

**DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHT USING HUMAN  
ACTIVITY RECOGNITION**

By  
PEH HONG BO

A REPORT  
SUBMITTED TO  
Universiti Tunku Abdul Rahman  
in partial fulfillment of the requirements  
for the degree of  
**BACHELOR OF COMPUTER SCIENCE (HONOURS)**  
Faculty of Information and Communication Technology  
(Kampar Campus)

MAY 2023

## REPORT STATUS DECLARATION FORM

**Title:** DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHT USING  
HUMAN ACTIVITY RECOGNITION


**Academic Session:** MAY 2023


I PEH HONG BO  
**(CAPITAL LETTER)**

declare that I allow this Final Year Project Report to be kept in  
Universiti 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)

  
\_\_\_\_\_  
(Supervisor's signature)

**Address:**  
30, Jalan 7/12D  
Kampung Sri Batu  
51100 K.L.

Dr. Aun Yichiet  
\_\_\_\_\_  
Supervisor's name

**Date:** 14 SEP 2023

**Date:** 15 SEP 2023

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY  
UNIVERSITI TUNKU ABDUL RAHMAN**


Date: 14<sup>th</sup> SEPTEMBER 2023

**SUBMISSION OF FINAL YEAR PROJECT**

It is hereby certified that **PEH HONG BO** (ID No: **19ACB06176**) has completed this final year project entitled "**DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHT USING HUMAN ACTIVITY RECOGNITION**" under the supervision of **Dr. Aun Yichiet** from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

  
\_\_\_\_\_  
(PEH HONG BO)

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**TITLE DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHTS USING HUMAN ACTIVITY RECOGNITION**” 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 : PEH HONG BO

Date : 13<sup>th</sup> SEPTEMBER 2023

## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Aun Yichiet who has given me this opportunity to engage in this IoT design project. It is my first step to establish a career in IoT development field. I would like to extend my gratitude to my moderator, Dr. Ooi Boon Yaik for pointing out my problems in FYP1 to improve this project.

Besides, I must thank my parents and my family for their love, support, and continuous encouragement. Without their support, physically, mentally and financially, I wouldn't have the chance and courage to complete the course.

## **ABSTRACT**

This is a development-based project to develop an automated smart lighting system that integrated a human activity recognition (HAR) model using computer vision. In recent years, smart homes have gradually become more popular in people's daily lives. An increasing number of smart home products have been introduced to the market and are widely embraced. Such market trend reflects people's demand for enhancing their quality of life by making use the advancement of technology in practical and real-life scenario. As one of the essential elements within a house, lighting systems have consistently been a popular cornerstone in the development of smart home systems to fulfil the requirement of flexible control and comfort use. However, none of the existing system could realise pure automation without any explicit involvement of human control. To fulfil the absence of such product, this project proposed an automated smart lighting prototype that apply human activity recognition model which work as the “brain” of the lighting system to understand what human is doing and to adjust the lighting condition accordingly. The prototype was built with a CSI camera, Jetson Nano and Yeelight smart light bulb to demonstrate the lighting system. A CNN-LSTM HAR model with evaluation accuracy at 74% is used as the backbone for activity recognition.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>xi</b>
<b>LIST OF TABLES</b>	<b>xiv</b>
<b>LIST OF SYMBOLS</b>	<b>xv</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xvi</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	1
1.2 Project Scope	2
1.3 Project Objectives	3
1.4 Contributions	4
1.5 Background Information	5
1.5.1 Human Activity Recognition	5
1.5.2 Convolutional Neural Network (CNN)	6
1.5.3 Recurrent Neural Network (RNN)	8
1.5.4 Long Short-Term Memory (LSTM)	10
1.5.5 Indoor Lighting	12
1.5.6 Docker	13
1.6 Report Organisation	14
<b>CHAPTER 2 LITERATURE REVIEWS</b>	<b>15</b>
2.1 Previous Works on Human Activity Recognition	15
2.1.1 Patient Monitoring by Abnormal Human Activity Recognition Based on CNN Architecture	15

2.1.2	Hockey Activity Recognition Using Pre-Trained Deep Learning Model	18
2.1.3	Long-term Recurrent Convolutional Networks for Visual Recognition and Description	20
2.1.4	Deep Neural Networks for Kitchen Activity Recognition	23
2.1.5	Summary of Reviewed Works	28
2.2	Previous Works on Lighting Adjustment Application	30
2.2.1	An IoT Based Smart Lighting System Based on Human Activity	30
2.2.2	Applications of Human Motion Tracking: Smart Lighting Control	33
2.3	Lighting for Each Activity Class	36
2.3.1	Dining	36
2.3.2	Drawing	36
2.3.3	Mopping Floor	36
2.3.4	Reading Books	37
2.3.5	Running on a Treadmill	37
2.3.6	Sleeping	37
2.3.7	Watching TV	37
2.3.8	Yoga	37
<b>CHAPTER 3 PROPOSED METHOD/APPROACH</b>		<b>38</b>
3.1	Methodologies	38
3.2	HAR Classification Model Network Architecture	40
3.3	Light Calibration Program	41
3.4	Timeline	42
<b>CHAPTER 4 SYSTEM DESIGN</b>		<b>43</b>
4.1	System Block Diagram	43
4.2	System Components Specifications	44
4.2.1	Configuration Program	44
4.2.2	HAR Model	46



4.2.3	Server Docker Container	47
4.2.4	Flask API	47
4.2.5	User Configuration Webpage	49
4.2.6	Database	50
 <b>CHAPTER 5 SYSTEM IMPLEMENTATION</b>		 <b>51</b>
5.1	Hardware Setup	51
5.2	Software Setup	52
5.3	HAR Model training	53
5.3.1	Dataset Collection	53
5.3.2	Data Pre-processing	53
5.3.3	Feature Extraction	59
5.3.4	Model Training	61
5.4	Server Docker	53
5.4.1	Flask API	63
5.4.2	User Configuration Webpage	67
5.5	MySQL Database Container	68
5.6	Configuration Program on Client	70
5.6.1	Configuration on Jetson Nano and CSI Camera	70
5.6.2	Configuration on IoT Light	70
5.6.3	Client Configuration Program	71
5.7	System Operation	73
 <b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>		 <b>77</b>
6.1	System Testing and Performance Metrics	77
6.2	Testing Setup and Result Discussion	79
6.3	Project Challenges	85
6.4	Objective Evaluation	86
 <b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>		 <b>87</b>
7.1	Conclusion	87
7.2	Recommendation	88

## **REFERENCES**

**89**

## **APPENDIX**

Appendix A: Training result of different model setting for 7 Class	A-1
Appendix B: Training result of different model setting for 8 Class	B-1
Appendix C: Final Year Weekly Report	C-1
Appendix D: Poster	D-1
Appendix E: Plagiarism Check Result	E-1
Appendix F: FYP2 Checklist	F-1

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 1.1.1	Setup of sensors in a smart home kitchen	1
Figure 1.5.2.1	Demonstration of filter sliding in convolutional layer	6
Figure 1.5.2.2	Visualisation of a general CNN architecture	7
Figure 1.5.3.1	Basic structure of a RNN cell	8
Figure 1.5.4.1	Visualisation of LSTM	10
Figure 1.5.5.1	Colour temperature scale	12
Figure 1.5.6.1	Interface of docker desktop	13
Figure 2.1.1.1	Labelling of dataset	16
Figure 2.1.1.2	Result of the model	16
Figure 2.1.2.1	Visualisation of VGG-16 architecture	18
Figure 2.1.2.2	Result of the model	19
Figure 2.1.3.1	Illustration of LRCN architecture for sequential input, static output	20
Figure 2.1.3.2	Result of the model	21
Figure 2.1.3.3	Difference between accuracy of LRCN and Single Frame Prediction	21
Figure 2.1.4.1	Pipeline of the proposed activity recognition architecture	23
Figure 2.1.4.2	Result of the model	26
Figure 2.2.1.1	Structure of the smart lighting system	30
Figure 2.2.1.2	Flow chart of the smart lighting system	31
Figure 2.2.1.3	Light intensity guideline for various daily activiry in different country	31
Figure 2.2.2.1	Estimation of human location by extracting the head area	33
Figure 2.2.2.2	Estimation of shoulder for heading direction estimation	34
Figure 2.2.2.3	Setup of the room	34
Figure 2.2.2.4	The global trajectory map	34

Figure 3.1.1	Project development methodology	38
Figure 3.2.1	Architecture visualisation of CNN-LSTM model	41
Figure 3.4.1	Proposed timeline for the project	42
Figure 4.1.1	System Block Diagram	43
Figure 4.2.1.1	Flow chart of the configuration program	44
Figure 4.2.4.1	Flow chart of flask API /upload route	47
Figure 4.2.4.2	Flow chart of flask API /batch_received route	48
Figure 4.2.4.3	Flow chart of flask API /configure route	48
Figure 5.3.2.1	Snippet of Extract_frame_report.csv	54
Figure 5.3.2.2	Snippet of data_frames_report.csv	54
Figure 5.3.2.3	Snippet of frames.csv	54
Figure 5.3.2.4	Snippet of Skip_vid_report.csv	54
Figure 5.3.2.5	Code of preprocess_data function	55
Figure 5.3.2.6	Code of extract_frames function	56
Figure 5.3.2.7	Code of preprocess_data function	58
Figure 5.3.3.1	Summary of MobileNetV2 (left) and DensNet201 (right)	59
Figure 5.3.3.2	Example code to create the CNN model	59
Figure 5.3.3.3	Code of extract_feature function	60
Figure 5.3.3.4	Code of train_test_loader function to split the dataset	60
Figure 5.3.4.1	Example code to create the LSTM model	61
Figure 5.3.4.2	Example code of model training and training result graph plotting	62
Figure 5.3.4.3	Example of training result graph	62
Figure 5.3.4.4	Example code of saving the best model	62
Figure 5.4.1.1	Code of /configure route	64
Figure 5.4.1.2	Code of /upload route	64
Figure 5.4.1.3	Code of /batch_received route	65
Figure 5.4.1.4	Flask API configuration	65
Figure 5.4.1.5	Code of Dockerfile	66
Figure 5.4.2.1	Dynamic rendering of the User Configuration Webpage	67
Figure 5.5.1	Docker-compose.yml file	68

Figure 5.5.1	Table Creation Scrip	69
Figure 5.6.2.1	Yeelight App interface to turn on the LAN control	70
Figure 5.6.3.1	Function to create bulb object	71
Figure 5.6.3.2	Code snipped for streaming and uploading images	71
Figure 5.6.3.3	Code snipped of requesting prediction result and light configuration	72
Figure 5.7.1	Log of starting the containers part 1	73
Figure 5.7.2	Log of starting the containers part 2	73
Figure 5.7.3	Log of starting the containers part 3	74
Figure 5.7.4	User Configuration Webpage	74
Figure 5.7.5	Updating light setting	74
Figure 5.7.6	Streaming logs on Jetson Nano and Server	75
Figure 5.7.7	Images saved in /uploads folder	75
Figure 5.7.8	Result log on Jetson Nano and Server	75
Figure 5.7.9	Yeelight setting changed. From left to right are original lighting, lighting on sleeping and lighting on reading book	76
Figure 6.1.1	Example of confusion matrix	77
Figure 6.2.1	Confusion matrix of DGAVG model with adam decay setting	83

## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 2.1.5.1	Summary of the reviewed works	28
Table 4.2.6.1	Colour temperature setting for each activity class	50
Table 4.2.6.1	Summary of table light_value in MySQL database	50
Table 5.1.1	Details of hardware requirement	51
Table 5.2.1	Software and libraries setup	52
Table 5.3.2.1	Summary of Training and Validation dataset	57
Table 5.3.2.2	Summary of Evaluation dataset	57
Table 5.3.3.1	Summary of shape of Train and Validation dataset	59
Table 5.3.4.1	Summary of Training Setting	61
Table 5.4.1.1	Summary of the docker image structure	63
Table 6.2.1	Summary of result for HAR model training	79

## LIST OF SYMBOLS

$\varphi$

Phi

## LIST OF ABBREVIATIONS

<i>IoT</i>	Internet of Things
<i>HAR</i>	Human Activity Recognition
<i>RFID</i>	Radio-Frequency Identification
<i>CNN</i>	Convolutional Neural Network
<i>RNN</i>	Recurrent Neural Network
<i>BPTT</i>	Back Propagation Through Time
<i>LSTM</i>	Long Short-Term Memory
<i>LED</i>	Light Emitting Diode
<i>YOLO</i>	You Only Look Once
<i>LIRIS</i>	Laboratoire d'InfoRmatique en Image et Systèmes d'information
<i>VoTT</i>	Visual Object Tagging Tool
<i>VGG</i>	Visual Geometry Group
<i>LRCN</i>	Long-Term Recurrent Convolutional Networks
<i>ILSVRC</i>	ImageNet Large Scale Visual Recognition Competition
<i>RGB</i>	Red Green Blue
<i>KSCGR</i>	Kitchen Scene Context based Gesture Recognition
<i>SVM</i>	Support Vector Machine
<i>HCF</i>	Hand-crafted Feature
<i>TV</i>	Television
<i>LAN</i>	Local Area Network
<i>RESNET</i>	Residual Network
<i>CSI</i>	Camera Serial Interface
<i>IIPSPW</i>	Introduction to Inventive Problem Solving and Proposal Writing
<i>STAIR</i>	Software Technology and Artificial Intelligence Research
<i>NUMPY</i>	Numerical Python
<i>CUDA</i>	Compute Unified Device Architecture
<i>cuDNN</i>	NVIDIA CUDA® Deep Neural Network library
<i>FYP</i>	Final Year Project
<i>MGAVG</i>	MobileNetV2 Global Average Pooling
<i>DGAVG</i>	DenseNet201 Global Average Pooling



# CHAPTER 1

## Introduction

This chapter is intended to get readers ready with a clearer picture of the proposed work. Background of the study, the objective and scope of project will be outlined with some explanation on the terminology.

### 1.1 Problem Statement and Motivation

The concept of Internet of Things (IoT) had been promoted with the rapid advancement of relevant hardware and network technology recently. IoT light, as part of the IoT member, had become more welcomed to cultivate an optimal surrounding based on different scenario. The commercialised smart lighting system that are currently available in the market would require the user to do the control through medium like mobile application, control panel or voice control. However, in a context of smart home, such action should be automated, i.e., allow the changes to be done without explicit involvement of users' effort. Automation could bring benefit to user, especially in those scenarios which the user is elderly or disabled that could have difficulties to perform the control action.

In current practice, implementation of a smart system that is fully automated will require deployment a number of sensors, e.g., RFID tag, motion sensor and light sensor to understand the surrounding [1]. However, such implementation is considered pretty inefficient as each sensor only serves a single detection purpose specifically for the connected device. A house renovation might also be required to implement the sensors and readers in a pleasant way, which is not cost effective. An example of implementation is done by L. Yao et al. [1] as shown in the figure below.

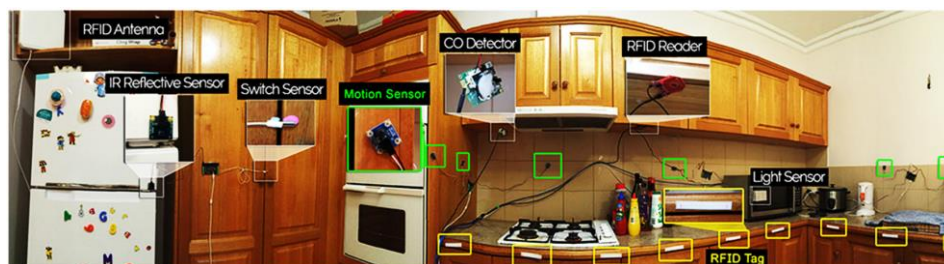


Figure 1.1.1: Setup of sensors in a smart home kitchen [1]

Lighting is an important element that defined the atmosphere in an indoor scene. Proper use of light could bring benefits for work productivity and mental health. This motivated the idea to develop an automated light configuration product using minimum sensor that adjust the indoor lighting according to a proper guideline based on the activity performed.

### **1.2 Project Scope**

This project is structured to build an automated light configuration prototype that react to the user activity in an indoor scenario. The prototype should cater 2 modules:

Module 1: A well-trained human activity recognition model that is able to perform real time recognition based on the captured video from camera.

Module 2: A program that have a complete guideline on suitable lighting parameters for each indoor activity class that can automatically adjust the IoT light setting without the need of explicit effort from user.

### 1.3 Project Objectives

The objective of the project includes:

- i. To train a multi-classes indoor activity recognition model for real-time inferences.

The core of the system is to perform recognition of the human activity in real time. Hence, it is essential to train an activity recognition model that can perform the recognition efficiently with an adequate level of accuracy. A multi-classes indoor activity recognition model that classifies 8 types of indoor activity should be delivered by the end of this project.

- ii. To develop a client-server automatic light calibrating program using Yeelight Library and Docker.

Another component of the system is the adjustment of lighting condition for the optimal indoor illumination base on the human activity detected. The system would integrate the Yeelight library which is built on top of the Yeelight Third-party Control Protocol Framework to perform the adjustment automatically based on the predefined parameter for the respective human action. User would also be allowed to save their own lighting preference other than the suggested setting for flexibility. The adjustment of light should be an automated process without the need of user to perform manual configuration.

### **1.4 Contributions**

The project is proposing a novel smart lighting system that address the pain point of automation control with the application of computer vision. This could benefit the users by making the control effortless to configure the lighting to the best condition, especially for elderly and disabled users. The lighting is configured with optimal parameters based on a proper guideline hence reduce the user's effort to perform self-study on light calibration standards. Suitable lighting is not only important to increase of work efficiency and maintain a stable mental state, but also to protect human's eyesight healthy. The proposed work could later be integrated in a smart home system to extend the power of smart home system and improve the quality of daily life.

### **1.5 Background Information**

#### **1.5.1 Human Activity Recognition**

Human activity recognition (HAR) is a subsection of Deep Learning research. It refers to the ability of machine to understand human's action with the implementation of information capturing sensors and interpret the retrieved data for determination [2]. It is an important research topic as such technology may widely be implemented in various sector as an assistive method, for instance, Internet of Things (IoT), surveillance system, sports training, construction safety, interactive gaming etc. Over past decade, this study field had achieved significant growth in line with the advance of deep learning and neural network which highly improved the accuracy rate and reduce the work of manual labelling feature compared to a general machine learning approach. The study had developed into branches including deployment of wearable sensor, non-wearable sensor and vision based. Wearable sensors require the user to wear certain sensor on the body to keep track on the movement of different body parts. Non-wearable sensor referred to the implementation of signal detection such as RFID, Wi-fi, radar, etc [3]. Vision based, on the other hand, is the implementation of computer vision technology to perform recognition of activity from the captured image or video data. It is a challenging study as it involved the consideration of multiple subjects, background subtraction, scene context, appearance, multiple view etc. The introduction of Convolutional Neural Network (CNN), a type of neural network that work excellent for image processing had greatly improved the accuracy of computer vision detection and lead to a rapid growth of the study.

### 1.5.2 Convolutional Neural Network (CNN)

CNN is a kind of neural network that widely implemented for deep learning tasks that involve processing and training on an image or video input. The network managed to perform well on top of its ability to learn on contextual information using the setup of 3 basic components, namely convolutional layer, pooling layer, and fully connected layer that are connected in sequence.

In convolutional layer, a fix size filter (a weight vector/matrix) slides through the input vector horizontally and vertically. Arithmetic operation take place between the filter and input to extract useful features as input for the next layer. As the weight of the filter is the same when sliding through the input, the feature map exported by a filter manage to capture similar feature at distinct location of the input, retaining the spatial information between the neighbouring pixels. Each filter used to detect one kind of target features, and a number of filters is applied to the input to extract distinct features.

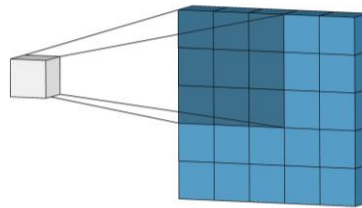


Figure 1.5.2.1: Demonstration of filter sliding in convolutional layer.

Pooling layer is connected next to the convolutional layer to down sample the feature map while preserving the most useful information. Similar filter using method such as maximum, average and summation slid through the feature map to extract the shrank output. This is important as it greatly reduced the number of parameters to train and introduces translation invariance (the same features are extracted from an object regardless of where it appears in the image), improving the model generalization for better performance on new object [4].

Fully connected layer is used at the end of the network to perform classification using the extracted high-level features. The output from previous section (that go through convolutional layer and pooling layer repetitively) is flattened as a 1-dimensional array and fed into the 1- dimensional linear layer. The dot product between

## CHAPTER 1

the input and the weight vector in the 1-dimensional linear layer is calculated and go through the activation function as final output [4].

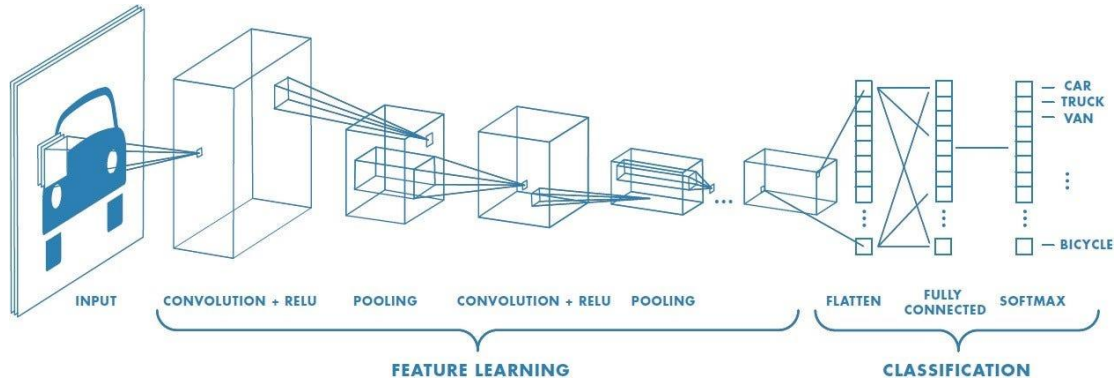


Figure 1.5.2.2: Visualisation of a general CNN architecture

### 1.5.3 Recurrent Neural Network (RNN)

While CNN is performing well in various machine learning task, the input sample is treated independently which cause CNN less effective on learning dataset that have time series information such as sentence and video. Recurrent Neural Network is introduced to address the need of learning temporal relationship in dataset [5].

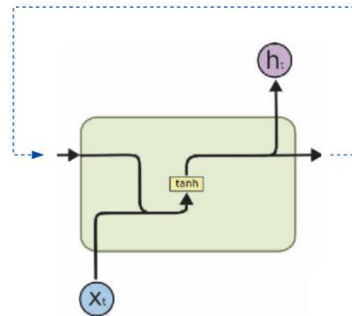


Figure 1.5.3.1: Basic structure of a RNN cell

As shown in figure above, a RNN cell takes in 2 inputs for calculation: the result from the previous timestep (the blue arrow), and the input of the current timestep ( $x_t$ ). The 2 inputs are combined and go through the activation function (which generally tanh) to get the final output. The output is then fed as input for prediction on the next timestep, and this loops until the last input. In other words, the output at current timestep is at certain level depends on the output of the previous timestep. This realises the ability of RNN to remember and process the features from past input throughout the whole temporal sequence for prediction on the subsequent timestep [6]. 2 outputs are generated: the predicted result for the current timestep ( $h$  in the figure, which output from the activation function will go through another softmax function to get the probability for the output) and the output to feed into the next step (without going through softmax layer).

As temporal information is important in RNN, the network used Back Propagation Through Time (BPTT) for the training process. It simply means that the loss calculated at each timestep is summed with the loss from timestep behind, and backpropagated to the timestep before [6].

However, RNN is facing difficulties in preserving the features over a long period of timestep. Considering the example below:



## CHAPTER 1

1. The boy is handsome.
2. The boy standing next to Lily and wearing a blue jacket is handsome.

In the first sentence, RNN could easily predict the use of “is” as it is right after “boy” which the output from boy is directly fed into the timestep at is. In the second sentence, RNN may not work as expected when there is a significant distance between “boy” and “is”, hence the output signals at timestep “boy” could have greatly changed when passing through the words in between before reaching “is” [5].

### 1.5.4 Long Short-Term Memory (LSTM)

To overcome the limitation in RNN, Long Short-Term Memory (LSTM) network is introduced which support flexible adjustment on preserving features for long and short-term.

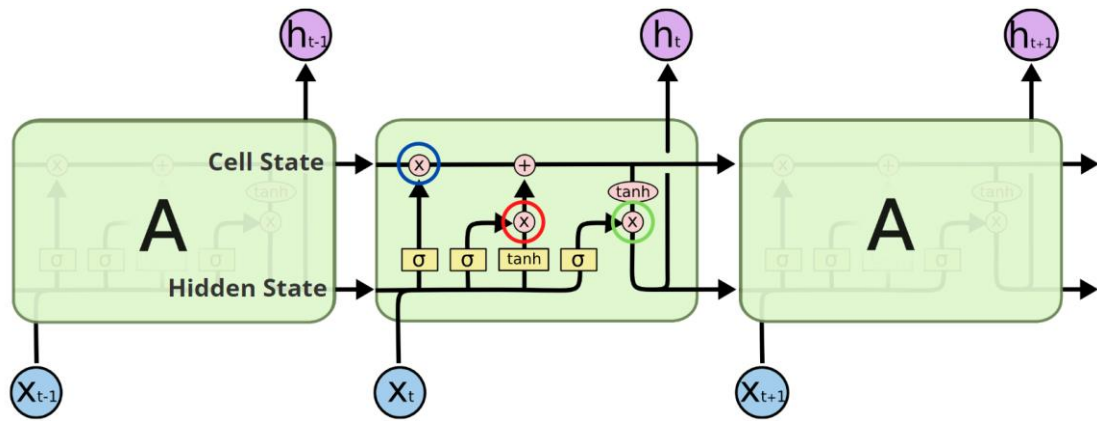


Figure 1.5.4.1: Visualisation of LSTM

The LSTM has 2 states and 3 gates, namely:

- Cell State** : Long term dependency in LSTM. Connected throughout the whole LSTM layer, allow information flow through the network while being selectively updated or ignored.
- Hidden State** : Short-term dependency in LSTM. the activation value of the LSTM layer at each time step, which summarizes the relevant information in the input sequence up to that point [7]. It is the combination of cell state and the activate value of the current input.
- Forget Gate** : (Blue circle in figure). Decide how much information from the previous cell state should be kept that impact the output of current timestep [8]. Value between 0 (to forget completely) and 1 (to remember completely).
- Input Gate** : (Red circle in figure). Decide to which extend the new content (output of tanh in the figure) of current input should be added into the cell state before passing to the next step [8]. Value between 0

## CHAPTER 1

(disable new content) and 1 (allow the full new content to be updated in the cell state). Summation of output from the forget gate and the input gate is the new cell state.

