology

There are three categories of methodologies known for a system analysis and design process: Structured Design, Rapid Application Development (RAD) and Agile Development. The methodology utilised in the project is prototyping, which is one of the subcategories stemmed from Rapid Application Development. Rapid Application Development indicates that the system is carried out or developed according to some of the basic concept of System Development Life Cycle (SDLC). The methodology is slightly altered from SDLC in order to speed up the system development process.

There are four phase in SDLC: planning, analysis, design and implementation. SDLC implements all of the phases in an orderly manner. Unlike SDLC, prototyping methodology is a methodology that executes the analysis phase, design phase and implementation phase simultaneously. The benefit of applying the prototyping methodology is that the developed application can fulfil user requirements quickly. Besides that, the methodology provides a good short time schedule as well as schedule visibility which allows the project is developed in a fast pace but the development progress is still able to be inspected easily.

1.8 Technology Involved in Project

Deep Learning

Deep learning is defined as ‘a subfield of machine learning concerned with algorithms that is inspired by the structure and function of the brain called artificial neural networks. Generally, in the area of artificial intelligence, deep learning is considered as a very advanced machine learning technique, typically involving neural network and many hidden layers. Several papers propose their application of deep learning such as anomaly-based intrusion detection system in area of network security, smart health services improvement, nuclei detection, retinal images classification and skin lesion analysis in medical area, leaves counting in plants and literature analysis (Mohammadi & Namadchian, 2017; Obinikpo & Kantarci, 2017; Sornapudi et al., 2018; Choi et al., 2017; Li & Shen, 2018; Ubbens et al., 2018; Liang et al., 2017).

RNN and LSTM

There are few types of neural network available for the deep learning researching. One of the well-known type of neural network is RNN. RNN is typically used for character recognition, plant identification and lithium-ion battery life prediction as shown in some of the following papers (Naz et al., 2018; Liu et al., 2018; Zhang et al., 2018). For once, a plain neural network is not considered for this project due to every input sequences are having different length. On the other hand, normal variant of RNN isn't considered for this project with the weakness of having vanishing gradient problem. Thus, for this project, a variant of RNN, that is LSTM model is going to be used for the hand activity recognition. Though LSTM requires much more difficult computation and needs more parameters compared to RNN, LSTM overcomes vanishing gradient problem as well as having the capability of handling very long sequences.

CHAPTER 2 LITERATURE REVIEW

The real question here is how to infer those action and in turn transform them into data input to be processed as an output. In this section, a few gesture inferring methods proposed by the related work will be discussed in depth regarding its pros, cons and its limitations.

Arenas and his colleagues (2017) recently proposed a CNN architecture for 3 types of hand gesture recognition which is ‘open’, ‘closed’ and ‘unknown’. Although they didn’t mention what sensor they used, they utilized MatConvNet library in MatLab as well as constantly tuning hyper parameters and observing behavioural changes of the architectures to determine the suitable CNN architecture to be finalized as their deliverable. Another paper studied about dynamic hand gesture recognition (Cai et al., 2017). They used Microsoft Kinect RGB-D sensor to obtain hand gesture input. Then, with the combination of Euclidean distance, EMG techniques as well as an improved DTW algorithm, they obtained the recognition output. They researched general traffic officer’s hand signals, which are ‘stop’, ‘straight’, ‘turn left’, ‘left turn waiting’, ‘turn right’, ‘lane change’, ‘slow down’ and ‘pull over’.

On the other hand, Benalcázar group’s study (2017) included a real-time gesture recognition model which used machine learning and a ‘non-invasive’ surface EMG sensor known as Myo Armband to capture their hand gesture input. The Myo Armband wore by the participant on their wrist and the armband can recognize 6 types of hand gesture, which are ‘pinch’, ‘fist’, ‘open’, ‘wave in’, ‘wave out’ and ‘no gesture’. Based on k-nearest neighbour and DTW algorithm, they then suggested a model which had better performance in terms of gesture recognition accuracy than the Myo system. Similar to the study of Cai and his team, they also used dynamic time warping algorithm. Real-time hand gesture recognition was explored by Jaramillo and Benalcázar’ work (2017). They used similar surface EMG as previously, that is the Myo Armband and practiced machine learning. Identical types of hand gesture as the previous paper were examined too. For Sakamoto and his associates’ paper, they studied about automatic hand gesture recognition with low-cost CW radar and CNN. The types of hand gesture they researched in were ‘arm moving back and forth’, ‘fist and palm’ and ‘palm rotating left and right’.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

To summarize the methods above, a comparison table is made as shown below:

Authors of the paper	Year of Work	Types of Sensors used	Methods / Algorithm used	Hand action / gestures recognition researched
Arenas' group	2017	Not mentioned	MatConvNet (MatLab), CNN	Open, closed, unknown
Cai L.Q et al.	2017	Microsoft Kinect RGB-D sensor	Euclidean distance, EMG, Improved DTW	Stop, straight, turn left, left turn waiting, turn right, lane change, slow down, pull over
Benalcazar et al.	2017	EMG (Myo Armband)	k-nearest neighbour, DTW	Pinch, fist, open, wave in, wave out, no gesture
Jaramillo & Benalcazar	2017	EMG (Myo Armband)	Machine learning	Pinch, fist, open, wave in, wave out, no gesture
Sakamoto T. et al.	2017	CW radar	CNN-based machine learning algorithm	Arm moving back and forth, fist and palm, palm rotating left and right

Table 2.1: Hand Gesture Study Comparison

However, the data acquisition process was said to be very time consuming and prone to human error as mentioned by Jaramillo and Benalcazar's paper (2017). Every study has a limited number of labels, which constitutes the concept of having dynamic labelling for this project. Additionally, the participant also needs to be present and wear sensors, preventing the convenience of collecting data everywhere. Thus, a mobile framework about automation of time series data collection process which uses smartphone sensor is proposed with the help of LSTM.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Moreover, a table comparing the features contained in the current Android application developed as well as the other Android application in the market is shown below:

Android apps/features	Wide sensor selections	Sensor graph visualization	Sensor delay option	Collecting interval option
SensorBoard	✓	✗	SENSOR_DELAY_FASTEST ONLY	✗
Sensor Data Collector	✓	✓	✓	✓
Sensor (CodeTech)	✓	✓	✗	✗
SensorLab	✓	✓	✗	✓
SensorData	✓	✓	✓	✓
G-Sensor Logger (Free ver)	✗	✓	✗	✓
Proposed work (Sensordyne)	✓	✗	✓	✓

Table 2.2: Mobile Applications' Comparison Part 1

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Android apps/features	Logging interval option	Background logging	Dynamic labelling	Additional features
SensorBoard	✓	✓	✗	✗
Sensor Data Collector	✗	✓	Must click to change	✗
Sensor (CodeTech)	✗	✗	✗	✗
SensorLab	✗	✓	✗	✗
SensorData	✓	✗	✗	✗
G-Sensor Logger (Free ver)	✗	✓	✗	✗
Proposed work (Sensordyne)	✓	✗	✓	Automated deep learning process

Table 2.3: Mobile Applications' Comparison Part 2

CHAPTER 3 SYSTEM DESIGN

3.1 Framework Overview

The framework is called automated time-series data collection framework. This framework helps facilitating the data collection process as well as the rest of the deep learning processes. For instance, the research of the hand gesture recognition. First of all, the framework allows the users to use a mobile application in data collection, labelling and processing as well as model building and training. Researcher is able to choose which type of sensor data to take account in data collecting process. Before data collecting process is initiated, the user chooses to add in labels for data, that is the types of gesture need to be taken account in. The researcher will be instructed to play a game which is going to be randomly changing labels.

The user is allowed to export the sample data collected to a file. Then, the file is uploaded to the cloud. The user will then choose to build and train model whose input results from the file being uploaded to cloud. The user is allowed to view training output as well as converting model for actual deployment and practice after the model building and training process has been completed by the deep learning instance hosted as a web server on the cloud.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

3.2 Use Case Diagram

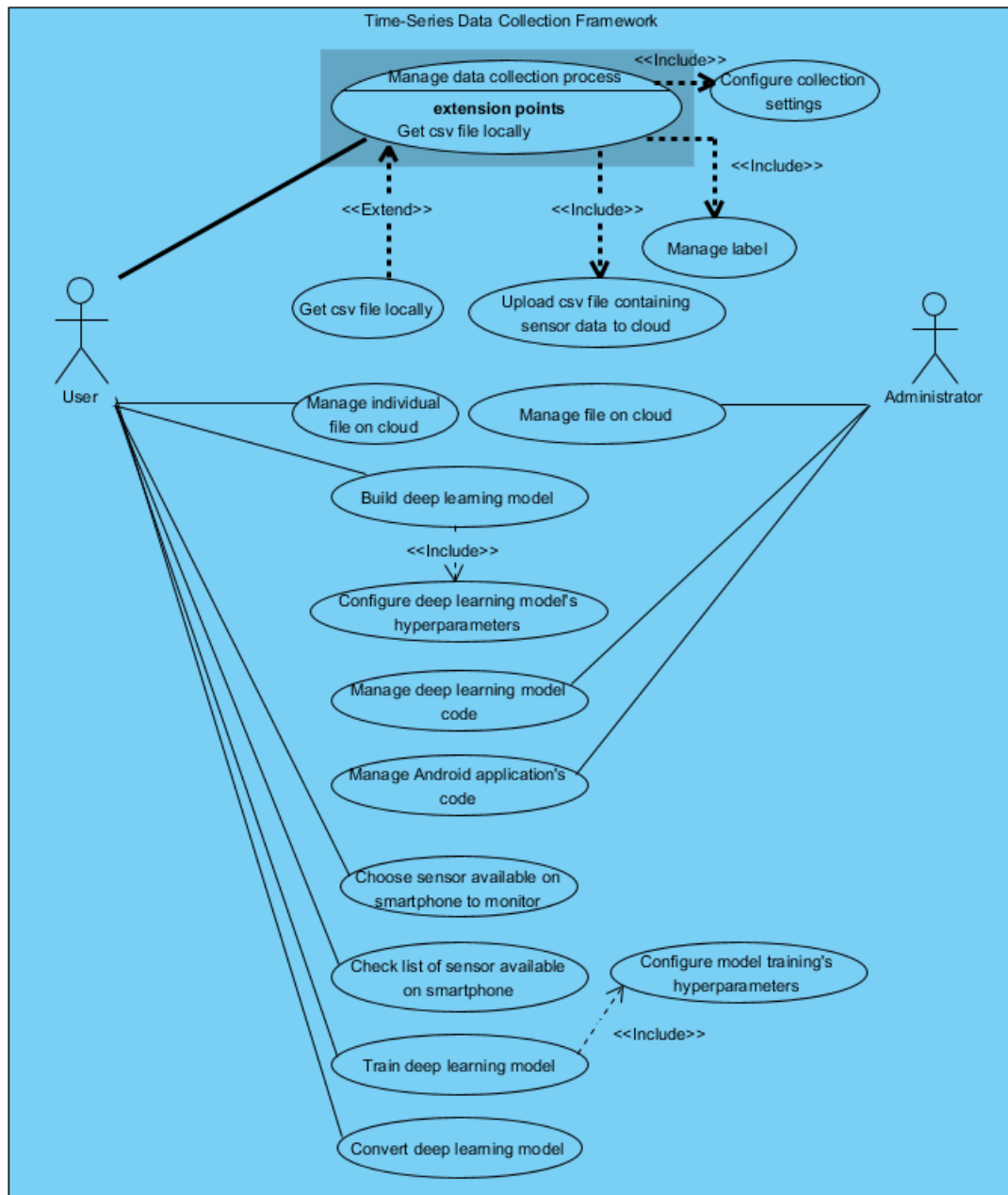


Figure 3.2.1: Use Case Diagram

3.3 Use Case Description

Manage Data Collection Process

Use Case Name: Manage Data Collection Process

ID: 101

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to manage his / her data collecting process

Brief Description: This use case describes how we handle the process of managing data collection process on user's smartphone

Trigger: User clicks the 'Start Collect' or 'Stop Collect' button

Type: External

Relationships: Association – ×, Include – ✓ (102, 103, 104), Extend – ✓ (101),

Generalisation – ×

Normal Flow of Events:

1. User enters the label for data.
2. A list of label for data is obtained and displayed on user interface.
3. User clicks the 'Start Collect' button, giving the orders to start the process.
4. The data collected is transferred into database which is created upon application start up.
5. User clicks the 'Stop Collect' button, giving the orders to stop the process.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 3a. List of labels is empty, showing the error message 'There is no label'.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Get CSV File Locally

Use Case Name: Get CSV File locally

ID: 102

Importance level: Medium

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to get the csv file locally

Brief Description: This use case describes how we handle the process of getting csv file locally

Trigger: User clicks the ‘Get local CSV’ button

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User clicks the ‘Get local CSV’ button.
2. ‘CSV was downloaded locally’ message is shown together with the file absolute path.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 1a. User clicks the ‘Get local CSV’ button. ‘Source file not exists’ message is shown.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Upload CSV File Containing Sensor Data to Cloud

Use Case Name: Upload CSV File Containing Sensor Data to Cloud

ID: 103

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to upload the csv file to cloud

Brief Description: This use case describes how we handle the process of uploading csv file to cloud

Trigger: User clicks the ‘Export to Cloud’ button

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User clicks the ‘Export to Cloud’ button. ‘CSV file uploaded to AWS S3’ message is shown.
2. The data from database is written and compiled into a local file in CSV format.
3. The local CSV file is taken from application data folder and uploaded to user’s directory on cloud.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 1a. There are no data collecting process prior to clicking the ‘Export to CSV’ button, thus no data is created on database. Error message ‘File does not exist’ is shown.
- 3a. Application doesn’t have internet access or permission, causing file upload process to fail.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manage Label

Use Case Name: Manage Label

ID: 104

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to manage the label for data collecting process

Brief Description: This use case describes how we handle the process of managing label for data collecting process

Trigger: User clicks 'Add' / 'Delete' / 'Clear' button in main interface

Type: External

Relationships: Association – ×, Include – ×, Extend – ×, Generalisation – ×

Normal Flow of Events:

1. User enters the text label for data.
2. User clicks the 'Add' button. 'Label added' message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 1a. User enters empty text label for data.
- 2a. User clicks the 'Delete' button. 'Label deleted' message is shown.
- 2b. User clicks the 'Clear' button. The user input is cleared in the text box.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Configure Collection Settings

Use Case Name: Configure Collection Settings

ID: 105

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to configure collection settings

Brief Description: This use case describes how we manage the process of configuring collection settings

Trigger: User accesses the ‘Settings’ interface.

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the ‘Settings’ interface.
2. User inputs value for label interval.
3. User inputs value for log data interval.
4. User inputs value for sensor sampling delay.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 2a. User inputs invalid value for label interval. The label interval falls back to default values.
- 3a. User inputs invalid value for log data interval. The log data interval falls back to default values.
- 4a. User inputs invalid value for sensor sampling delay. The sensor sampling delay falls back to default values.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manage Individual File on Cloud

Use Case Name: Manage Individual File on Cloud

ID: 106

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to manage the file on cloud

Brief Description: This use case describes how we manage the process of managing file on cloud

Trigger: User accesses the ‘Manage File’ interface on the side bar of the application

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the ‘Manage File’ interface on the side bar of the application
2. User inputs file name on text box.
3. User clicks on ‘Download’ button. File is added to the cloud storage.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 3a. User clicks on ‘Delete’ button. File is removed from the cloud storage.
- 3b. User clicks on ‘Share’ button. File is shared to the folder of the other user.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manage File on Cloud

Use Case Name: Manage File on Cloud

ID: 107

Importance level: High

Primary Actor: Developer/Administrator

Use Case Type: Detail, Essential

Stakeholders and Interests: Developer/Administrator – wants to manage the file on cloud according to user's request

Brief Description: This use case describes how we manage the process of managing file on cloud

Trigger: Developer want to manage file according to user's request

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. Developer accesses the cloud storage.
2. Developer manages the file according to user's preferences.

Sub Flows: Not applicable

Alternate / Exceptional Flows: Not applicable

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Build Deep Learning Model

Use Case Name: Build Deep Learning Model

ID: 108

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to build the deep learning model according to his/her own preference

Brief Description: This use case describes how we manage the process of building deep learning model and send the model configuration to web server for building

Trigger: User clicks ‘Save Model’ button

Type: External

Relationships: Association – ✕, Include – ✓ (109), Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the ‘Model and Neural Network Settings’ interface.
2. User clicks ‘Save Model’ button. Success result message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 2a. User clicks ‘Save Model’ button. ‘No layer in the model created’ message is shown.
- 2b. User clicks ‘Save Model’ button. ‘No internet access’ message is shown.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Configuring Deep Learning Model's Hyperparameters

Use Case Name: Configuring Deep Learning Model's Hyperparameters

ID: 109

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to build the deep learning model according to his/her own preference

Brief Description: This use case describes how we manage the process of building deep learning model and send the model configuration to web server for building

Trigger: User clicks 'Add Layer to Model' button

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the 'Model and Neural Network Settings' interface.
2. Application shows the list of hyperparameters available.
3. Application updates activation list, optimizer list and losses list from web server.
4. User inputs values for the corresponding hyperparameters.
5. User clicks 'Add Layer to Model' button. 'Layer added' message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 4a. User inputs wrong format for corresponding hyperparameters.
- 4b. User inputs missing values for corresponding hyperparameters.
- 5a. User clicks 'Add Layer to Model' button. 'Incorrect format for input shape' message is shown.
- 5b. User clicks 'Add Layer to Model' button. 'Missing input' message is shown.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manage Deep Learning Model Code

Use Case Name: Manage Deep Learning Model Code

ID: 110

Importance level: High

Primary Actor: Developer/Administrator

Use Case Type: Detail, Essential

Stakeholders and Interests: Developer/Administrator – wants to refine or modify the code on web server according to user's feedback

Brief Description: This use case describes how we manage the process of managing deep learning model code

Trigger: Developer/Administrator refines the programming code hosted on web server.

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. The code is refined and modified.
2. The code is uploaded to web server's directory.
3. The user can get functionality updates in real time.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 3a. Code may have syntax error which breaks the build model functionality on application. The output couldn't be shown on user interface.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manage Android Application's Code

Use Case Name: Manage Android Application's Code

ID: 111

Importance level: High

Primary Actor: Developer/Administrator

Use Case Type: Detail, Essential

Stakeholders and Interests: Developer/Administrator – wants to add new functionality or fix known bugs on application according to user's feedback.

Brief Description: This use case describes how we manage the process of managing Android application's code

Trigger: Developer/Administrator modifies application code.

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. The application code is refined and modified.
2. The application is deployed.
3. The user can install the new application deployed.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 3a. Code may have syntax error which breaks certain functionality on application or causing the application unable to start.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Choose Sensor Available on Smartphone to Monitor

Use Case Name: Choose Sensor Available on Smartphone to Monitor

ID: 112

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to check list of sensor's available on smartphone

Brief Description: This use case describes how we manage the process of checking list of sensors available on smartphone.

Trigger: User accesses the 'Settings' interface.

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

5. User accesses the 'Settings' interface.
6. User chooses which sensor to keep track of by toggling the switch for the sensor.

Sub Flows: Not applicable

Alternate / Exceptional Flows: Not applicable

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Check List of Sensors Available on Smartphone

Use Case Name: Check List of Sensors Available on Smartphone

ID: 113

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to check list of sensor's available on smartphone

Brief Description: This use case describes how we manage the process of checking list of sensors available on smartphone.

Trigger: User accesses the 'Check Sensors' interface.

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

7. User accesses the 'Check Sensors' interface.
8. Application gets the name and type of sensors available on the smartphone.
9. Application shows the list of sensors consisting of the name and type on user interface.
10. The user can know which sensors' data are available to collect on the smartphone.