**Output Gate :** (Green circle in figure). Decide to which extend the new cell state should be passed to the next timestep. The new cell state (summation of previous cell state and tanh of current input) first go through a tanh activation, then is multiplied with the output gate to output the hidden state of the current timestep. Value between 0 (omit whole cell state) and 1 (reveal the whole cell state).

At a single timestep, the cell state of the previous step first goes through the forget gate to get output C1. The input of the current step goes through a tanh activation and combine with the input gate to get output I1. C1 and I1 is added as the new cell state C2 which will be passed to the next step. C2 will also go through the output gate, result in O1 as the hidden state (activation value, or the output) of the current timestep.

By using different combination of gate value, the cell state can be updated accordingly to retain useful information throughout all timestep to achieve a learning performance and inference accuracy.

### 1.5.5 Indoor Lighting

Lighting is the deliberate use of artificial or natural light to illuminate a space or an object [9]. It involves the use of lighting fixtures, such as lamps, light bulbs, and LED strips, to provide a desired level of illumination in a particular area. Lighting plays an essential role as setting of light could have direct impact on one's emotion, productivity and even health condition [10]. Hence, proper lighting is important when designing an interior space for activity.

Brightness (or luminance) is defined as the amount of light emitted by a light source or reflected from a source that is being perceived by a human eye, measured in lumens. A simple example is when human say the room is dark (or dim) or bright when the light is off (0 lumens) and on (lumens > 0). Higher brightness provides better visibility and motivate our feelings, while a lower brightness usually refers to a scene that is more moody or relaxing.

On the other hand, colour temperature refers to the “warm” or “cold” of a light source. It is measured in kelvin (k), where lower value corresponds to a warmer setting while higher kelvin shows a cooler tone. The difference of value also reflects in the colour of light. Having 5000K as the median which shows a white light that close to a daylight, lower temperature changes colour from yellow, orange to red, while higher temperature illuminates in cold blue tone. Warmer colour temperatures are often used in areas designed for relaxation, such as bedrooms or living rooms, while cooler colour temperatures are commonly used in areas designed for work or task performance, such as offices or workshops.

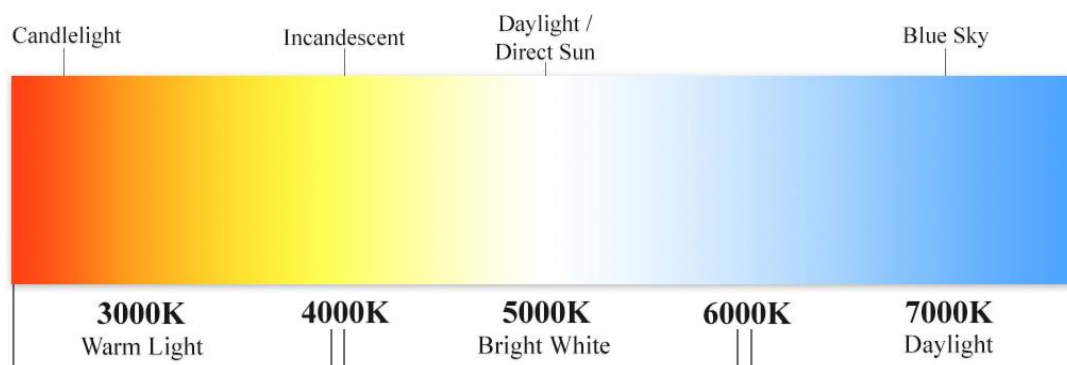


Figure 1.5.5.1: Colour temperature scale

## 1.5.6 Docker

Docker is an open-source platform designed to simplify the process of building, shipping, and running applications, using the concept of “container”. Unlike traditional virtual machines which run on hardware virtualization, docker container is implemented on OS-level virtualization which remove the need to include a complete operating system within the machine. This implementation makes docker container incredibly lightweight and resource-efficient, which multiple containers can run on a single host without the overhead associated with full virtualization.

Besides that, docker containers encapsulate all the necessary components and dependencies required to run an application, making it easier to ensure consistency and reproducibility across different environments. Developer will need to export the docker image using dockerfile script, and the application can easily be deployed on other devices/environments by pulling the image to create the docker container. This ensure the consistency of the application and greatly reduced the effort to maintain the code, runtime, libraries, and system tools etc..

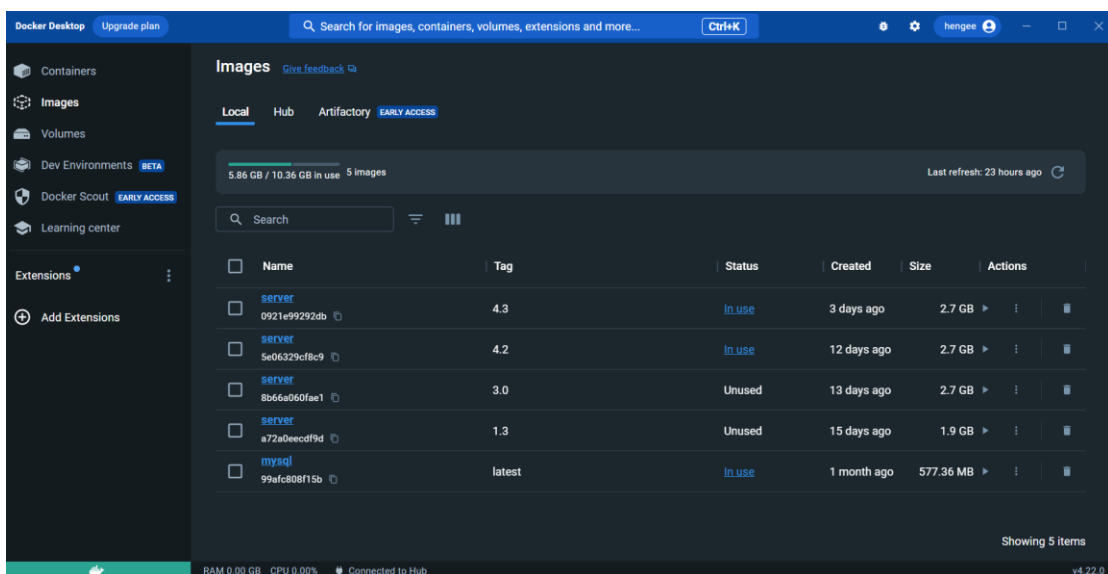


Figure 1.5.6.1: Interface of docker desktop

## CHAPTER 1

### **1.6 Report Organisation**

The whole report is organized into 7 chapters.

In Chapter 1, the overview of the project is provided by stating the problem statement, objectives and project scope. Some background information is also provided for better understanding on the remaining sections.

Chapter 2 is reviewing some similar project carried out by other researchers to understand the project background and discuss the possible improvement in current project.

In Chapter 3, the methodology to develop the project is proposed. The detailed design is discussed in Chapter 4 with suitable diagrams such as system architecture and flow chart.

In Chapter 5, the actual implementation of the project is recorded in detail with the code snippet and the screenshot of the system. It helps readers to understand the project by going through the whole implementation and the actual operation of system. The functionality of the system is tested here and proven to be workable.

In Chapter 6, the performance of the model training and the challenges faced during the project implementation are discussed. The report wrapped by at Chapter 7 by giving the conclusion and future recommendations to the project.

## CHAPTER 2

### Literature Reviews

The literature review is carried out in a way to explore the existing application of human activity recognition model with computer vision based, which cover the core of the proposed system which is to classify the type of indoor activity of the user. It also looked into 2 smart light configuring project which related to the second part of the proposed system to provide automated illumination base on activity type.

#### 2.1 Previous Works on Human Activity Recognition

##### 2.1.1 Patient Monitoring by Abnormal Human Activity Recognition Based on CNN Architecture [11]

[11] had proposed a multi-class abnormal action recognition model that apply deep learning method using You Only Look Once (YOLO) network of the Convolutional Neural Network (CNN). The trained model was aimed to perform recognition on the patient's abnormal activity for the application in hospital patient care field.

The CNN model applied for the system was built on top of YOLO. It has an architecture of 24 convolutional layers and connected with 2 fully connected layers. When an image is fed into the system, the CNN will divide the image into  $N \times N$  grid cells of random size (normally  $19 \times 19$ ) and then go through the network. The object(s) detected in the image will then be bounded with a bounding box and a confidence score. Each bounding box was labelled with  $x, y, w, h, c$ .  $X$  and  $y$  is the centre point of the object,  $w, h$  indicate the width and hight, while  $c$  is the class probability of the detected object (define how probably the object is belong to the detected class). YOLO was chosen as the backbone of the system because its characteristic of detecting object in a single run, which minimize the time and suitable for real time monitoring. The CNN model is retrained with LIRIS dataset (a dataset that have 10 human activities) and an 8 abnormal actions dataset for this project.

To train the suggested CNN model for the use of patient's abnormal action recognition, the author had built their own dataset that focusing on the abnormal activity of the patients due to the absence of suitable dataset in the current field. Focusing the common abnormal activity, 8 categories of abnormal activity as follow was selected

## CHAPTER 2

from the dataset to train the model: backward fall, chest pain, coughing, faint, forward fall, headache, vomiting and taking medicine. The dataset was built by 8 volunteer, each recording 3 videos of 4-5 seconds length for each particular abnormal action. Each video is segmented into 30 frames per second (fps), which develop a total of 2880 frame for each action, results in a final dataset of 23,040 frames. The whole dataset was manually labelled with the aid of Visual Object Tagging Tool (VoTT) by dragging the bounding box around the patient and specifying the class id of such action as the diagram below. Before testing, the dataset is separated into training and testing sets in 2 different ratio – 60 : 40 and 70 : 30 to evaluate the effect of training set size on the accuracy of the model.



Figure 2.1.1.1 Labelling of dataset

In the testing for the group of 60 : 40, the model performed well which manage to reach an average of 85.246 for precision, 85.264 for recall, 85.057 for F1 score and 95.77 as the final accuracy. While for the group of 70 : 30, the result is even better which report an average of 90.134 for precision, 88.421 for recall, 89.869 for F1 score and 96.8 of accuracy.

Sr. No.	Parameters → Abnormalities ↓	Precision	Recall	F1-Score	Accuracy
1	Backward Fall	83.004	92.105	87.318	96.296
2	Chest Pain	86.638	75.281	80.561	94.236
3	Coughing	87.037	80.342	83.556	95.542
4	Faint	74.131	88.073	80.503	94.461
5	Forward Fall	86.726	81.328	83.94	95.485
6	Headache	77.406	78.39	77.895	93.791
7	Taking Medicine	91.703	96.33	93.96	98.326
8	Vomiting	95.327	90.265	92.727	98.022
Average		85.246	85.264	85.057	95.77

Sr. No.	Parameters → Abnormalities ↓	Precision	Recall	F1-Score	Accuracy
1	Chest Pain	91.25	81.716	86.22	95.982
2	Coughing	88.393	85.714	87.033	96.592
3	Faint	84.339	88.889	83.843	95.762
4	Forward Fall	89.13	85.774	87.42	96.592
5	Headache	87.845	83.193	83.019	95.379
6	Taking Medicine	95.217	96.476	95.842	98.876
7	Vomiting	95.475	91.342	93.363	98.237
Average		90.134	88.421	89.269	96.8

Figure 2.1.1.2 Result of the model

The strength of the project is it introduces a fast-recognising model that could accurately recognise the abnormal action of a patient. The model also implemented by



## CHAPTER 2

a single camera, which distinguish from the other works that would require multiple camera viewpoints which reduces the cost significantly. This could be particularly useful to support the real time monitoring work in patient care to improve the efficiency and lifesaving especially in a big health institution.

Even though the proposed model performed well, the dataset used are simple which only have a single patient in the frame and the background are more consistent. The performance may be degraded when placing in a real-life scenario where multiple individuals may appear in a single frame. Also, the proposed model has a difficulty to handle scenario of overlapping object in group and small project.

### 2.1.2 Hockey Activity Recognition Using Pre-Trained Deep Learning Model

The proposal by Rangasamy et. al. [12] tends to address the activity recognition in the field of sports activity. The proposed work adapted deep neural network approach of VGG-16 model to perform recognition on a self-collected hockey dataset. It aimed to recognize 4 actions: free hit, goal, long corner and penalty corner which are most essential in the hockey game for scoring.

This project has chosen the method of transfer learning to set up the convolutional neural network. VGG-16 was chosen to be the CNN and was pretrained with the full dataset from ImageNet. The architecture has 16 convolutional layers, connected with 3 fully connected layers and completed with a SoftMax layer at the end [12]. To suit the purpose of this project, the final layer of fully connected layer in the original VGG-16 model was replaced with a new layer for the classification of hockey activity.

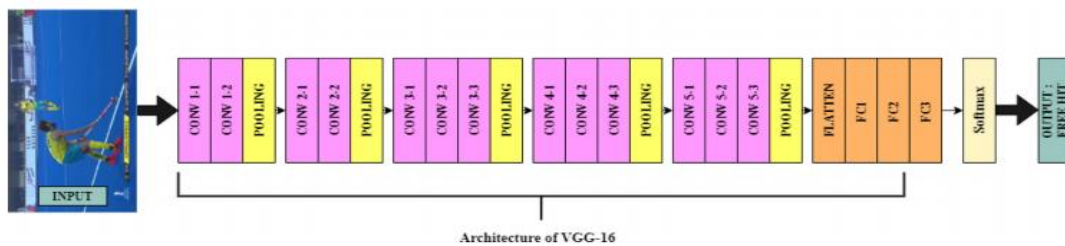


Figure 2.1.2.1: Visualisation of VGG-16 architecture

The actual model of the proposed work was implemented with Keras v2.3.1 and Tensorflow library. Training on the pretrained model with the collected dataset is performed with 3 epoch sizes (100, 200, 300) and a default batch size of 32 to evaluate the performance of epoch on the result. For the optimization of model, Adam optimizer from Keras was used with hyperparameter of learning rate 0.001, beta 1 0.9 and beta 2 0.999.

Due to the absence of publicly accessible benchmarked dataset for the hockey sport, the authors had collected the dataset of the mentioned activity themselves from the hockey match, e.g., Hockey World Cup 2018 from the International Hockey Federation broadcast YouTube channel. The dataset is challenging for recognition as it involved multiple camera view and appearance of the recognising object. The videos collected were having 1280 x 720p resolution and converted into frames of 25fps. The

frames of the chosen activities where then manually annotated for training the model. To ensure the balance of data, 100 frames were selected for each of the 4 chosen activity, which constructed a database of 400 frames for the subsequent work. The frames were then reorganized into size of 224 x 224 to suit the VGG-16 model.

Accuracy Matrix of proposed Model.

No. of epochs	Class	Precision	Recall	F1-Score	Accuracy
100	Free Hit	1.00	0.80	0.89	0.90
	Goal	0.93	0.93	0.93	
	Long Corner	0.82	0.90	0.86	
	Penalty Corner	0.86	1.00	0.92	
200	Free Hit	1.00	0.80	0.89	0.95
	Goal	1.00	1.00	1.00	
	Long Corner	0.83	1.00	0.91	
	Penalty Corner	1.00	1.00	1.00	
300	Free Hit	1.00	0.90	0.95	0.98
	Goal	1.00	1.00	1.00	
	Long Corner	0.91	1.00	0.95	
	Penalty Corner	1.00	1.00	1.00	

Figure 2.1.2.2: Result of the model

The result of the model was tabulated in the figure above. Precision is referring to the accuracy of positive prediction (true positive rate out of true positive + false positive estimation), recall is referring to the rate of sample that has been correctly recognised as their actual class , and F1 is the harmonic mean of the two. Hence, F1 score is taken as the key evaluation for the model. It clearly shows that epochs of 300 get the best F1 score in overall and obtain the highest accuracy of 0.98.

The proposed project had achieved a compromising result of 0.98 accuracy on recognising the hockey activity. Such method could be adapted in the real world to evaluate the performance of the players for training and boosting in an automated manner which highly reduced the burden of coach and trainer. Besides that, the model was implemented with the concept of transfer learning, hence greatly minimized the requirement of large dataset and effort for training from scratch, at the same time receive a high accuracy rate which is more cost effective. However, the model still not to be perfect as it only trained for 4 activities and does not take temporal feature into consideration. Hence, the work may be further improved by adapting Long Short-Term Memory (LSTM) in the model for spatiotemporal evaluation and train with more actions to raise the performance and able to commercialize in sports training field.

### 2.1.3 Long-term Recurrent Convolutional Networks for Visual Recognition and Description

In [13], the authors addressed the need of learning spatial and temporal features in video deep learning model by proposing the Long-term Recurrent Convolutional Networks (LRCNs) which combined convolutional layers and long-range temporal recursion for tasks including video recognition and video captioning. The authors suggested 3 setups of LRCN for 3 problems: sequential input, static output (video recognition); static input, sequential output (image captioning); sequential input, sequential output (video description).

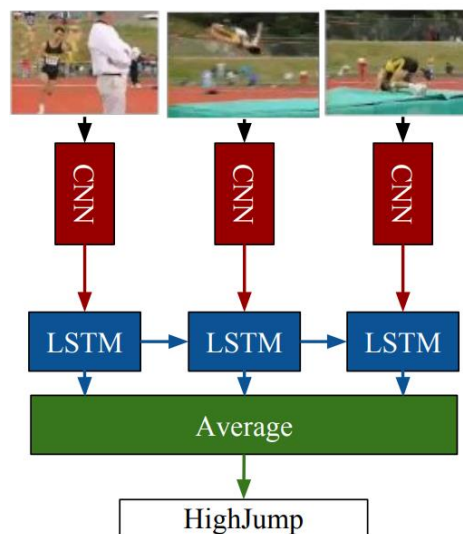


Figure 2.1.3.1: Illustration of LRCN architecture for sequential input, static output

The model setup for video recognition task is shown in figure 2.1.3.1. Considering an instance to the model is of  $T$  inputs  $(x_1, x_2, \dots, x_T)$  (i.e.,  $T$  frames or  $T$  timestep). The model first takes in frame  $x_t$  in sequence from the instance. Each frame is processed by the CNN for feature transformation that product a fixed-shape vector  $\phi V(x_t)$ . Then, the outputs of fixed timestep  $\phi V$  is fed into the LSTM network which output the classification result for each  $\phi V(x_t)$ . Late fusion approach is applied by calculating the average of per-timestep prediction as a single predicted output for the input instance. The weight of CNN and LSTM is shared all across.

The model was tested with the UCF101 video dataset. It is a dataset with realistic videos collected from YouTube, having 101 categories with average 130

## CHAPTER 2

samples each, total up to 13320 videos that range from 1 seconds to 10 seconds long at 25 fps and resolution of 320 x 240 [14].

For the training of this model, the videos are resized to 240 x 320 and augmented using 227 x 227 crops and mirroring. Instead of taking the full extracted frames as one sample, video clips of 16 frames were extracted from each video. Another dataset is generated based on the optical flow of frames. The optical flow is calculated using method proposed by [15] which the flow value redistributed to range (-128, 128) and added with channel by determining the flow magnitude [13].

The CNN used in the model is a hybrid of CaffeNet (single-GPU version of AlexNet) reference model that pretrained with ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC-2012) dataset [13]. Performance of the LRCN model is compared to the baseline of Single Frame classification using the same pretrained CNN, where each frame of the video clips is treated as an image classification task without the LSTM layer.

<b>Model</b>	<b>Single Input Type</b>	
	RGB	Flow
Single frame	67.37	74.37
LRCN-fc <sub>6</sub>	<b>68.20</b>	<b>77.28</b>

Figure 2.1.3.2: Result of the model

Label	$\Delta$	Label	$\Delta$
BoxingPunchingBag	40.82	BoxingSpeedBag	-16.22
HighJump	29.73	Mixing	-15.56
JumpRope	28.95	Knitting	-14.71
CricketShot	28.57	Typing	-13.95
Basketball	28.57	Skiing	-12.50
WallPushups	25.71	BaseballPitch	-11.63
Nunchucks	22.86	BrushingTeeth	-11.11
ApplyEyeMakeup	22.73	Skijet	-10.71
HeadMassage	21.95	Haircut	-9.10
Drumming	17.78	TennisSwing	-8.16

Figure 2.1.3.3: Difference between accuracy of LRCN and Single Frame Prediction

Based on the result, LRCN achieved a better accuracy of 68.20 on RGB dataset compared to single frame prediction at 67.37. Meanwhile, the optical flow dataset outperformed RGB dataset with accuracy of 77.28 which shows that flow dataset could be more suitable for video classification task. The LRCN model also recorded a great improvement in accuracy of each class with greatest improvement of 40.82 in the

## CHAPTER 2

BoxingPunchingBag class. The research also tried with different model parameters which found that changes in LSTM units (256, 512 and 1024) only shows a little impact on the performance when testing with RGB input.

The research successfully showed that combination of CNN and LSTM could leads to a better performance in video classification task. However, the performance of the model is only considered acceptable for RGB input compared to Optical Flow input. Fortunately, the proposed model is convenient to test with different setup as the CNN layer can be replaced with any state-of-the-art CNN model for a better performance. Total training time is also relatively shorter as it only train on the LSTM layer.

### 2.1.4 Deep Neural Networks for Kitchen Activity Recognition

This research done by Monteiro et. al. [16] had focus their aspect of human action recognition research in addressing indoor environment activity recognition with single static camera. The authors proposed a novel deep neural architecture which evaluated on the Kitchen Scene Context based Gesture Recognition dataset (KSCGR) and achieved a significant accuracy compared to the state-of-art.

The authors believe that different CNN will capture different patterns on the same data. Also, different perspective of view on the data will have a better support on the classification. Combination of the above instance may result in a better performance. With such believe, this research proposed an architecture which implement different combinations of data, CNNs and fusion methods, as illustrated in the following section. Figure 2.1.4.1 shows the pipeline of the proposed architecture.

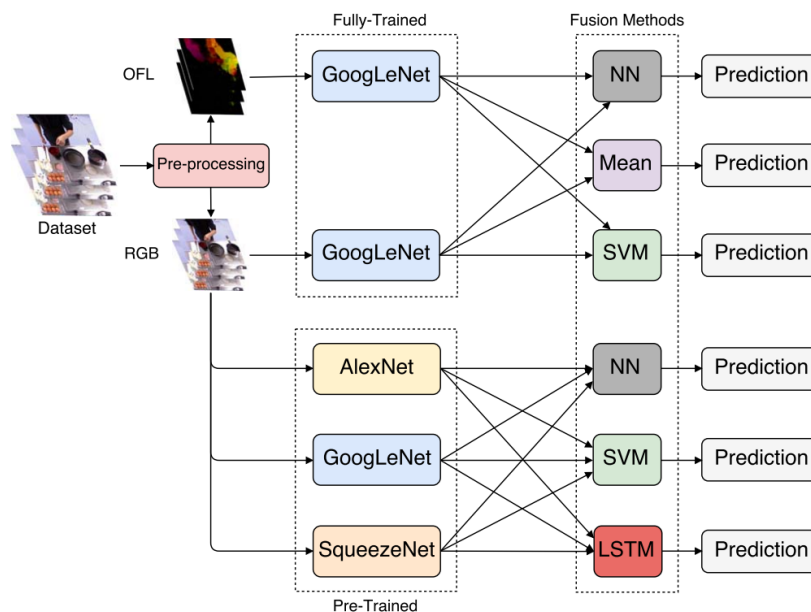


Figure 2.1.4.1: Pipeline of the proposed activity recognition architecture

The proposed architecture can be distributed into 3 segments, including (1) data pre-processing, (2) CNNs for recognition, (3) Fusion methods, and an additional step (4) post processing [16].

## CHAPTER 2

### **(1) Data pre-processing**

This stage involved 2 actions – image resizing and optical flow generation. First, all inputs are resized to a fixed resolution of 256 x 256. Next, the system generates a dense optical flow (OFL) representation for each adjacent frame. This is a representation that trace the displacement of each pixel between 2 adjacent frames when moving from the first to the second and by generating a corresponding vector [16]. The pre-processing stage delivered 2 sets of input data – the RGB version (original and resized) and the OFL for all the frames.

### **(2) CNNs layer**

This segment is the core which perform the activity recognition on the input data. The layer is separated into 2 segments – The full-trained CNNs and the pre-trained CNNs.

The fully trained segment consists of 2 GoogLeNet that remain the same architecture of 22 layers deep and configured with the same training hyperparameters. The 2 networks are assigned with different input, one to handle the RGB input and the other to process OFL input. These 2 CNNs were trained from scratch with the KSCGR dataset.

The pre-trained CNNs is a composition of 3 pre-trained models – GoogLeNet, AlexNet and SqueezeNet. These models are pre-trained with the ILSVRC 2012 ImageNet dataset. The models were configured with a learning rate to allow better adaption to the classify the new input from KSCGR dataset. This segment was responsible to perform classification only on the RGB dataset.

At the end of the classification, each network in each segment will generate their own classification probability vector on the inputs, which is passed to the fusion methods for final classification.

### **(3) Fusion methods**

This segment is responsible to fuse the probability vectors received from the networks for the classification.

Before fusion, the classification score vector from each network is merged as one as the input of the fusion methods. In other words, the fusion methods for full-trained network will take an input vector which is the merged results from GoogLeNet



RGB and GoogLeNet OFL. Similarly, the full-trained segment will have an input vector which is the merge of results from AlexNet, GoogLeNet and SqueezeNet.

For input from full-trained networks, 3 fusion methods were adopted including Neural Network (NN), Mean and Support Vector Machine (SVM). While the pre-trained networks also adopted 3 fusion methods, including NN, SVM and Long Short-Term Memory (LSTM).

#### **(4) Post Processing**

As the input of an action was broken into series of frames which treated individually, performance may be affected when a certain frame within the series was incorrectly classified. After testing, the author had proposed that a frame that sit in the middle of a series of 20 frame, which have the different classification compared to the series can be noted as misclassification and can be removed.

The dataset used in this research is Kitchen Scene Context based Gesture Recognition dataset (KSCGR). This is a dataset used in the ICPR 2012 challenge. The dataset is a collection of recoding of preparing 5 menus in a kitchen at bird's eye view, captured with a position fixed Kinect sensor [16]. Each menu preparation is a single video that last 5 – 10 minutes, reflected as 900 – 1800 frames per video. 8 actions are performed in the video, include breaking, mixing, baking, turning, cutting, boiling, seasoning, peeling and none. The menus are prepared by 7 different individuals, result in a dataset of  $7 * 5 = 35$  videos.