Sub Flows: Not applicable

Alternate / Exceptional Flows: Not applicable

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Train Deep Learning Model

Use Case Name: Train Deep Learning Model

ID: 114

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to build the deep learning model according to his/her own preference

Brief Description: This use case describes how we manage the process of training deep learning model

Trigger: User clicks the 'Fit Model' button

Type: External

Relationships: Association – ×, Include – ✓ (115), Extend – ×, Generalisation – ×

Normal Flow of Events:

1. User accesses the 'Fit Model' interface.
2. User clicks 'Fit Model' button.
3. Application sends request to web server.
4. Web server processes the request.
5. 'Successful' message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 5a. 'Failed' message is shown.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Configuring Model Training's Hyperparameters

Use Case Name: Configuring Model Training's Hyperparameters

ID: 115

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to train the deep learning model according to his/her own preference

Brief Description: This use case describes how we manage the process of building deep learning model and send the training hyperparameters to web server for training

Trigger: User clicks 'Save Settings' button

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the 'Model and Neural Network Settings' interface.
2. Application shows the list of hyperparameters available.
3. User inputs values for the corresponding training hyperparameters.
4. User clicks 'Save Settings' button. 'Successfully save settings' message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 4a. User clicks 'Save Settings' button. 'Settings file is not created' message is shown.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Convert Deep Learning Model

Use Case Name: Convert Deep Learning Model

ID: 116

Importance level: High

Primary Actor: User

Use Case Type: Detail, Essential

Stakeholders and Interests: User – wants to convert the deep learning model

Brief Description: This use case describes how we manage the process of converting deep learning model

Trigger: User clicks the ‘Convert Model’ button

Type: External

Relationships: Association – ✕, Include – ✕, Extend – ✕, Generalisation – ✕

Normal Flow of Events:

1. User accesses the ‘Fit Model’ interface.
2. User clicks ‘Fit Model’ button.
3. Application sends request to web server.
4. Web server processes the request.
5. ‘Successful’ message is shown.

Sub Flows: Not applicable

Alternate / Exceptional Flows:

- 4a. ‘Failed’ message is shown.

3.4 Activity Diagram

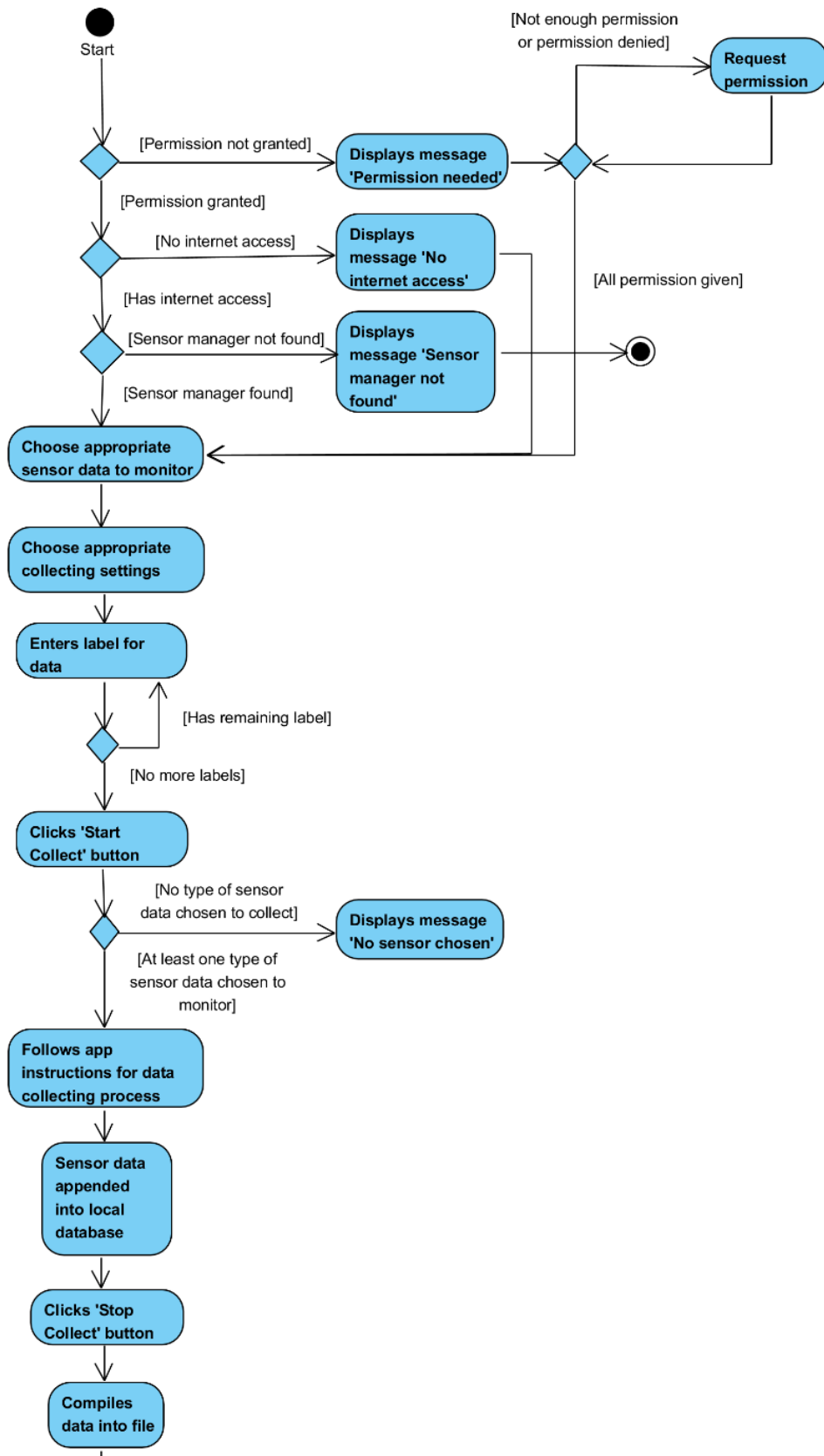


Figure 3.4.1: Activity Diagram Part 1

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

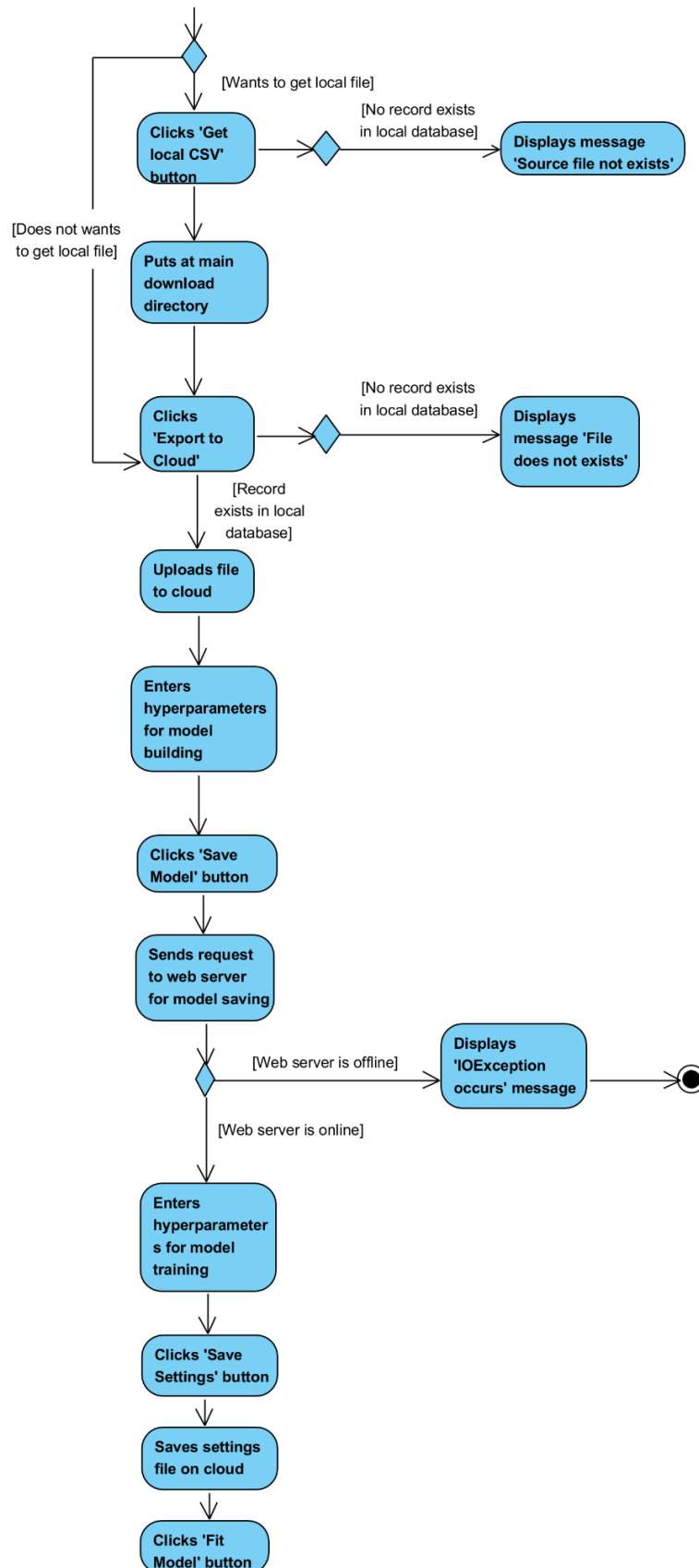


Figure 3.4.2: Activity Diagram Part 2

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

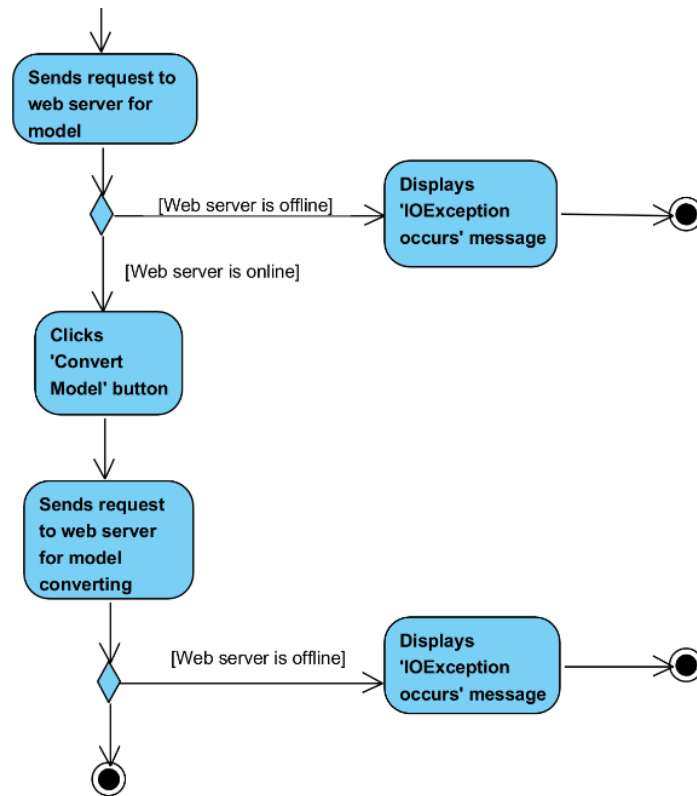


Figure 3.4.3: Activity Diagram Part 3

3.5 Class Diagram

Relationship between Classes

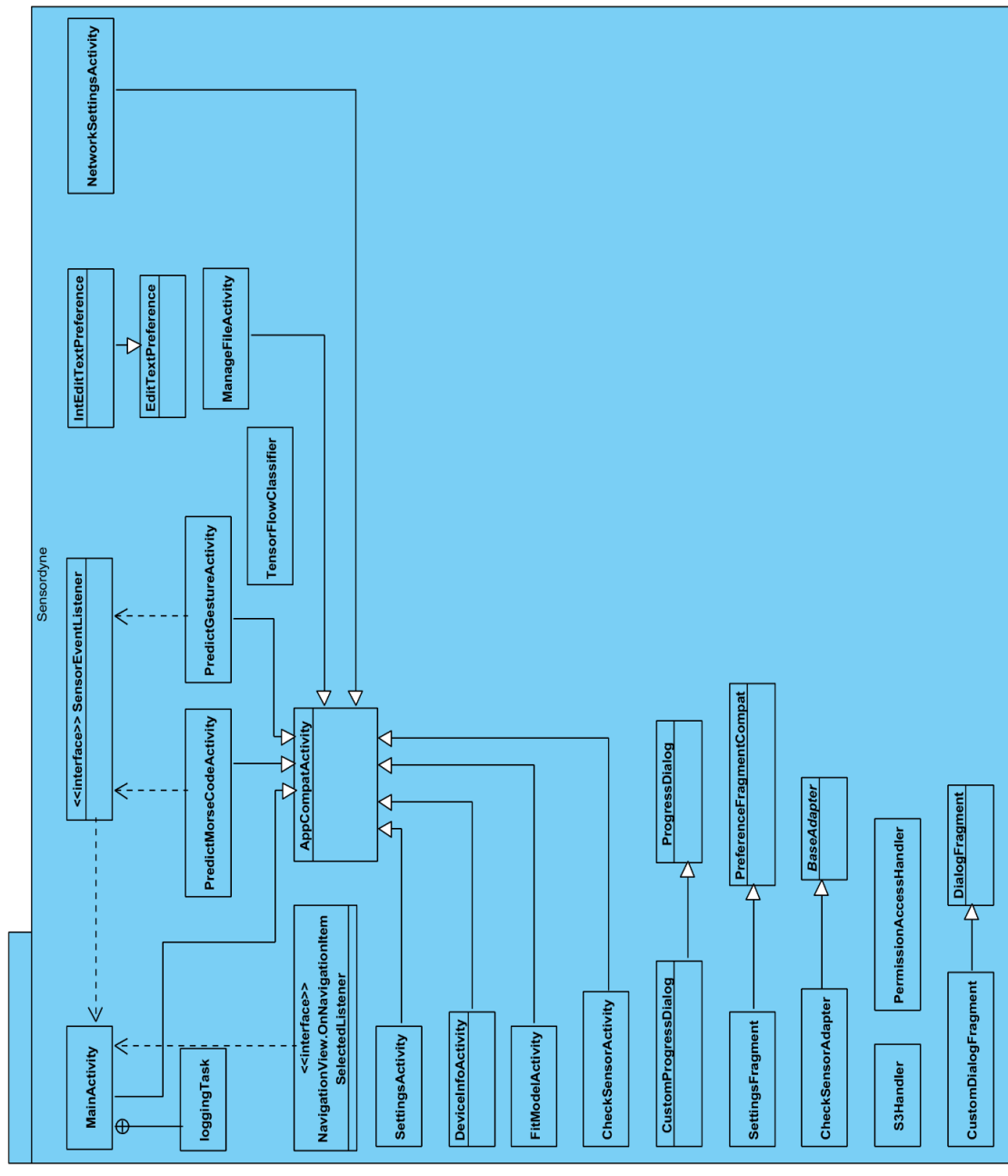


Figure 3.5.1: Relationship of Class Diagram

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

CheckSensorActivity

CheckSensorActivity
-sensorNames : ArrayList<String> = new ArrayList<>()
#onCreate(savedInstanceState : Bundle) : void

Figure 3.5.2: CheckSensorActivity

CheckSensorAdapter

CheckSensorAdapter
-names : ArrayList<String> -context : Context -layoutInflater : LayoutInflater -value : String
+getCount() : int +getItem(position : int) : Object +getItemId(position : int) : long +getView(position : int, view : View, parent : ViewGroup) : View

Figure 3.5.3: CheckSensorAdapter

CustomDialogFragment

CustomDialogFragment
-dialogCode : int -message : String -positiveButtonText : String -negativeButtonText : String -title : String -alertDialog : AlertDialog
<u>newInstance(dialogCode : int)</u> +onCreateDialog(savedInstanceState : Bundle) : Dialog +onCancel(dialog : DialogInterface) : void

Figure 3.5.4: CustomDialogFragment

CustomProgressDialog

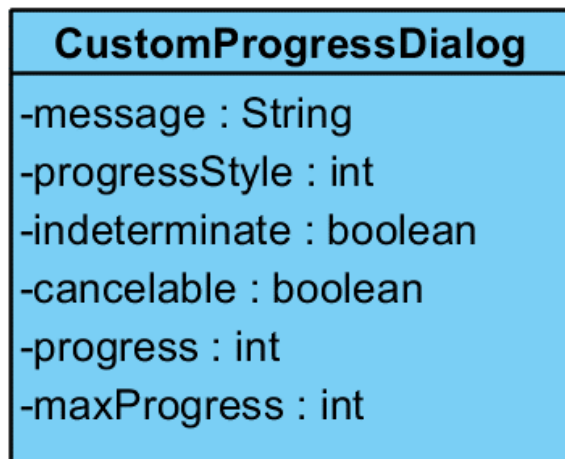


Figure 3.5.5: CustomProgressDialog

DeviceInfoActivity

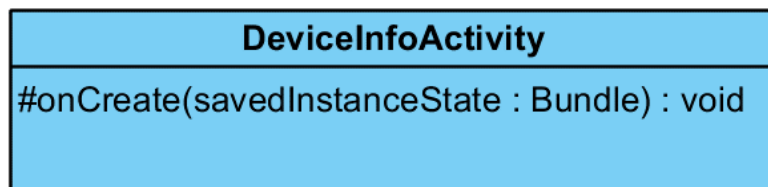


Figure 3.5.6: DeviceInfoActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

FitModelActivity

FitModelActivity
-permissionHandler : PermissionAccessHandler -s3Handler : S3Handler -androidID : String -fitModelButton : Button -graphVisualization : String -example : String
#onCreate(savedInstanceState : Bundle) : void +fitModel(view : View) : void +convertModel(view : View) : void +wantGraph(graphVisualization : String) : void +useExample(example : String) : void

Figure 3.5.7: FitModelActivity

IntEditTextPreference

IntEditTextPreference
+IntEditTextPreference(context : Context) +IntEditTextPreference(context : Context, attrs : AttributeSet) +IntEditTextPreference(context : Context, attrs : AttributeSet, defStyle : int) #getPersistedString(defaultReturnValue : String) : String #persistString(value : String) : boolean

Figure 3.5.8: IntEditTextPreference

loggingTask

loggingTask
-activityReference : WeakReference<MainActivity>
+loggingTask(context : MainActivity) #doInBackground(voids : ..) : Void

Figure 3.5.9: loggingTask

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

MainActivity



Figure 3.5.10: MainActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

ManageFileActivity

ManageFileActivity
-permissionAccessHandler : PermissionAccessHandler -s3Handler : S3Handler -filesInfo : TextView -fileInfo : EditText -downloadFileButton : Button -deleteFileButton : Button
#onCreate(savedInstanceState : Bundle) : void +deleteFile(view : View) : void +downloadFile(view : View) : void +shareFile() : void +shareFileFunc(targetAndroidID : String) : void

Figure 3.5.11: ManageFileActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

NetworkSettingsActivity

NetworkSettingsActivity
-permissionAccessHandler : PermissionAccessHandler -s3Handler : S3Handler -unitInput : EditText -inputShapelInput : EditText -dropoutInput : EditText -layerTypeSpinner : Spinner -activationSpinner : Spinner -lossSpinner : Spinner -optimizerSpinner : Spinner -batchSizeInput : EditText -epochInput : EditText -timeStepInput : EditText -addLayerButton : Button -saveModelButton : Button -saveSettingsButton : Button -layerType : String -optimizer : String -loss : String -activation : String -dropout : float -unit : int -batchSize : int -epoch : int -timeStep : int -pyCodeList : ArrayList<String> = new ArrayList<>()
#onCreate(savedInstanceState : Bundle) : void #onDestroy() : void +addLayer(view : View) : void +saveModel(view : View) : void -getOptimizersList() : void -getActivationsList() : void -getLossesList() : void +onItemSelected(parent : AdapterView<?>, v : View, position : int, id : long) : void +onNothingSelected(parent : AdapterView<?>) : void +saveFitModelSettings(view : View)

Figure 3.5.12: NetworkSettingsActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

PermissionAccessHandler

PermissionAccessHandler
-activity : Activity
+PERMISSION_ALL : int
-permissionDialog : CustomDialogFragment
-permissionSettingsDialog : CustomDialogFragment
-internetAccessDialog : CustomDialogFragment
+PermissionAccessHandler(activity : Activity)
+operation()
+getPermissionDialog() : CustomDialogFragment
+setPermissionDialog(permissionDialog : CustomDialogFragment) : void
+getPermissionSettingsDialog() : CustomDialogFragment
+setPermissionSettingsDialog(permissionSettingsDialog : CustomDialogFragment) : void
-convertArrayListStringToStringOfArray(source : ArrayList<String>) : String[]
+responseToNoPermission() : void
+isInternetAccessAvailable() : boolean
+responseToNoInternetAccess() : void

Figure 3.5.13: PermissionAccessHandler

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

PredictGestureActivity

PredictGestureActivity
<pre> -N_SAMPLES : int = 15 -leftTextView : TextView -rightTextView : TextView -onTableTextView : TextView -resultTextView : TextView -sensorManager : SensorManager -gyroscope : Sensor -accelerometer : Sensor -magnetometer : Sensor -gravimeter : Sensor -gyroX : List<Float> -gyroY : List<Float> -gyroZ : List<Float> -accX : List<Float> -accY : List<Float> -accZ : List<Float> -magX : List<Float> -magY : List<Float> -magZ : List<Float> -azimuth : List<Float> -pitch : List<Float> -roll : List<Float> -gravX : List<Float> -gravY : List<Float> -gravZ : List<Float> -gyroscopeVal : float[] -acceleroVal : float[] -magnetoVal : float[] -orientationVal : float[] -gravVal : float[] -r : float[] -results : float[] -classifier : TensorFlowClassifier -data : List<Float> = new ArrayList<>() -collectTimer : Timer -logTimer : Timer -collectTimerTask : TimerTask -logTimerTask : TimerTask -startPredictButton : Button -stopPredictButton : Button -acceleroEmpty : boolean = true -magnetoEmpty : boolean = true -tts : TextToSpeech #onCreate2(savedInstanceState : Bundle) : void #onStart2() : void #onResume2() : void #onPause2() : void #onStop2() : void +onSensorChanged2(event : SensorEvent) : void +onAccuracyChanged2(sensor : Sensor, accuracy : int) : void -speak(speech : String) : void +findHighestProb(results : float[]) : String +startPredict(view : View) : void +stopPredict(view : View) : void -toFloatArray(list : List<Float>) : float[] +round(d : float, decimalPlace : d) : float -getSensorManager() : SensorManager </pre>

Figure 3.5.14: PredictGestureActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

PredictMorseCodeActivity

```
PredictMorseCodeActivity
-N_SAMPLES : int = 15
-tTextView : TextView
-mTextView : TextView
-nTextView : TextView
-resultTextView : TextView
-sensorManager : SensorManager
-linearAccelerometer : Sensor
-linearAccX : List<Float>
-linearAccY : List<Float>
-linearAccZ : List<Float>
-linearAcceleroVal : float[]
-results : float[]
-classifier : TensorFlowClassifier2
-data : List<Float> = new ArrayList<>()
-collectTimer : Timer
-logTimer : Timer
-collectTimerTask : TimerTask
-logTimerTask : TimerTask
-startPredictButton : Button
-stopPredictButton : Button
-tts : TextToSpeech

#onCreate(savedInstanceState : Bundle) : void
#onStart() : void
#onPause() : void
#onResume() : void
#onStop() : void
+onSensorChanged(event : SensorEvent) : void
+onAccuracyChanged2(sensor : Sensor, accuracy : int) : void
-speak(speech : String) : void
+startPredict(view : View) : void
+stopPredict(view : View) : void
-toFloatArray(list : List<Float>) : float[]
+round(d : float, decimalPlace : d) : float
-getSensorManager() : SensorManager
```

Figure 3.5.15: PredictMorseCodeActivity

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

SettingsActivity

SettingsActivity
+KEY_PREF_GYROSCOPE_SWITCH : String = "gyroscope_switich" +KEY_PREF_ACCELEROMETER_SWITCH : String = "accelerometer_switich" +KEY_PREF_MAGNETOMETER_SWITCH : String = "magnetometer_switch" +KEY_PREF_ORIENTATION_SWITCH : String = "orientation_switch" +KEY_PREF_GRAVITY_SWITCH : String = "gravity_switch" +KEY_PREF_LINEAR_ACCELEROMETER_SWITCH : String = "linear_accelerometer_switch" +KEY_PREF_LPF_LINEAR_ACC_SWITCH : String = "lpf_linear_acc_switch" +KEY_PREF_PROXIMITY_SWITCH : String = "proximity_switch" +KEY_PREF_TIME_LABEL_INTERVAL : String = "time_label_interval" +KEY_PREF_TIME_LOG_INTERVAL : String = "time_logging_interval" +KEY_PREF_SENSOR_SAMPLING_DELAY : String = "sensor_sampling_delay" +KEY_PREF_COLLECT_MODE : String = "collect_mode" +KEY_PREF_LOG_TIMER_SWITCH : String = "log_timer_switch" +KEY_PREF_TIMER_MODE : String = "timer_mode" +KEY_PREF_TEXT_TO_SPEECH : String = "texttospeech_switch"
#onCreate(savedInstanceState : Bundle) : void #onStop() : void

Figure 3.5.16: SettingsActivity

SettingsFragment

SettingsFragment
-gyroscopeSetting : SwitchPreferenceCompat -accelerometerSetting : SwitchPreferenceCompat -magnetometerSetting : SwitchPreferenceCompat -orientationSetting : SwitchPreferenceCompat -gravitySetting : SwitchPreferenceCompat -linearaccelerometerSetting : SwitchPreferenceCompat -proximitySetting : SwitchPreferenceCompat -logTimerSetting : SwitchPreferenceCompat -lowPassFilterAccSetting : CheckBoxPreference -linearAccUnsupported : boolean -timeLabelIntervalSetting : IntEditTextPreference -timeLoggingIntervalSetting : IntEditTextPreference -sensorSamplingDelaySetting : IntEditTextPreference -collectModeSetting : ListPreference -timerModeSetting : ListPreference -textToSpeechSetting : SwitchPreferenceCompat -sensorManager : SensorManager -gyroscope : Sensor -accelerometer : Sensor -magnetometer : Sensor -gravmeter : Sensor -linearaccelerometer : Sensor -proximity : Sensor -prefChangeListener : OnPreferenceChangeListener
+onCreatePreference(savedInstanceState : Bundle, rootKey : String) : void

Figure 3.5.17: SettingsFragment

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

S3Handler

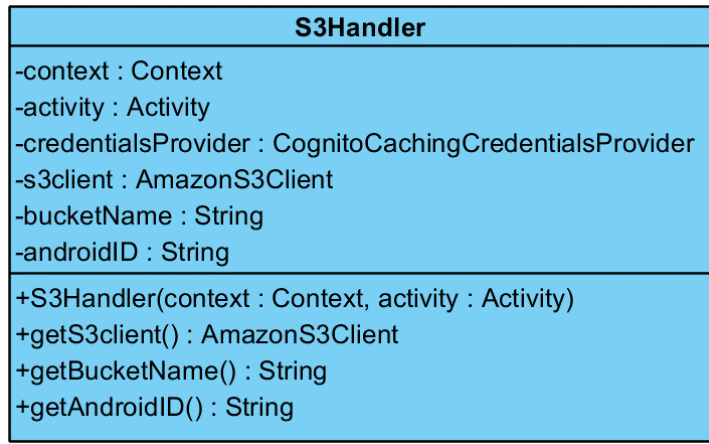


Figure 3.5.18: S3Handler

TensorFlowClassifier

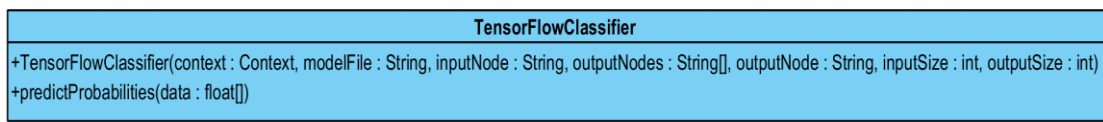
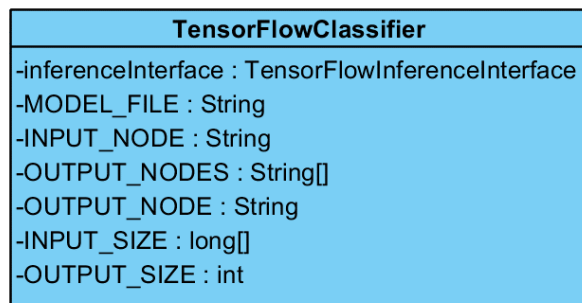


Figure 3.5.19: TensorFlowClassifier

3.6 Entity Relationship Diagram




















sensordata		
	ID	integer(10) U
	timestamp	integer(10)
	label	varchar(20)
	gyroX	real(30)
	gyroY	real(30)
	gyroZ	real(30)
	accX	real(10)
	accY	real(10)
	accZ	real(10)
	magX	real(10)
	magY	real(10)
	magZ	real(10)
	azimut	real(10)
	pitch	real(10)
	roll	real(10)
	linearAccX	real(10)
	linearAccY	real(10)
	linearAccZ	real(10)
	proximity	integer(10)

Figure 3.6.1: Entity Relationship Diagram

A table is created after creating a new database upon application start up. The application will determine data elements / fields to be written to the table through the user settings. For instance, if the user had ticked the gyroscope switch in the user settings, the database is created with the additional data elements of gyroX, gyroY and gyroZ. In the other hands, if the user had ticked the magnetometer switch and linear acceleration switch in the user settings, the table would have the data elements of magX, magY and magZ as well as linearAccX, linearAccY and linearAccZ. The database, namely 'CurInstSensorDB' is created under SQLite database version 3.0 and above, depending on which Android version the smartphone has.

Additionally, the data types allocated for the attributes within the entities were different as stated above. Noticed that there is only fixed size length for each data types in SQLite database. Thus, the data type length on the diagram above can be safely ignored. Also, in the software (Visual Paradigm) that the diagram was created, each of the sensor data type is of REAL (except Proximity) while in the application, they are treated as REAL because float value is considered as REAL in SQLite database.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

3.7 Sequence Diagram

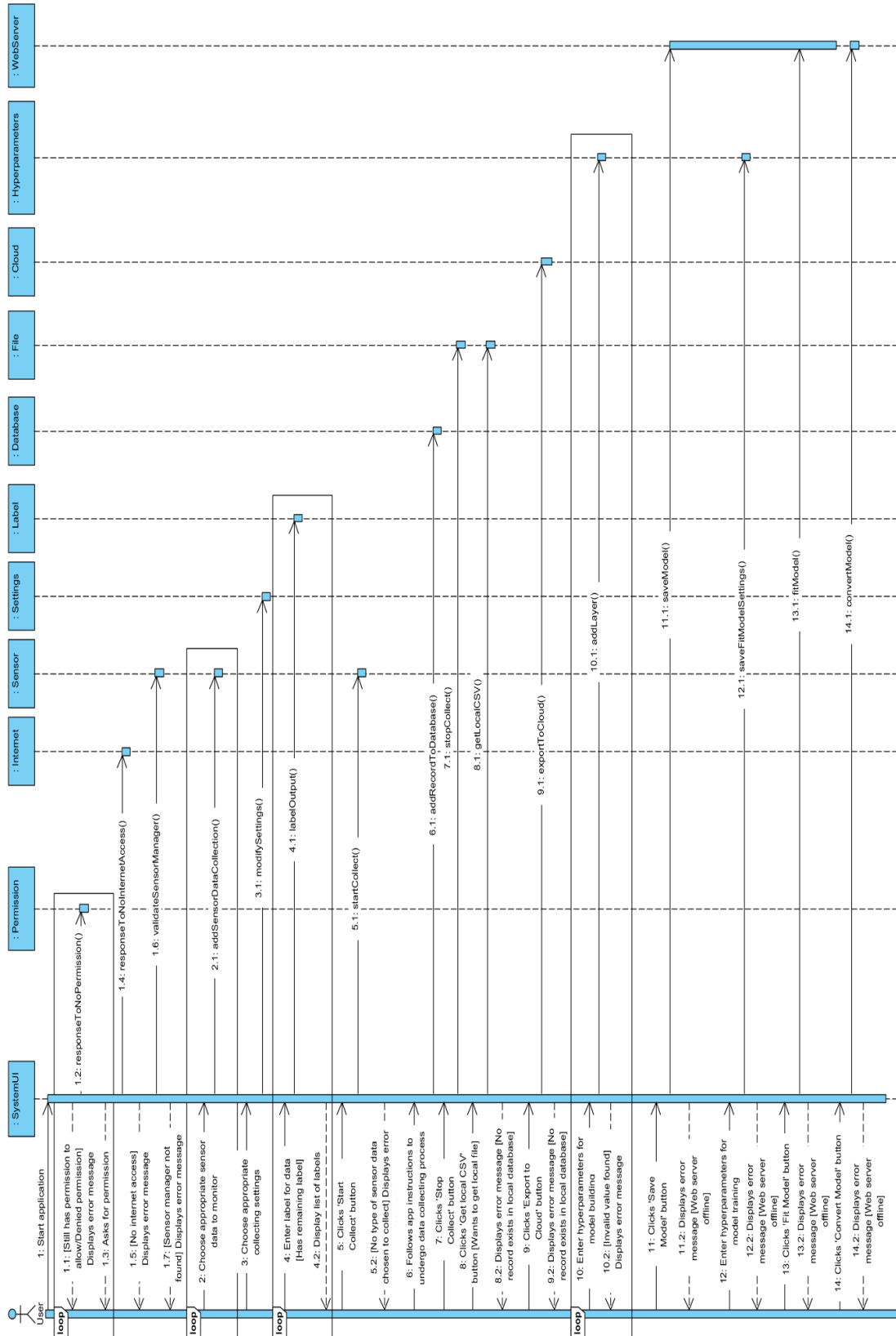


Figure 3.7.1: Sequence Diagram

3.8 Communication Diagram

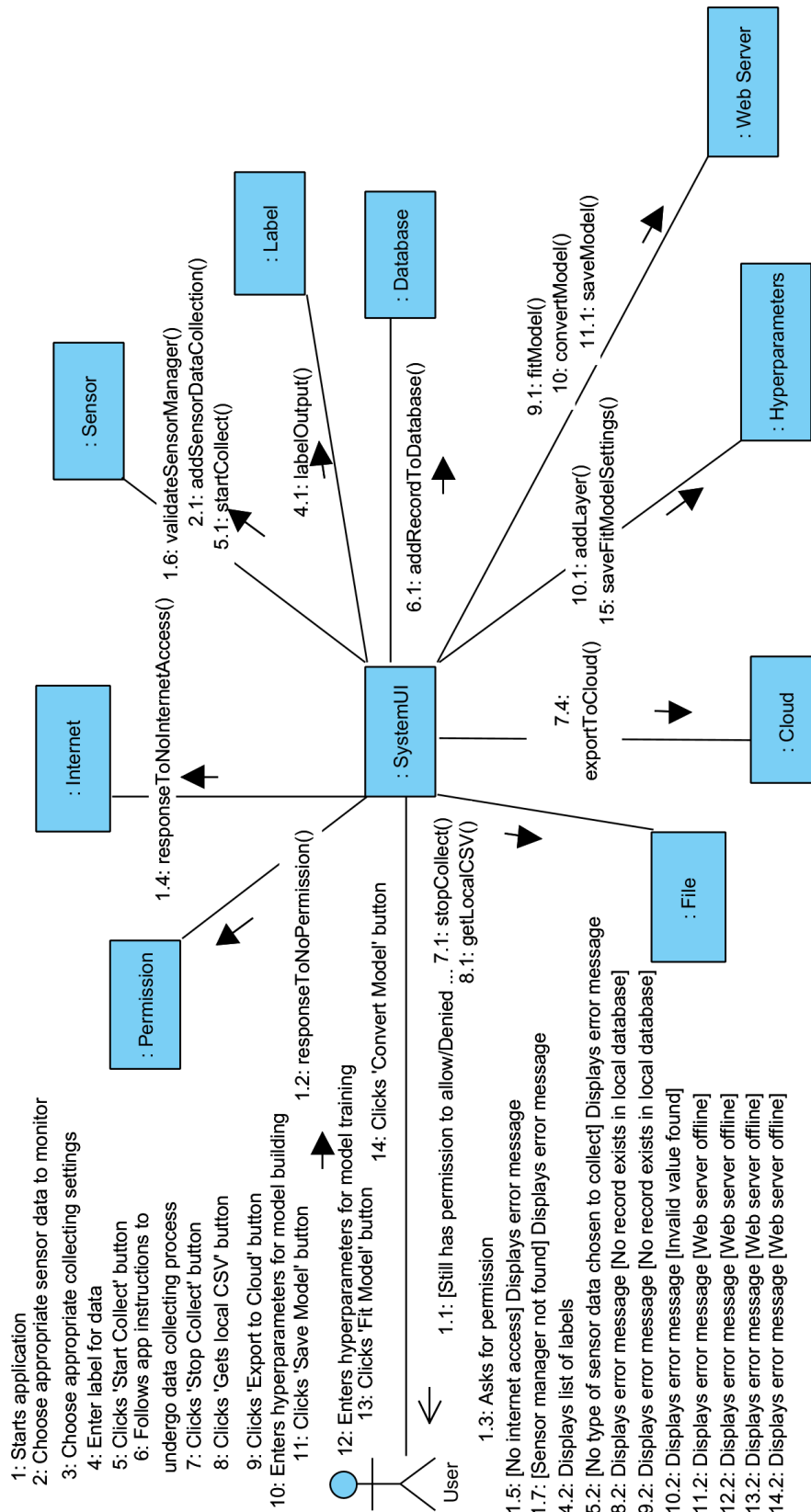


Figure 3.8.1: Communication Diagram

3.9 State Diagram

Sensor

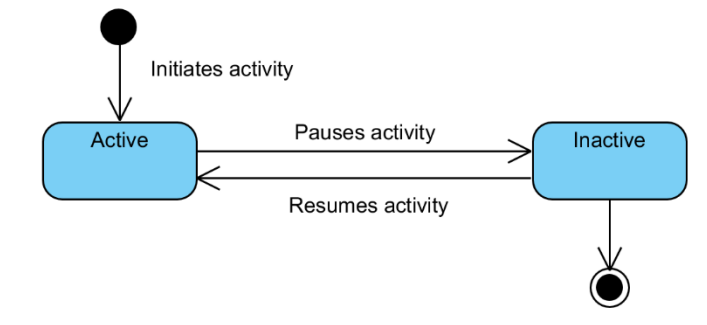


Figure 3.9.1: Sensor State Diagram

The sensors available in the smartphone is active when initiating activities such as CheckSensorActivity, MainActivity, PredictGestureActivity, PredictMorseCodeActivity and SettingsFragment. Then, the sensors become inactive once the activities are stopped.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Database

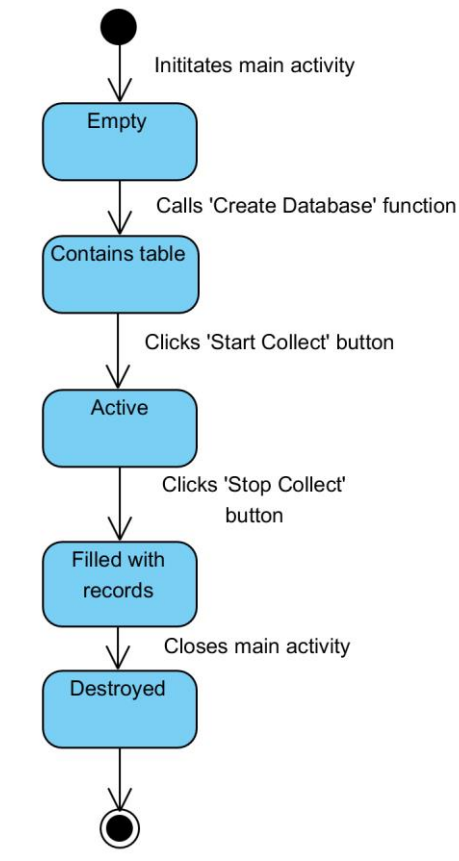


Figure 3.9.2: Database State Diagram

An empty database is created after initiating MainActivity. When the Create Database function is being called, the database creates a table called 'CurInstSensorDB'. The table would later be storing thousands or millions of records when the collection process starts with 'Start Collect' button and stops after the 'Stop Collect' button is clicked. The database will then be destroyed once the MainActivity is stopped.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Dataset File