To better understand the performance of the multiple-network architecture, the figure 2.1.4.2 shows the summarization of all combination of architecture, and its respective result.

Approach	Precision	Recall	F-measure	Accuracy
HCF [7]	0.62	0.63	0.61	0.64
HCF + PP [7]	<b>0.68</b>	<b>0.68</b>	<b>0.68</b>	<b>0.72</b>
GoogLeNet <sub>[RGB]</sub>	0.69	0.68	0.69	0.69
GoogLeNet <sub>[OFL]</sub>	0.64	0.63	0.63	0.63
GoogLeNet <sub>[Off-the-shelf]</sub>	0.61	0.61	0.61	0.61
GoogLeNet <sub>[RGB+OFL]</sub> + Mean	0.71	0.69	0.70	0.69
GoogLeNet <sub>[RGB+OFL]</sub> + SVM	0.67	0.72	0.70	0.72
GoogLeNet <sub>[RGB+OFL]</sub> + NN	0.72	0.73	0.72	0.73
AlexNet <sub>[Fine-tuned]</sub>	0.77	0.75	0.76	0.75
GoogLeNet <sub>[Fine-tuned]</sub>	0.73	0.71	0.72	0.71
SqueezeNet <sub>[Fine-tuned]</sub>	0.70	0.66	0.68	0.66
AlexNet <sub>[Fine-tuned]</sub> + SVM	0.76	0.72	0.74	0.72
GoogLeNet <sub>[Fine-tuned]</sub> + SVM	0.72	0.71	0.68	0.71
SqueezeNet <sub>[Fine-tuned]</sub> + SVM	0.64	0.59	0.61	0.59
3 CNNs + SVM	0.73	0.67	0.70	0.67
3 CNNs + NN	0.77	0.75	0.76	0.75
3 CNNs + NN + PP	0.77	0.76	0.75	0.76
3 CNNs + LSTM	0.78	0.78	0.78	0.78
3 CNNs + LSTM + PP	<b>0.80</b>	<b>0.78</b>	<b>0.79</b>	<b>0.79</b>

Figure 2.1.4.2: Result of the model

\* The HCF (Hand-crafted Feature) on the top is the state-of-the-art (at that moment of research) in recognizing the same piece of dataset.

Referring to the table above, the architectures can be generally separated into the following groups:

- Single view of data (full-trained CNN) (GoogLeNet<sub>[RGB]</sub>, GoogLeNet<sub>[OFL]</sub>)
- Combined view of data (full-trained CNN) + fusion methods (GoogLeNet<sub>[RGB + OFL]</sub> + Mean/SVM/NN)
- Single CNN (AlexNet<sub>[Fine-tuned]</sub>, GoogLeNet<sub>[Fine-tuned]</sub>, SqueezeNet<sub>[Fine-tuned]</sub>)
- Single CNN + SVM
- Combined CNN + fusion methods (3CNNs + SVM/NN/LSTM)
- Combined CNN + fusion methods + post processing (3CNNs + SVM/NN/LSTM + PP)

Based on the result in F-measure and Accuracy, it shows that combined CNN have a better performance over single CNN. Combined data view also result in a better performance over the single data view. Also, the combined pre-trained CNN (3CNNs) have a better performance over the combined full-trained CNN. 3CNNs that go through the post processing also result in a slightly better performance compared to the respective 3CNNs that did not perform post processing. Comparing with the state-of-

## CHAPTER 2

the-art model at that time, the proposed architecture had well performed over it with 0.78 accuracy compared to the HCF (0.64) without post processing.

The pro of the proposed deep neural network structure is the application of the combination of multiple CNNs. Such architecture allowed the collaboration between CNNs to achieve a higher accuracy. However, the con of such proposal is also obvious which reflect in the load and processing time and requirement on the processing environment.

### 2.1.5 Summary of Reviewed Works

The following is a summary of the reviewed works in the previous section. As the project only aimed on classification on human activity and the dataset used does not include bounding box annotation, YOLO which mean more for object detection and require additional effort for manual annotation in not chosen. Meanwhile, the architecture proposed in 2.1.4 is too heavy for inference on Jetson Nano. Hence, Setup in 2.1.2 was chosen by extending the work to append VGG CNN with LSTM as proposed in 2.1.3.

Table 2.1.5.1: Summary of the reviewed works

Section No.	Project Type	Architecture / Technology	Dataset	Accuracy
2.1.1	HAR for patient monitoring	YOLO	Type: Patient abnormal action No. of action: 8 Details: 30fps 23,040 frames	Set 1: 95.77 Set 2: 96.80
2.1.2	HAR for hockey action training	VGG-16	Type: Hockey World Cup 2018 No. of action: 4 Details: 1280x720p 25fps 400 frames	0.98
2.1.3	HAR using LRCN	LRCN (CoffeeNet + LSTM)	Type: UCF101 No. of action: 101 Details: 320x240x3 25fps	RGB: 0.68 Flow: 0.77

CHAPTER 2

2.1.4	HAR for kitchen activity	Novel composite architecture (Referring to Figure 2.1.4.1)	Type: Kitchen activities (KSCGR) No. of action: 8 Details: 35 videos, 900 – 1800 frames per video	0.79
Proposed work	HAR for smart lighting	LRCN (VGG-16 + LSTM)	-	-

## 2.2 Previous Works on Lighting Adjustment Application

### 2.2.1 An IoT Based Smart Lighting System Based on Human Activity

This project by Adnan, Kamal and Chellappan [17] is attempted to provide a solution on the lighting in house. The proposed system allow user to change their lighting setting according to the type of activity that they would perform based on comparison with the ambient lighting condition.

The structure of the system was illustrated in figure 2.2.1.1. The system consists of an Aduino Uno microcontroller as the centre of the connection with other components, a mobile application for the user control via interface that communicate with Arduino via Bluetooth, a light sensor for detecting the intensity of surrounding currently upon adjustment request, a dimmer circuit to control the intensity of light bulb, a bulb that react to the adjustment and the power source to support the light bulb and microcontroller.

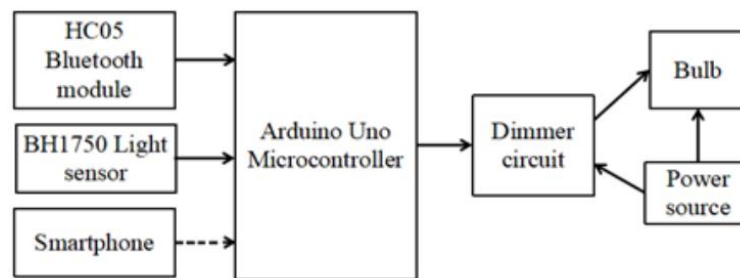


Figure 2.2.1.1: Structure of the smart lighting system

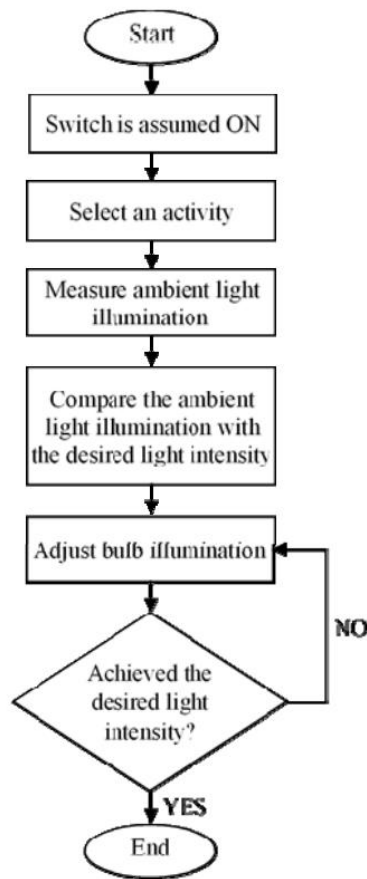


Figure 2.2.1.2: Flow chart of the smart lighting system

Activity	America (IES)	Europe (EN 12464)	Malaysia (JKR)	Japan (JIS Z9110)	British (BS EN 12464)	This work
Watching television	50	300	50	-	100	50
Relaxing	50	100	50	50	200	50
Reading	300	500	300	500	500	300
Cooking	-	500	-	-	400	450
Bathing	100	200	100	100	150	100
Laundry	300	300	200	100	300	200
Eating	100	200	300	300	-	300
Exercising	500	300	300	-	-	300
Sleeping	0.1	-	-	2	-	0

Figure 2.2.1.3: Light intensity guideline for various daily activity in different country

Figure 2.2.1.2 illustrate the flow of using the system. Before using the system, user is required to install a mobile application that was designed for the control of the system. 9 activities including eat, watch tv, sleep, bath, relax, exercise, cook, laundry and reading with its respective lighting guideline has been included in the application. From here, user can choose the activity that they would like to do. Once selected, the light intensity value will be passed to the microcontroller. The light sensor will then detect the light intensity of current surrounding and pass to microcontroller. The 2 intensity values are compared. If the ambient lighting is greater or equal than the lighting requirement of the intended action, the light bulb will not response to the request. If it is lesser, then the bulb will light up and adjusted to the difference of the 2 intensity values to ensure that the lighting condition meet the required standard.

The proposed work is a brilliant idea that take ambient lighting into account in defining the illumination, rather than solely adjust the light base on the standard

## CHAPTER 2

intensity. This could keep the illumination meet the standard and avoid the lighting to go beyond the requirement which is a waste energy and may be harmful to eyesight healthy. However, the proposed work also has its limitation as it required user to manually choose the activity to do through the mobile phone. In a case where the phone is not around or missing, the use will lose control towards the light. The system could be improved by adapting activity recognition to classify the type of activity and adapt the lighting to change accordingly in an automated manner to make the system towards the concept of “smart” system.



### 2.2.2 Applications of Human Motion Tracking: Smart Lighting Control

Chun and Lee [18] has applied the human motion tracking technology to implement a smart lighting system that adjust based on human activity.

The human detection applied in the system can be divided into 3 segments: The detection of human location, the estimation of heading direction and the global trajectory of human.

In the first segment, the detect of human and the location estimation was realised with the help of Kinect camera. The existence of human was captured by the mentioned cameras to obtain the depth and thermal information. In the use of Kinect camera, the camera was placed in the ceiling to perform detection of human and positioning by using depth information and extracting the head area of human from the captured image as in figure 2.2.2.1. The location of the man was recorded in x, y and z which referring to the centre of head, and the height which measured by the distance between the top of head and the camera.

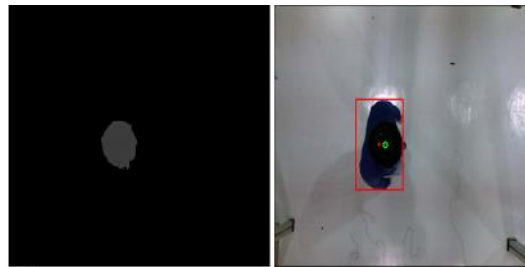


Figure 2.2.2.1: Estimation of human location by extracting the head area

In the second segment, the heading direction of the man was estimated by shoulder detecting. As illustrated in figure 2.2.2.2, the heading direction of a man could be estimated by defining the centre of the shoulder. A bounding box was applied to bound the human from image captured by Kinect camera. The information of the head area was removed from the image. Then, a straight line is drawn by connecting the 2 points on edge and the centre point is defined. Using the straight line as a guide, the larger area is considered as the shoulder, and the opposite area is the head, and the direction is where our face is facing and moving towards.

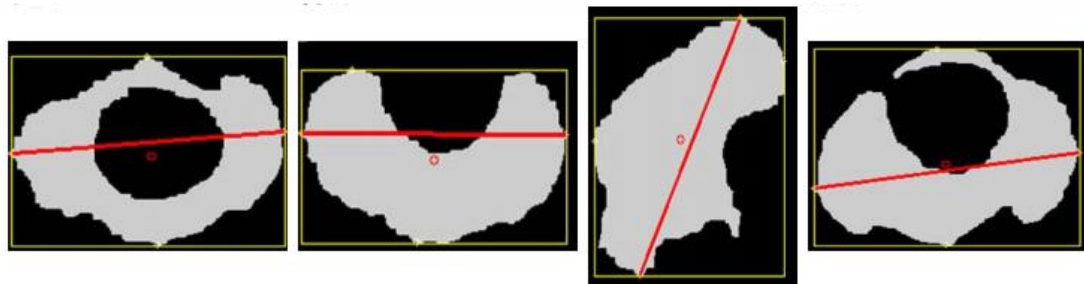


Figure 2.2.2.2: Estimation of shoulder for heading direction estimation

In the third segment, a global trajectory estimation and tracking system was implemented with a global map image. The system has a global trajectory map that traces the current location of the person. The image captured by the Kinect camera was converted into a 2D plane, and the location information was mapped to the global map to keep track of the person's current location in the monitoring area. This allowed for a large area of coverage by implementing multiple cameras to cover the entire area. By using the centre location, height, and clothing colour information of the person extracted from the depth image, an identity was created to keep track of the person as they passed through from one camera coverage area to another, as well as to keep track of multiple individuals appearing in the same area.

By incorporating the above techniques, a smart lighting system was realised that allow the adjust of illumination and lighting area.



Figure 2.2.2.3: Setup of the room

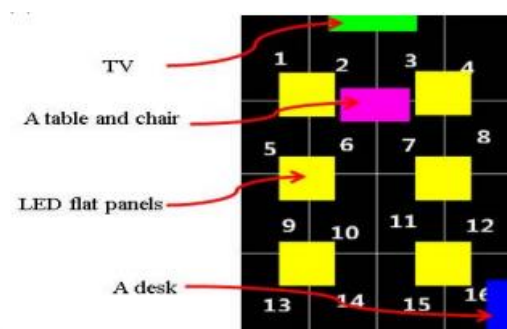


Figure 2.2.2.4: The global trajectory map

First of all, with the heading estimation and the global map (where the placement of light and furniture are also marked in the map), the system can perform adjustment by making the rear lighting (the back of the man) dimmer, and the front area (where the man is facing and walking towards) brighter.

## CHAPTER 2

Secondly, the system can classify human activity by identifying the individual's location and current height (Z axis), and adjust the lighting accordingly based on the activity. For instance, if a man is sitting in front of the television, the system can classify it as "watching TV" and adjust the lighting accordingly. The system supports four types of activity recognition: study mode (when the person stays close to the desk), dialog mode (when two people are in close proximity near the table), moving around mode (when the person is walking around the area), and watching TV mode (when the person is sitting on the sofa and staying close to the TV). The lighting was adjusted to 7000K, 5000K, and 6500K for the first three actions, respectively. For the "watching TV" mode, a web camera was used to capture the dominant colour of the current TV screen, and the lighting is adjusted to match the colour.

The proposed smart lighting system is considered a complete system which can be applied in real life event. It supports the automation of lighting control and tracking of multiple people in the same area which is useful in the actual lighting control activity. Besides, it provided a human optimized the usage of energy by focusing the lighting in the area where the man appeared and adjust the illumination according to the guideline.

However, the system possessed a weakness in the classification of human activity. The activity recognition is simply based on the location of the man without considering what the man is actually doing. For example, instead of having a conversation, 2 men remain near at the table may also be 2 men sitting at the table and reading separately without any conversation. A man remains close to a desk may also be doing something else such as playing game, doing some handicraft etc. which not necessary to be reading.

To overcome the issue, the system may implement human activity recognition to perform the classification of action accurately. As vision-based recognition system is analysing based on the captured image, it could effectively guarantee that the action recognized match the actual activity performed by the user, instead of simply based on the location of the user.

### **2.3 Lighting for Each Activity Class**

Lighting is always an essential aspect of creating a comfortable and functional indoor environment. The setting of illumination shows a relationship with psychological state which brings impact on the performance on the activity performed [19]. For example, study support that indoor lighting with high correlated colour temperature (17000 K) in an indoor office floor setting shows positive impact on the workers wellbeing and also supporting their performance [20]. General parameters of adjusting an illumination include brightness, colour, and colour temperature. The following discussed suitable setting for the 7 activity classes to be classified in this project.

#### **2.3.1 Dining**

People tend to be more relax and comfortable when they are enjoying their food. Hence, warmer lighting is preferred as it helps to develop a cosy and intimate feel that stimulate appetite. It is suggested to have a colour temperature of around 2700K to 3000K which is a warm white light for best dining experience [21].

#### **2.3.2 Drawing**

Lighting is really important in drawing as the activity deals with paint colour which shows a vast contrast under different light setting. Daylight is most preferred as cool and sharp light help to ensure the colour is presented in the most accurate form [22]. Natural light is suggested at a range of 4000k – 6500k [23], while sunlight has a temperature of 5900k [24]. [25] also suggested that colour temperature around 5000k is optimum for visual appreciation of paintings.

#### **2.3.3 Mopping Floor**

The light setting for household activity, i.e., mopping floor, should cater both productivity and visibility. Productivity is in terms of finish the task faster and more efficient, while visibility is important as it help man to clearly identify all the tiny dirt to ensure that the floor is cleaned thoroughly. With that being said, a cold blue light is most suitable as it guarantees visibility at the same time keeping man focus and improve productivity [26].

### **2.3.4 Reading Books**

The light setting for reading should provide adequate illumination that minimizes eye strain and enhances visual clarity. Ideally, the light should be bright enough to clearly illuminate the reading material, but not so bright that it causes glare or reflections. Considering the general scene where the reader is reading with focus to take in information, a cool white light at 4000K to 6500K that assimilate the daylight can help to ensure the reading material is clearly visible and also keeping the reader awake for better reading experience [27].

### **2.3.5 Running on a Treadmill**

This class activity represents the exercise class. The aim of lighting for exercising is to create an ambient that promote motivation and focus that can motivate the user to keep working out [28]. Referring to the lighting of a gym room, cool light within 4000K – 6000K are optimum for exercise.

### **2.3.6 Sleeping**

Research suggested that secretion of melatonin, a hormone that regulates sleep-wake cycle, is directly proportioned to the darkness level [29]. With warmer light providing a relaxing environment, 1000K with minimum brightness level is used for sleeping class in this project. If the class is continuously detected for over 30 minutes, power off command will be issued to turn off the light.

### **2.3.7 Watching TV**

The lighting consideration for watching television is to balance between developing a comfort and relaxing environment while ensuring proper light levels to reduce eyes strain. Hence, the warm white light with temperature 3000K is suggested with maximum brightness level to balance the light from television.

### **2.3.8 Yoga**

Yoga is a kind of meditation exercise that bring human to inner peace. Hence, lighting should be defined to create a calming and balanced atmosphere that promotes relaxation, focus, and tranquillity. Therefore, the lighting temperature is suggested to 3500K for a warmth tone light with sufficient brightness.

## CHAPTER 3

### Proposed Method/Approach

#### 3.1 Methodologies

The proposed system can be separated into 3 segments: training the HAR model and developing the light calibration program.

Based on the general machine learning pipeline, the proposed system can be implemented following the methodology illustrated below:

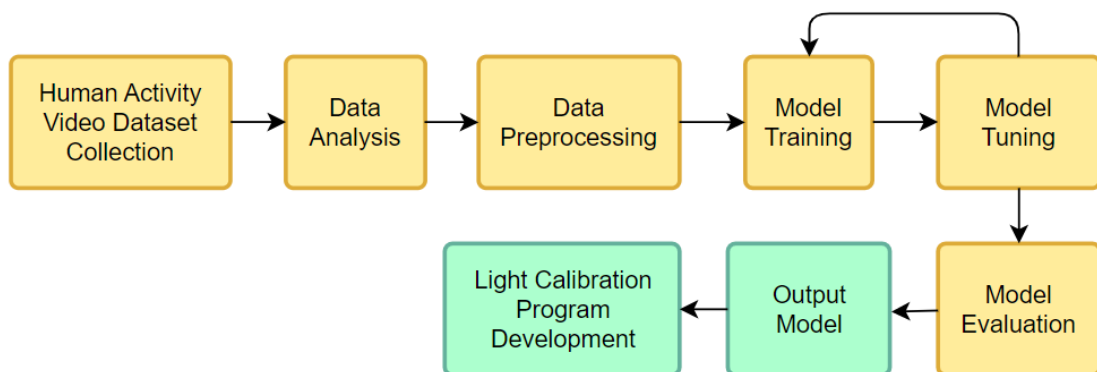


Figure 3.1.1: Project development methodology

Data acquisition can be done by searching for existing human activity recognition from that are publicly shared on the Internet. As the proposed system is using computer vision for classification, only video dataset is targeted. Most of the public dataset included both indoor and outdoor activity hence analysis on the dataset is needed to filter only the dataset and the respective class(es) that are suitable for the proposed indoor activity classification system.

Once the dataset ready, preprocessing is required to clean up and reconstruct the dataset in a way that fit the model input requirement without unintended noise. Steps of preprocessing includes splitting the dataset into train-validation-test set, extracting video into sequence of frames and reformat the frames into data that fit the model input size.

## CHAPTER 3

The finalized dataset can then be fed into the designed model for training. Training and validation dataset is used in the training phase which the first set is used by the model for learning while the second set is to use as a control group to monitor the training performance. Loss/accuracy graph is plotted after each training to visualize the performance throughout the training process, together with confusion matrix to evaluate the trained model. Then, the model setup and hyperparameter are adjusted to improve the model performance from issues such as underfitting or overfitting.

The best model achieved after tuning is used for model evaluation, which test set is used to evaluate the model actual performance on unseen data. The final model is used as the classification backbone to develop the lighting configuration program.

The final system is proposed to use client-server architecture. Under such setup, the image capturing and light controlling will be carry out on client side, while the prediction of user activity will be performed on the server side. The server also holds a database which allow user to save their own preferences on light condition for each actions in case the suggested setting is not suitable for them.

### 3.2 HAR Classification Model Network Architecture

This project proposed an CNN-LSTM architecture model. It is a fusion model which use Convolutional Network (CNN) as the feature extractor, and a Long Short-Term Memory (LSTM) layer that takes in the extracted feature from CNN that are stacked on sequence to learn on the temporal information.

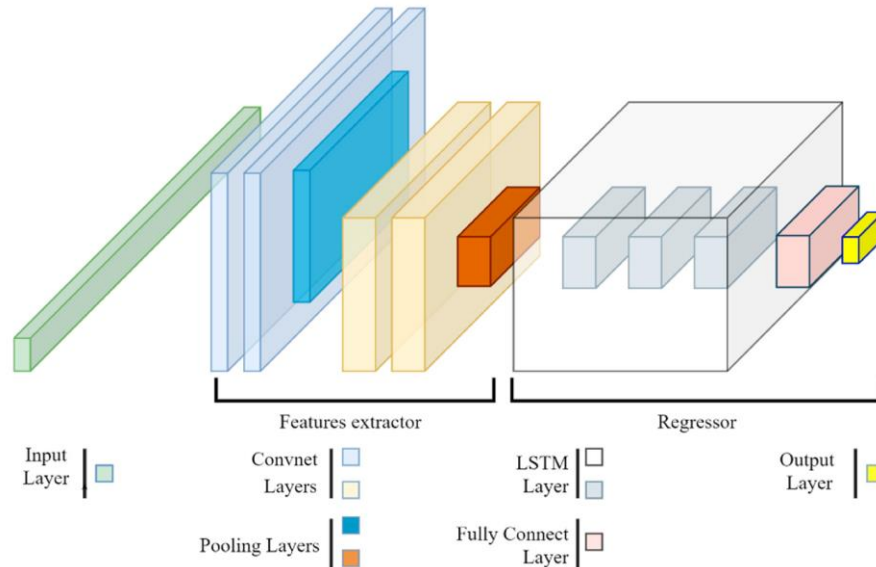


Figure 3.2.1: Architecture visualisation of CNN-LSTM model [30]

#### CNN:

The CNN is used as a feature extractor which process the input data and output the extracted features as feed for the LSTM. CNN is a model that only processing on batch of images (batch, width, height, channel) instead of batch of images that are stacked with based on timestep sequence (batch, timestep, width, height, channel). Since the model is performing inference on video input, the video dataset is pre-processed as input of shape (frame\_count, width, height, channel). To accommodate with the temporal information, the CNN model is wrapped with a Time Distributed Layer which increase the input dimension from (b, w, h, c) to (b, t, w, h, c). Published CNN model (e.g., VGG-16 and Resnet) were imported as the feature extractor with all layer weight frozen as these are well trained weight on the ImageNet dataset. The top layer is removed as it does not need to perform classification. Considering the limitation of hardware computing capability and to increase training speed, a pooling layer is attached after the last convolutional layer to reduce the output dimension, i.e., total number of parameters to train. The output is finally flattened and fed to LSTM for subsequent training.



### **LSTM:**

The LSTM model is where the proposed model begins to learn from the dataset. It takes in 2-d array input of shape (timestep, features). Timestep corresponded to the number of frames per sample, while features referred to the length of features extracted from the CNN. The proposed model used 1 layer LSTM ended by a Dense Layer of 8 neurons (correspond to the number of classes to classify) with SoftMax activation function to carry out the classification task.

### **3.3 Light Calibration Program**

The controlling system is proposed to operate using client-server architecture. The prediction of user action is implemented on the server side, while the streaming of user action and light controlling is implemented on the client side. This architecture overcome the limitation of computing resources on microcontroller, allowing the model to further upgraded with better complexity and more supporting action class.

Microcontroller which connected to the camera and light is served as the client to capture and forwarding the image to server, then perform light calibration according to the predicted result. A composite docker container integrating the trained model, a database and a HTML webpage is served as the server. The trained model is used for prediction on the input images from microcontroller, while the webpage allow user to configure their personal preferences on the lighting conditions which is saved in the database container.

### 3.4 Timeline

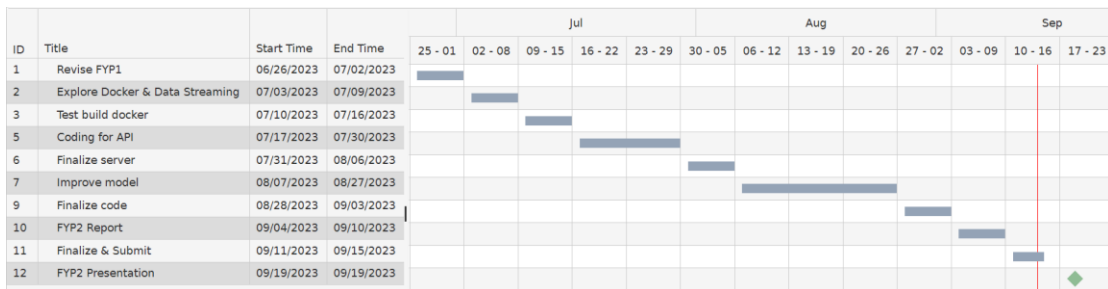


Figure 3.4.1: Proposed timeline for the project

The Gantt Chart above showed the planning of the project timeline. It started with revision on the completed works in FYP1. Approximately 1 weeks was allocated to explore on docker and data streaming. The subsequent week was used to try out the development with docker. Then, 2 weeks were reserved to complete the code for server API. The next week was assigned to finalize the whole server docker. After completing the server, 3 weeks were planned to explore the possible improvement on the HAR model on top of the model from FYP1. Another 1 week was planned to combine the final model and finalize all the coding work. The subsequent week is assigned for report writing and submission. After that, it was time to prepare for the presentation, and that wrapped up the final year project 2.