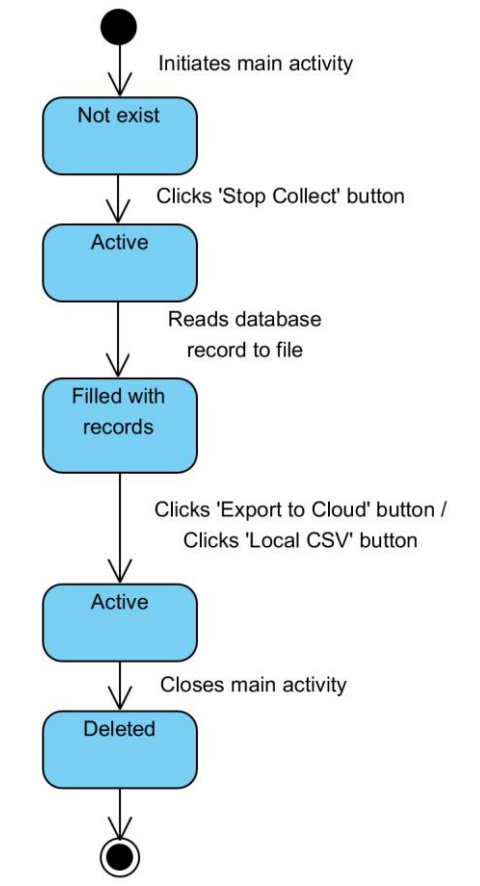


Figure 3.9.3: Dataset File State Diagram

A record file does not create right after initiating MainActivity. After clicking 'Stop Collect' button, the record file will be created and reading database records and the records will be written to file. By clicking 'Export to Cloud' button or clicking 'Local CSV' button, the record file will be sent to the cloud storage. The record file is deleted after closing MainActivity.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Python Code File / Settings File

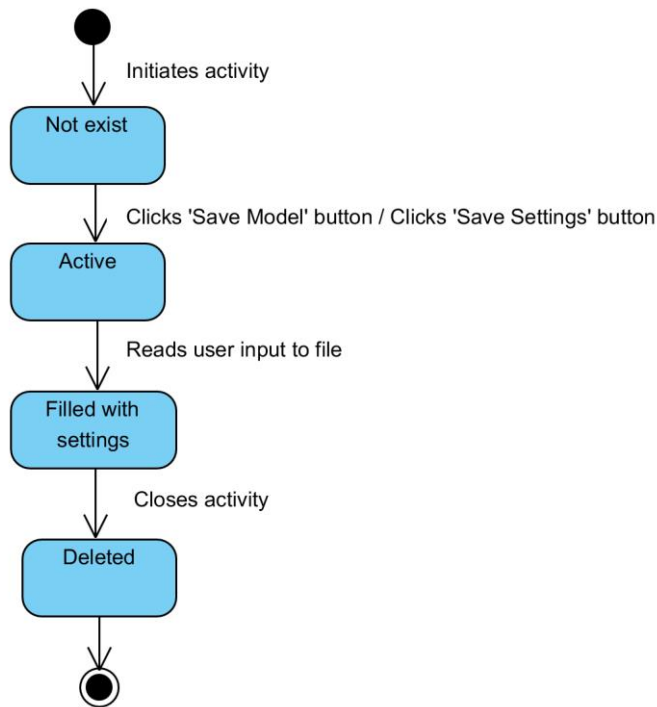


Figure 3.9.4: Python Code File / Settings File State Diagram

A python code file or settings file does not create right after initiating SettingsActivity. For the python code file, after clicking 'Save Model' button, the python code will be constructed line by line and sent to the python code file which then will be used to tell the online web server to run the following code. Each of the line contains model configuration. On the other hand, for the settings file, after clicking 'Save Settings' button, the settings file will be created, storing the training hyperparameters. The record file is deleted after closing MainActivity.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Cloud

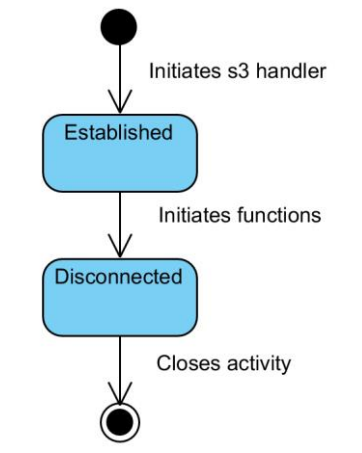


Figure 3.9.5: Cloud State Diagram

Once the S3Handler class is being initialized, the application will establish connection with the cloud storage which then be used in the functions in multiple activities such as MainActivity, ManageFileActivity and NetworkSettingsActivity. When the activity closes, the S3Handler class is destroyed, disconnecting the connection between the application and the cloud storage.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Web Server

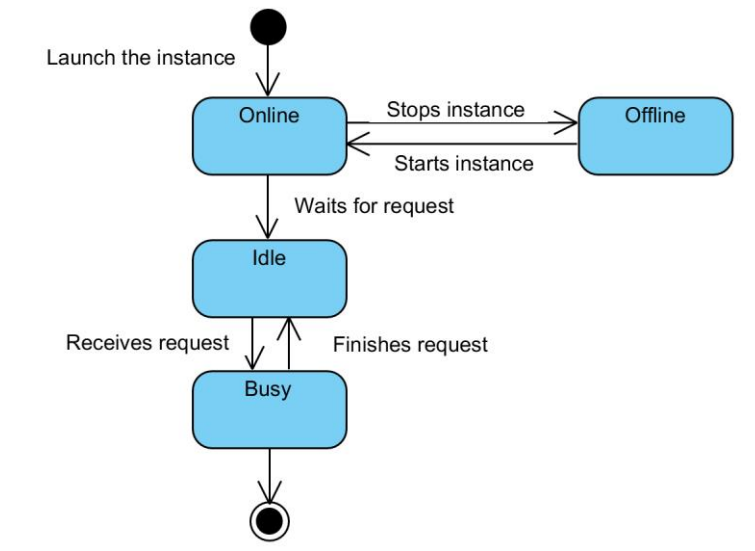


Figure 3.9.5: Web Server State Diagram

The web server becomes online once the instance/web server is launched or starts and can become offline if the instance is stopped. The instance waits for request in an idle state. Then, it receives request from some functions in FitModelActivity and NetworkSettingsActivity and becomes busy, and once it finishes request, it goes back to idle state.

CHAPTER 4 IMPLEMENTATION

4.1 Tools

The tools used in the project are Android Studio, an IDE for the development of the Android application. During the development of the Android application, a temporary SQLite database is constructed to store the raw sensor data of the smartphone and the database records will be deleted upon closing the application. OkHttpClient acts as a HTTP client for the android application in order to submit the GET request to the Amazon S3 cloud storage for raw dataset storage as well as to the deep learning Amazon EC2 cloud instance which is an Apache web server with Flask web application. For the development of the Python deep learning script, PyScripter on Window is used with the Tensorflow and Keras backend. The Python deep learning script is then deployed on the deep learning cloud instance to serve multiple users. Amazon Cognito acts as a credential provider service which provides each smartphone users with an unauthenticated credential which helps them to create an individual folder depending on their Android ID. Additionally, to speed up the model training, the Google Colab is used.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

4.2 Requirements

For the Android application, it supports a minimum operating system version of Android 5.0 (API level 21) up to maximum operating system version of Android 8.1 (API level 27) on the smartphone. Other than that, the Android application requires working sensor and internet connectivity on the Android smartphone since the Android application needs to collect data about the sensor as well as upload the raw sensor data in a csv format to the cloud. To prevent any issues during the development, the Android Studio used to develop the application is of same version from the beginning of the project development until the end.

```
compileSdkVersion 27
defaultConfig {
    applicationId "unclesave.example.com.test2"
    minSdkVersion 21
    targetSdkVersion 27
}
```

Figure 4.2.1: SDK Version

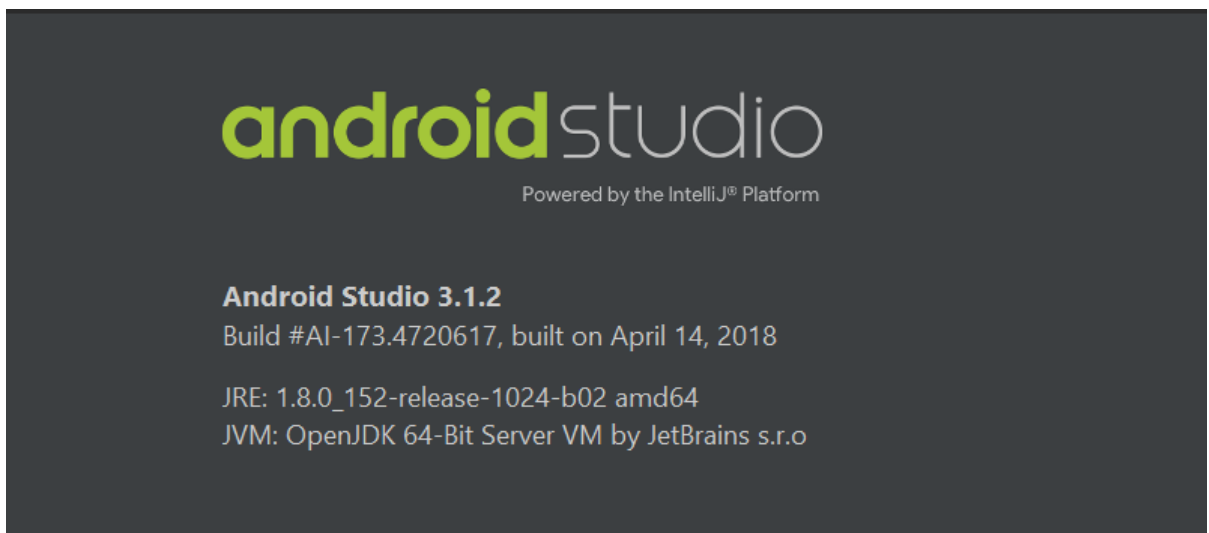


Figure 4.2.2: Android Studio Version

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Packages or libraries used for the development of the mobile application are as below:

```
implementation 'com.amazonaws:aws-android-sdk-core:2.2.+'
implementation 'com.amazonaws:aws-android-sdk-s3:2.6.+'
implementation 'com.amazonaws:aws-android-sdk-cognito:2.6.+'
implementation('com.amazonaws:aws-android-sdk-cognitoauth:2.6.+@aar') { transitive = true }
implementation('com.amazonaws:aws-android-sdk-mobile-client:2.6.+@aar') { transitive = true }
implementation 'com.amazonaws:aws-android-sdk-apigateway-core:2.6.+'
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'org.tensorflow:tensorflow-android:+'
```

Figure 4.2.3: Used Android libraries

As observed in the previous diagram, the version of the OKHttpClient used is 3.11.0. For the SQLite database embedded on the Android application, each of the SQLite database is of different version depending on the Android version of the smartphone. The table for SQLite version corresponding to specified Android version is shown below:

Android API version	SQLite version
API 27 (8.1)	3.19.4
API 26 (8.0)	3.18.2
API 25/24 (7.1.1/7.0)	3.9.2
API 23 (6.0)	3.8.10.2
API 22 (5.1.1)	3.8.6.1
API 21 (5.0)	3.8.6

Table 4.2.4: Android API/SQLite Version

```
(tensorflow_p36) ubuntu@ip-172-31-25-132:~/flaskproject$ python -V
Python 3.6.6 :: Anaconda, Inc.
```

Figure 4.2.5: Python Version

```
(tensorflow_p36) ubuntu@ip-172-31-25-132:~/flaskproject$ apache2 -v
Server version: Apache/2.4.18 (Ubuntu)
Server built: 2018-06-07T19:43:03
```

Figure 4.2.6: Apache Version

```
(tensorflow_p36) ubuntu@ip-172-31-25-132:~/flaskproject$ dpkg -l | grep wsgi
ii  libapache2-mod-wsgi-py3      4.3.0-1.1build1
    amd64                    Python 3 WSGI adapter module for Apache
```

Figure 4.2.7: mod_wsgi Version

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

For the programming languages, Python 3.6.4 is used on the web server running the script. For the Tensorflow and Keras, Tensorflow is of version 1.9.0 and Keras is of version 2.2.0. For the deep learning web server instance hosted on EC2, it is of type t2.micro and has 2.5GHz unknown Intel Xeon model CPU and 1GB memory. For the Flask web application, it is of 1.0.2. Apache is of version 2.4.18 while mod_wsgi is of version 4.3.0-1.1build1.

4.3 Framework Architecture

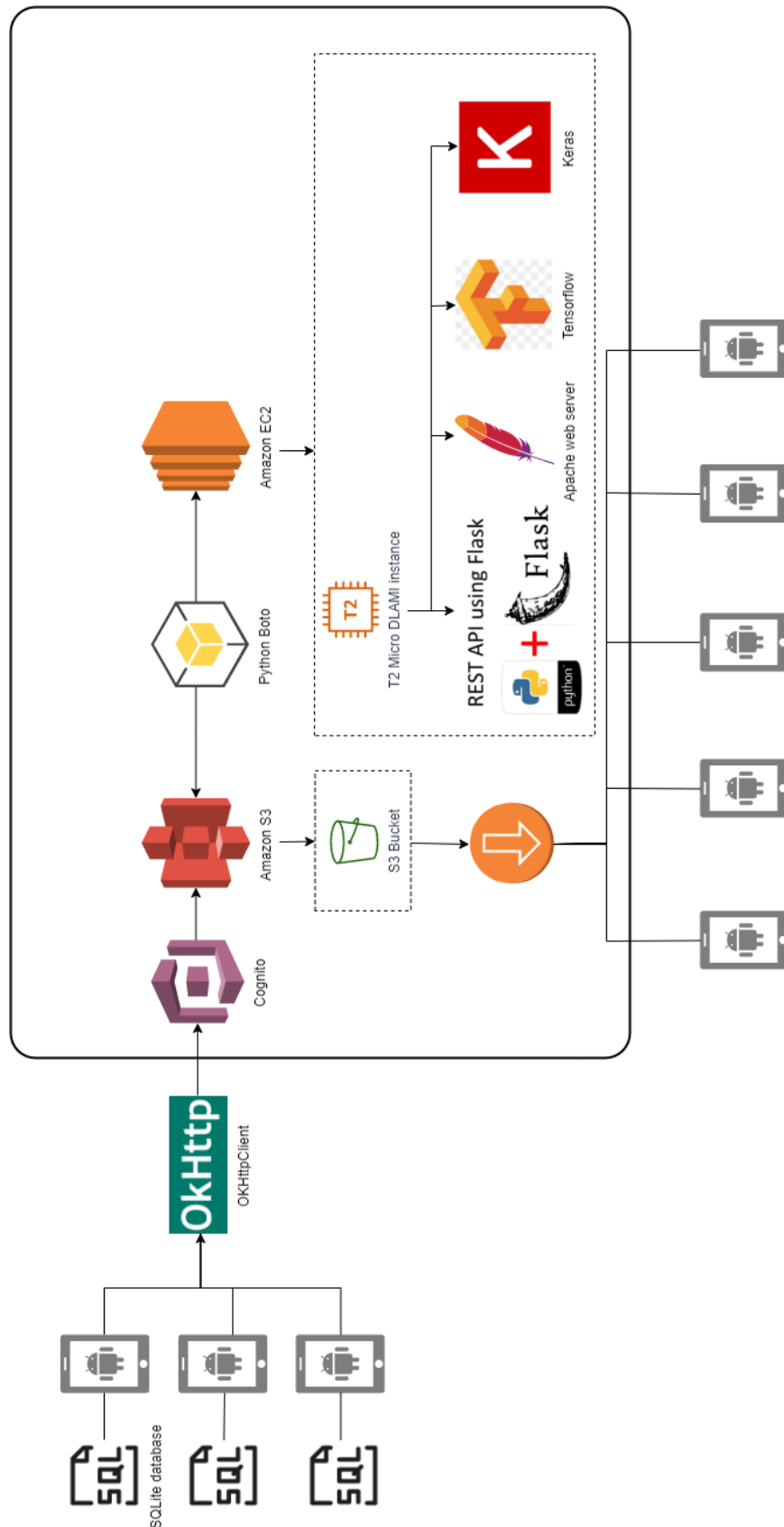


Figure 4.3.1: Framework Architecture

CS (Hons) Computer Science

Faculty of Information and Communication Technology (Perak Campus), UTAR.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

There are several platforms implemented for the project, Amazon Web Services (EC2, S3), Java (Android and SQLite Database) and Python (Tensorflow and Keras backend). First, the application starts. A new SQLite local database is created upon application start up. When the data collecting process starts, the records are appended into the SQLite local database. Followed by newly created records, a new file in csv format is created, all of the records are copied from SQLite local database to the file. After that, the file is being uploaded into one of the user folders in Amazon Simple Storage Service (S3) provided by Amazon Web Service. The user folder containing uploaded file is named after the Android smartphone ID. Afterwards, the application sends a simple HTTP GET request containing the smartphone ID with the help of OKHTTPClient to the deep learning instance setup on the Amazon Elastic Compute Cloud provided by Amazon Web Service, giving the script the capability of finding dataset in their user folder on cloud. The deep learning instance acts as an Apache web server which consists of Flask, a Python micro framework/web application. The deep learning instance executes the script which uses the dataset uploaded on the cloud to build and training the neural network model. After the model is being trained, the training output is logged and sent back to S3. If the training output is good, the Keras model is ready to be converted to Tensorflow model for actual deployment.

4.4 Resolved Implementation Issues and Challenges

Few issues and challenges were faced during the implementation as shown below:

1. IntEditTextPreferences and EditTextPreferences

EditTextPreferences is used in Settings. Originally, for EditText to be accepting the integer input, `android:inputType=number` needs to be specified in xml file. But for EditTextPreferences, the specified line does not do anything. Thus, IntEditTextPreferences functioned as the custom extension library of the original one. It provides the capability to allow integer input instead of String input.

2. Background scripting

Originally, the python script was meant to be executed by one of the services handled by Amazon Web Service, that is Lambda. However, AWS Lambda provided a size limit on deployment packages/libraries. In order for the python script to be executed, several libraries or packages need to be included along with the script. The size of python libraries needed for the script is exceeded in a way that AWS Lambda couldn't support it. The maximum size of the deployment package supported by AWS Lambda must be less than or equal to 250mb while the size of python libraries needed for the script is approximately 300-450mb. Thus, this can't be implemented due to the factor of storage size limit.

Another way to implement the python script execution in the background is by using the AWS Cloudwatch. AWS Cloudwatch acts as a listener for the AWS S3, a cloud storage which stores the csv file uploaded through the Android application. As soon as the csv file is uploaded, the AWS Cloudwatch will follow the procedure, calling an instance specified for Deep Learning from AWS EC2 to execute the python script. However, the idea is hindered by the AWS Cloudwatch. AWS Cloudwatch tends to have delayed notifications on the uploaded file, which leads to delayed execution of the python script. Thus, this too can't be implemented due to the factor of timing.

Finally, a solution to that is to use a web server to host a web application full of scripting which the mobile application can request the web server to do specified task.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

3. AWS Mobile Hub and AWS Cognito

Due to migration of AWS Mobile Hub to AWS Amplify during the middle of the project development, unauthenticated AWS Cognito is used instead. This allows the mobile application to gain temporary credential to use some AWS services such as access to S3 cloud storage and EC2 web server instance.

4. Slow insertion of record into SQLite database

The slow insertion of record causes trouble such as writing to file performance issues and record duplication issues. In order to solve this, a precompiled SQL insert statement is created first before the execution as well as the statement of `beginTransaction()/setTransactionSuccessful()/endTransaction()` is used. Other than that, `StringBuilder` is also used to replace `String` to speed up the performance.

5. Multiple execution of deep learning codes on web server causing Internal Server Error 500

This error is 'Tensor (...)' is not an element of this graph. This usually happens after first time executing the code. This can be solved by encapsulating the deep learning code with code shown below:

```
global graph
with graph.as_default():
```

```
graph = tf.get_default_graph()
```


4.5 Unresolved Implementation Issues and Challenges

1. Predict Morse Code

The data collecting processes went well until the model training part. The prediction is not accurate due to some possible reasons: overfitting, lack of training data or lack of useful features.

CHAPTER 5 TESTING AND EXPERIMENTAL RESULTS

5.1 Features to be Tested

1. UC105 - Configure collecting settings
2. UC109 - Configure deep learning model's hyperparameters

5.2 Features not to be Tested

Other aspects of the system are not covered in this testing section. This includes:

1. Operation procedure - This testing section is for system level test, operation procedure is not covered here.
2. Network security - This testing section is designed for functional test, security is not covered here.