## CHAPTER 4

### System Design

#### 4.1 System Block Diagram

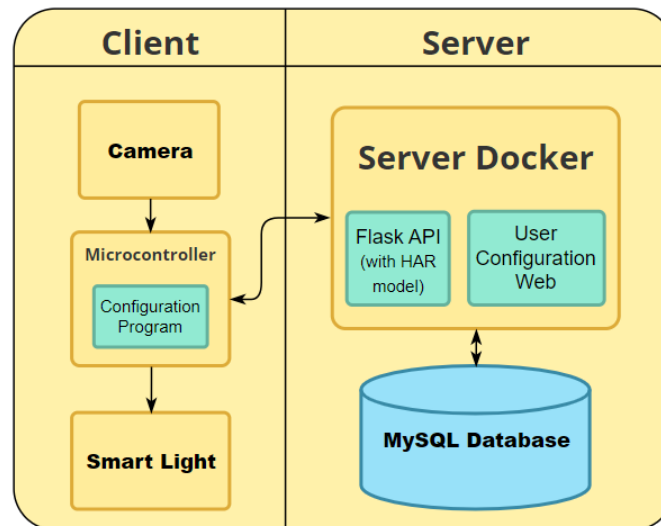


Figure 4.1.1: System Block Diagram

The overview of the system is illustrated as above. The whole system is set up based on the client server architecture. On the client side, the camera is used to capture user activity, while smart light is listening to the microcontroller for configuration command. Microcontroller act as the interface to pass the images from camera to HAR docker and receiving the prediction result and configuration parameter to post the configuration command to smart light device. On the server side, the Server docker hold the flask API with HAR model and User Configuration webpage. The flask API will receive the images from microcontroller, perform predictions, grab the corresponding configuration parameter from the database then send the result back to microcontroller. The User Configuration webpage act as the interface to get user input for their preferred light setting and save in the database. Lastly, the MySQL Database docker is used to save the supported activity and relevant light setting. The microcontroller will hold a python script for all the activity mentioned above, while the server docker will run a flask app for prediction and database communication. The camera, microcontroller and smart light are supposed to be within the same LAN for communication, while the sever and database can be held local or cloud.

## 4.2 System Components Specifications

### 4.2.1 Configuration Program

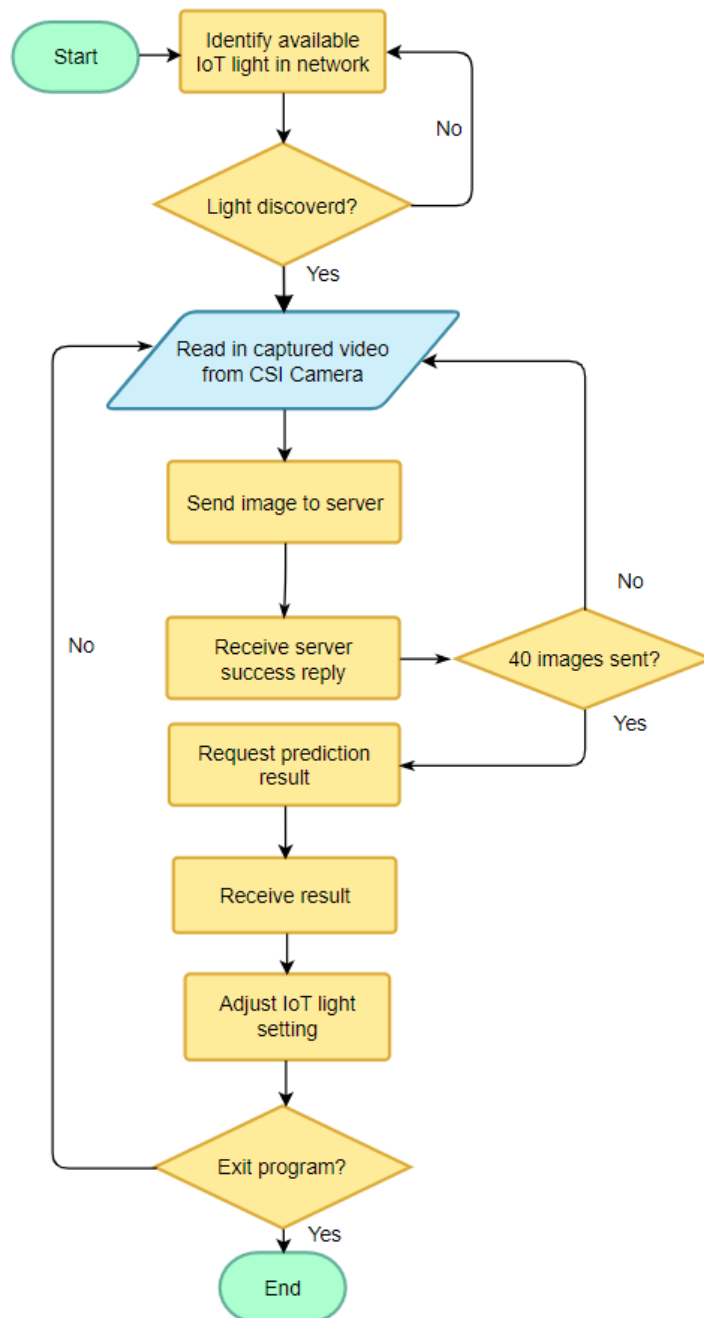


Figure 4.2.1.1: Flow chart of the configuration program

The microcontroller is the centre of communication between the camera, light and HAR server. A configuration program written in python is running on the microcontroller to support connection. The flow of the program is defined as below:

## CHAPTER 4

- Use Yeelight library to discover available Yeelight devices in the network and establish a connection.
- Use OpenCV to capture streaming images from the connected Camera.
- Establish the connection with server docker.
- Stream the images from camera to the flask API on server for model prediction using Requests library.
- After sending 40 images (size of an instance), pause streaming and request for prediction result and light configuration parameter.
- Receive response from server and configure the light accordingly.
- Loop the process until user manually stops the program.

### 4.2.2 HAR Model

As justified earlier, the HAR model is proposed to use the CNN-LSTM architecture. A typical setup would be stacking up the CNN and LSTM as a whole model for prediction. However, to preserve the usage of computing resources, CNN and LSTM were deployed separately in this project.

In training phase, the pre-processed data is pumped into the CNN for feature extraction. The extracted features saved as numpy arrays are then used for training on LSTM with different setting. In the flask API, the CNN extractor is created based on the best performing CNN from the training result, then the trained LSTM is loaded in for real-time prediction.

Besides preserving computing resources, such setting also improved the training efficiency as feature extraction only need to be performed once, then the extracted result can be stored and load in again for LSTM training whenever needed instead of doing feature extraction in every training epoch. Besides that, it also provided flexibility to setup the feature extractor using different CNN model to test for the optimal performance.

The input data are pre-processed to shape (batch, 40, 112, 112, 3), using 40 frames for one sample and the shape resized to 112x112x3. The extracted output would be in the shape of (batch, 40, feature length) where the feature length is dependent on the output layer of the CNN. The output shape is then used as the input shape of the LSTM input layer. The LSTM is defined with 128 units for the hidden layer with dropout 0.5 for regularization. The input then goes through a dense layer with 8 unit to produce the final result in shape (batch, 8).

### 4.2.3 Server Docker Container

The server side is designed to deal with the model performance and accepting user input. Considering the limitation of computing capability in microcontroller, the model prediction is moved to the server to support model with more activity class and better performance which go beyond the computing capability of microcontroller. The server container will hold the trained model, the Flask API as controlling program and User Configuration Webpage for user frontend. It is exported as docker image to ease the process of python library installation when setting up the API on new server.

### 4.2.4 Flask API

The Flask API is a RESTful API using flask library to perform the model prediction, webpage rendering and database update. Upon startup, the API will create the CNN extractor and load in trained LSTM, then wait for HTTP request from the client. The following route were defined in the API:

i. **@app.route('/upload', methods=['POST'])**

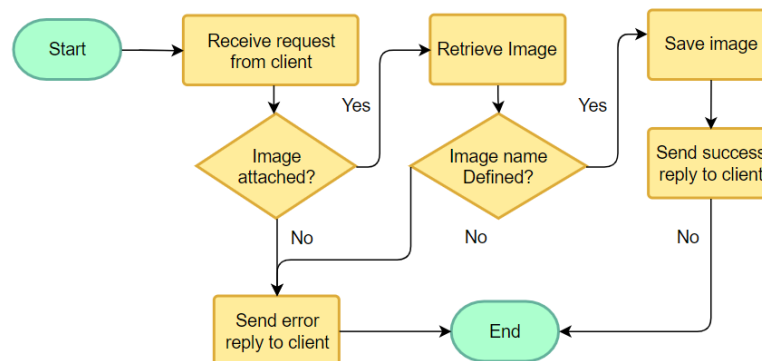


Figure 4.2.4.1: Flow chart of flask API /upload route

This route is triggered when the client sends an image to the server at route /upload using POST method. The API first check if image is included in the packet. Then, the image data is retrieved and further checking if the name of the image is specified. The file name is named based on the sequence of the image being captured (i.e. first image name 0.jpg). Having both items received, the API will store the image in specified /upload folder and reply to the client with a success message. Else, error message is replied to the client.

ii. `@app.route('/batch_received', methods=['GET'])`

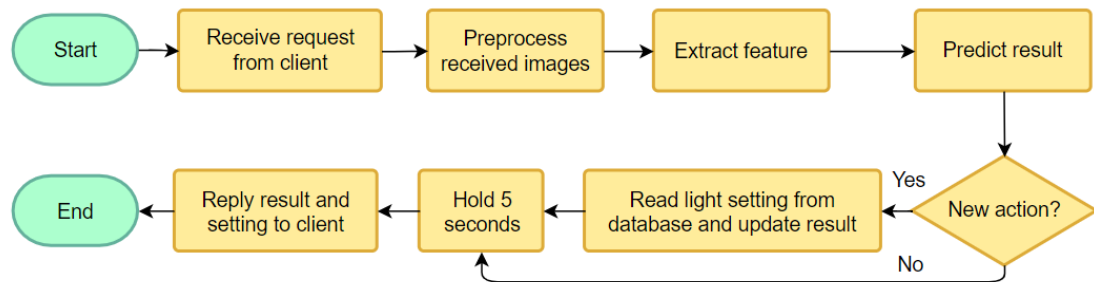


Figure 4.2.4.2: Flow chart of flask API /batch\_received route

This route is triggered when client request for prediction result after sending 40 images. The API will read in the uploaded images and preprocess them into single numpy array. The feature is extracted and then predicted by the trained LSTM model. If the result is a new action, the API will read the light setting from the database. Else, the previous setting together with prediction result is sent to the client. 5 seconds sleep time is added before response to avoid flooding the client.

iii. `@app.route('/configure', methods=['GET', 'POST'])`

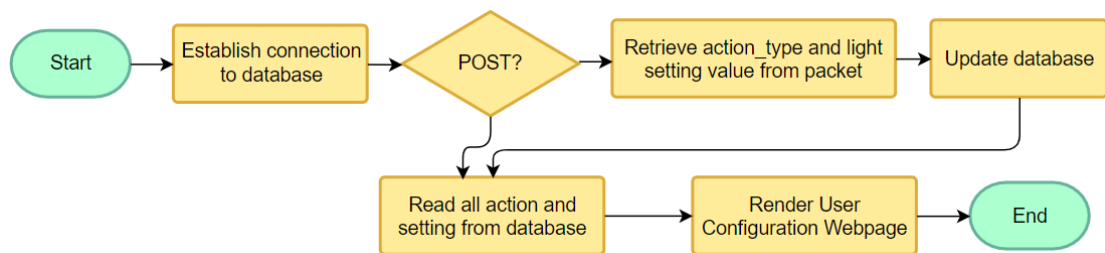


Figure 4.2.4.3: Flow chart of flask API /configure route

This route is triggered when user access the User Configuration webpage. The route look for 2 http method: GET and POST. It first establish the connection to the MySQL database. On GET method (i.e. user just trying to visit the webpage), it will read all the action type and light setting stored in the database and render the configure.html file (the User Configuration Webpage code) with data. If POST is used, this means that user is trying to update the setting of a particular action. The API will retrieve user's input for action type and the setting value and update the database accordingly, then render the website with the latest data.



### **4.2.5 User Configuration Webpage**

The html page is held in the server container as the front end for user to input their own preferences. The webpage can be accessed by the IP of the server with the route /configure. It displays all the supported action and its corresponding light setting recorded in the database. The values are dynamically read from the database. Based on the current setting, user can choose a desired action type and update the setting accordingly by summiting the html form. The webpage is written using HTML and CSS.

### 4.2.6 Database

The database is a separated docker container used to store the supported activities and light configuration parameters. The database container is created in MySQL with a default schema named “yeelight”. On first access, a table with column “action\_type” and “temperature” is created to store the data. Referring to the information from section 2.3, the light configuration for each activity class is defined as below:

Table 4.2.6.1: Colour temperature setting for each activity class

Class Name	Light Colour Temperature
Sleeping	1000K with minimum brightness
Dining	3000K
Watching TV	3000K
Yoga	3500K
Reading Book	4500K
Running on a Treadmill	5000K
Drawing	5000K
Mopping Floor	6500K

Table 4.2.6.2: Summary of table light\_value in MySQL database

Light_value		
PK	ID	INT Auto-increment
	action_type	Varchar(255)

The database container is built with docker volume to hold the data permanently. This avoid the table reset to empty whenever it restarts which is the default behaviour of docker without volume. A docker-compose file is written to start the server container and database container within the same Virtual Docker network. Then, the database is ready to be accessed by the server container.

## CHAPTER 5

### System Implementation

#### 5.1 Hardware Setup

The hardware required by the project included desktop, microcontroller, CSI camera and configurable IoT light. The details are listed in the table below.

Table 5.1.1: Details of hardware requirement

No.	Description	Specifications
1.	Desktop	
	Operating System	Windows 10
	Processor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
	Graphic	NVIDIA GeForce GTX 1080 8 GB GDDR5X
	Memory	16GB DDR4 RAM
	Storage	1TB SATA HDD
	Usage	Data pre-processing and model training
2.	Microcontroller	
	Model	Jetson Nano B01
	Operating System	Ubuntu 18.04
	Processor	Quad-core ARM A57 @ 1.43 GHz
	Graphic	128-core Maxwell
	Memory	4 GB 64-bit LPDDR4 25.6 GB/s
	Storage	64GB MicroSD
	Usage	Run the client program
3.	CSI Camera	
	Model	CSI IMX219
	Pixels	8,000,000
	Resolution	3280 x 2464
	Usage	Capture human action and pass to jetson nano for inference
4.	IoT Light	
	Model	Yeelight LED Strips 1S
	Supported Colour	Razer Chorma RGB, 16 million colours
	Control Framework	Yeelight 3 <sup>rd</sup> -party Control Protocol
	Usage	Adjust indoor lighting based on detected activity

## 5.2 Software Setup

The software and libraries required by the project are listed below. The libraries are installed on top of the anaconda virtual environment:

Table 5.2.1: Software and libraries setup

<b>Software</b>		
Name	Usage	Installation Guide
Anaconda	To manage virtual environment and run Jupyter Notebook	<a href="#">Anaconda documentation</a>
Jupyter Notebook	Python IDE for HAR Model training	Included in Anaconda
Visual Studio Code	Python IDE for server docker python scripting	<a href="#">Visual Studio Code documentation</a>
Docker Engine / Docker Desktop	To export the server Flask API as an Operating System Virtualized Container	<a href="#">Docker documentation</a>
MySQL Workbench	Connect and execute sql script on MySQL database	<a href="#">MySQL Workbench documentation</a>
<b>Library</b>		
Name	Version	Installation Guide
Tensorflow with Keras	2.13	<a href="#">Tensorflow installation guide</a>
CUDA	11.8.0	Included in tensorflow guide
cuDNN	8.6.0.163	Included in tensorflow guide
OpenCV	4.6.0	<a href="#">pip install opencv-python</a>
Matplotlib	3.7.1	<a href="#">pip install matplotlib</a>
Scikit-learn	1.2.0	<a href="#">pip install scikit-learn</a>
Numpy	1.24.2	<a href="#">pip install numpy</a>
Yeelight	0.7.10	<a href="#">pip install yeelight</a>
Flask	2.3.3	<a href="#">pip install flask</a>
Flask-MySQL	1.5.2	<a href="#">pip install flask-mysql</a>
Cryptography	41.0.3	<a href="#">pip install cryptography</a>
Pillow	10.0.0	<a href="#">pip install pillow</a>

### 5.3 HAR Model Training

#### 5.3.1 Dataset Collection

A few published human activity video datasets were explored thoroughly to identify the activity classes that suit the project theme. Two suitable datasets were nominated, including STAIR Actions and Kinetic 700\_2020. However, STAIR Actions dataset requires a crawler program to crawl the video from YouTube, which the program having unfixed bug. Hence, Kinetic 700\_2020 dataset was chosen for this project.

7 classes were elected to be classified by the HAR model: dining, drawing, mopping floor, reading book, running on treadmill, sleeping, watching tv. The dataset can be downloaded from the AWS server directly. The download link to each action class is listed in the train\_path.txt posted on this [Github](#) site [31].

#### 5.3.2 Data Pre-processing

##### **Extract Frames from Video:**

The downloaded train dataset was stored in a main folder called “Train”. All videos of each activity class were stored in the subfolder under “Train”, named by the name of the class. Another folder called “Train2” is created to store the extracted frames.

A python script was written to perform frame extraction on the videos using OpenCV. The “Train2” folder will be the root directory for processed data. Then, it reads in the video and calculates the total frames available (e.g., 10 seconds video at 30 fps results in 300 frames). If the available frame is less than 80, the video is skipped and will not be used in training. Then, cv2 extracts 80 frames from the video. A subfolder is created for the processing class, and the processed frames are stored in the 3<sup>rd</sup> level subfolder in that class folder (“Train2/Class\_Name\_Folder/Video\_Name\_Folder”). Each frame is named from 0 to 79 according to sequence in the video. 4 csv files are created as below:

- **Extract\_frame\_report:** Keep track of the processing category, number of videos loaded, number of videos skipped and number of video loaded with error (in case).

	A	B	C	D
1	Category	load_count	fail_count	load_error
2	dining	750	38	0
3	drawing	738	44	0
4	mopping_floor	844	44	0

Figure 5.3.2.1: Snippet of Extract\_frame\_report.csv

- **Data\_frames\_report:** record the total frame count of each video that successfully loaded for extraction.

	A	B	C
1	dining	Data\dining\--UW3CcUqaU_000020_000030.mp4	250
2	dining	Data\dining\--1z6wTJmBIU_000002_000012.mp4	300
3	dining	Data\dining\--cfCZm3Qp6E_000030_000040.mp4	300
4	dining	Data\dining\--chadEau4KM_000021_000031.mp4	150

Figure 5.3.2.2: Snippet of data\_frames\_report.csv

- **Frames:** Record the path to all the extracted frames

	A	B	C	D	E	F	G
82	dining	-1z6wTJmBIU_000002_000012	Data2\dining\--1z6wTJmBIU_000002_000012\1.jpg				
83	dining	-1z6wTJmBIU_000002_000012	Data2\dining\--1z6wTJmBIU_000002_000012\2.jpg				
84	dining	-1z6wTJmBIU_000002_000012	Data2\dining\--1z6wTJmBIU_000002_000012\3.jpg				

Figure 5.3.2.3: Snippet of frames.csv

- **Skip\_vid\_report:** Keep track of the video being skipped and its frame count.

	A	B	C
1	dining	Data\dining\--Pc62CZZMII_000015_000025.mp4	69
2	dining	Data\dining\0f66UEeLLeM_000310_000320.mp4	47
3	dining	Data\dining\0tTGBCCl8j0_000060_000070.mp4	75

Figure 5.3.2.4: Snippet of Skip\_vid\_report.csv

The same processes were performed on the validation dataset.

```

def preprocess_data(CATEGORIES):

    with open(os.path.join(DATA2_DIR, 'Extract_frame_report.csv'), mode='a', newline='') as file:
        writer3 = csv.writer(file)
        writer3.writerow(['Category', 'load_count', 'fail_count', 'load_error'])

    for i, category in enumerate(CATEGORIES):

        load_count = 0
        fail_count = 0
        load_error = 0
        category_dir = os.path.join(DATA_DIR, category)

        print('Loading ', category, ' : \n')
        start_time = datetime.datetime.now()
        print('Start at: ', start_time.strftime("%c"))

        for video_file in os.listdir(category_dir):
            video_path = os.path.join(category_dir, video_file)
            frames, loaded = extract_frames(video_path, category)
            if not loaded:
                fail_count += 1
                print('Fail to load ', video_file)
            else:
                if len(frames) == SEQUENCE_LENGTH:
                    print('Succeed to load ', video_file)
                    load_count += 1
                    if load_count % 50 == 0:
                        gc.collect()
                        time.sleep(60)
                else:
                    load_error += 1
                    print(video_file, ' Loaded with error')
        writer3.writerow([category, load_count, fail_count, load_error])
        end_time = datetime.datetime.now()
        print('End at: ', end_time.strftime("%c"))

        time_difference = (end_time - start_time).total_seconds() / 60
        print(f"Time taken: {time_difference:.2f} minutes")
        gc.collect()
        time.sleep(100)

    return 'Finish loading!'

```

Figure 5.3.2.5: Code of preprocess\_data function

The preprocess\_data function will take in the CATEGORIES parameter, which is a list of the action class. It loops through the list to generate the path to videos in each action class folder and pass the path to extract\_frames function for extraction.

```

def extract_frames(video_file, category):
    cap = cv2.VideoCapture(video_file)
    frames = []
    vid_name = os.path.splitext(os.path.basename(video_file))[0]
    frame_dir = os.path.join(DATA2_DIR, category, os.path.splitext(os.path.basename(video_file))[0])
    loaded = True

    with open(os.path.join(DATA2_DIR, 'Skip_vid_report.csv'), mode='a', newline='') as file1:
        with open(os.path.join(DATA2_DIR, 'Dataset_frames_report.csv'), mode='a', newline='') as file2:
            writer1 = csv.writer(file1)
            writer2 = csv.writer(file2)
            video_frames_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

            if video_frames_count < SEQUENCE_LENGTH:
                writer1.writerow([category, video_file, str(video_frames_count)])
                loaded = False
                return 0, loaded

            else:
                os.makedirs(frame_dir, exist_ok=True)
                writer2.writerow([category, video_file, str(video_frames_count)])

                skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

                with open(os.path.join(DATA2_DIR, 'frames.csv'), mode='a', newline='') as file3:
                    writer3 = csv.writer(file3)

                    for frame_counter in range(SEQUENCE_LENGTH):

                        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

                        success, frame = cap.read()

                        if not success:
                            print("Fail to read")
                            break

                        cv2.imwrite(os.path.join(frame_dir, f'{frame_counter}.jpg'), frame)
                        writer3.writerow([category, vid_name, os.path.join(frame_dir, f'{frame_counter}.jpg')])

                        frames.append(frame)

                    cap.release()

    return frames, loaded

```

Figure 5.3.2.6: Code of extract\_frames function

The extract\_frames function will read in the video and extract 80 frames using OpenCV. The frames are then stored to the corresponding folder under “Training2”.

### Data Pre-processing:

After finish extraction, another python function is written to pre-process the data. From each video file, 40 frames were first read in, resized to (112 x 112 x 3), format as a NumPy array and divided by 255 to normalize the pixels value between 0 and 1. The output is of shape (40, 112, 112, 3) which is the shape of an input sample. The output sample is then appended to a python list, and the class label of the sample is appended to another python list. Same processes were carried out on the remaining 40 frames, and all the remaining videos. The resulted Dataset list and Label list is then converted to NumPy array and go through feature extraction before splitting into Train and Test



## CHAPTER 5

set at 80:20 with a custom function. The processed array of each class is stored in a single folder call “Dataset\_processed\_full”.

Table 5.3.2.1: Summary of Training and Validation dataset

Category	# of Video Loaded	Total Sample (40f * 2)	Train Size	Validation Size	# of Video Skipped
Dining	750	1500	1200	300	38
Drawing	738	1476	1181	295	44
Mopping floor	844	1688	1350	338	44
Reading book	936	1872	1498	374	46
Running on treadmill	815	1630	1304	326	42
Sleeping	726	1452	1162	290	43
Watching tv	671	1342	1074	268	40
Yoga	723	1446	1157	289	28
<b>Total</b>	<b>5480</b>	<b>12406</b>	<b>9926</b>	<b>2480</b>	<b>297</b>

Table 5.3.2.2: Summary of Evaluation dataset

Category	# of Video Loaded	Total Sample (40f * 2)	# of Video Skipped
Dining	49	98	0
Drawing	46	92	0
Mopping floor	50	100	0
Reading book	49	98	0
Running on treadmill	48	96	0
Sleeping	48	96	2
Watching tv	49	98	1
Yoga	50	100	0
<b>Total</b>	<b>389</b>	<b>778</b>	<b>3</b>

```

def preprocess_data(category, GPU):
    tf.debugging.set_log_device_placement(True)
    tf.config.set_soft_device_placement(False)
    with tf.device(GPU):
        X = []
        y = []
        exampleCount = 1
        category_dir = os.path.join(DATA2_DIR, category)
        print("Loading ", category)

        for video in os.listdir(category_dir):
            video_dir = os.path.join(category_dir, video)

            if len(os.listdir(video_dir)) > 0:
                print("Loading video ", str(exampleCount), ': ', str(video))
                frames = []

                for frame_file in range(40):
                    img_path = os.path.join(video_dir, str(str(frame_file) + '.jpg'))
                    img = image.load_img(img_path, target_size=(112, 112))
                    x = image.img_to_array(img)
                    x = x/255
                    x = np.expand_dims(x, axis=0)
                    frames.append(x)
                X.append(np.concatenate(frames))
                y.append(category)
                exampleCount += 1
                frames.clear()
                gc.collect()

                for frame_file in range(40,80):
                    img_path = os.path.join(video_dir, str(str(frame_file) + '.jpg'))
                    img = image.load_img(img_path, target_size=(112, 112))
                    x = image.img_to_array(img)
                    x = x/255
                    x = np.expand_dims(x, axis=0)
                    frames.append(x)
                X.append(np.concatenate(frames))
                y.append(category)
                exampleCount += 1
                frames.clear()
                gc.collect()

            else:
                print('Skip vid: ', video)
        print("Finished loading ", category)
        print("Formating input X")
        X = np.array(X)
        print("Formating label y")
        y = np.array(y)

        print('\nSummary: \n')
        print(str(category), ': ', str(exampleCount))

    return X, y

```

Figure 5.3.2.7: Code of preprocess\_data function

### 5.3.3 Feature Extraction

The function `extract_feature` was created for feature extraction. As the project was trying out on different CNN, all CNN to be used is created before calling the function. Then, the function call pass in the numpy array of the class to be processed, the CNN and the number of samples to be processed at once. The samples are processed using `model.predict_on_batch()`. Then, the processed data and label array is splitted into 80:20 for training and testing set. The label is further processed into one-hot array using the python function `to_categorical`. The final numpy for data and label are saved for training.