5.3 Approach

This test will also be a black-box test where test cases are derived using black box testing techniques such as:

1. Equivalence partitioning
2. Boundary value analysis

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

5.4 Environment and Infrastructure

Hardware

Hardware Name	Mi Max 2
Manufacturer	Xiaomi
Processor	Qualcomm Snapdragon 625 2.0GHz
Graphics Card	Adreno 506
RAM	4GB
Storage	64GB

Table 5.4.1: Hardware Environment

Software

Application Name	Sensordyne
Version	1.0
Type of File	Installable Android Application Package (.apk)
Size of Application	43.9 MB
Size on Disk	44.7 MB
Operating System	Android 7.1.1 Nougat

Table 5.4.2: Software Environment

5.5 Configure Collecting Settings

Equivalence Partitioning

Equivalence partitioning	Boundary value analysis
<u>timeLabelIntervalSettings</u>	
Valid	timeLabelIntervalSettings > 2000 – Valid
Valid	timeLabelIntervalSettings = 2000 - Valid
Invalid	timeLabelIntervalSettings < 2000 - Invalid
<u>timeLoggingIntervalSettings</u>	
Valid	timeLoggingIntervalSettings > 10 – Valid
Valid	timeLoggingIntervalSettings = 10 - Valid
Invalid	timeLoggingIntervalSettings < 10 - Invalid
<u>sensorSamplingDelaySettings</u>	
Valid	sensorSamplingDelaySettings > 0 – Valid
Valid	sensorSamplingDelaySettings = 0 - Valid
Invalid	sensorSamplingDelaySettings < 0 - Invalid

Table 5.5.1: Equivalence Partitioning for Configuring Collecting Settings Part 1

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Test Condition	Test Coverage	Test Data
-ve < timeLabelIntervalSettings < 2000	-ve < timeLoggingIntervalSettings < 2000 (Invalid)	-1
timeLabelIntervalSettings = 2000	timeLabelIntervalSettings = 2000 (Valid)	2000
2000 < timeLabelIntervalSettings < +ve	2000 < timeLabelIntervalSettings < +ve (Valid)	5000
Valid timeLabelIntervalSettings	Valid timeLabelIntervalSettings (Valid output)	7000
Invalid timeLabelIntervalSettings	Invalid timeLabelIntervalSettings (Valid output)	1500
-ve < timeLoggingIntervalSettings < 10	-ve < timeLoggingIntervalSettings < 10 (Invalid)	-5
timeLoggingIntervalSettings = 10	timeLoggingIntervalSettings = 10 (Valid)	10
10 < timeLoggingIntervalSettings < +ve	10 < timeLoggingIntervalSettings < +ve (Valid)	50
Valid timeLoggingIntervalSettings	Valid timeLoggingIntervalSettings (Valid output)	20
Invalid timeLoggingIntervalSettings	Invalid timeLoggingIntervalSettings (Valid output)	5
-ve < sensorSamplingDelaySettings < 0	-ve < sensorSamplingDelaySettings < 0 (Invalid)	-100
sensorSamplingDelaySettings = 0	sensorSamplingDelaySettings = 0	0
0 < sensorSamplingDelaySettings < +ve	0 < sensorSamplingDelaySettings < +ve (Valid)	5000
Valid sensorSamplingDelaySettings	Valid sensorSamplingDelaySettings (Valid output)	10000
Invalid sensorSamplingDelaySettings	Invalid sensorSamplingDelaySettings (Invalid output)	-
		10000

Table 5.5.2: Equivalence Partitioning for Configuring Collecting Settings Part 2

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Boundary Value Analysis

Test Condition	Test Coverage	Test Data
-ve < timeLabelIntervalSettings < 2000	-ve < timeLabelIntervalSettings < 2000 (Invalid)	1999
timeLabelIntervalSettings = 2000	timeLabelIntervalSettings = 2000 (Valid)	2000
2000 < timeLabelIntervalSettings < +ve	2000 < timeLabelIntervalSettings < +ve (Valid)	2001
Valid timeLabelIntervalSettings	Valid timeLabelIntervalSettings (Valid output)	7000
Invalid timeLabelIntervalSettings	Invalid timeLabelIntervalSettings (Valid output)	1500
-ve < timeLoggingIntervalSettings < 10	-ve < timeLoggingIntervalSettings < 10 (Invalid)	9
timeLoggingIntervalSettings = 10	timeLoggingIntervalSettings = 10 (Valid)	10
10 < timeLoggingIntervalSettings < +ve	10 < timeLoggingIntervalSettings < +ve (Valid)	11
Valid timeLoggingIntervalSettings	Valid timeLoggingIntervalSettings (Valid output)	20
Invalid timeLoggingIntervalSettings	Invalid timeLoggingIntervalSettings (Valid output)	5
-ve < sensorSamplingDelaySettings < 0	-ve < sensorSamplingDelaySettings < 0 (Invalid)	-1
sensorSamplingDelaySettings = 0	sensorSamplingDelaySettings = 0	0
0 < sensorSamplingDelaySettings < +ve	0 < sensorSamplingDelaySettings < +ve (Valid)	1
Valid sensorSamplingDelaySettings	Valid sensorSamplingDelaySettings (Valid output)	10000
Invalid sensorSamplingDelaySettings	Invalid sensorSamplingDelaySettings (Invalid output)	- 10000

Table 5.5.3: Boundary Value Analysis for Configuring Collecting Settings

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Test Procedure Setup

1. Press on the Label interval (millisecond) box in the Settings section.
2. User inputs -1 (-ve < timeLabelInterval < 2000).
3. User presses OK.
4. Repeat Steps 1 to 3 with user input 1999 (-ve < timeLabelInterval < 2000).
5. Repeat Steps 1 to 3 with user input 2000 (timeLabelInterval = 2000).
6. Repeat Steps 1 to 3 with user input 2001 (2000 < timeLabelInterval < +ve).
7. Repeat Steps 1 to 3 with user input 5000 (2000 < timeLabelInterval < +ve).
8. Repeat Steps 1 to 3 with user input 7000 (Valid timeLabelInterval).
9. Repeat Steps 1 to 3 with user input 1500 (Invalid timeLabelInterval).
10. Press on the Log data interval (millisecond) box in the Settings section.
11. User inputs -5 (-ve < timeLoggingInterval < 10).
12. User presses OK.
13. Repeat Steps 10 to 12 with user input 9 (-ve < timeLoggingInterval < 10).
14. Repeat Steps 10 to 12 with user input 10 (timeLoggingInterval = 10).
15. Repeat Steps 10 to 12 with user input 11 (10 < timeLoggingInterval < +ve).
16. Repeat Steps 10 to 12 with user input 50 (-ve < timeLoggingInterval < 10).
17. Repeat Steps 10 to 12 with user input 9 (-ve < timeLoggingInterval < 10).
18. Repeat Steps 10 to 12 with user input 20 (Valid timeLoggingInterval).
19. Repeat Steps 10 to 12 with user input 5 (Invalid timeLoggingInterval).
20. Press on the Sensor sampling delay (microsecond) box in the Settings section.
21. User inputs -100 (-ve < sensorSamplingDelaySettings < 0).
22. User presses OK.
23. Repeat Steps 20 to 22 with user input -1 (-ve < sensorSamplingDelaySettings < 0).
24. Repeat Steps 20 to 22 with user input 0 (sensorSamplingDelaySettings = 0).
25. Repeat Steps 20 to 22 with user input 1 (0 < sensorSamplingDelaySettings < +ve).
26. Repeat Steps 20 to 22 with user input 5000 (0 < sensorSamplingDelaySettings < +ve).
27. Repeat Steps 20 to 22 with user input 10000 (Valid sensorSamplingDelaySettings).
28. Repeat Steps 20 to 22 with user input -10000 (Invalid sensorSamplingDelaySettings).

5.6 Configure Deep Learning Model's Hyperparameters

Equivalence Partitioning

Equivalence partitioning	Boundary value analysis
<u>unit</u>	
Valid	unit > 0 – Valid
Invalid	unit = 0 - Invalid
<u>dropout</u>	
Valid	dropout < 1.0 – Valid
Valid	dropout = 1.0 - Valid
Invalid	dropout > 1.0 - Invalid

Table 5.6.1: Equivalence Partitioning for Configuring Deep Learning Model's Hyperparameters Part 1

Test Condition	Test Coverage	Test Data
0 < unit < +ve	0 < unit < +ve (Valid)	64
unit = 0	unit = 0 (Invalid)	0
0 < dropout < 1.0	0 < dropout < 1.0 (Valid)	0.5
dropout = 1.0	dropout = 1.0 (Valid)	1.0
1.0 < dropout < +ve	1.0 < dropout < +ve (Invalid)	1.5

Table 5.6.2: Equivalence Partitioning for Configuring Deep Learning Model's Hyperparameters Part 1

Boundary Value Analysis

Test Condition	Test Coverage	Test Data
0 < unit < +ve	0 < unit < +ve (Valid)	1
unit = 0	unit = 0 (Invalid)	0
0 < dropout < 1.0	0 < dropout < 1.0 (Valid)	0.99
dropout = 1.0	dropout = 1.0 (Valid)	1.0
1.0 < dropout < +ve	1.0 < dropout < +ve (Invalid)	1.01

Table 5.6.3: Boundary Value Analysis for Configuring Deep Learning Model's Hyperparameters

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Test Procedure Setup

1. User inputs 1 for Number of unit in Model and Neural Network Settings Section ($0 < \text{unit} < +ve$).
2. User presses 'Add Layer to Model' button.
3. Repeat Steps 1 to 2 with user input 64 ($0 < \text{unit} < +ve$).
4. Repeat Steps 1 to 2 with user input 0 ($\text{unit} = 0$).
5. User inputs appropriate value for Number of unit in Model and Neural Network Settings Section.
6. User inputs 0.5 for Dropout in Model and Neural Network Settings Section ($0 < \text{dropout} < 1.0$).
7. User presses 'Add Layer to Model' button.
8. Repeat Steps 5 to 7 with user input 0.99 ($0 < \text{dropout} < 1.0$).
9. Repeat Steps 5 to 7 with user input 1.0 ($\text{dropout} = 1.0$).
10. Repeat Steps 5 to 7 with user input 1.01 ($1.0 < \text{dropout} < +ve$).
11. Repeat Steps 5 to 7 with user input 1.5 ($1.0 < \text{dropout} < +ve$).

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Testing with Input Shape

There are multiple test cases with different combinations when testing input shape.

Test cases are shown below:

Test Case/Layer Type	Dense	LSTM
()	x	x
(,)	x	x
(,2)	x	x
(,2,)	x	x
(,2,2)	x	x
(2)	x	x
(20)	x	x
(2,)	✓	x
(20,)	✓	x
(200,,)	x	x
(200,,2)	x	x
(20,2)	✓	✓
(2,20)	✓	✓
(20,2,)	✓	✓
(20,2,2)	✓	✓
(20,20,2)	✓	✓
(20,20,20)	✓	✓

Table 5.6.4: List of Test Cases for Input Shape

During the model building phase, the input shape must be specified on the first layer. The first layer depending on which layer type it is, has different dimension requirement. The format of the layers is verified through regular expression stated in the code. For example, the Dense layer requires 2 dimensions in minimum, while the LSTM layer requires 2 dimensions in minimum too. The Dense layer does not need the second dimension to be specified, but the LSTM layer does.

5.7 Experimental Results

Unoptimized vs Optimized Data Collection

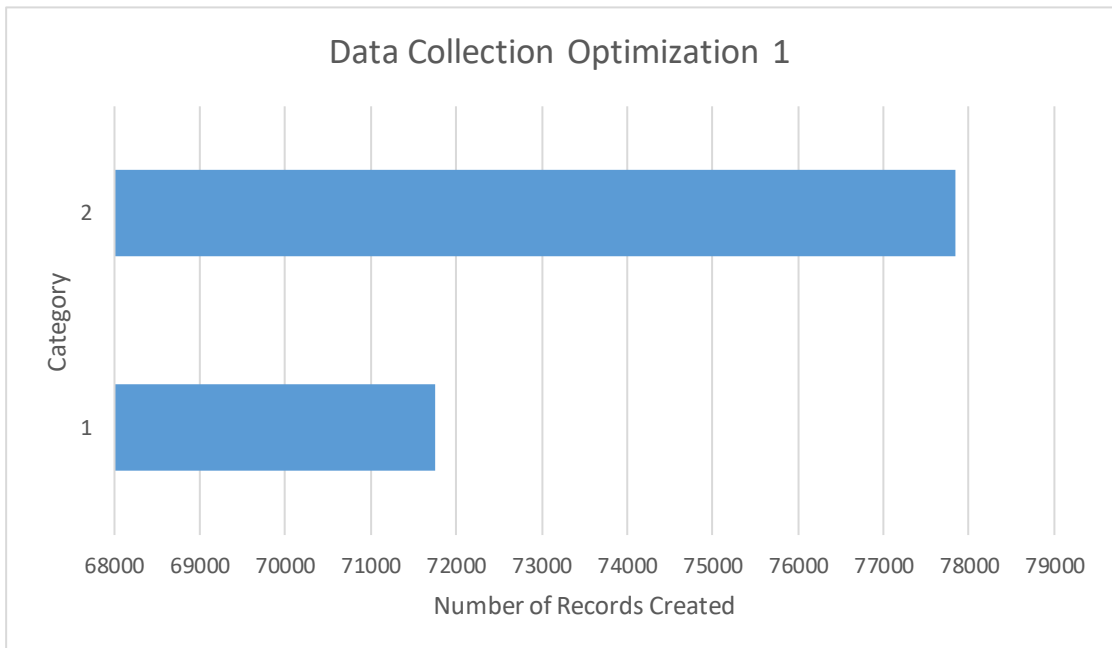


Figure 5.7.1: Records created in database over 15 minutes duration

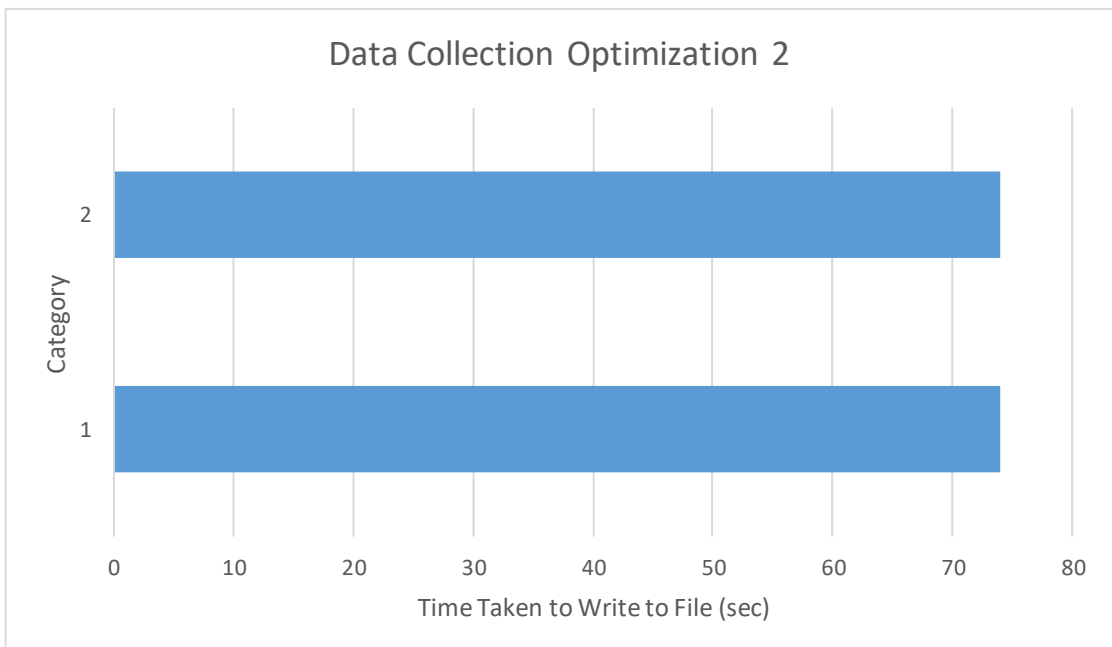


Figure 5.7.2: Time consumed to write file over 15 minutes duration

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

In terms of sensor data collection in the mobile application, the first category '1' is done using StringBuilder data type with precompiled SQL statement while the second category '2' is done using String data type without precompiled SQL statement.

Though the time taken to write a file for both categories is identical, it takes 0.0009507 sec to write a record to file for category '1' while it takes 0.0010174 sec to write a record to file for category '2', efficiently showing performance improvement about 7%.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Manual vs Automated Labelling

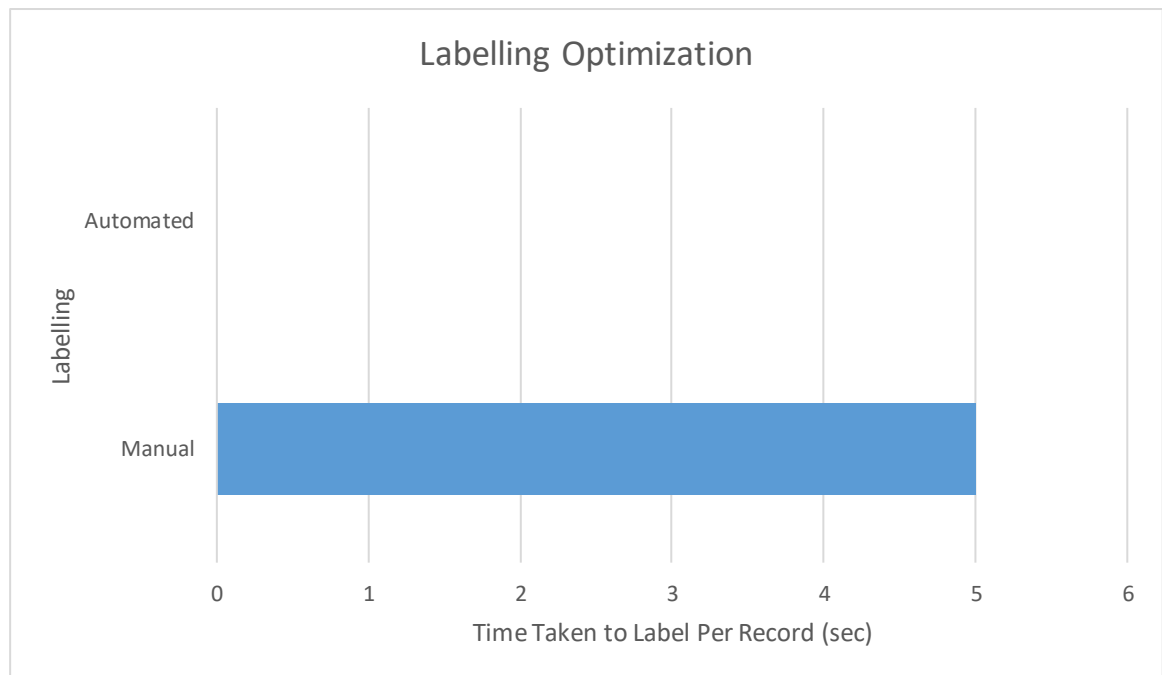


Figure 5.7.3: Time Taken to Label Single Record

The manual labelling is done through highlighting the data points as well as typing the label for each record in Microsoft Excel and labelling one by one while the automated labelling is done through the application itself. It not only minimizing the human labelling error, it also reduces the time for labelling to zero, leaving more time for the other deep learning process.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Model Building

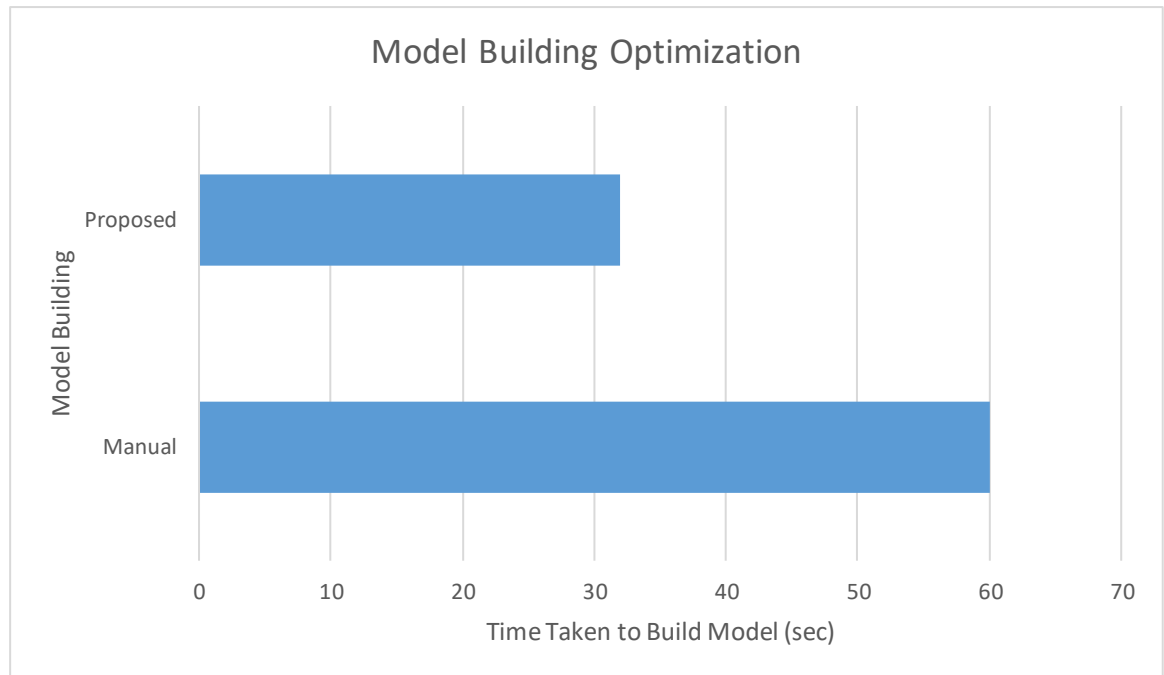


Figure 5.7.4: Time Taken to Build Model

The manual model building process is done by logging in to the web server and coding the structure of the deep learning model, while the automated model building process is done by inputting the values in the application interface, presses save button after everything is done. The proposed process actually saves the necessity of coding and more time for other phases in deep learning.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Model Training