The CNN to be used in FYP2 are MobileNetV2 and DenseNet201. Feature extractors are created as below. The “trainable” parameter of each CNN layers needs to be configured to “False” to avoid update in model weight:

Layer (type)	Output Shape	Param #	Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 40, 112, 112, 3)]	0	input_2 (InputLayer)	[(None, 40, 112, 112, 3)]	0
time_distributed (TimeDistributed)	(None, 40, 4, 4, 1280)	2257984	time_distributed (TimeDistributed)	(None, 40, 3, 3, 1920)	18321984
time_distributed_1 (TimeDistributed)	(None, 40, 1280)	0	time_distributed_1 (TimeDistributed)	(None, 40, 1920)	0
Total params: 2,257,984 Trainable params: 0 Non-trainable params: 2,257,984			Total params: 18,321,984 Trainable params: 0 Non-trainable params: 18,321,984		

Figure 5.3.3.1: Summary of MobileNetV2 (left) and DensNet201 (right)

```
# Load DenseNet201 model
dens201 = DenseNet201(weights='imagenet', include_top=False, input_shape=(112, 112, 3))

# Freeze ResNet50 layers
for layer in dens201.layers:
    layer.trainable = False

# Extract features using ResNet50
inputs = Input(shape=INPUT_SHAPE)
l1 = TimeDistributed(dens201)(inputs)
l2 = TimeDistributed(GlobalAveragePooling2D())(l1)
#outputs = TimeDistributed(Flatten())(l2)

# Create model
dens201_gavg = Model(inputs, l2)
dens201_gavg.summary()
```

Figure 5.3.3.2: Example code to create the CNN model

Table 5.3.3.1: Summary of shape of Train and Validation dataset

Feature Extractor	X_train	y_train	X_test	y_test
MobileNetV2	(9926, 40, 1280)	(9926, 8)	(2480, 40, 1280)	(2480,8)
DenseNet201	(9926, 40, 1920)	(9926, 8)	(2480, 40, 1920)	(2480,8)

```

def extract_feature(X, model, step):
    tf.debugging.set_log_device_placement(False)
    tf.config.set_soft_device_placement(True)
    with tf.device('/device:GPU:1'):

        start = 0
        end = 0
        while end < X.shape[0]:
            end = start + step
            if start != 0:
                print('Processing video ', start, ' - ', end)
                X_sliced = X[start:end,:,:,:]
                X_train = model.predict_on_batch(X_sliced)
                X_train_concat = np.concatenate((X_train_concat,X_train))
                start = end
                gc.collect()
            else:
                print('Start processing...')
                print('Processing video ', start, ' - ', end)
                X_sliced = X[start:end,:,:,:]
                X_train = model.predict_on_batch(X_sliced)
                X_train_concat = X_train
                start = end
                gc.collect()

            print('Processed ', end, ' video')

        return X_train_concat

```

Figure 5.3.3.3: Code of extract\_feature function.

```

def train_test_loader(classname, X_train, X_test, y_train, y_test, model_name, main_dir):

    if len(X_train) != 0 and len(y_train) != 0:
        x_processed = np.load(main_dir + model_name + '/' + model_name + '_' + classname + '_processed.npy')
        split = round(len(x_processed)*0.8)
        X_train = np.concatenate((X_train, x_processed[0:split, :, :]))
        X_test = np.concatenate((X_test, x_processed[split:len(x_processed), :, :]))

        y_processed = np.load(main_dir + model_name + '/' + model_name + '_' + classname + '_label.npy')
        y_train = np.concatenate((y_train, y_processed[0:split]))
        y_test = np.concatenate((y_test, y_processed[split:len(y_processed)]))

        return X_train, X_test, y_train, y_test, x_processed.shape, y_processed.shape, split
    else:
        x_processed = np.load(main_dir + model_name + '/' + model_name + '_' + classname + '_processed.npy')
        split = round(len(x_processed)*0.8)
        X_train = x_processed[0:split, :, :]
        X_test = x_processed[split:len(x_processed), :, :]

        y_processed = np.load(main_dir + model_name + '/' + model_name + '_' + classname + '_label.npy')
        y_train = y_processed[0:split]
        y_test = y_processed[split:len(y_processed)]

    return X_train, X_test, y_train, y_test, x_processed.shape, y_processed.shape, split

```

Figure 5.3.3.4: Code of train\_test\_loader function to split the dataset

### 5.3.4 Model Training

The LSTM described in section 3.2 was compiled to train on the extracted features. The model used Adam optimizer with default initial learning rate at 0.001 and ReduceLRonPlateau scheduler at factor of 0.2 and patient of 5 to reduce the learning rate by 80% if the validation loss does not improve after every 5 epochs. Dropout of 0.5 is applied on the LSTM layer. This randomly initialize half of the input to 0 before passing to the next iteration which helps to regularize the data and tackle possible overfitting. Model checkpoint monitoring on the validation loss is used to save only the trained weight of the best performing model with minimum validation loss over the full 30 epochs. Detail evaluation on the result will be discussed in the following section.

To find the optimum setup for model training, 4 different setting on batch size and LSTM unit size is applied to the training:

Table 5.3.4.1: Summary of Training Setting

Parameter	Setting 1	Setting 2	Setting 3	Setting 4
LSTM unit	128	128	128	10
Epoch	30	30	30	30
Batch size	4	16	4	4
Optimizer	Adam – 0.001 lr	Adam – 0.001 lr	Adam – 0.001 lr (decay 0.001)	Adam – 0.001 lr
Scheduler	ReduceLR OnPlateau (factor 0.2, patient 5)	ReduceLR OnPlateau (factor 0.2, patient 5)	ReduceLR OnPlateau (factor 0.2, patient 5)	ReduceLR OnPlateau (factor 0.2, patient 5)

```

INPUT_SHAPE_lstm = (X_train.shape[1], X_train.shape[2])
INPUT_SHAPE_lstm

tf.keras.backend.clear_session()

inputs = Input(shape=INPUT_SHAPE_lstm)

# LSTM layer
lstm = LSTM(128, return_sequences=False, dropout=0.5)(inputs)

# Output layer
outputs = Dense(NUM_CLASSES, activation='softmax')(lstm)

# Create model
model = Model(inputs, outputs)
model.summary()

mcp = ModelCheckpoint(f'lstm_{Extractor}_{Layer}_{BatchSize}_{Optimizer}_{Scheduler}', save_best_only=True, monitor='val_loss', mode='min', verbose=1)
reduce_lr = ReduceLRonPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

```

Figure 5.3.4.1: Example code to create the LSTM model

```

history = model.fit(X_train, y_train, batch_size=BATCH_SIZE, epochs=EPOCHS, validation_data=(X_test, y_test), callbacks=[mcp, reduce_lr])

plt.plot(history.history["accuracy"])
plt.plot(history.history["val_accuracy"])
plt.plot(history.history["loss"])
plt.plot(history.history["val_loss"])
plt.title(f"lstm_{Extractor}_{Layer}_{Optimizer}_{Scheduler} Accuracy-Loss Chart")
plt.ylabel("Accuracy / Loss")
plt.xlabel("Epoch")
plt.xticks(np.arange(0, 30, 1))
plt.yticks(np.arange(0, 2.8, 0.2))
plt.legend(["Accuracy", "Validation Accuracy", "loss", "Validation Loss"])
plt.show()

```

Figure 5.3.4.2: Example code of model training and training result graph plotting

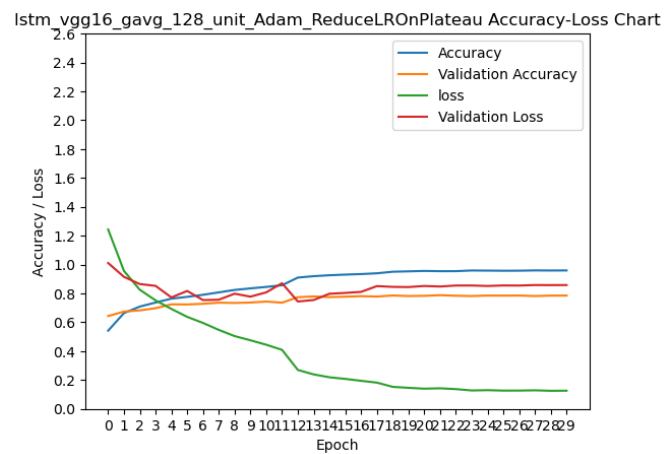


Figure 5.3.4.3: Example of training result graph

After training, the best model was loaded from the checkpoint and to export the architecture and weights for production use.

```

model_temp = tf.keras.models.load_model('/home/ycaun/Desktop/jeheng/lstm_dens201_gavg8class_128_unit_4_Adam_with_decay_ReduceLRonPlateau')

setup = 'adam_16'
name = 'dense201_gavg'
classname = '7class'

model_json = model_temp.to_json()
with open(setup + "_" + name + "_" + classname + "_" + "model.json", "w") as json_file:
    json_file.write(model_json)

model_temp.save_weights(setup + "_" + name + "_" + classname + "_" + "model.h5")
print("Saved model to disk")

```

Figure 5.3.4.4: Example code of saving the best model.

## 5.4 Server Docker

### 5.4.1 Flask API

The flask API docker is created with the file structure below:

- /Docker\_App
  - /model\_param
    - Model.json
    - Model.h5
  - /static
    - Styles.css
  - /templates
    - Config.html
  - /uploads
    - 0.jpg
    - 1.jpg
    - .....
  - Docker\_compose.yml
  - Dockerfile
  - Requirements.txt
  - Server.py

Table 5.4.1.1: Summary of the docker image structure

Item	Description
/model_param	The folder holding the trained LSTM
/static	The folder holding CSS for user configuration webpage
/templates	The folder holding the user configuration webpage html code
/uploads	The folder to store the images from client
Docker_compose.yml	The script to run composite docker container (server container and database container)
Dockerfile	The file to build the server docker
Requirements.txt	TXT file which lists the python libraries required in the docker image
Server.py	The main code of Flask API

The Flask API is used for feature extraction and model prediction. The following routes are defined to carry out the functionality as described in section 4.2.4:

- i. **@app.route('/configure', methods=['GET', 'POST']):**  
To render the User Configuration Webpage

```
@app.route('/configure', methods=['GET', 'POST'])
def configure():
    conn = mysql.connect() # Get a connection to the MySQL database
    cursor = conn.cursor()

    if request.method == 'POST':
        action_type = request.form.get('action-type')
        temperature = request.form.get('temperature')

        # Update the database with the new numeric value
        query = "UPDATE light_value SET value = %s WHERE action_type = %s"
        cursor.execute(query, (temperature, action_type))
        conn.commit()

    # Retrieve data from the database for rendering in the template
    query = "SELECT action_type, value FROM light_value"
    cursor.execute(query)
    action_values = {row[0]: row[1] for row in cursor.fetchall()}

    conn.close() # Close the connection

    return render_template('config.html', action_values=action_values)
```

Figure 5.4.1.1: Code of /configure route

- ii. **@app.route('/upload', methods=['POST']):**  
To save image uploaded from server.

```
@app.route('/upload', methods=['POST'])
def upload():
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400

    file = request.files['file']

    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400

    if file:
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return jsonify({'message': 'File uploaded successfully'}), 200
```

Figure 5.4.1.2: Code of /upload route



iii. **@app.route('/batch\_received', methods=['GET']):**

To perform prediction, retrieve light setting from database and return the result to client.

```
@app.route('/batch_received', methods=['GET'])
def batch_received():
    global expected
    global frames
    global predictor
    global extractor
    global prev_result
    class_list = ['dining', 'drawing', 'mopping_floor', 'reading_book', 'running_on_treadmill', 'sleeping', 'watching_tv']
    frame_file = './uploads/'
    setting = 4004

    print('preprocessing')
    frames = preprocess_data(frame_file)
    print('extracting')
    X = extraction(frames, extractor)
    print('predicting')
    result = prediction_temp(X, predictor, class_list)
    if result != prev_result:
        print("new action detected")
        setting = get_light_config(class_list[result])
        prev_result = result

    time.sleep(5)
    return jsonify({'message': 'Batch of images received ' + str(result) + ' ' + str(setting), 'result': str(result), 'setting':
str(setting)}), 200
```

Figure 5.4.1.3: Code of /batch\_received route

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'uploads' # Specify the upload folder
app.config['MYSQL_DATABASE_HOST'] = 'db' # The hostname of your MySQL container
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'root'
app.config['MYSQL_DATABASE_DB'] = 'yeelight'
mysql = MySQL()
mysql.init_app(app)
```

Figure 5.4.1.4: Flask API configuration

The Flask API was configured with the variables above to establish the connection with MySQL database container. When the docker start, the API will be bound to port 5000 on local host. Access to the API was done by browsing the server host IP address with port 5000.

```
# Use an official Python runtime as the base image
FROM python:3.8-slim-buster

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
#RUN pip install --no-cache-dir -r requirements.txt
RUN pip install -r requirements.txt

# Install MySQL client library
RUN apt-get update && apt-get install -y default-mysql-client

# Set environment variable for Flask
ENV FLASK_APP=server.py

# Expose the port that the app runs on
EXPOSE 5000

# Run the command to start the app
#CMD ["flask", "run", "--host=0.0.0.0"]
CMD ["python", "-u", "server.py"]
```

Figure 5.4.1.5: Code of Dockerfile

Finally, the server image is exported by running docker build command. The Dockerfile above specified the image to build on top of a python image and installed with necessary libraries. The last CMD line specify the container to run the Flask API (code saved as server.py) when it is turned on for service.

### 5.4.2 User Configuration Webpage

The user configuration webpage is written in html and held inside the server docker container. Access to the page can be done by visiting server host IP at port 5000 with route /configure (e.g., 192.168.100.177:5000/configure). The webpage is rendered dynamically on /configure route based on the data read from MySQL database and upon user input.

```

<table>
  <tr>
    <th>Action Type</th>
    <th>Numeric Value</th>
  </tr>
  {% for action, value in action_values.items() %}
  <tr>
    <td>{{ action }}</td>
    <td>{{ value }}</td>
  </tr>
  {% endfor %}
</table>
<form method="POST" action="/configure">
  <div class="form-row">
    <div class="form-column">
      <label for="action-type">Action Type:</label>
      <select id="action-type" name="action-type">
        {% for action, value in action_values.items() %}
        <option value="{{ action }}">{{ action }}</option>
        {% endfor %}
      </select>
    </div>
    <div class="form-column">
      <label for="temperature">Temperature:</label>
      <input type="text" id="temperature" name="temperature">
    </div>
  </div>
  <div class="form-row" id="submit">
    <input type="submit" value="Update" >
  </div>
</form>

```

Action Type	Numeric Value
dining	3000
drawing	5000
sleeping	2700
reading_book	4500
watching_tv	3000
running_on_treadmill	5000
mopping_floor	6500

Action Type:  Temperature:

Figure 5.4.2.1: Dynamic rendering of the User Configuration Webpage

## 5.5 MySQL Database Container

The database container was created by specifying the parameters in the `docker-compose.yml` file.

```
version: '3'
services:
  app:
    build: .
    ports:
      - "5000:5000"
    image: server:4.3
    depends_on:
      - db

  db:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: yeelight
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql # Use a named volume for data storage

volumes:
  mysql_data: # Define the named volume
    driver: local
```

Figure 5.5.1: Docker-compose.yml file

In the `db` section, the “`image: mysql:latest`” declared the docker engine to pull the latest version MySQL docker image from docker hub. The database is created using the root user password “`root`” and a default scheme called “`yeelight`”. The database is bound to the port 3306 at local host. Lastly, the named volume is specified to store the database permanently.

The `app` section is declaring the Flask API container which named as `server:4.3`. The service is running depend on the MySQL database container, meaning that the server is only ready to run after the MySQL database is fully up. This is necessary for the connection between server container and database container. Running the command `docker-compose up` will turn on the container service accordingly.

## CHAPTER 5

```
use yeelight;

CREATE TABLE light_value (
  id INT AUTO_INCREMENT PRIMARY KEY,
  action_type VARCHAR(255),
  value INT
);

INSERT INTO light_value (action_type, value) VALUES ('dining', 3000);
INSERT INTO light_value (action_type, value) VALUES ('drawing', 5000);
INSERT INTO light_value (action_type, value) VALUES ('sleeping', 2700);
INSERT INTO light_value (action_type, value) VALUES ('reading_book', 4500);
INSERT INTO light_value (action_type, value) VALUES ('watching_tv', 3000);
INSERT INTO light_value (action_type, value) VALUES ('running_on_treadmill', 5000);
INSERT INTO light_value (action_type, value) VALUES ('mopping_floor', 6500);

select * from light_value;
```

Figure 5.5.2: Table creation script

On first startup, the above SQL script is used to create the table with default light setting.

## 5.6 Configuration Program on Client

The client configuration program will run on Jetson Nano to stream images using camera and configure light on the Yeelight device. To run the code, Jetson Nano and the IoT light need to be properly configured.

### 5.6.1 Configuration on Jetson Nano and CSI Camera

To use the Jetson Nano, a microSD with 64GB storage size was flashed with the official operating system Jetpack 4.6.1. SD Card Formatter and Etcher were required to format the microSD card and flash the operating system. The official OS was suggested as it was pre-built with CUDA and cuDNN for GPU processing, and OpenCV which could speed up the configuration. The device was restarted with the CSI Camera plugged in, and it was ready to run the program.

### 5.6.2 Configuration on IoT Light

Yeelight 1S is used as the configurable IoT light in this project. To configure the light for wireless control over LAN, it needs to be first connected with the mobile application Yeelight and Mi Home. With the connection established, find the light under LAN Control option in the Yeelight app and toggle the allow option to on. This allow the light to be discovered within the connected LAN. Communication with the light from Jetson Nano is realised using the Yeelight python library and calling the corresponding function such as change colour, brightness and on off.

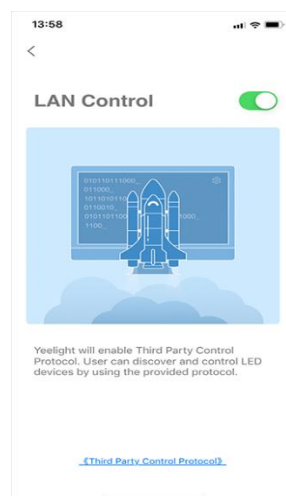


Figure 5.6.2.1: Yeelight App interface to turn on the LAN control

### 5.6.3 Client Configuration Program

The python script was developed with 2 objectives: capturing user action for classification and issuing command for light control.

The program first started by discovering available Yeelight device within the LAN. Discover\_bulbs function from Yeelight python library is used for the discovery which return the IP address of the device. The IP was used to create a bulb object, which later used for light configuration.

```
def create_yeelight():
    bulb_info = discover_bulbs()
    if len(bulb_info) > 0:
        print('Bulb detected')
        bulb = Bulb(bulb_info[0]['ip'])

        return True, bulb
    else:
        print('No bulb discovered')
        return False, None
```

Figure 5.6.3.1: Function to create bulb object

OpenCV with Nvidia Gstreamer is used to stream the video input captured by the CSI camera. The image streamed by OpenCV is in numpy format. It was pre-processed by dividing the with 225 and resize to 112x112. Then, the image numpy was encoded in JPEG format and converted to byte in order to be posted to the server as JSON object using requests library.

```
video_capture = cv.VideoCapture(gstreamer_pipeline(flip_method=0), cv.CAP_GSTREAMER)
window_title = "CSI Camera"

if video_capture.isOpened():

    try:
        window_handle = cv.namedWindow(window_title, cv.WINDOW_AUTOSIZE)
        i = 1
        while True:
            ret_val, img = video_capture.read()
            # Check to see if the user closed the window
            # Under GTK+ (Jetson Default), WND_PROP_VISIBLE does not work correctly. Under Qt it does
            # GTK - Substitute WND_PROP_AUTOSIZE to detect if window has been closed by user
            if cv.getWindowProperty(window_title, cv.WND_PROP_AUTOSIZE) >= 0:
                cv.imshow(window_title, img)
                if not ret_val:
                    print('fail to stream')
                    break

                img = cv.resize(img,(112,112))
                _, img_encoded = cv.imencode('.jpg', img)
                image_data = img_encoded.tobytes()

                files = {'file': (str(i) + '.jpg', image_data, 'image/jpeg')}
                response = requests.post(server_url, files=files)

                if response.status_code == 200:
                    print("Image " + str(i) + " uploaded successfully.")
                    i += 1
                else:
                    print("Error uploading image:" + str(i), response.json())
```

Figure 5.6.3.2: Code snipped for streaming and uploading images

After 40 successful uploads, the program will send request for the prediction result. The program pauses and wait until the response received. After getting the response from server, program will retrieve the result and light setting. If new action detected, light configuration command is issued to update Yeelight with the new setting value.

```

if i == batch_size + 1:
    response = requests.get(server_url.replace("upload", "batch_received"))

    if response.status_code == 200:
        data = response.json()
        result = data['result']
        setting = data['setting']
        if result != prev_result:
            print('New action ' + class_list[int(result)] + ' detected')
            if discovered:
                light_adjustment(yeelight, int(result), int(setting))
            prev_result = result
        print("Batch of images received. \n" + 'Result: ' + class_list[int(result)] + '\nSetting: ' + setting)
        i = 1
    else:
        print("Error confirming batch received:", response.json())
        i = 1
    time.sleep(1) # Add a delay to avoid overwhelming the server

def light_adjustment(bulb, result, setting):
    if bulb is None:
        print('No IoT light detected for configuration')
        return 0

    print('Activity ', str(result), ' detected. Start configure light')
    if result != 5:
        bulb.set_brightness(100)
        bulb.set_color_temp(setting)
    elif result == 5:
        bulb.set_brightness(1)
        bulb.set_color_temp(setting)
    else:
        print('Activity to be supported in the future')

```

Figure 5.6.3.3: Code snippet of requesting prediction result and light configuration



## 5.7 System Operation

This section will go through the operation on the server and client.

The system should begin from the server side. Start the server container and database container by running “docker-compose up” within the working directory where the docker-compose file is stored.

```

E:\Uni Notes\FYP\FYP2\Docker_yeelight_v2>docker-compose up
[+] Running 2/0
  Container docker_yeelight_v2-db-1   Created           0.0s
  Container docker_yeelight_v2-app-1  Created           0.0s
Attaching to docker_yeelight_v2-app-1, docker_yeelight_v2-db-1
docker_yeelight_v2-db-1 | 2023-09-12 21:15:39+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.e18 started
docker_yeelight_v2-db-1 | 2023-09-12 21:15:39+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
docker_yeelight_v2-db-1 | 2023-09-12 21:15:39+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.1.0-1.e18 started
docker_yeelight_v2-db-1 | ' /var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
docker_yeelight_v2-db-1 | 2023-09-12T21:15:39.851878Z 0 [System] [MY-015015] [Server] MySQL Server - start.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.054689Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Please use SET GLOBAL host_cache_size=0 instead.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.359444Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.1.0) starting as process 1
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.062774Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.162206Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.359398Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.359444Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.361031Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.384325Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
docker_yeelight_v2-db-1 | 2023-09-12T21:15:40.384381Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.1.0' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
docker_yeelight_v2-app-1 | 2023-09-12 21:15:41.169667: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU will not be used.
docker_yeelight_v2-app-1 | 2023-09-12 21:15:41.210743: I tensorflow/tsl/cuda/cudart_stub.cc:28] Could not find cuda drivers on your machine, GPU will not be used.

```

Figure 5.7.1: Log of starting the containers part 1

Based on the log, it could be seen that the server container is started only after the database container is ready.

```

docker_yeelight_v2-app-1 | loading lstm
docker_yeelight_v2-app-1 | model loaded
docker_yeelight_v2-app-1 | Model: "model"
docker_yeelight_v2-app-1 |
|-----|-----|-----|
| Layer (type) | Output Shape | Param # |
|-----|-----|-----|
| input_1 (InputLayer) | [(None, 40, 1920)] | 0 |
|-----|-----|-----|
| lstm (LSTM) | (None, 128) | 1049088 |
|-----|-----|-----|
| dense (Dense) | (None, 8) | 1032 |
|-----|-----|-----|
| Total params: 1050120 (4.01 MB) |
| Trainable params: 1050120 (4.01 MB) |
| Non-trainable params: 0 (0.00 Byte) |
|-----|-----|-----|
| None |
docker_yeelight_v2-app-1 | loading extractor
docker_yeelight_v2-app-1 | Model: "model"
docker_yeelight_v2-app-1 |
|-----|-----|-----|
| Layer (type) | Output Shape | Param # |
|-----|-----|-----|
| input_2 (InputLayer) | [(None, 40, 112, 112, 3 | 0 |
| | )] | |
|-----|-----|-----|
| time_distributed (TimeDist | (None, 40, 3, 3, 1920) | 18321984 |
| | tributed) | |
|-----|-----|-----|
| time_distributed_1 (TimeDi | (None, 40, 1920) | 0 |
| | stributed) | |
|-----|-----|-----|
| Total params: 18321984 (69.89 MB) |
| Trainable params: 0 (0.00 Byte) |
| Non-trainable params: 18321984 (69.89 MB) |
|-----|-----|-----|
| None |

```

Figure 5.7.2: Log of starting the containers part 2

## CHAPTER 5

The trained LSTM is loaded, and the feature extractor is created.

```
docker_yeelight_v2-app-1 | * Serving Flask app 'server'
docker_yeelight_v2-app-1 | * Debug mode: off
docker_yeelight_v2-app-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSG
I server instead.
docker_yeelight_v2-app-1 | * Running on all addresses (0.0.0.0)
docker_yeelight_v2-app-1 | * Running on http://127.0.0.1:5000
docker_yeelight_v2-app-1 | * Running on http://172.19.0.3:5000
docker_yeelight_v2-app-1 | Press CTRL+C to quit
```

Figure 5.7.3: Log of starting the containers part 3

The server container is now up and running on local host, bound to port 5000.

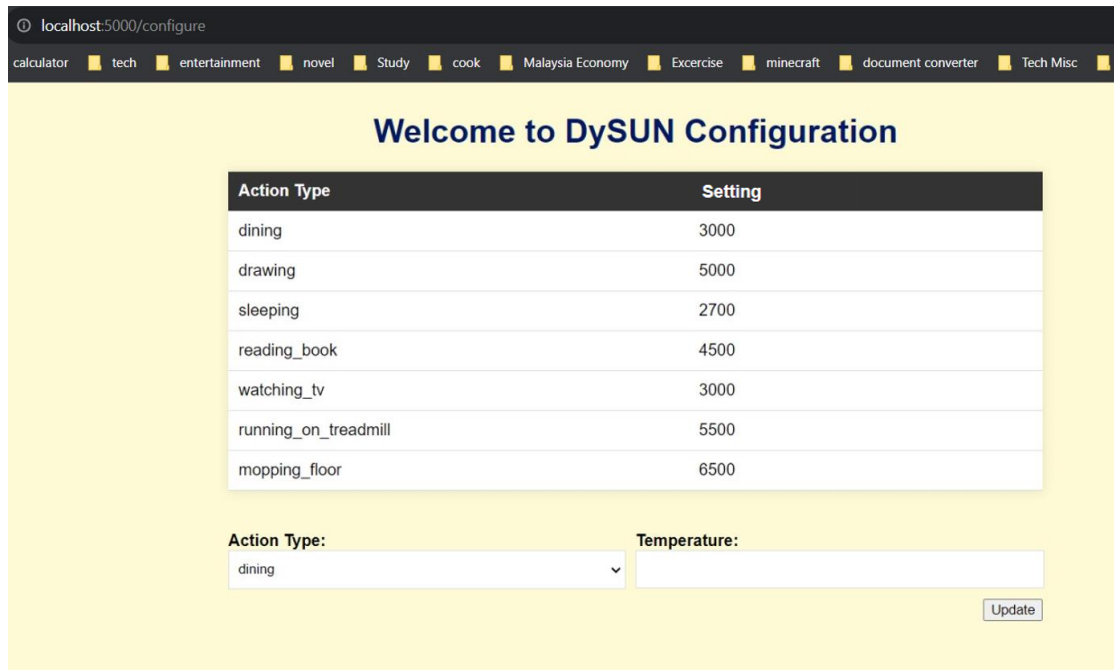


Figure 5.7.4: User Configuration Webpage

The User Configuration Webpage is accessible at localhost IP:5000/configure

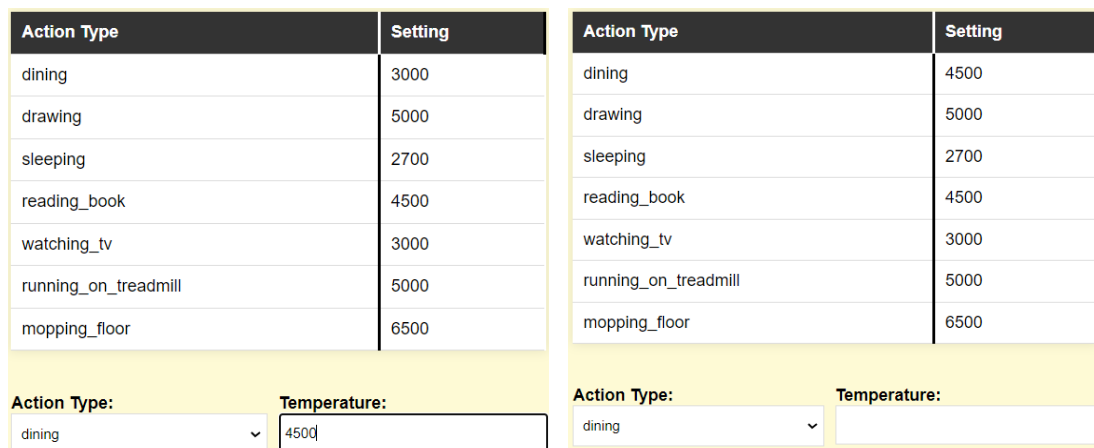


Figure 5.7.5: Updating light setting

By submitting the update request, setting of dining is changed from 3000 to 4500.



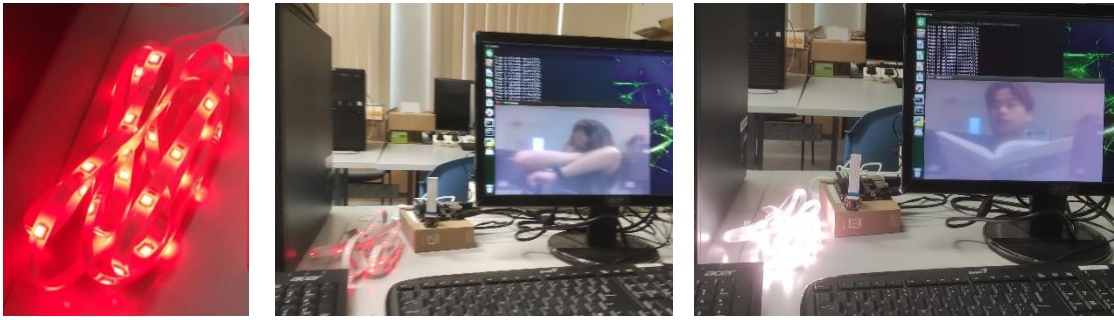


Figure 5.7.9: Yeelight setting changed. From left to right are original lighting, lighting on sleeping and lighting on reading book

## CHAPTER 6

### System Evaluation and Discussion

#### 6.1 System Testing and Performance Metrics

The following metrics is used to discuss the performance of HAR model training:

**i. Accuracy**

Accuracy is calculated by dividing the number of instance being correctly predicted/classified by the total number of all instances. It simply means that out of all sample, how many of them are being correctly classified. This is mainly used to evaluate the performance of a classifier.

**ii. Loss**

Loss is the metric to quantify the discrepancy between the model's predictions and the true target values. In model training, the target is typically to achieve the highest accuracy with minimum loss.

**iii. Confusion Matrix**

Confusion matrix helps to visualize the prediction result of a model. It is further interpreted using the following terms:

		Model Predictions			
		A	B	C	D
Actual Classes	A	9	1	0	0
	B	1	15	3	1
	C	5	0	24	1
	D	0	4	1	15

Figure 6.1.1: Example of confusion matrix

- **True Positive (TP):** The number of instances that are correctly classified as its own class. In the example above, Class A has 9 TP.
- **False Positive (FP):** The number of instances from other classes that are misclassified as the target class. For example, Class A has 6 FP (1B, 5C, 0D)
- **False Negative (FN):** The number of instances from the target class that are misclassified as other class(es). For example, Class A has 1 FN misclassified as Class B

- **True Negative (TN):** The number of instances that are from other classes and correctly predicted as classes other than the target class. All the prediction highlighted in blue in the above example are TN.
- **Precision:** The number of instances that are correctly predicted as target class, out of the total number of instances that are being classified as the target class. The formular is:

$$\mathbf{TP / (TP + FP)}$$

- **Recall:** The number of instances that are correctly predicted as the target class, out of the total number of instances of the target class. It also defined as the confidentiality of the model to predict a positive result. The formular is:

$$\mathbf{TP / (TP + FN)}$$

- **F1 Score:** F1 is always referred as the harmonic mean between precision and recall. The higher the F1, the more balance between precision and recall, the better the performance of the model. The formular is:

$$\frac{\mathbf{TP}}{\mathbf{TP + 1/2 (FN + FP)}}$$

## 6.2 Testing Setup and Result Discussion

Using the extracted features from the 3 CNN model, 4 training setting is used to compare and identify the optimal setting of CNN-LSTM model to be used for this project. All setting were trained for 30 epochs with Adam optimizer and ReduceLRonPlateau scheduler.

Table 6.2.1: Summary of result for HAR model training

Setting	Model	Train Accuracy	Validation Accuracy	Validation Loss	Evaluation Accuracy	Precision	Recall	MCP Epoch
<b>7 Class (FYP1)</b> LSTM unit: 128 Epoch: 30 Batch size: 4 Optimizer: Adam – 0.001 lr Scheduler: ReduceLRonPlateau (factor 0.2, patient 5)	<b>DGAVG</b>	0.8401	0.7796	0.6425	0.71	0.75	0.71	4
	<b>MGAVG</b>	0.8605	0.7723	0.6766	0.73	0.74	0.73	5
	<b>VGAVG</b>	0.9098	0.7737	0.7440	0.67	0.67	0.67	13

CHAPTER 6

LSTM unit: 128 Epoch: 30 Batch size: 4 Optimizer: Adam – 0.001 lr Scheduler: ReduceLROnPlateau (factor 0.2, patient 5)	<b>DGAVG</b>	0.8331	0.7448	<b>0.7528</b>	<b>0.73</b>	0.73	0.75	4
	<b>MGAVG</b>	0.8124	0.7476	0.7655	0.70	0.71	0.71	3
	<b>VGAVG</b>	0.7407	0.6472	1.0547	0.60	0.61	0.60	5
LSTM unit: 128 Epoch: 30 Batch size: 16 Optimizer: Adam – 0.001 lr Scheduler: ReduceLROnPlateau (factor 0.2, patient 5)	<b>DGAVG</b>	0.9248	0.7681	<b>0.6905</b>	<b>0.74</b>	0.76	0.74	9
	<b>MGAVG</b>	0.8085	0.7637	0.7095	0.70	0.71	0.71	3



CHAPTER 6

LSTM unit: 128 Epoch: 30 Batch size: 4 Optimizer: Adam – 0.001 lr (decay 0.001) Scheduler: ReduceLROnPlateau (factor 0.2, patient 5)	<b>DGAVG</b>	0.8632	0.7617	<b>0.7117</b>	<b>0.73</b>	0.73	0.73	29
	<b>MGAVG</b>	0.8572	0.7552	0.7344	0.72	0.72	0.72	30
LSTM unit: 10 Epoch: 30 Batch size: 4 Optimizer: Adam – 0.001 lr Scheduler: ReduceLROnPlateau (factor 0.2, patient 5)	<b>DGAVG</b>	0.8354	0.6940	<b>0.9104</b>	<b>0.67</b>	0.69	0.67	25
	<b>MGAVG</b>	0.7533	0.7020	0.9373	0.66	0.68	0.66	6

## CHAPTER 6

The best model achieved in FYP1 was using VGG16 with Global Average Pooling (VGAVG) as feature extractor, which recorded the evaluation accuracy at 0.67. The 2 new model trained in FYP2 is first compared to the previous best model on 7 action classes dataset with default settings (row 1). The result showed that both MobileNetV2 (MGAVG) and DesneNet201 (DGAVG) achieved better performance in validation loss (0.67 and 0.64, compared to 0.74) and in evaluation accuracy (0.73 and 0.71, compared to 0.67). 3 models are further compared on the 8 action classes dataset. Which the performance of VGAVG dropped to 0.60 while MGAVG and DGAVG remained at 0.7. It could be justified that the 2 new models are performing better on the dataset compared to VGAVG, hence remaining testing on the 8 action classes dataset is performed on the 2 new models.

Comparing on batch size, increasing value from 4 (row 2) to 16 (row 3) leads to a slight improvement in both model where the validation accuracy increased by 0.02 and validation loss decreased by 0.06. The evaluation accuracy of DGAVG also remarked at 0.74 (increased by 0.01) which is the highest evaluation accuracy of all variations. Even though larger batch size in general leads to a weaker generalization, it could still be justified as the dataset used in relatively small compared to a concrete HAR model training project, where chances are there for a larger batch size to achieve a better performance on small dataset.

Similar improvement shown when the models were trained with batch size 4 together with Adam optimizer at a decay of 0.01 throughout the iteration. Worth to mentioned that the validation loss and accuracy curve were smoother compared to other training without Adam decay, showing the model is steadily improving throughout the training. However, the validation loss and accuracy where still capping around 0.7. The best model checkpoint epoch also suggested that the model already reached their best performance in the first 3 – 5 epochs. As the training accuracy continued to grow till 0.9 while validation accuracy remains capped, overfitting happened where larger dataset with data augmentation is required to improve the model generalization.

Different from the result in FYP1, reducing the learning unit in LSTM from 128 to 10 promote negative impact on the performance where the validation loss spike by 0.2 and evaluation accuracy dropped by 0.15. To maintain the accuracy rate, LSTM is deployed with 128 units.

Looking into the F1 accuracy (evaluation accuracy), precision and recall, the result is pretty balance between precision and recall which shows that the model does not obviously bias to any of the activity class. However, it still noticeable that the models are having weaker performance in classifying “reading book” confidently with a general precision around 0.5 in all training setting. This false positive are mainly contributed by the class “dining”, “sleeping” and some from “watching tv” as these activities also involve sitting still in front of something.

```

[[72  2  4 12  4  2  1  1]
 [ 1 76  3  7  1  4  0  0]
 [ 1  2 74  2  7  2  7  5]
 [ 6  6  1 64  5  9  4  3]
 [ 4  0  3  3 75  1  3  7]
 [ 0  2  1 15  5 65  5  3]
 [ 1  0  6 10  3  5 70  3]
 [ 6  6  5  1  7  4  2 69]]

```

Figure 6.2.1: Confusion matrix of DGAVG model with adam decay setting.

As shown in figure 6.2.1, in column 4 (predicted: reading book), dining, sleeping and watching TV contributed 12, 15 and 10 false positive on reading book respectively.

The models were able to achieve the highest accuracy of 0.74. This is an acceptable accuracy, but also implied that the model still has a significant improvement space in terms of learning from the dataset. A reasonable explanation could be that the use of an input shape of (112, 112, 3) was less than the expected (224, 224, 3), which caused the loss of useful features for generalization. The trade-off of using global average pooling at the end to deal with the limitation of training resources could have also removed some useful features that could have improved performance. Overfitting also discovered in VGAP model which difference of 0.3 and 0.2 shown between the training and evaluation accuracy, indicate the trained model is lacking generality hence failed to adapt to unseen data in real world.

In overall, the performance of MGAVG and DGAVG were actually pretty close where DGAVG slightly better than MGAVG. However, the size of the extracted feature using MGAVG is only 1280, smaller than the size of DGAVG of 1920. The Time Per Inference Step (ms) of MGAVG is also much faster (CPU: 25.9; GPU: 3.8) compared to DGAVG (CPU: 127.2; GPU: 6.7), which could greatly reduce the time for feature extraction and LSTM training [32]. MGAVG could be the optimum choice for

## CHAPTER 6

production considering the model efficiency. However, in this project, DGAVG trained on batch size 16 is used for the prototype for demonstrating the best accuracy achieved. Anyway, the 2 models are considered interchangeable based on user's consideration.

To summarize the above, DGAVG models are performing slightly better than MGAVG models, The best evaluation accuracy is considered acceptable but could still be further improved. Difference in batch size and LSTM unit count promoted significant impact on the performance in this project setup. MGAVG is slightly weaker but is faster and require fewer computing resources. As the project is running on a server with less restriction on computing resources, DGAVG is used in the prototype to demonstrate with the best accuracy.

\*Source Code for HAR model and Lighting Control Program is available [here](#).

### 6.3 Project Challenges

The first issue to implement the project is the dataset collection. All along, there has been insufficient human activity video dataset compared to other video classification task. The activity class supported by each dataset are also relatively less, which makes it hard to gather one set of data having all kind of activity that have a reasonable lighting guideline to suit the indoor activity recognition theme of this project. Some of the dataset also required raw crawling from YouTube, which is more time consuming and require special care as some video may have been removed.

Besides that, hardware resource is another constraint as limitation in GPU memory restricted the size of feature for LSTM training. This constraint caused the project to implement the feature extractor and LSTM model with an input size of 112 x 112 that is even smaller than the suggested input shape (e.g., 224 x 224). Such actions may cause loss of feature from the original data, which leads to unfavourable learning and performance on the model inference.

When testing with the actual system, the performance of the model is somehow unstable which the result may be flickering if an action is being performed continuously. The prediction of the model is also impacted by the angle and distance to the camera. The more information captured within a frame (e.g., a person with many things surrounded), the greater the models tend to be impacted. The model also having difficulties when multiple people is presented in the same frame as the dataset is more on single actor with close third-person camera view or first-person view.

## 6.4 Objective Evaluation

The proposed system had achieved the 2 objectives state in Chapter 1.

- i. Objective 1 was achieved with the proposed DGAVG model. The project had deployed the CNN-LSTM network architecture as the HAR model and trained the model using the Kinetic 700\_2020 dataset. Several training settings was tested with the best result reported at F1 score of 74%. The model is capable of classifying 8 class of human activities, included dining, drawing, mopping floor, reading, running on treadmill, sleeping, watching tv and yoga.
- ii. Objective 2 was achieved by deploying the classification server and controlling client. The Flask API docker was developed to perform classification on the sample uploaded by the client, while the client is responsible for uploading the captured image and controlling the IoT light. The control of light is automated based on the activity type classified by the classification server.

## CHAPTER 7

### Conclusion and Recommendation

#### 7.1 Conclusion

The correct use of lighting had been proven to have positive impact on human work productivity and also maintain a healthy state of visual acuity and mental stability. Even though highly configurable IoT light had been introduced to the market for a long time, there isn't an application that could fully automate the control of such IoT product and provide lighting suggestion based on the activity that is being performed.

Hence, this project had proposed a fully automated lighting control system that react to 8 classes of human activity with the use of IoT and HAR. The project used the CNN-LSTM network model for the HAR system, where DenseNet201 as the feature extractor and LSTM layer for temporal learning. The model was able to achieve an acceptable accuracy at 0.74. Using the trained model, a lighting configuration program is coded in python using the Yeelight framework to control the IoT light using client-server architecture. The project proved that the idea of fully automated lighting configuration program is a realisable idea and subject to further improvement for better performance.

### 7.2 Recommendation

The project has proven the concept of HAR for lighting control is applicable. However, the accuracy of trained model is not highly satisfied. Improvement could be achieved by trying out different combination of pretrained CNN and LSTM to extract more useful spatial features for temporal learning.

Besides that, the camera angle, distance, and number of people presented in the frame impact the performance of the model when deploying the system. This could be due to the lack of diversity in training dataset. Hence, it is suggested to develop a more comprehensive dataset that considered the elements mentioned above.

Last but not least, the proposed model only able to support 8 activity class at the moment. The future work could consider training the model with more activity class to support a wider range of activity with necessary lighting parameters for better production value.



## REFERENCES

- [1] L. Yao *et al.*, “WITS: an IoT-endowed computational framework for activity recognition in personalized smart homes,” *Computing*, vol. 100, no. 4, pp. 369–385, 2018, doi: 10.1007/s00607-018-0603-z.
- [2] O. C. Ann and L. B. Theng, “Human activity recognition: A review,” *Proc. - 4th IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2014*, no. March, pp. 389–393, 2014, doi: 10.1109/ICCSCE.2014.7072750.
- [3] S. Deep and X. Zheng, “Leveraging CNN and Transfer Learning for Vision-based Human Activity Recognition,” *2019 29th Int. Telecommun. Networks Appl. Conf. ITNAC 2019*, pp. 31–34, 2019, doi: 10.1109/ITNAC46935.2019.9078016.
- [4] S. Indolia, A. K. Goswami, S. P. Mishra, and P. Asopa, “Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach,” *Procedia Comput. Sci.*, vol. 132, pp. 679–688, 2018, doi: 10.1016/j.procs.2018.05.069.
- [5] S. W. Perry, “Handbook of Neural Network Signal Processing,” *J. Acoust. Soc. Am.*, vol. 111, no. 6, pp. 2525–2526, Jun. 2002, doi: 10.1121/1.1480419.
- [6] K. Mani, “Understanding Recurrent Neural Networks (RNNs) from Scratch,” *Medium*. <https://medium.com/@kaushikmani/understanding-recurrent-neural-networks-aea0078defc6> (accessed Apr. 23, 2023).
- [7] C. Olah, “Understanding LSTM Networks,” *colah.github.com*, Aug. 27, 2015. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (accessed Apr. 23, 2023).
- [8] K. Mani, “GRU’s and LSTM’s,” *Medium.com*, Feb. 17, 2019. <https://towardsdatascience.com/grus-and-lstm-s-741709a9b9b1> (accessed Apr. 23, 2023).
- [9] M. Ruth, “Lighting ( illumination ).,” *Salem Press Encyclopedia*. 2022.
- [10] O. Osibona, B. D. Solomon, and D. Fecht, “Lighting in the home and health: A systematic review,” *Int. J. Environ. Res. Public Health*, vol. 18, no. 2, pp. 1–20, 2021, doi: 10.3390/ijerph18020609.
- [11] M. A. Gul, M. H. Yousaf, S. Nawaz, Z. U. Rehman, and H. Kim, “Patient monitoring by abnormal human activity recognition based on CNN architecture,” *Electron.*, vol. 9, no. 12, pp. 1–14, 2020, doi: 10.3390/electronics9121993.
- [12] K. Rangasamy, M. A. As’ari, N. A. Rahmad, and N. F. Ghazali, “Hockey activity recognition using pre-trained deep learning model,” *ICT Express*, vol. 6, no. 3, pp. 170–174, 2020, doi: 10.1016/j.ict.2020.04.013.
- [13] J. Donahue *et al.*, “Long-Term Recurrent Convolutional Networks for Visual Recognition and Description,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4,

## REFERENCES

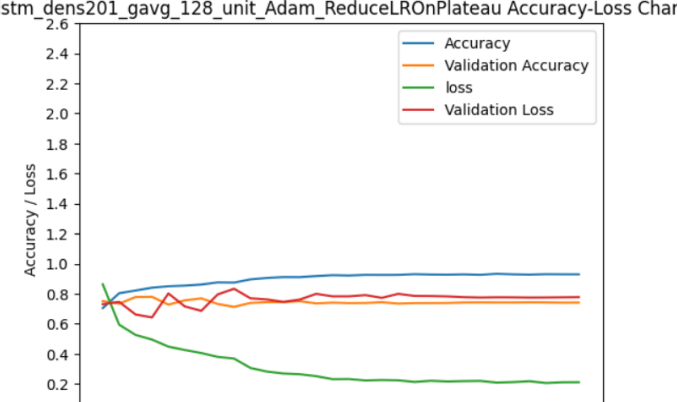
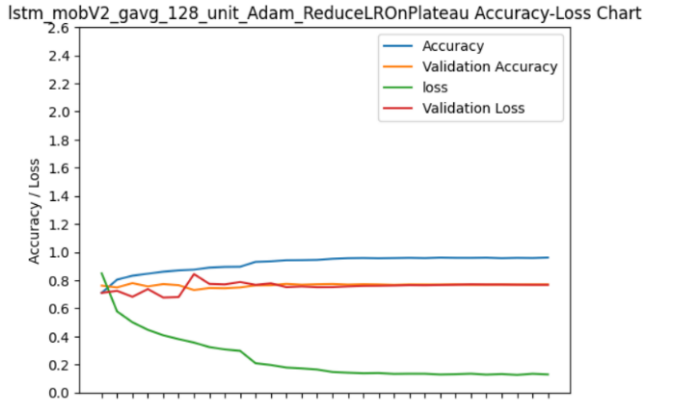
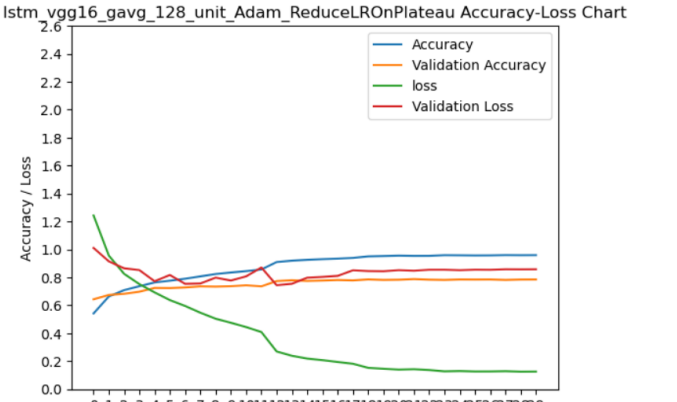
- pp. 677–691, 2017, doi: 10.1109/TPAMI.2016.2599174.
- [14] Y. Peng, Y. Zhao, and J. Zhang, “Two-Stream Collaborative Learning with Spatialoral Attention for Video Classification,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 773–786, 2019, doi: 10.1109/TCSVT.2018.2808685.
- [15] T. Brox, A. Bruhn, N. Papenbergh, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3024, pp. 25–36, 2004, doi: 10.1007/978-3-540-24673-2\_3.
- [16] J. Monteiro, R. Granada, R. C. Barros, and F. Meneguzzi, “Deep neural networks for kitchen activity recognition,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017-May, pp. 2048–2055, 2017, doi: 10.1109/IJCNN.2017.7966102.
- [17] N. Adnan, N. Kamal, and K. Chellappan, “An IoT based smart lighting system based on human activity,” *2019 14th IEEE Malaysia Int. Conf. Commun. Emerg. Technol. IoE 5G, MICC 2019*, no. December, pp. 65–68, 2019, doi: 10.1109/MICC48337.2019.9037601.
- [18] S. Y. Chun and C. S. Lee, “Applications of human motion tracking: Smart lighting control,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 387–392, 2013, doi: 10.1109/CVPRW.2013.65.
- [19] Y. Li, T. Ru, Q. Chen, L. Qian, X. Luo, and G. Zhou, “Effects of illuminance and correlated color temperature of indoor light on emotion perception,” *Sci. Rep.*, vol. 11, no. 1, pp. 1–12, 2021, doi: 10.1038/s41598-021-93523-y.
- [20] P. R. Mills, S. C. Tomkins, and L. J. M. Schlangen, “The effect of high correlated colour temperature office lighting on employee wellbeing and work performance,” *J. Circadian Rhythms*, vol. 5, pp. 1–9, 2007, doi: 10.1186/1740-3391-5-2.
- [21] “Which colour temperature for which room? ,” *ANY-LAMP*. <https://www.any-lamp.com/blog/which-color-temperature-for-what-room> (accessed Apr. 23, 2023).
- [22] Matt Fussell, “Lighting Your Art Studio,” *Thevirtualinstructor*. <https://thevirtualinstructor.com/blog/lighting-your-art-studio> (accessed Apr. 23, 2023).
- [23] “What color temperature best approximates natural sunshine?,” *Waveform Lighting*. <https://www.waveformlighting.com/knowledgebase/full-spectrum/what-color-temperature-best-approximates-natural-sunshine> (accessed Apr. 23, 2023).
- [24] S. C. Liew, “Optical Remote Sensing,” *Centre for Remote Imaging, Sensing and Processing, National University of Singapore*. <https://crisp.nus.edu.sg/~research/tutorial/optical.htm> (accessed Apr. 23, 2023).
- [25] S. M. C. Nascimento and O. Masuda, “Best lighting for visual appreciation of artistic paintings—experiments with real paintings and real illumination,” *J. Opt. Soc. Am. A*,

## REFERENCES

- vol. 31, no. 4, p. A214, 2014, doi: 10.1364/josaa.31.00a214.
- [26] F. Team, “How to Choose the Right Light Temperature for your Bulb or Fixture,” *Feit electric*, Apr. 05, 2017. <https://www.feit.com/blogs/inspire/how-to-choose-the-right-color-light-temperature-for-your-bulb-or-fixture> (accessed Apr. 23, 2023).
- [27] M. Cheong, “The Best Lighting for Studying and Reading,” *Taotronics*, Feb. 11, 2020. <https://blog.taotronics.com/home-garden/lighting/the-best-lighting-for-studying-and-reading/> (accessed Apr. 23, 2023).
- [28] “How to get the best home gym lighting,” *Philips Hue US*, Jun. 10, 2020. <https://www.philips-hue.com/en-us/explore-hue/blog/best-home-gym-lighting> (accessed Apr. 23, 2023).
- [29] P. Boyce and D. J. Kennaway, “Effects of light on melatonin production,” *Biol. Psychiatry*, vol. 22, no. 4, pp. 473–478, 1987, doi: 10.1016/0006-3223(87)90169-7.
- [30] C. H. Yang and P. Y. Chang, “Forecasting the demand for container throughput using a mixed-precision neural architecture based on cnn–lstm,” *Mathematics*, vol. 8, no. 10, pp. 1–17, 2020, doi: 10.3390/math8101784.
- [31] kinetics-cvdf, “Kinetics Datasets Downloader,” *Github*. <https://github.com/cvdfoundation/kinetics-dataset> (accessed Apr. 24, 2023).
- [32] Keras, “Keras Applications.” <https://keras.io/api/applications/> (accessed Sep. 14, 2023).

**APPENDIX**

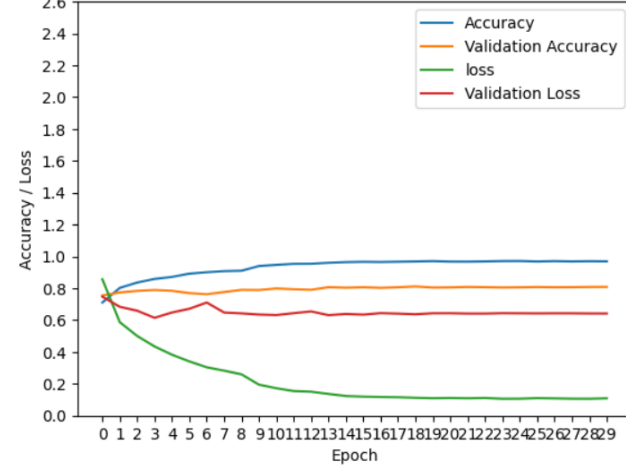
**Appendix A: Training result of different model setting for 7 Class**

Setup	Dens201_GAVG	MobV2_GAVG	VGG16_GAVG																																																																																																																																																																					
LSTM unit: 128 Epoch: 30 Batch size: 4 Optimizer: Adam – 0.001 lr Scheduler: ReduceLRonPlateau (factor 0.2, patient 5)																																																																																																																																																																								
	<pre> [[70 4 2 10 8 4 0]  [ 0 82 1 5 4 0 0]  [ 0 2 56 7 29 3 3]  [ 3 8 0 61 16 10 0]  [ 2 0 0 1 87 5 1]  [ 0 6 0 18 8 62 2]  [ 1 0 5 14 9 5 64]]                     </pre>	<pre> [[77 1 3 9 5 2 1]  [ 1 78 2 7 1 3 0]  [ 5 2 77 7 4 2 3]  [ 9 12 2 66 0 6 3]  [ 3 4 7 7 70 2 3]  [ 2 6 6 18 3 58 3]  [ 1 2 7 13 5 3 67]]                     </pre>	<pre> [[69 1 2 10 2 8 6]  [ 2 74 1 6 6 1 2]  [ 7 3 71 2 12 2 3]  [13 9 6 55 4 7 4]  [ 6 4 8 9 60 3 6]  [ 3 4 6 15 6 60 2]  [ 7 0 7 13 6 2 63]]                     </pre>																																																																																																																																																																					
	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.92</td><td>0.71</td><td>0.80</td><td>98</td></tr> <tr><td>1</td><td>0.80</td><td>0.89</td><td>0.85</td><td>92</td></tr> <tr><td>2</td><td>0.88</td><td>0.56</td><td>0.68</td><td>100</td></tr> <tr><td>3</td><td>0.53</td><td>0.62</td><td>0.57</td><td>98</td></tr> <tr><td>4</td><td>0.54</td><td>0.91</td><td>0.68</td><td>96</td></tr> <tr><td>5</td><td>0.70</td><td>0.65</td><td>0.67</td><td>96</td></tr> <tr><td>6</td><td>0.91</td><td>0.65</td><td>0.76</td><td>98</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.71</td><td>678</td></tr> <tr><td>macro avg</td><td>0.75</td><td>0.71</td><td>0.72</td><td>678</td></tr> <tr><td>weighted avg</td><td>0.75</td><td>0.71</td><td>0.72</td><td>678</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.92	0.71	0.80	98	1	0.80	0.89	0.85	92	2	0.88	0.56	0.68	100	3	0.53	0.62	0.57	98	4	0.54	0.91	0.68	96	5	0.70	0.65	0.67	96	6	0.91	0.65	0.76	98	accuracy			0.71	678	macro avg	0.75	0.71	0.72	678	weighted avg	0.75	0.71	0.72	678	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.79</td><td>0.79</td><td>0.79</td><td>98</td></tr> <tr><td>1</td><td>0.74</td><td>0.85</td><td>0.79</td><td>92</td></tr> <tr><td>2</td><td>0.74</td><td>0.77</td><td>0.75</td><td>100</td></tr> <tr><td>3</td><td>0.52</td><td>0.67</td><td>0.59</td><td>98</td></tr> <tr><td>4</td><td>0.80</td><td>0.73</td><td>0.76</td><td>96</td></tr> <tr><td>5</td><td>0.76</td><td>0.60</td><td>0.67</td><td>96</td></tr> <tr><td>6</td><td>0.84</td><td>0.68</td><td>0.75</td><td>98</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.73</td><td>678</td></tr> <tr><td>macro avg</td><td>0.74</td><td>0.73</td><td>0.73</td><td>678</td></tr> <tr><td>weighted avg</td><td>0.74</td><td>0.73</td><td>0.73</td><td>678</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.79	0.79	0.79	98	1	0.74	0.85	0.79	92	2	0.74	0.77	0.75	100	3	0.52	0.67	0.59	98	4	0.80	0.73	0.76	96	5	0.76	0.60	0.67	96	6	0.84	0.68	0.75	98	accuracy			0.73	678	macro avg	0.74	0.73	0.73	678	weighted avg	0.74	0.73	0.73	678	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.64</td><td>0.70</td><td>0.67</td><td>98</td></tr> <tr><td>1</td><td>0.78</td><td>0.80</td><td>0.79</td><td>92</td></tr> <tr><td>2</td><td>0.70</td><td>0.71</td><td>0.71</td><td>100</td></tr> <tr><td>3</td><td>0.50</td><td>0.56</td><td>0.53</td><td>98</td></tr> <tr><td>4</td><td>0.62</td><td>0.62</td><td>0.62</td><td>96</td></tr> <tr><td>5</td><td>0.72</td><td>0.62</td><td>0.67</td><td>96</td></tr> <tr><td>6</td><td>0.73</td><td>0.64</td><td>0.68</td><td>98</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.67</td><td>678</td></tr> <tr><td>macro avg</td><td>0.67</td><td>0.67</td><td>0.67</td><td>678</td></tr> <tr><td>weighted avg</td><td>0.67</td><td>0.67</td><td>0.67</td><td>678</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.64	0.70	0.67	98	1	0.78	0.80	0.79	92	2	0.70	0.71	0.71	100	3	0.50	0.56	0.53	98	4	0.62	0.62	0.62	96	5	0.72	0.62	0.67	96	6	0.73	0.64	0.68	98	accuracy			0.67	678	macro avg	0.67	0.67	0.67	678	weighted avg	0.67	0.67	0.67	678
	precision	recall	f1-score	support																																																																																																																																																																				
0	0.92	0.71	0.80	98																																																																																																																																																																				
1	0.80	0.89	0.85	92																																																																																																																																																																				
2	0.88	0.56	0.68	100																																																																																																																																																																				
3	0.53	0.62	0.57	98																																																																																																																																																																				
4	0.54	0.91	0.68	96																																																																																																																																																																				
5	0.70	0.65	0.67	96																																																																																																																																																																				
6	0.91	0.65	0.76	98																																																																																																																																																																				
accuracy			0.71	678																																																																																																																																																																				
macro avg	0.75	0.71	0.72	678																																																																																																																																																																				
weighted avg	0.75	0.71	0.72	678																																																																																																																																																																				
	precision	recall	f1-score	support																																																																																																																																																																				
0	0.79	0.79	0.79	98																																																																																																																																																																				
1	0.74	0.85	0.79	92																																																																																																																																																																				
2	0.74	0.77	0.75	100																																																																																																																																																																				
3	0.52	0.67	0.59	98																																																																																																																																																																				
4	0.80	0.73	0.76	96																																																																																																																																																																				
5	0.76	0.60	0.67	96																																																																																																																																																																				
6	0.84	0.68	0.75	98																																																																																																																																																																				
accuracy			0.73	678																																																																																																																																																																				
macro avg	0.74	0.73	0.73	678																																																																																																																																																																				
weighted avg	0.74	0.73	0.73	678																																																																																																																																																																				
	precision	recall	f1-score	support																																																																																																																																																																				
0	0.64	0.70	0.67	98																																																																																																																																																																				
1	0.78	0.80	0.79	92																																																																																																																																																																				
2	0.70	0.71	0.71	100																																																																																																																																																																				
3	0.50	0.56	0.53	98																																																																																																																																																																				
4	0.62	0.62	0.62	96																																																																																																																																																																				
5	0.72	0.62	0.67	96																																																																																																																																																																				
6	0.73	0.64	0.68	98																																																																																																																																																																				
accuracy			0.67	678																																																																																																																																																																				
macro avg	0.67	0.67	0.67	678																																																																																																																																																																				
weighted avg	0.67	0.67	0.67	678																																																																																																																																																																				

APPENDIX

LSTM unit: 128  
 Epoch: 30  
 Batch size: 16  
 Optimizer: Adam – 0.001 lr  
 Scheduler:  
 ReduceLRonPlateau (factor 0.2, patient 5)

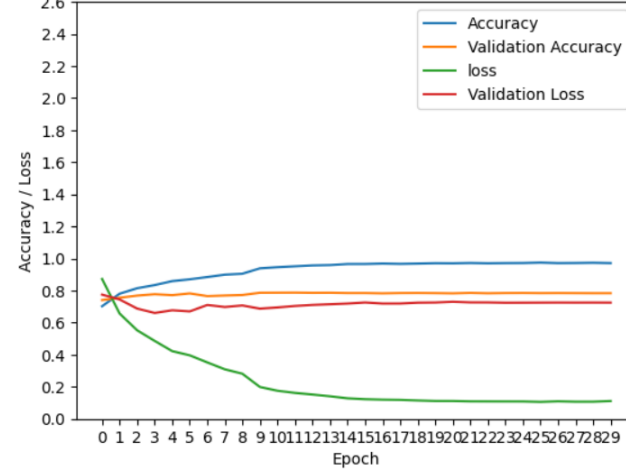
lstm\_dens201\_gavg\_128\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[65 1 6 14 6 3 3]
 [ 0 81 3 6 0 2 0]
 [ 1 2 80 8 6 3 0]
 [ 3 5 4 73 3 8 2]
 [ 3 1 9 3 72 4 4]
 [ 0 1 2 20 6 63 4]
 [ 0 0 6 8 5 3 76]]
```