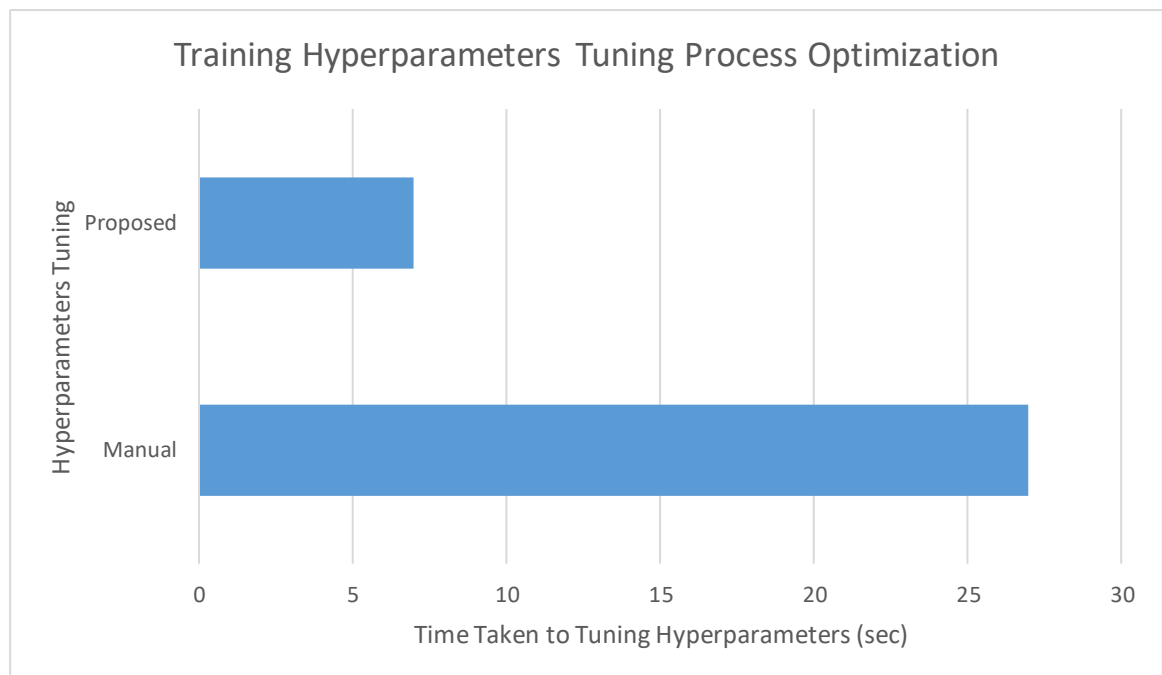


Figure 5.7.5: Time Taken to Tune Training Hyperparameters

Similarly, as the section discussed before, the manual hyperparameters tuning for model training process is done by logging in to the web server and changing the hyperparameters in the coding part while the proposed hyperparameters tuning process is done by inputting the values in the application interface and presses save button. The proposed process actually saves the necessity of coding and more time for other phases in deep learning.

5.8 Mobile Application Testing

As seen at the following figure, a clear user interface can be seen, showing multiple buttons in the applications. There are buttons such as export to cloud, build model, data labelling. First, the user can input a label for the sensor data the Android applications collected. The label inputted by the user becomes a list of labels which then allows the user to start the data collecting process. Afterwards, the list of labels becomes a randomly generated event for data collecting process later. The user can then click the start collect button in order to start the process. During the data collecting process, the user may be instructed to perform a different gesture as the instruction changes from time to time. As the user follows the instruction, the sensor data is collected then compiled in a csv file. When the user clicks the stop collecting button, the data collecting process is stopped. This leads to the csv file being uploaded to the cloud storage that is AWS S3.

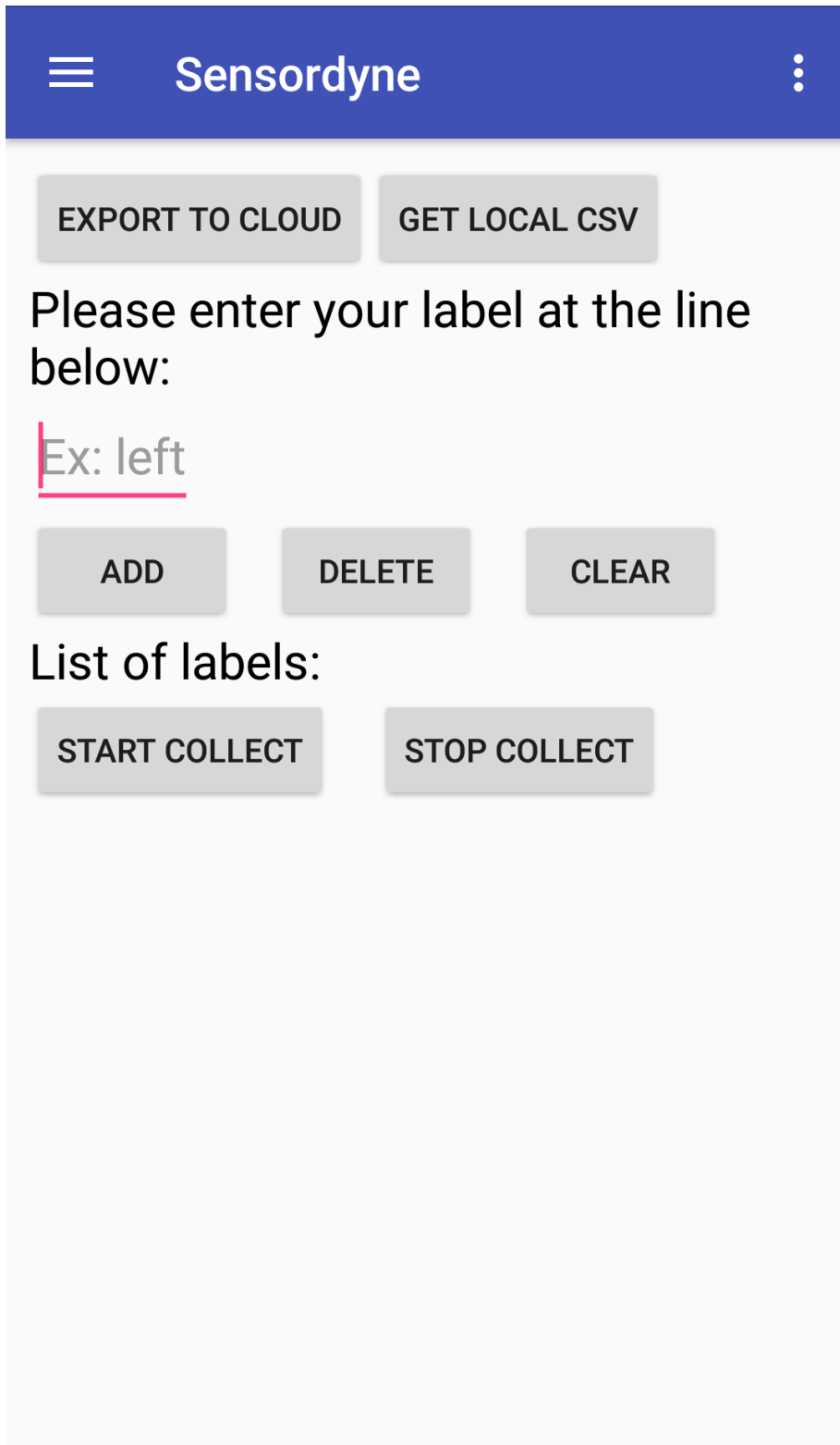


Figure 5.8.1: Main Interface

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Before the collecting process starts, the collecting settings can be freely chosen from the settings section. There are wide range of choice, such as choosing sensor data to monitor, altering label change interval, log interval, sensor delays and collect modes.

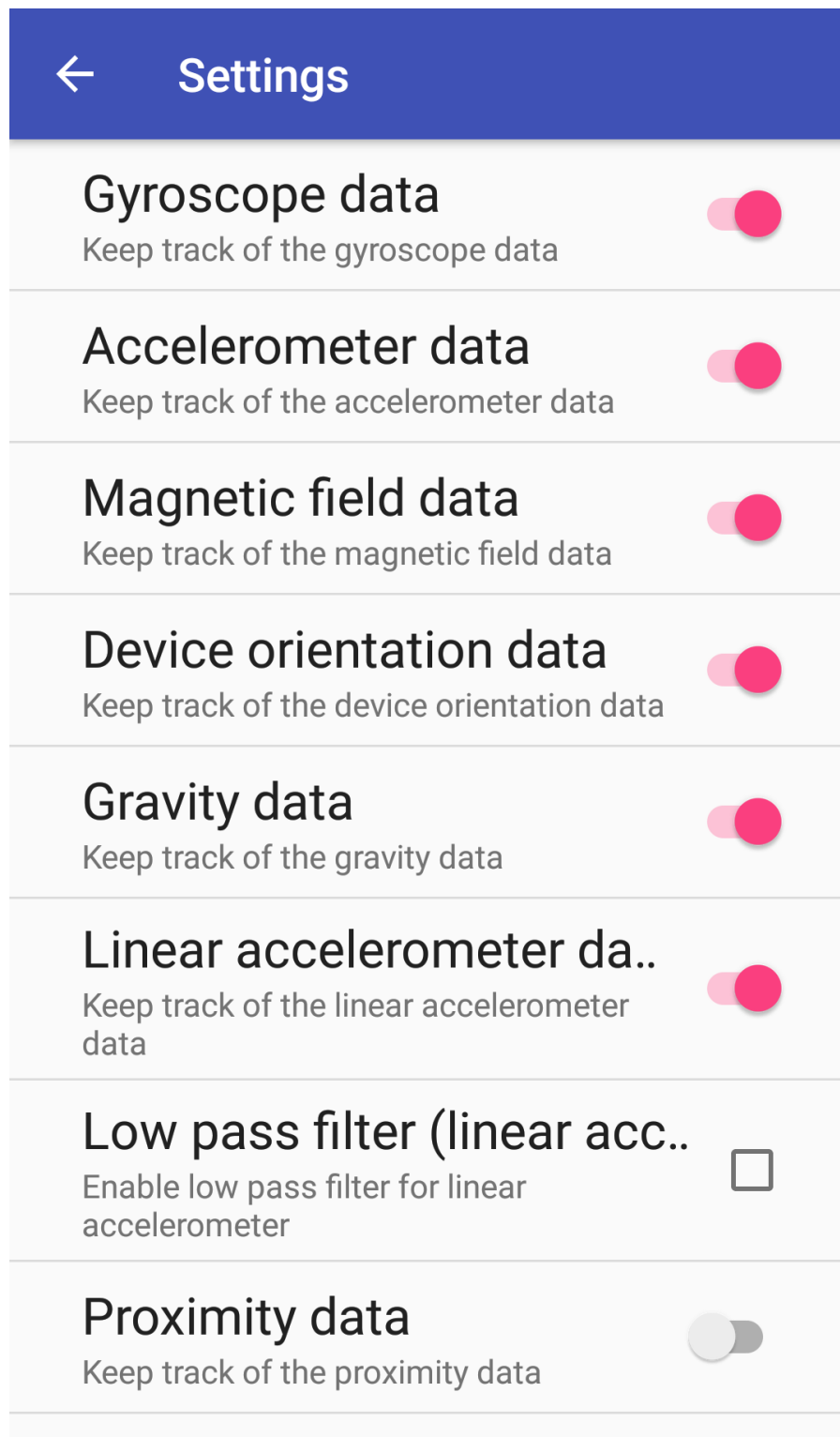


Figure 5.8.2: Settings Part 1

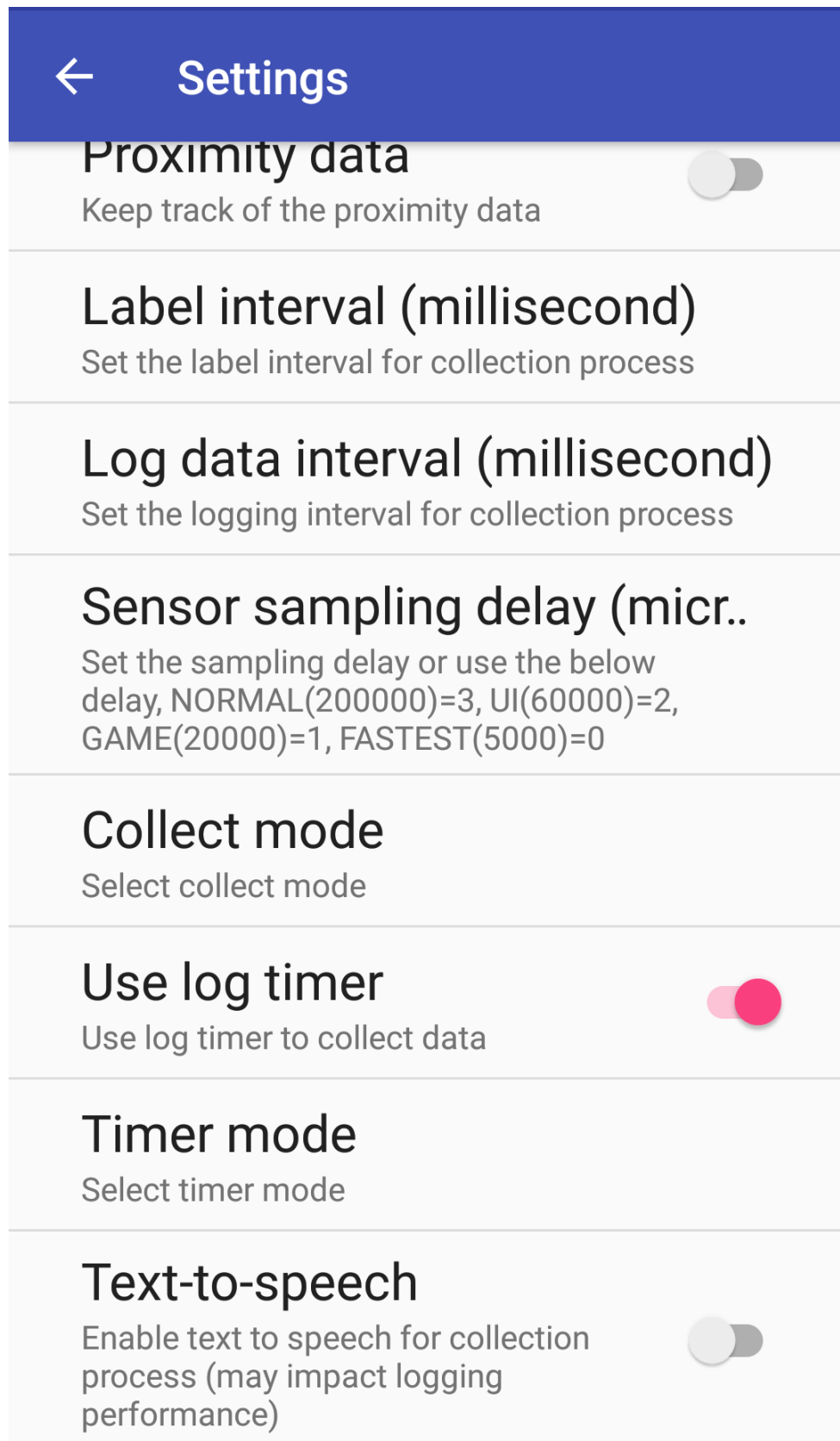


Figure 5.8.3: Settings Part 2

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

The application allows the dynamic labelling feature. Any label can be entered, giving the opportunity to use any label for the sensor records.

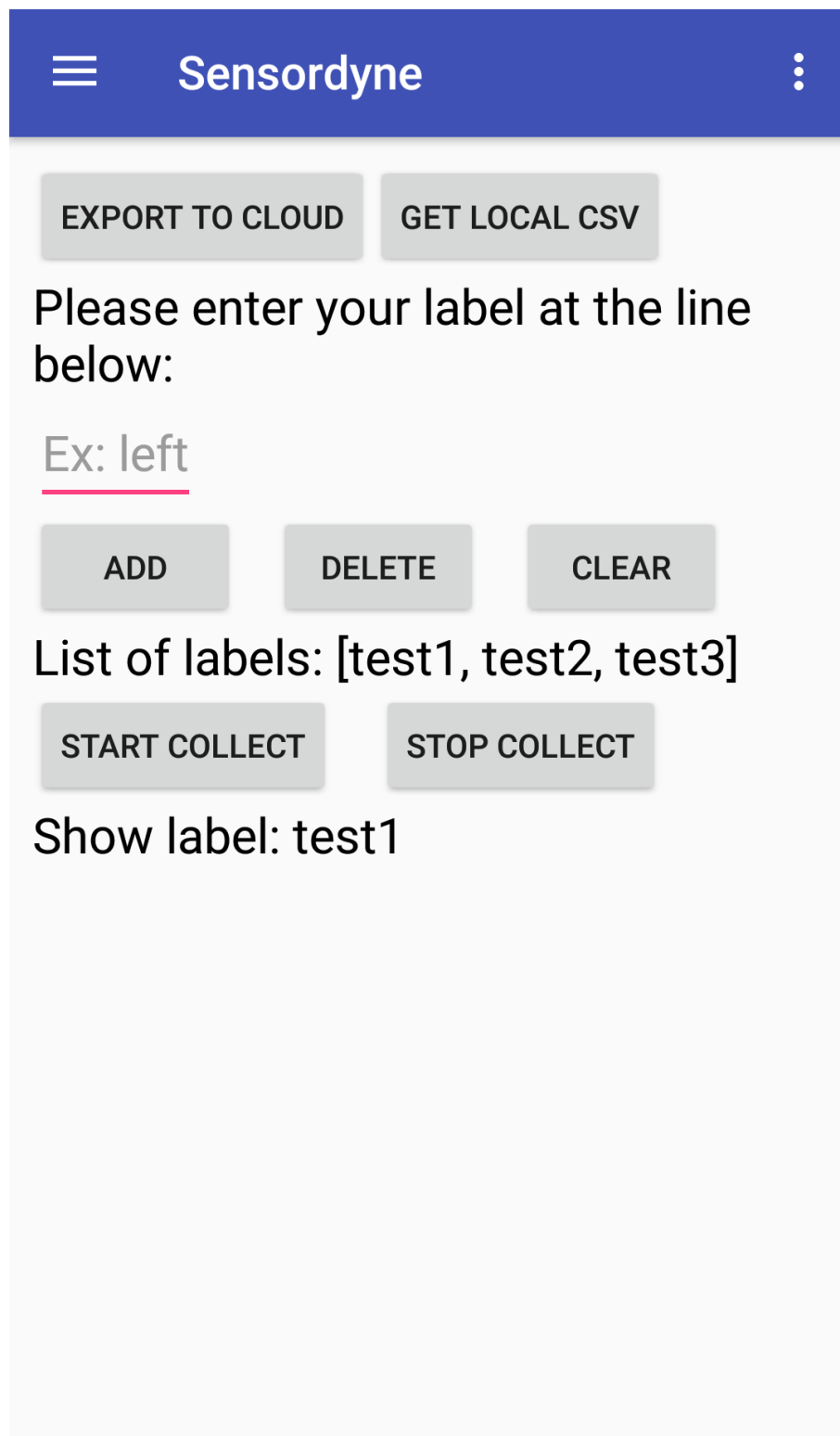


Figure 5.8.4: Dynamic Labelling

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

After finishing the collecting process, the file can be exported to cloud. AWS S3 cloud storage can then be accessed, showing all datasets collected before. This can allow you to choose which one to keep and which one to delete.

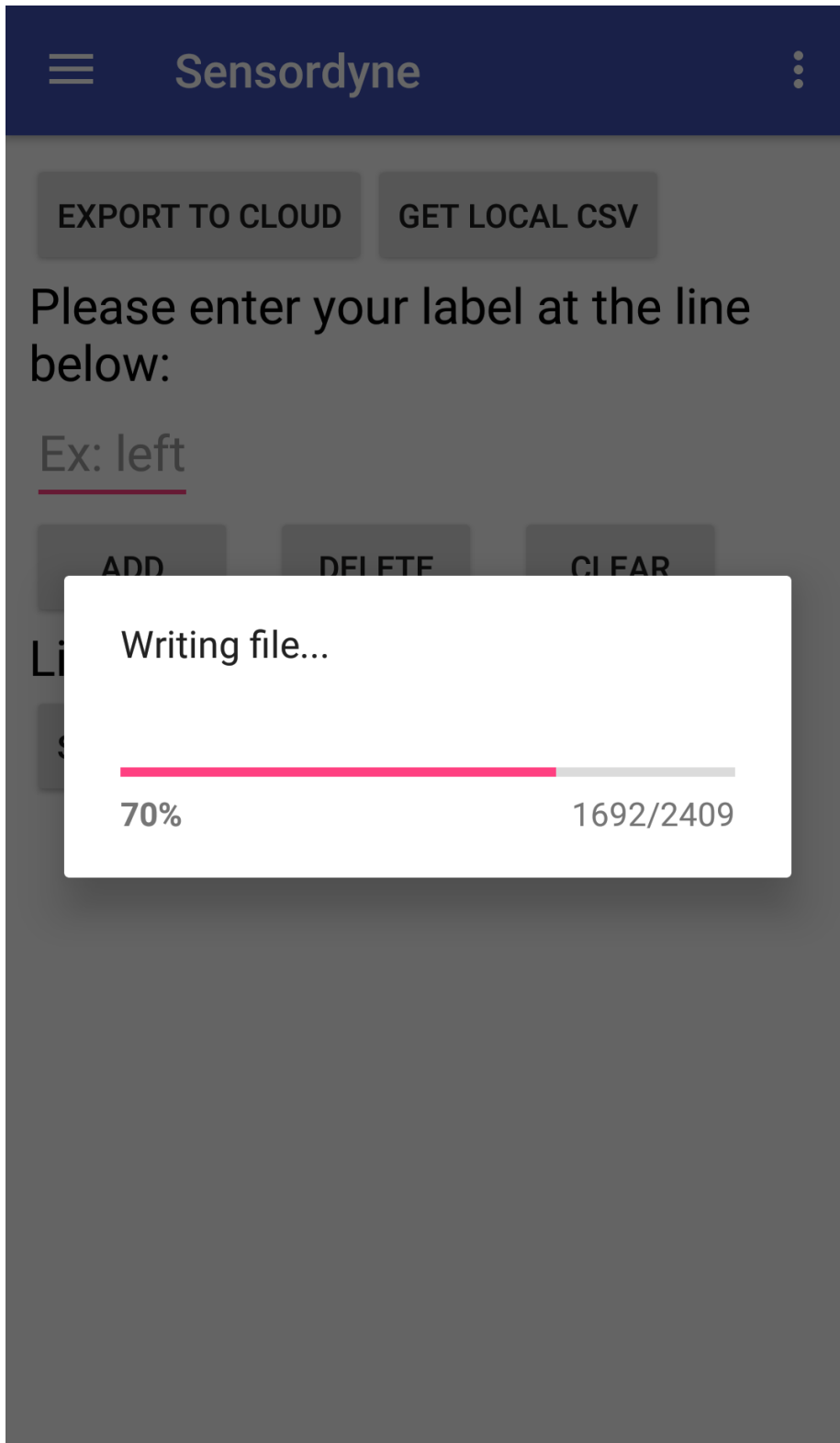


Figure 5.8.5: After Clicking Stop Collect Button

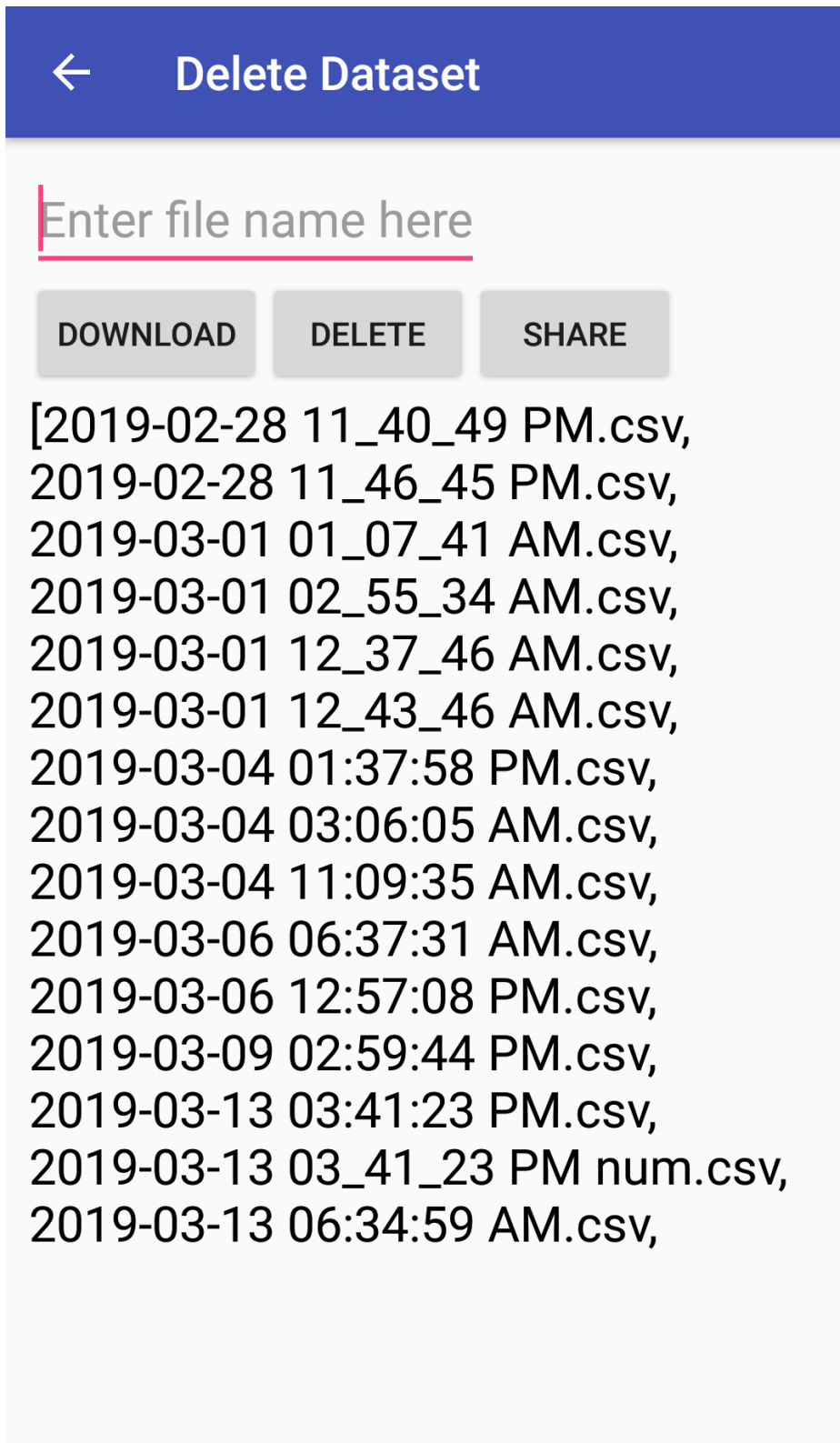


Figure 5.8.6: Manage File in Cloud Storage

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Alternatively, to share dataset with the others, device info page can be accessed. Other can give out their smartphone ID, so that the dataset can be shared to their individual folder on cloud storage.

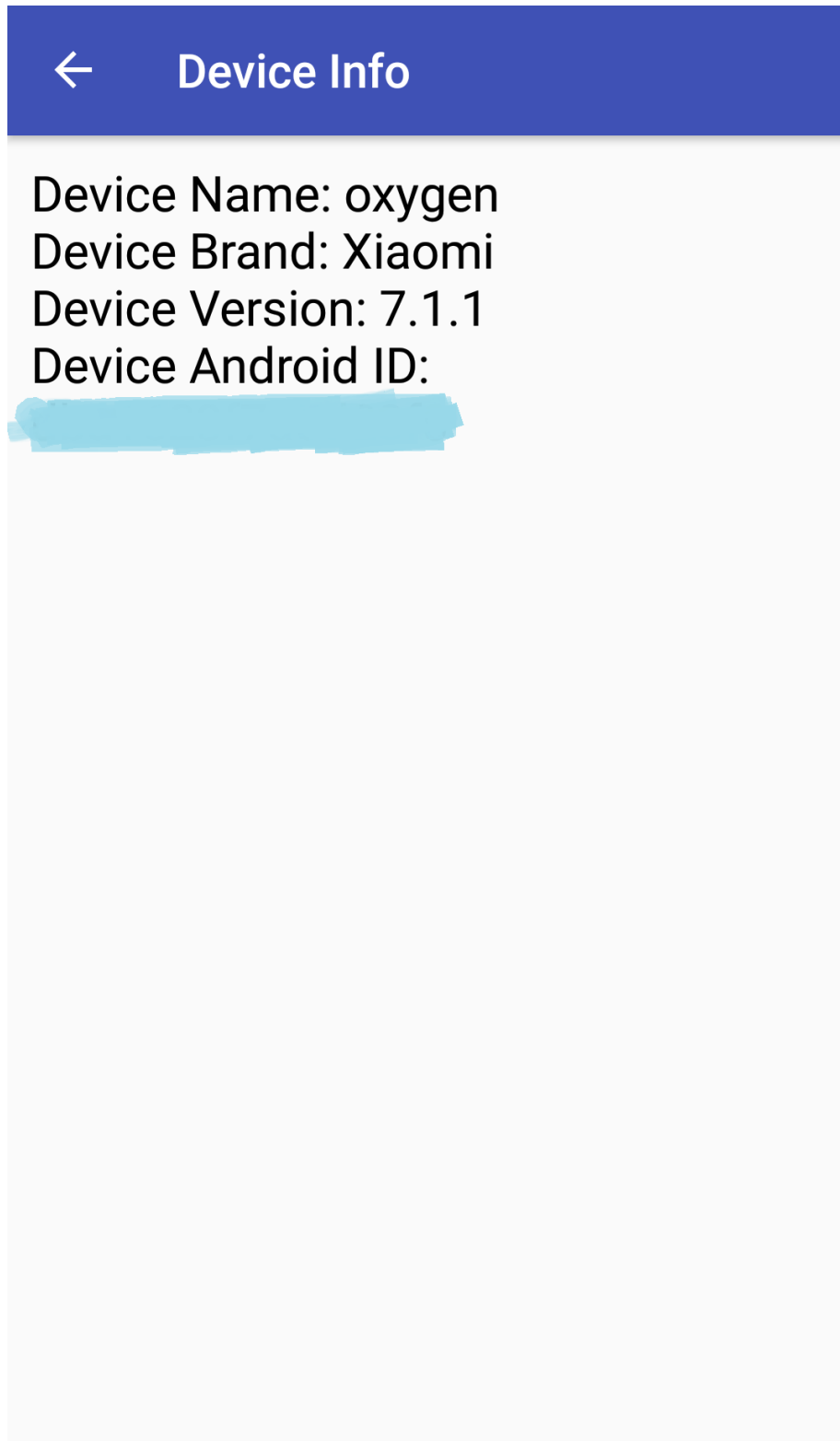


Figure 5.8.7: Accessing Device Info

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Then, model also can be built in the following section. The first subsection below allows the freedom to choose hyperparameters and customize model to own preferences. The second subsection depicts the training hyperparameters to be used for training/fitting model.

The screenshot shows a mobile application interface for configuring a neural network model and training settings. The interface is titled "Model and Neural Network..." and is divided into two main sections: "Model Settings" and "Training settings".

Model Settings:

- Layer type: Dense
- Number of unit: 64
- Input shape info: (20,2)
- Activation function: None
- Dropout: 0.16
- Optimizer: adagrad
- Loss function: categorical_hinge

Buttons: ADD LAYER TO MODEL, SAVE MODEL

Training settings:

- Batch size: 5
- Epoch: 5
- Time step: 20

Button: SAVE SETTINGS

Figure 5.8.8: Tune Model Building and Training Hyperparameters

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Finally, the fit model button can be clicked. This makes the application to submit a HTTP GET request to web server hosted in AWS EC2. The instance acts as a web server which helps to accept the request and executes the deep learning python script. Then, the script executed logged the training output to the S3 cloud storage. If the training result is satisfied, the convert model button can be clicked, converting the model for actual mobile deployment. The training output can be seen in the same figures the dataset is shown too.

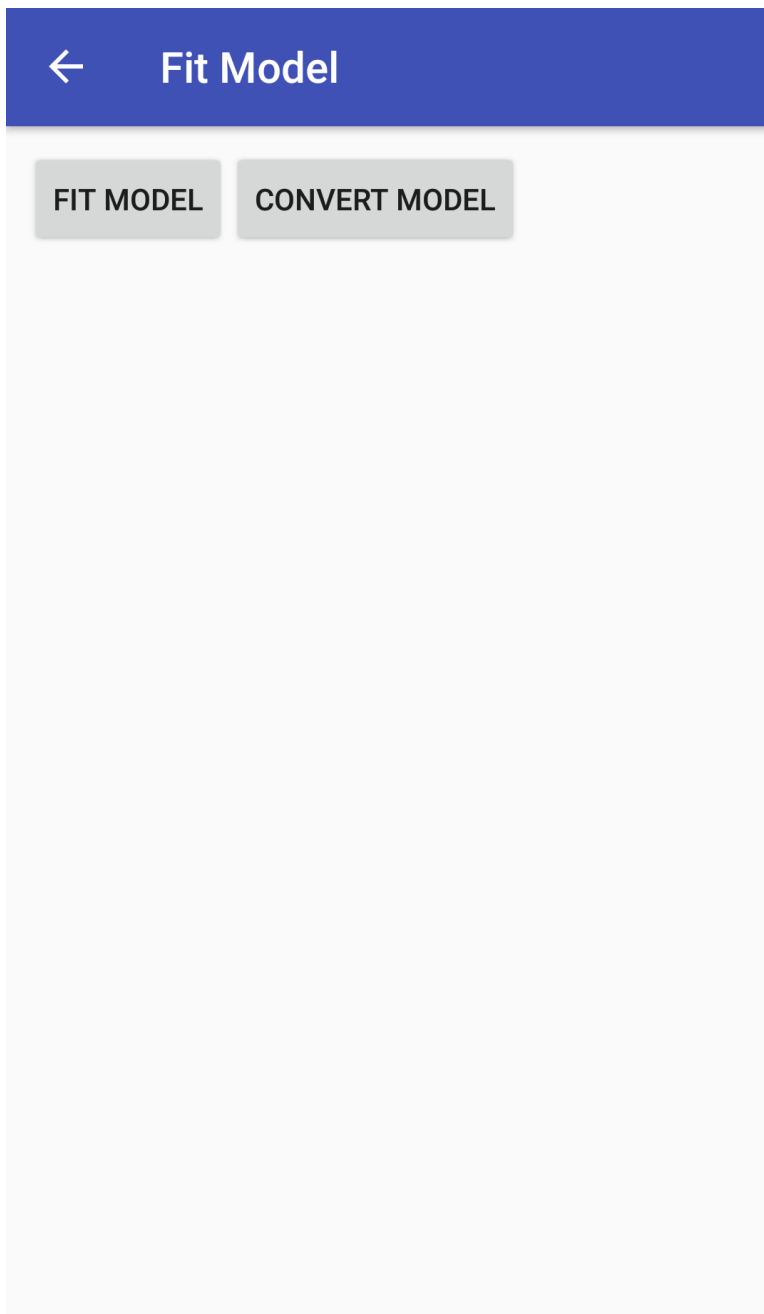


Figure 5.8.9: Fit Model and Convert Model function

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

The following files resulted from clicking fit model button and convert model button are stored at the cloud storage as shown below.

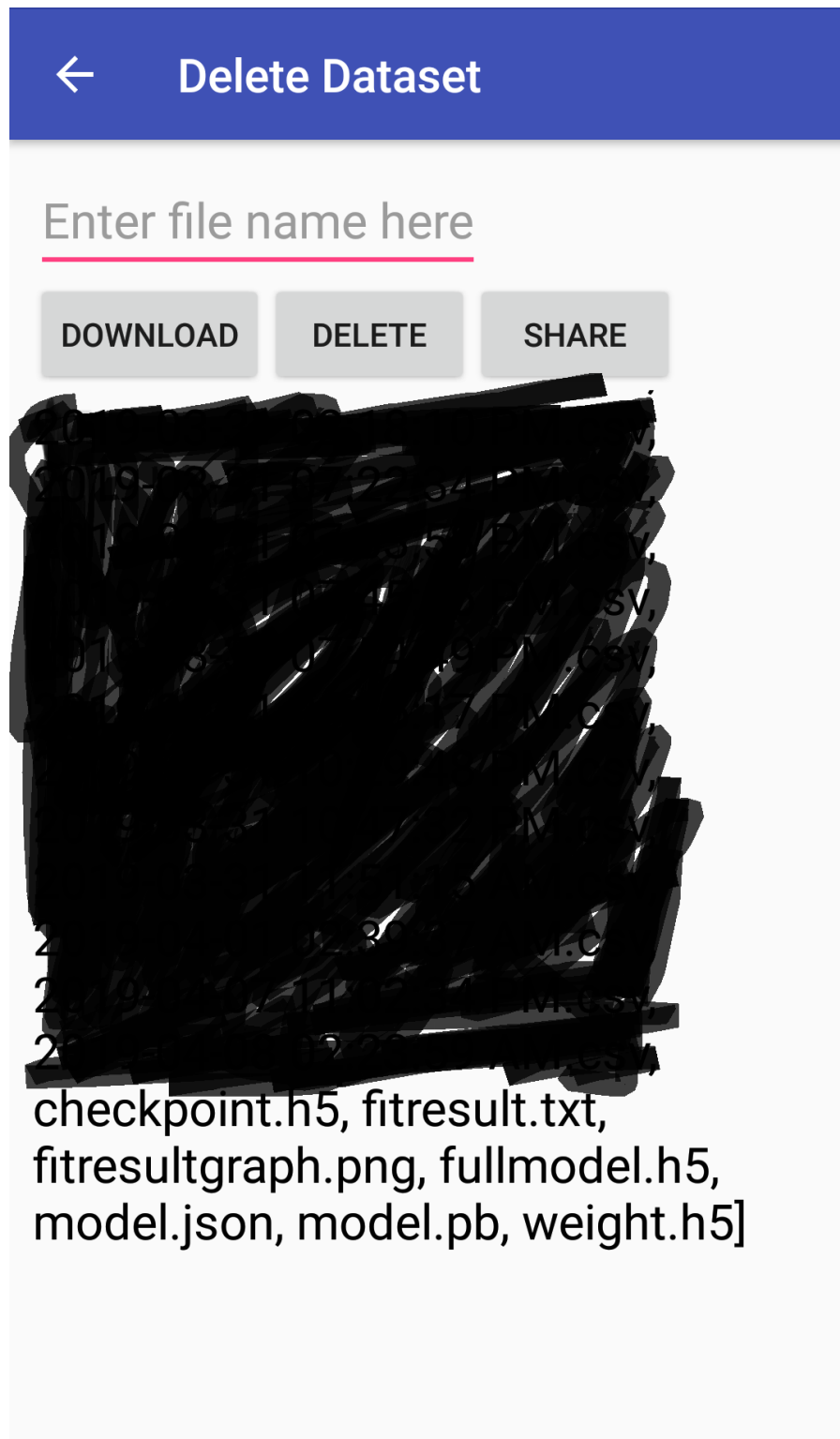


Figure 5.8.10: Training Results and Converted Model

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Actual mobile deployment can be seen in the following figures. The start predict and stop predict button can be clicked, initiating and ending the prediction.