	precision	recall	f1-score	support
0	0.90	0.66	0.76	98
1	0.89	0.88	0.89	92
2	0.73	0.80	0.76	100
3	0.55	0.74	0.63	98
4	0.73	0.75	0.74	96
5	0.73	0.66	0.69	96
6	0.85	0.78	0.81	98
accuracy			0.75	678
macro avg	0.77	0.75	0.76	678
weighted avg	0.77	0.75	0.76	678

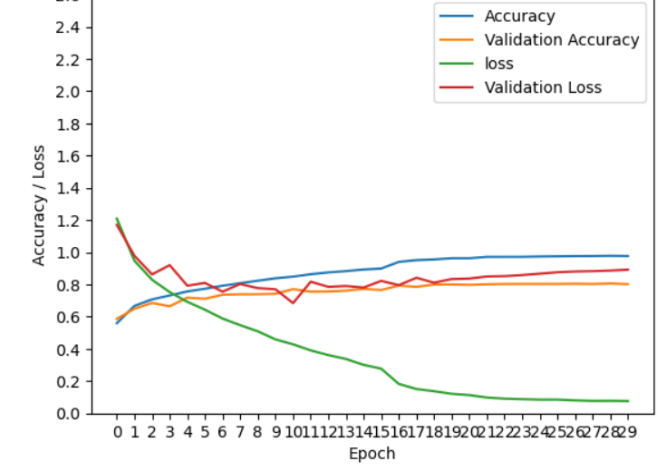
lstm\_mobV2\_gavg\_128\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[80 1 3 9 3 2 0]
 [ 1 74 4 8 1 4 0]
 [ 2 3 72 1 15 2 5]
 [ 5 9 6 59 7 10 2]
 [ 4 1 8 1 77 3 2]
 [ 1 4 2 16 7 63 3]
 [ 3 0 5 7 11 7 65]]
```

	precision	recall	f1-score	support
0	0.83	0.82	0.82	98
1	0.80	0.80	0.80	92
2	0.72	0.72	0.72	100
3	0.58	0.60	0.59	98
4	0.64	0.80	0.71	96
5	0.69	0.66	0.67	96
6	0.84	0.66	0.74	98
accuracy			0.72	678
macro avg	0.73	0.72	0.72	678
weighted avg	0.73	0.72	0.72	678

lstm\_vgg16\_gavg\_128\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



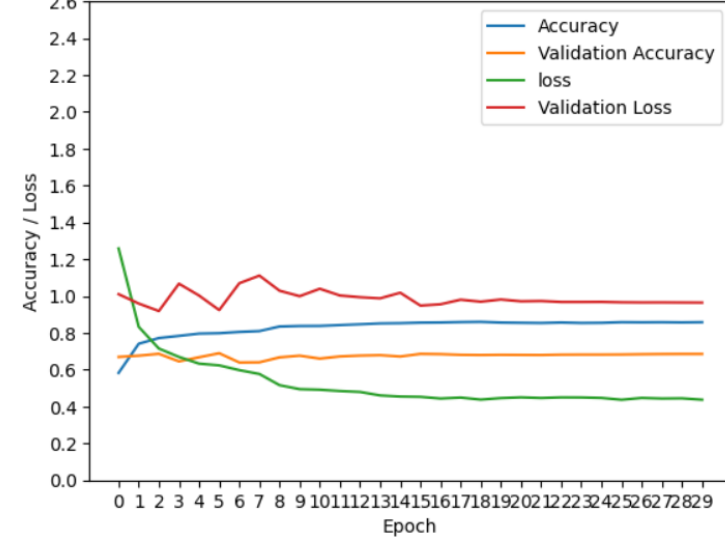
```
[[67 0 2 18 2 4 5]
 [ 2 72 5 4 5 3 1]
 [ 9 7 67 3 8 0 6]
 [15 8 3 51 7 8 6]
 [ 3 3 9 8 64 4 5]
 [ 2 7 7 14 6 55 5]
 [ 6 2 9 8 5 3 65]]
```

	precision	recall	f1-score	support
0	0.64	0.68	0.66	98
1	0.73	0.78	0.75	92
2	0.66	0.67	0.66	100
3	0.48	0.52	0.50	98
4	0.66	0.67	0.66	96
5	0.71	0.57	0.64	96
6	0.70	0.66	0.68	98
accuracy			0.65	678
macro avg	0.65	0.65	0.65	678
weighted avg	0.65	0.65	0.65	678

APPENDIX

LSTM unit: 10  
 Epoch: 30  
 Batch size: 4  
 Optimizer: Adam – 0.001 lr  
 Scheduler:  
 ReduceLRonPlateau (factor  
 0.2, patient 5)

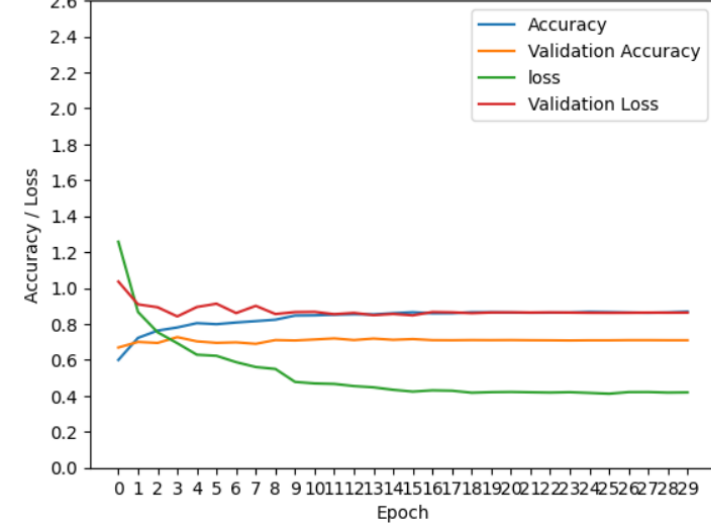
lstm\_dens201\_gavg\_10\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[75 1 2 5 14 1 0]
 [ 4 77 1 2 7 1 0]
 [ 5 2 64 1 27 1 0]
 [20 12 1 31 24 8 2]
 [ 5 0 1 0 87 1 2]
 [ 2 7 0 5 19 62 1]
 [ 3 0 8 3 19 5 60]]
```

	precision	recall	f1-score	support
0	0.66	0.77	0.71	98
1	0.78	0.84	0.81	92
2	0.83	0.64	0.72	100
3	0.66	0.32	0.43	98
4	0.44	0.91	0.59	96
5	0.78	0.65	0.71	96
6	0.92	0.61	0.74	98
accuracy			0.67	678
macro avg	0.73	0.67	0.67	678
weighted avg	0.73	0.67	0.67	678

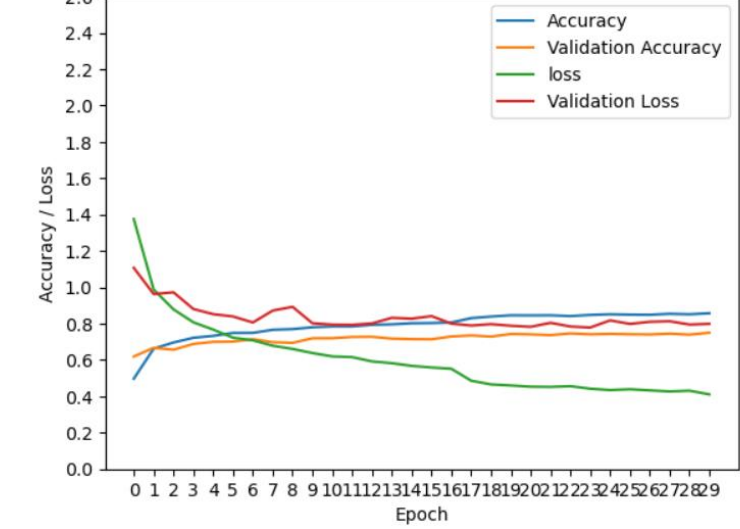
lstm\_mobV2\_gavg\_10\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[70 2 0 8 9 8 1]
 [ 0 78 0 3 5 6 0]
 [ 3 4 61 3 24 2 3]
 [ 3 7 2 53 10 19 4]
 [ 0 2 5 3 72 10 4]
 [ 0 1 0 10 6 79 0]
 [ 0 0 2 14 16 4 62]]
```

	precision	recall	f1-score	support
0	0.92	0.71	0.80	98
1	0.83	0.85	0.84	92
2	0.87	0.61	0.72	100
3	0.56	0.54	0.55	98
4	0.51	0.75	0.61	96
5	0.62	0.82	0.71	96
6	0.84	0.63	0.72	98
accuracy			0.70	678
macro avg	0.74	0.70	0.71	678
weighted avg	0.74	0.70	0.71	678

lstm\_vgg16\_gavg\_10\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



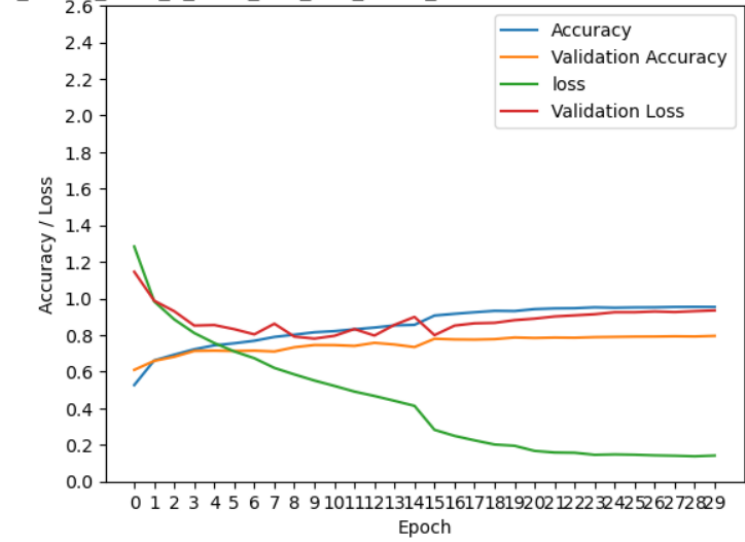
```
[[61 0 0 18 4 8 7]
 [ 5 71 1 7 3 4 1]
 [ 6 11 66 1 13 0 3]
 [11 9 4 60 3 6 5]
 [10 1 7 10 59 4 5]
 [ 1 2 7 18 4 62 2]
 [ 7 0 8 5 10 4 64]]
```

	precision	recall	f1-score	support
0	0.60	0.62	0.61	98
1	0.76	0.77	0.76	92
2	0.71	0.66	0.68	100
3	0.50	0.61	0.55	98
4	0.61	0.61	0.61	96
5	0.70	0.65	0.67	96
6	0.74	0.65	0.69	98
accuracy			0.65	678
macro avg	0.66	0.65	0.66	678
weighted avg	0.66	0.65	0.66	678

APPENDIX

LSTM layer 1 unit: 128  
 LSTM layer 2 unit: 64  
 Epoch: 30  
 Batch size: 4  
 Optimizer: Adam – 0.001 lr  
 Scheduler:  
 ReduceLRonPlateau (factor  
 0.2, patient 5)

lstm\_vgg16\_gavg\_2\_layer\_128\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```

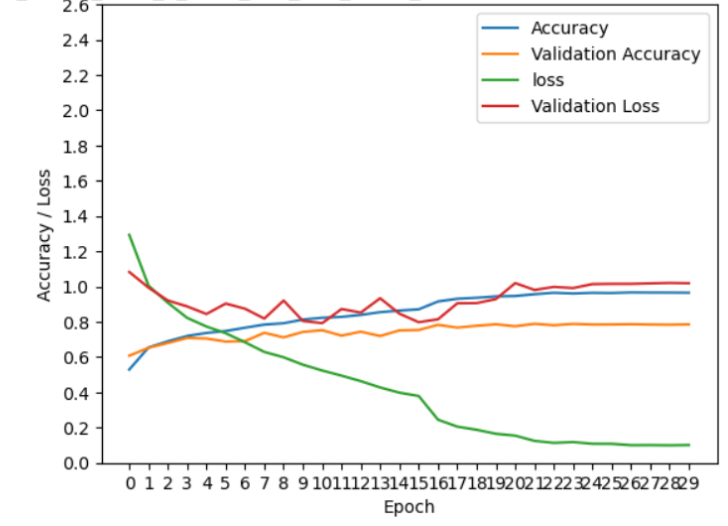
[[66 1 3 12 2 6 8]
 [ 2 69 4 9 3 3 2]
 [ 7 5 63 3 16 2 4]
 [10 8 4 58 9 5 4]
 [ 5 4 8 5 62 7 5]
 [ 3 4 5 11 8 62 3]
 [ 6 1 6 10 8 2 65]]
    
```

	precision	recall	f1-score	support
0	0.67	0.67	0.67	98
1	0.75	0.75	0.75	92
2	0.68	0.63	0.65	100
3	0.54	0.59	0.56	98
4	0.57	0.65	0.61	96
5	0.71	0.65	0.68	96
6	0.71	0.66	0.69	98
accuracy			0.66	678
macro avg	0.66	0.66	0.66	678
weighted avg	0.66	0.66	0.66	678

APPENDIX

LSTM layer 1 unit: 256  
 LSTM layer 2 unit: 128  
 Epoch: 30  
 Batch size: 4  
 Optimizer: Adam – 0.001 lr  
 Scheduler:  
 ReduceLRonPlateau (factor  
 0.2, patient 5)

lstm\_vgg16\_gavg\_2\_layer\_128\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```

[[64 2 2 14 1 6 9]
 [ 1 69 5 6 6 5 0]
 [ 7 4 73 4 7 1 4]
 [12 10 7 52 7 6 4]
 [ 5 1 11 7 65 3 4]
 [ 6 5 6 15 6 53 5]
 [ 9 1 7 9 2 3 67]]
    