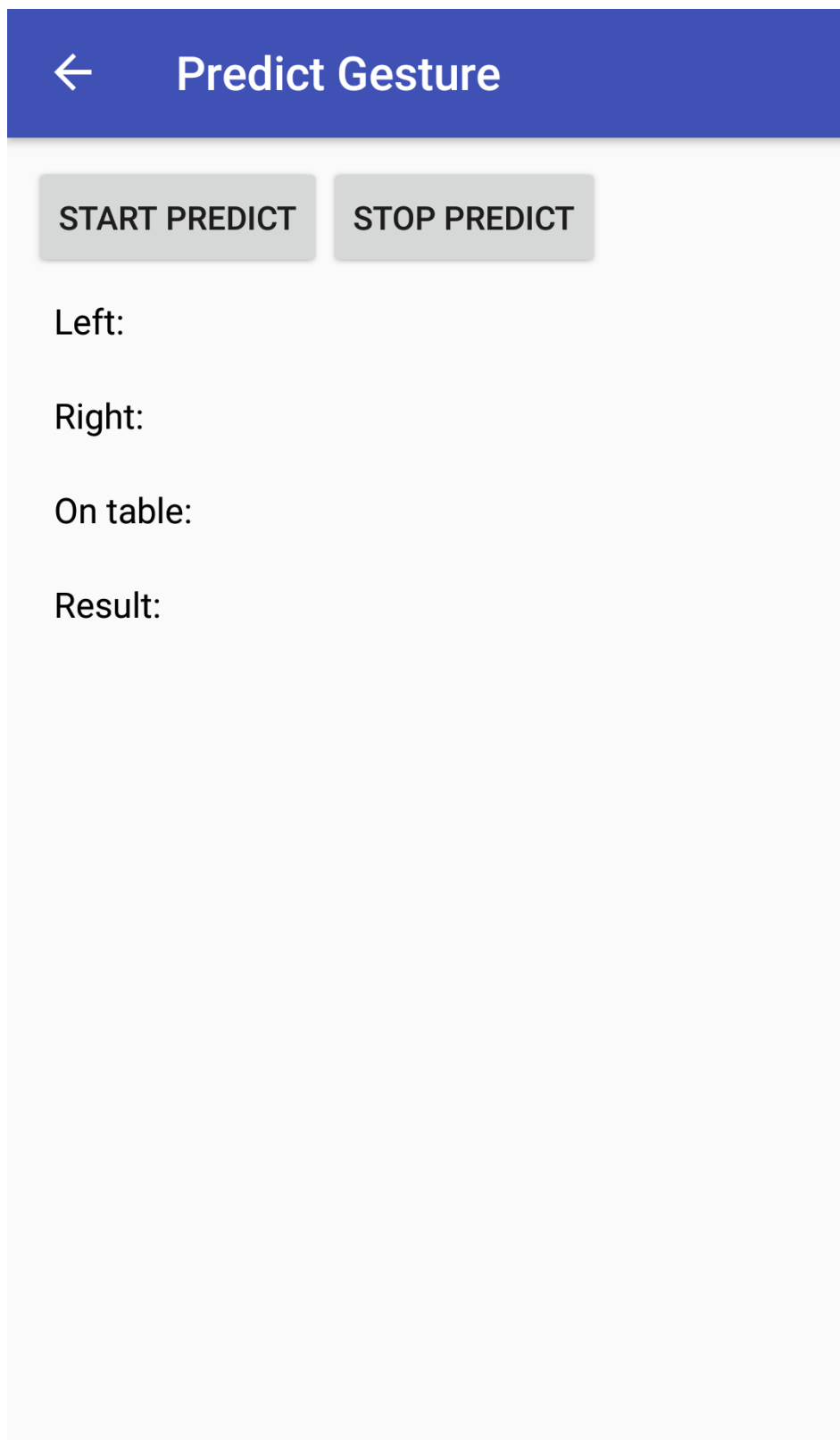


Figure 5.8.11: Predict Hand Gesture

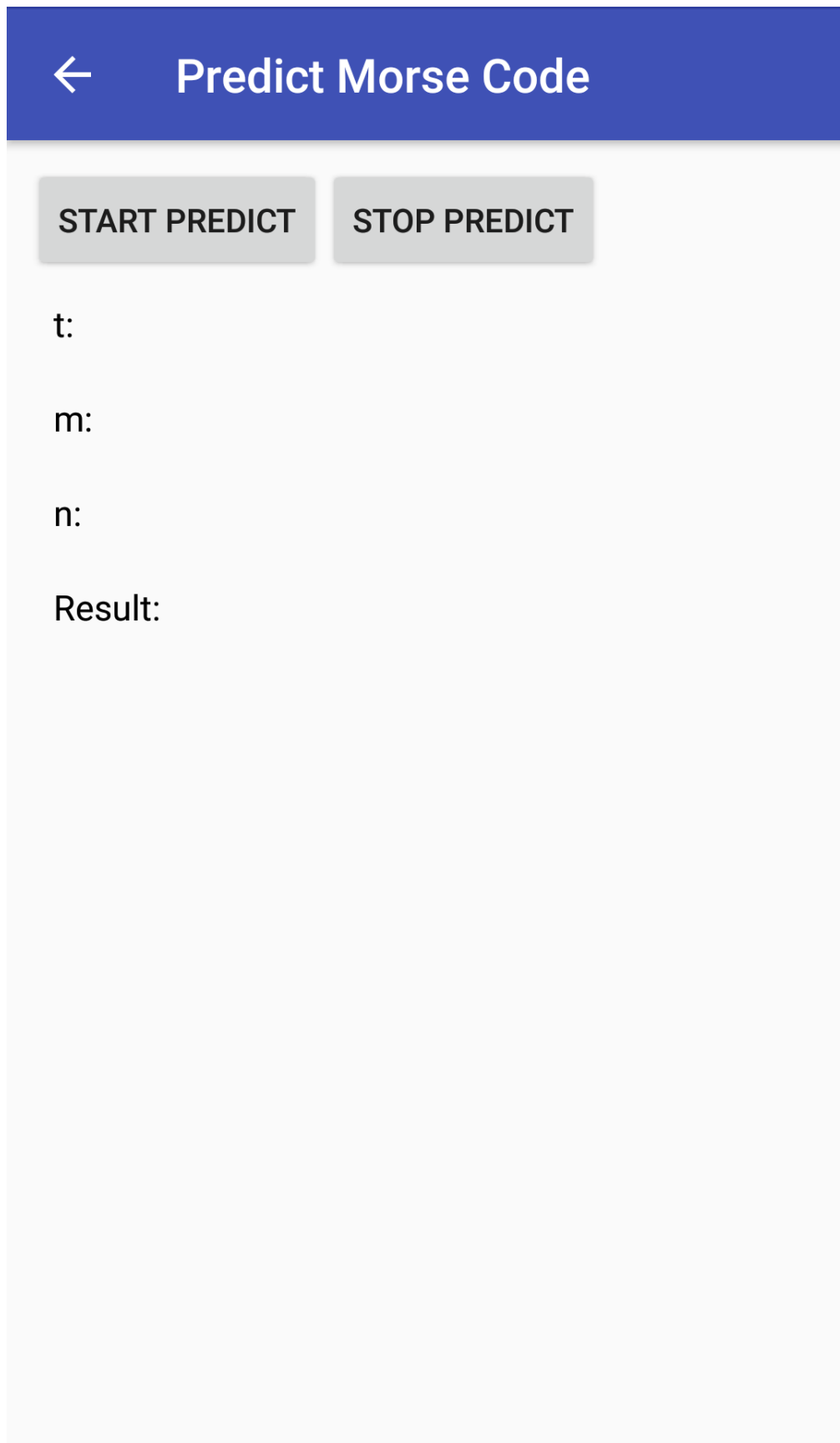


Figure 5.8.12: Predict Morse Code

CHAPTER 6 CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this paper, several issues surrounding deep learning researches were discussed. It is shown that the data collecting process and model building and training process in deep learning research required further optimisation in terms of usability. In corresponding to that, an Android mobile application that implements the automated data collection framework was developed. The mobile application is able to collect multiple sensor data in the smartphone and supports dynamic labelling of the data according to user's preferences. Then, it sends the sensor data over the cloud. Not only that, the users can share the dataset between them through the mobile application interface. Then, on the cloud, model is built and trained before logging the training output back to the cloud storage.

6.2 Future Work

However, the current work may still need some improvement in terms of usability and security. The current mobile application needed to implement a much more secured authentication method as well as a much more usable user interface. Also, the web server hosted on AWS EC2 instance is able to serve users under light load. As for a user that needs to utilise a complicated model and large dataset training service, the web server may need to be upgraded to a higher tier to accompany the load.

REFERENCES

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M. and Steggles, P., 1999, September. Towards a better understanding of context and context-awareness. In *International Symposium on Handheld and Ubiquitous Computing* (pp. 304-307). Springer, Berlin, Heidelberg.
2. Arenas, J.O.P., Murillo, P.C.U. and Moreno, R.J., 2017, August. Convolutional neural network architecture for hand gesture recognition. In *Electronics, Electrical Engineering and Computing (INTERCON), 2017 IEEE XXIV International Conference on* (pp. 1-4). IEEE.
3. Benalcázar, M.E., Jaramillo, A.G., Zea, A., Páez, A. and Andaluz, V.H., 2017, August. Hand gesture recognition using machine learning and the Myo armband. In *Signal Processing Conference (EUSIPCO), 2017 25th European* (pp. 1040-1044). IEEE.
4. Bergman, E. and Johnson, E. (2001). Towards accessible human-computer interaction. *Sun Microsystems Laboratories The First Ten Years*.
5. Cai, L., Cui, S., Xiang, M., Yu, J. and Zhang, J., 2017. Dynamic hand gesture recognition using RGB-D data for natural human-computer interaction. *Journal of Intelligent & Fuzzy Systems*, 32(5), pp.3495-3507. A New Deep Learning Approach for Anomaly Base IDS using Memetic Classifier
6. Casino, F, Batista, E, Borrás, F, Martínez-Balleste, A, & Patsakis, C 2017, 'Healthy Routes in the Smart City: A Context-Aware Mobile Recommender', *IEEE Software*, 34, 6, p. 42-47, Scopus®, EBSCOhost, viewed 3 April 2018.
7. Choi, J.Y., Yoo, T.K., Seo, J.G., Kwak, J., Um, T.T. and Rim, T.H., 2017. Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database. *PloS one*, 12(11), p.e 0187336.
8. Hibbeln, M., Jenkins, J.L., Schneider, C., Valacich, J.S. and Weinmann, M., 2017. How is your user feeling? Inferring emotion through human-computer interaction devices. *Group*, 1000, p.248.
9. Jaramillo, A.G. and Benalcázar, M.E., 2017, October. Real-time hand gesture recognition with EMG using machine learning. In *Ecuador Technical Chapters Meeting (ETCM), 2017 IEEE* (pp. 1-5). IEEE.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

10. Khozouie, N., Fotouhi-Ghazvini, F. and Minaei-Bidgoli, B., "Developing a context-aware mobile patient monitoring framework with an ontology-based approach," 2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA), Putrajaya, 2017, pp. 1-5.
11. Li, Y. and Shen, L., 2018. Skin lesion analysis towards melanoma detection using deep learning network. *Sensors*, 18(2), p.556.
12. LlamaLab, 2018, *Google Playstore*, version 1.10.7, mobile app, viewed 1 April 2018
<<https://play.google.com/store/apps/details?id=com.llamalab.automate&hl=en>>
13. Liang, H., Sun, X., Sun, Y. and Gao, Y., 2017. Text feature extraction based on deep learning: a review. *EURASIP journal on wireless communications and networking*, 2017(1), p.211.
14. Liu, X., Xu, F., Sun, Y., Zhang, H. and Chen, Z. (2018). Convolutional Recurrent Neural Networks for Observation-Centered Plant Identification. *Journal of Electrical and Computer Engineering*, 2018, pp.1-7.
15. Mohammadi, S. & Namadchian, A. (2017), 'A New Deep Learning Approach for Anomaly Base IDS using Memetic Classifier', *International Journal Of Computers, Communications & Control*, 12, 5, pp. 677-688, *Computers & Applied Sciences Complete, EBSCOhost*, viewed 3 April 2018.
16. Moore, P. and Van Pham, H., 2017, October. On wisdom and rational decision-support in context-aware systems. In *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on* (pp. 1982-1987). IEEE.
17. Naz, S., Umar, A., Ahmed, S., Ahmad, R., Shirazi, S., Razzak, M. and Zaman, A. (2018). Statistical Features Extraction for Character Recognition Using Recurrent Neural Network. *Pakistan Journal of Statistics*, 34(1), pp.47-53.
18. Obinikpo, A.A. and Kantarci, B., 2017. Big Sensed Data Meets Deep Learning for Smarter Health Care in Smart Cities. *Journal of Sensor and Actuator Networks*, 6(4), p.26.
19. Sakamoto, T., Gao, X., Yavari, E., Rahman, A., Boric-Lubecke, O. and Lubecke, V.M., 2017, December. Radar-based hand gesture recognition using IQ echo plot and convolutional neural network. In *Antenna Measurements & Applications (CAMA), 2017 IEEE Conference on* (pp. 393-395). IEEE.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

20. Salah, N.B., Saâdi, I.B. and Ghezala, H.B., 2017, October. A comprehensive view of u-accessibility in context-aware learning systems for disabled learners. In *Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on* (pp. 41-48). IEEE.
21. Schmidt, A. (2000). Implicit human computer interaction through context. *Personal Technologies*, [online] 4(2-3), pp.191-199. Available at: <https://link.springer.com/article/10.1007/BF01324126> [Accessed 1 Apr. 2018].
22. Schmidt, A. (2003). *Ubiquitous computing-computing in context* (Doctoral dissertation), April 4, pp. 212.
23. Sornapudi, S., Stanley, R.J., Stoecker, W.V., Almubarak, H., Long, R., Antani, S., Thoma, G., Zuna, R. and Frazier, S.R., 2018. Deep Learning Nuclei Detection in Digitized Histology Images by Superpixels. *Journal of Pathology Informatics*, 9(1), p.5.
24. Ubbens, J., Cieslak, M., Prusinkiewicz, P. and Stavness, I., 2018. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant Methods*, 14(1), p.6.
25. Wang, Z, Yu, Z, Zhou, X, Chen, C, & Guo, B 2016, 'Towards Context-Aware Mobile Web Browsing', *Wireless Personal Communications*, 91, 1, pp. 187-203, Computers & Applied Sciences Complete, EBSCOhost, viewed 3 April 2018.
26. Zhang, X., Liu, X., Yuan, S. and Lin, S. (2017). Eye Tracking Based Control System for Natural Human-Computer Interaction. *Computational Intelligence and Neuroscience*, [online] 2017(2017), pp.1-9. Available at: <https://doi.org/10.1155/2017/5739301> [Accessed 1 Apr. 2018].
27. Zhang, Y., Xiong, R., He, H. and Pecht, M. (2018). Long Short-Term Memory Recurrent Neural Network for Remaining Useful Life Prediction of Lithium-Ion Batteries. *IEEE Transactions on Vehicular Technology*, 67(7), pp.5695-5705.



A Framework to Automate Time Series Data Collection Using Mobile Phone for Deep Learning Training and Deployment

By Jason Lim Jing Wei, Bachelor of Computer Science (HONS),
University Tunku Abdul Rahman, Faculty of Information and
Communication Technology

Introduction

- Targeted for deep learning enthusiasts and hobbyists
- Focusing on simplification of data collecting process
- Deep learning feature on smartphone application
- Dynamic labelling for data collecting process
- Tuning model building and training hyperparameters

Discussions

- Tools used: AWS S3, AWS EC2, Android Studio, Jupyter Notebook, SQLite, OKHTTPClient, Flask, TensorFlow, Keras
- Programming language: Python, Java

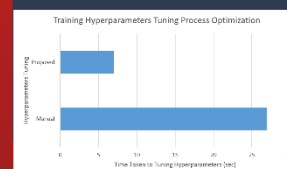
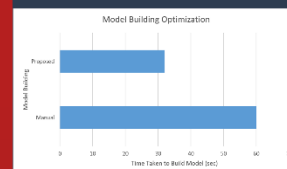
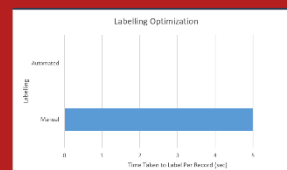
Methods

- Collect and label time series data from smartphone sensor
- Send time series sample data to cloud
- Send requests to cloud web server
- Cloud web server builds and train model
- Training result stored in cloud storage
- Convert model for mobile deployment and actual practices

Conclusion

- Helps speeding up progress and process of deep learning
- Parallel data collection, model building and training
- Share deep learning model with each other

Results



Acknowledgments

- Special thanks to my project supervisor, Dr. Ooi Boon Yaik for providing advices throughout the project development

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

PLAGIARISM CHECK RESULT

The screenshot shows the Turnitin Feedback Studio interface. The main content area displays a 'LIST OF FIGURES' table with the following data:

Figure Number	Title	Page
3.2.1	Use Case Diagram	32
3.4.1	Activity Diagram Part 1	49
3.4.2	Activity Diagram Part 2	50
3.4.3	Activity Diagram Part 3	51
3.5.1	Relationship between Classes	52
3.5.2	CheckSensorActivity	53
3.5.3	CheckSensorAdapter	53
3.5.4	CustomDialogFragment	53
3.5.5	CustomProgressDialog	54

The right sidebar shows a 'Match Overview' with a total similarity index of 19%. The matches listed are:

- 1 Submitted to Universiti... Student Paper 3%
- 2 digitalcommons.lasalle... Internet Source 1%
- 3 eprints.utar.edu.my Internet Source 1%
- 4 Submitted to Laureate... Student Paper 1%
- 5 Submitted to Rocheste... Student Paper 1%
- 6 Submitted to University... Student Paper <1%
- 7 eprints.utm.my Internet Source <1%
- 8 Submitted to University... Student Paper <1%

The screenshot shows the Turnitin Originality Report for the document 'A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLL... By Jason Jing Wei Lim'. The report indicates a similarity index of 19%.

Similarity Index	Similarity by Source
19%	Internet Sources: 12%
	Publications: 6%
	Student Papers: 17%

The report lists the following matches:

- 1% match (Internet from 19-Mar-2016) <http://digitalcommons.lasalle.edu>
- 1% match (Internet from 25-Jul-2017) <http://eprints.utar.edu.my>
- 1% match (student papers from 14-Nov-2013) [Submitted to Universiti Tunku Abdul Rahman on 2013-11-14](#)
- 1% match (student papers from 25-Sep-2006) [Submitted to Rochester Institute of Technology on 2006-09-25](#)
- 1% match (student papers from 13-Jul-2016) [Submitted to Laureate Education Inc. on 2016-07-13](#)
- <1% match (student papers from 08-Sep-2017) [Submitted to University of New South Wales on 2017-09-08](#)
- <1% match (student papers from 06-Aug-2015) [Submitted to Universiti Tunku Abdul Rahman on 2015-08-06](#)
- <1% match (student papers from 11-Sep-2018) [Submitted to University of Liverpool on 2018-09-11](#)
- <1% match (student papers from 16-Sep-2013) [Submitted to De Montfort University on 2013-09-16](#)

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

Universiti Tunku Abdul Rahman			
Form Title: Supervisor’s Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	Supervisor’s Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: _____ % Similarity by source Internet Sources: _____ % Publications: _____ % Student Papers: _____ %	
Number of individual sources listed of more than 3% similarity: _____	
Parameters of originality required and limits approved by UTAR are as Follows: <ul style="list-style-type: none"> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <p><i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i></p>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

CS (Hons) Computer Science
Faculty of Information and Communication Technology (Perak Campus), UTAR.

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT

*Based on the above results, I hereby declare that I am satisfied with the originality of the
Final Year Project Report submitted by my student(s) as named above.*

Signature of Supervisor

Name:

Date:

Signature of Co-Supervisor

Name: _____

Date: _____

A FRAMEWORK TO AUTOMATE TIME SERIES DATA COLLECTION USING
MOBILE PHONE FOR DEEP LEARNING TRAINING AND DEPLOYMENT



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	15ACB02540
Student Name	Jason Lim Jing Wei
Supervisor Name	Dr. Ooi Boon Yaik

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <p>_____</p> <p>(Signature of Student)</p> <p>Date:</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <p>_____</p> <p>(Signature of Supervisor)</p> <p>Date:</p>
---	---