```

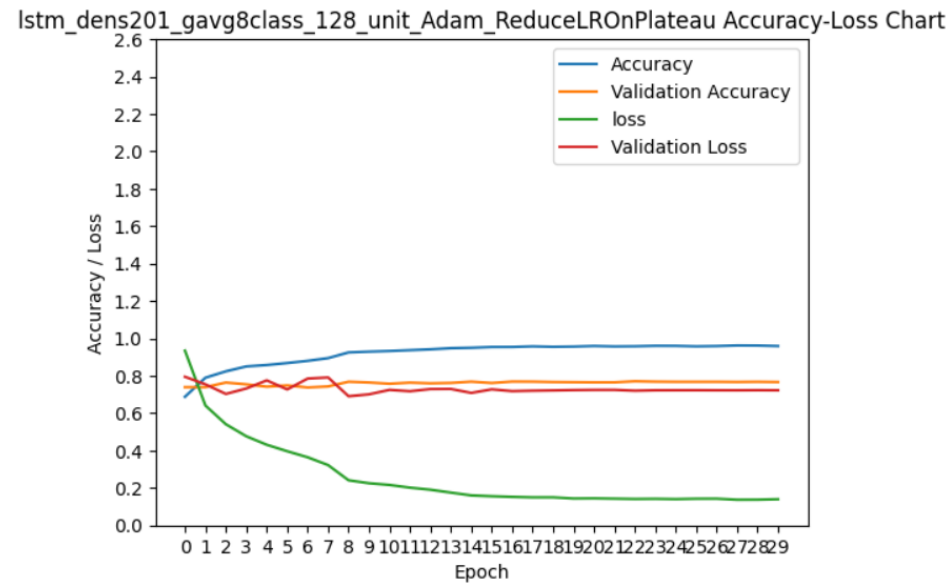
	precision	recall	f1-score	support
0	0.62	0.65	0.63	98
1	0.75	0.75	0.75	92
2	0.66	0.73	0.69	100
3	0.49	0.53	0.51	98
4	0.69	0.68	0.68	96
5	0.69	0.55	0.61	96
6	0.72	0.68	0.70	98
accuracy			0.65	678
macro avg	0.66	0.65	0.65	678
weighted avg	0.66	0.65	0.65	678



Appendix B: Training result of different model setting for 8 Class

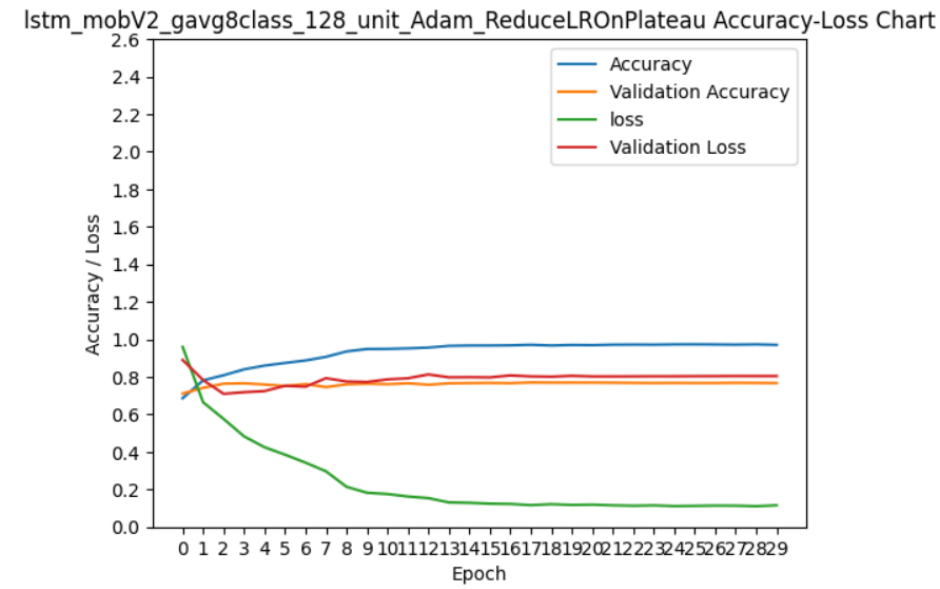
Setup	Dens201_GAVG	MobV2_GAVG	VGG16_GAVG																																																																																																																																																																																				
<b>LSTM unit: 128</b> <b>Epoch: 30</b> <b>Batch size: 4</b> <b>Optimizer: Adam – 0.001 lr</b> <b>Scheduler: ReduceLRonPlateau (factor 0.2, patient 5)</b>																																																																																																																																																																																							
	<pre> [[62 1 1 16 5 8 1 4]  [ 0 74 2 6 2 5 0 3]  [ 0 2 68 3 9 4 3 11]  [ 2 8 1 62 4 15 0 6]  [ 1 0 0 3 73 5 4 10]  [ 0 0 2 14 3 75 0 2]  [ 0 0 3 5 6 9 69 6]  [ 3 2 2 4 3 3 0 83]]                     </pre>	<pre> [[71 2 0 14 4 0 1 6]  [ 0 71 4 8 2 4 0 3]  [ 0 4 66 5 6 2 3 14]  [ 3 9 2 60 9 6 4 5]  [ 0 2 5 7 70 1 4 7]  [ 0 5 1 11 5 68 4 2]  [ 3 1 3 8 6 6 66 5]  [ 1 4 5 8 3 8 1 70]]                     </pre>	<pre> [[65 0 2 13 3 8 6 1]  [ 1 70 3 7 4 4 1 2]  [ 6 5 65 3 8 3 3 7]  [11 8 2 49 6 10 8 4]  [ 3 3 12 9 54 1 6 8]  [ 5 0 6 21 7 50 6 1]  [ 4 0 8 7 7 6 61 5]  [ 6 3 6 7 11 3 8 56]]                     </pre>																																																																																																																																																																																				
	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.91</td><td>0.63</td><td>0.75</td><td>98</td></tr> <tr><td>1</td><td>0.85</td><td>0.80</td><td>0.83</td><td>92</td></tr> <tr><td>2</td><td>0.86</td><td>0.68</td><td>0.76</td><td>100</td></tr> <tr><td>3</td><td>0.55</td><td>0.63</td><td>0.59</td><td>98</td></tr> <tr><td>4</td><td>0.70</td><td>0.76</td><td>0.73</td><td>96</td></tr> <tr><td>5</td><td>0.60</td><td>0.78</td><td>0.68</td><td>96</td></tr> <tr><td>6</td><td>0.90</td><td>0.70</td><td>0.79</td><td>98</td></tr> <tr><td>7</td><td>0.66</td><td>0.83</td><td>0.74</td><td>100</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.73</td><td>778</td></tr> <tr><td>macro avg</td><td>0.75</td><td>0.73</td><td>0.73</td><td>778</td></tr> <tr><td>weighted avg</td><td>0.75</td><td>0.73</td><td>0.73</td><td>778</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.91	0.63	0.75	98	1	0.85	0.80	0.83	92	2	0.86	0.68	0.76	100	3	0.55	0.63	0.59	98	4	0.70	0.76	0.73	96	5	0.60	0.78	0.68	96	6	0.90	0.70	0.79	98	7	0.66	0.83	0.74	100	accuracy			0.73	778	macro avg	0.75	0.73	0.73	778	weighted avg	0.75	0.73	0.73	778	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.91</td><td>0.72</td><td>0.81</td><td>98</td></tr> <tr><td>1</td><td>0.72</td><td>0.77</td><td>0.75</td><td>92</td></tr> <tr><td>2</td><td>0.77</td><td>0.66</td><td>0.71</td><td>100</td></tr> <tr><td>3</td><td>0.50</td><td>0.61</td><td>0.55</td><td>98</td></tr> <tr><td>4</td><td>0.67</td><td>0.73</td><td>0.70</td><td>96</td></tr> <tr><td>5</td><td>0.72</td><td>0.71</td><td>0.71</td><td>96</td></tr> <tr><td>6</td><td>0.80</td><td>0.67</td><td>0.73</td><td>98</td></tr> <tr><td>7</td><td>0.62</td><td>0.70</td><td>0.66</td><td>100</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.70</td><td>778</td></tr> <tr><td>macro avg</td><td>0.71</td><td>0.70</td><td>0.70</td><td>778</td></tr> <tr><td>weighted avg</td><td>0.71</td><td>0.70</td><td>0.70</td><td>778</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.91	0.72	0.81	98	1	0.72	0.77	0.75	92	2	0.77	0.66	0.71	100	3	0.50	0.61	0.55	98	4	0.67	0.73	0.70	96	5	0.72	0.71	0.71	96	6	0.80	0.67	0.73	98	7	0.62	0.70	0.66	100	accuracy			0.70	778	macro avg	0.71	0.70	0.70	778	weighted avg	0.71	0.70	0.70	778	<table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.64</td><td>0.66</td><td>0.65</td><td>98</td></tr> <tr><td>1</td><td>0.79</td><td>0.76</td><td>0.77</td><td>92</td></tr> <tr><td>2</td><td>0.62</td><td>0.65</td><td>0.64</td><td>100</td></tr> <tr><td>3</td><td>0.42</td><td>0.50</td><td>0.46</td><td>98</td></tr> <tr><td>4</td><td>0.54</td><td>0.56</td><td>0.55</td><td>96</td></tr> <tr><td>5</td><td>0.59</td><td>0.52</td><td>0.55</td><td>96</td></tr> <tr><td>6</td><td>0.62</td><td>0.62</td><td>0.62</td><td>98</td></tr> <tr><td>7</td><td>0.67</td><td>0.56</td><td>0.61</td><td>100</td></tr> <tr><td>accuracy</td><td></td><td></td><td>0.60</td><td>778</td></tr> <tr><td>macro avg</td><td>0.61</td><td>0.60</td><td>0.61</td><td>778</td></tr> <tr><td>weighted avg</td><td>0.61</td><td>0.60</td><td>0.61</td><td>778</td></tr> </tbody> </table>		precision	recall	f1-score	support	0	0.64	0.66	0.65	98	1	0.79	0.76	0.77	92	2	0.62	0.65	0.64	100	3	0.42	0.50	0.46	98	4	0.54	0.56	0.55	96	5	0.59	0.52	0.55	96	6	0.62	0.62	0.62	98	7	0.67	0.56	0.61	100	accuracy			0.60	778	macro avg	0.61	0.60	0.61	778	weighted avg	0.61	0.60	0.61	778
		precision	recall	f1-score	support																																																																																																																																																																																		
	0	0.91	0.63	0.75	98																																																																																																																																																																																		
1	0.85	0.80	0.83	92																																																																																																																																																																																			
2	0.86	0.68	0.76	100																																																																																																																																																																																			
3	0.55	0.63	0.59	98																																																																																																																																																																																			
4	0.70	0.76	0.73	96																																																																																																																																																																																			
5	0.60	0.78	0.68	96																																																																																																																																																																																			
6	0.90	0.70	0.79	98																																																																																																																																																																																			
7	0.66	0.83	0.74	100																																																																																																																																																																																			
accuracy			0.73	778																																																																																																																																																																																			
macro avg	0.75	0.73	0.73	778																																																																																																																																																																																			
weighted avg	0.75	0.73	0.73	778																																																																																																																																																																																			
	precision	recall	f1-score	support																																																																																																																																																																																			
0	0.91	0.72	0.81	98																																																																																																																																																																																			
1	0.72	0.77	0.75	92																																																																																																																																																																																			
2	0.77	0.66	0.71	100																																																																																																																																																																																			
3	0.50	0.61	0.55	98																																																																																																																																																																																			
4	0.67	0.73	0.70	96																																																																																																																																																																																			
5	0.72	0.71	0.71	96																																																																																																																																																																																			
6	0.80	0.67	0.73	98																																																																																																																																																																																			
7	0.62	0.70	0.66	100																																																																																																																																																																																			
accuracy			0.70	778																																																																																																																																																																																			
macro avg	0.71	0.70	0.70	778																																																																																																																																																																																			
weighted avg	0.71	0.70	0.70	778																																																																																																																																																																																			
	precision	recall	f1-score	support																																																																																																																																																																																			
0	0.64	0.66	0.65	98																																																																																																																																																																																			
1	0.79	0.76	0.77	92																																																																																																																																																																																			
2	0.62	0.65	0.64	100																																																																																																																																																																																			
3	0.42	0.50	0.46	98																																																																																																																																																																																			
4	0.54	0.56	0.55	96																																																																																																																																																																																			
5	0.59	0.52	0.55	96																																																																																																																																																																																			
6	0.62	0.62	0.62	98																																																																																																																																																																																			
7	0.67	0.56	0.61	100																																																																																																																																																																																			
accuracy			0.60	778																																																																																																																																																																																			
macro avg	0.61	0.60	0.61	778																																																																																																																																																																																			
weighted avg	0.61	0.60	0.61	778																																																																																																																																																																																			

**LSTM unit: 128**  
**Epoch: 30**  
**Batch size: 16**  
**Optimizer: Adam – 0.001 lr**  
**Scheduler: ReduceLRonPlateau (factor 0.2, patient 5)**



```
[[65  2  0 21  4  3  2  1]
 [ 0 82  2  5  1  1  0  1]
 [ 1  2 70  5  7  3  5  7]
 [ 3  4  0 68  4 11  2  6]
 [ 1  0  1  6 73  3  4  8]
 [ 0  4  0 13  4 68  3  4]
 [ 0  0  4  8  5  4 74  3]
 [ 4  3  1  8  6  1  1 76]]
```

	precision	recall	f1-score	support
0	0.88	0.66	0.76	98
1	0.85	0.89	0.87	92
2	0.90	0.70	0.79	100
3	0.51	0.69	0.59	98
4	0.70	0.76	0.73	96
5	0.72	0.71	0.72	96
6	0.81	0.76	0.78	98
7	0.72	0.76	0.74	100
accuracy			0.74	778
macro avg	0.76	0.74	0.75	778
weighted avg	0.76	0.74	0.74	778

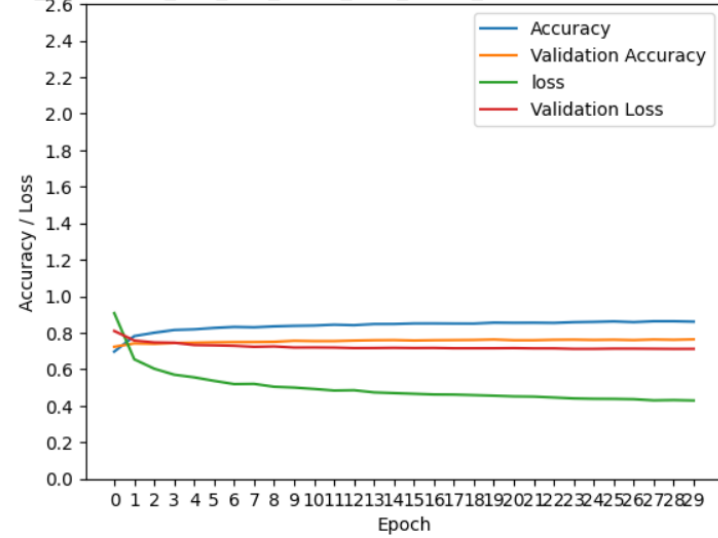


```
[[73  0  2  9  3  3  3  5]
 [ 1 77  3  4  3  3  0  1]
 [ 2  2 63  2 18  3  2  8]
 [ 5 10  2 56 10  5  1  9]
 [ 3  4  5  2 74  2  2  4]
 [ 0  3  3 12  4 66  6  2]
 [ 3  2  4  8  7  4 66  4]
 [ 1  5  4  7  4  5  1 73]]
```

	precision	recall	f1-score	support
0	0.83	0.74	0.78	98
1	0.75	0.84	0.79	92
2	0.73	0.63	0.68	100
3	0.56	0.57	0.57	98
4	0.60	0.77	0.68	96
5	0.73	0.69	0.71	96
6	0.81	0.67	0.74	98
7	0.69	0.73	0.71	100
accuracy			0.70	778
macro avg	0.71	0.71	0.71	778
weighted avg	0.71	0.70	0.71	778

**LSTM unit: 10**  
**Epoch: 30**  
**Batch size: 4**  
**Optimizer: Adam –**  
**0.001 lr (decay 0.001)**  
**Scheduler:**  
**ReduceLRonPlateau**  
**(factor 0.2, patient 5)**

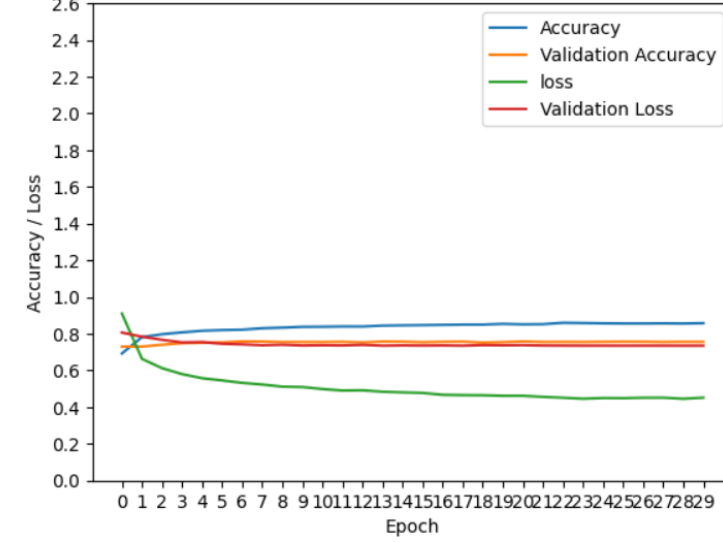
lstm\_dens201\_gavg8class\_128\_unit\_Adam\_with\_decay\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[72  2  4 12  4  2  1  1]
 [ 1 76  3  7  1  4  0  0]
 [ 1  2 74  2  7  2  7  5]
 [ 6  6  1 64  5  9  4  3]
 [ 4  0  3  3 75  1  3  7]
 [ 0  2  1 15  5 65  5  3]
 [ 1  0  6 10  3  5 70  3]
 [ 6  6  5  1  7  4  2 69]]
```

	precision	recall	f1-score	support
0	0.79	0.73	0.76	98
1	0.81	0.83	0.82	92
2	0.76	0.74	0.75	100
3	0.56	0.65	0.60	98
4	0.70	0.78	0.74	96
5	0.71	0.68	0.69	96
6	0.76	0.71	0.74	98
7	0.76	0.69	0.72	100
accuracy			0.73	778
macro avg	0.73	0.73	0.73	778
weighted avg	0.73	0.73	0.73	778

lstm\_mobV2\_gavg8class\_128\_unit\_Adam\_with\_decay\_ReduceLRonPlateau Accuracy-Loss Chart

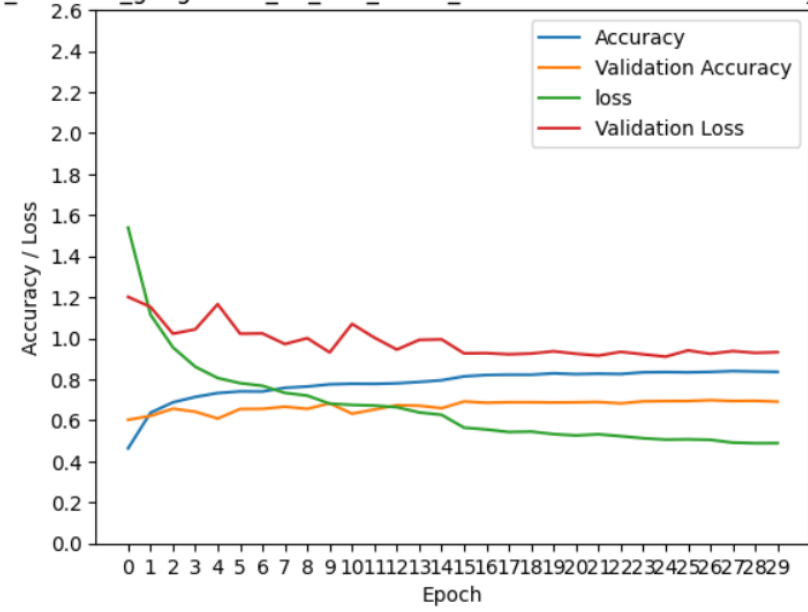


```
[[74  0  3 12  2  1  3  3]
 [ 0 77  3  8  1  3  0  0]
 [ 6  2 75  0  5  2  4  6]
 [ 9  8  2 58  5  2  6  8]
 [ 3  2  4  3 70  4  3  7]
 [ 1  4  3 15  6 64  1  2]
 [ 2  0  4 10  5  4 70  3]
 [ 2  5  6  4  5  3  3 72]]
```

	precision	recall	f1-score	support
0	0.76	0.76	0.76	98
1	0.79	0.84	0.81	92
2	0.75	0.75	0.75	100
3	0.53	0.59	0.56	98
4	0.71	0.73	0.72	96
5	0.77	0.67	0.72	96
6	0.78	0.71	0.74	98
7	0.71	0.72	0.72	100
accuracy			0.72	778
macro avg	0.72	0.72	0.72	778
weighted avg	0.72	0.72	0.72	778

**LSTM unit: 10**  
**Epoch: 30**  
**Batch size: 4**  
**Optimizer: Adam – 0.001 lr**  
**Scheduler: ReduceLRonPlateau (factor 0.2, patient 5)**

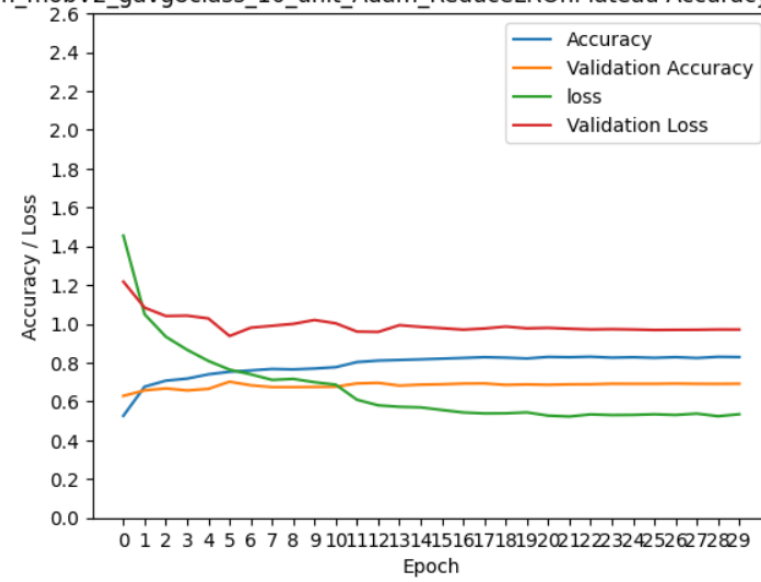
Istm\_dens201\_gavg8class\_10\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[72 0 1 10 4 4 3 4]
 [ 1 63 3 8 4 10 0 3]
 [ 5 1 60 2 19 3 5 5]
 [12 3 0 62 5 8 4 4]
 [ 7 0 1 4 73 3 5 3]
 [ 3 0 1 17 5 59 9 2]
 [ 6 0 3 6 4 4 73 2]
 [15 4 1 8 7 2 5 58]]
```

	precision	recall	f1-score	support
0	0.60	0.73	0.66	98
1	0.89	0.68	0.77	92
2	0.86	0.60	0.71	100
3	0.53	0.63	0.58	98
4	0.60	0.76	0.67	96
5	0.63	0.61	0.62	96
6	0.70	0.74	0.72	98
7	0.72	0.58	0.64	100
accuracy			0.67	778
macro avg	0.69	0.67	0.67	778
weighted avg	0.69	0.67	0.67	778

Istm\_mobV2\_gavg8class\_10\_unit\_Adam\_ReduceLRonPlateau Accuracy-Loss Chart



```
[[72 2 3 3 8 4 1 5]
 [ 0 75 4 2 3 4 0 4]
 [ 4 3 70 3 5 3 0 12]
 [ 7 18 4 29 7 21 3 9]
 [ 2 3 12 0 59 5 1 14]
 [ 2 6 6 1 3 73 0 5]
 [ 2 2 6 5 5 10 64 4]
 [ 2 8 4 1 4 7 1 73]]
```

	precision	recall	f1-score	support
0	0.79	0.73	0.76	98
1	0.64	0.82	0.72	92
2	0.64	0.70	0.67	100
3	0.66	0.30	0.41	98
4	0.63	0.61	0.62	96
5	0.57	0.76	0.65	96
6	0.91	0.65	0.76	98
7	0.58	0.73	0.65	100
accuracy			0.66	778
macro avg	0.68	0.66	0.66	778
weighted avg	0.68	0.66	0.65	778

**Appendix C: Final Year Project Weekly Report**

**FINAL YEAR PROJECT WEEKLY REPORT**

*(Project II)*

<b>Trimester, Year:</b> T3Y4	<b>Study week no.:</b> 4
<b>Student Name &amp; ID:</b> Peh Hong Bo & 19ACB06176	
<b>Supervisor:</b> Dr. Aun Yichiet	
<b>Project Title:</b> DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHTS USING HUMAN ACTIVITY RECOGNITION	

**1. WORK DONE**

[Please write the details of the work done in the last fortnight.]

- Revision on work done in FYP1.
- Explore development using docker.

**2. WORK TO BE DONE**

- Explore data streaming library and API coding.

**3. PROBLEMS ENCOUNTERED**

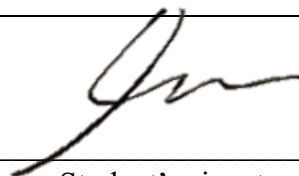
- NA

**4. SELF EVALUATION OF THE PROGRESS**

- Progress on track.



Supervisor's signature



Student's signature

## FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> T3Y4	<b>Study week no.:</b> 8
<b>Student Name &amp; ID:</b> Peh Hong Bo & 19ACB06176	
<b>Supervisor:</b> Dr. Aun Yichiet	
<b>Project Title:</b> DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHTS USING HUMAN ACTIVITY RECOGNITION	

### 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Explored on docker and API coding.
- Started writing the server program

### 2. WORK TO BE DONE

- Complete the server program
- Improve HAR model performance

### 3. PROBLEMS ENCOUNTERED

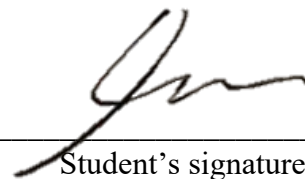
- Tried to retrain the network with 224x224 input but still running out of GPU RAM.

### 4. SELF EVALUATION OF THE PROGRESS

- Progress delayed.



Supervisor's signature



Student's signature

## FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> T3Y4	<b>Study week no.:</b> 11
<b>Student Name &amp; ID:</b> Peh Hong Bo & 19ACB06176	
<b>Supervisor:</b> Dr. Aun Yichiet	
<b>Project Title:</b> DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHTS USING HUMAN ACTIVITY RECOGNITION	

### 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Completed server docker and client program
- Trained LSTM with new CNN as feature extractor

### 2. WORK TO BE DONE

- Improve model performance
- Report writing

### 3. PROBLEMS ENCOUNTERED

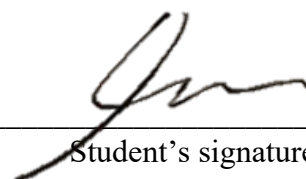
- NA

### 4. SELF EVALUATION OF THE PROGRESS

- Progress delayed




\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

**Appendix D: POSTER**




# DySUN

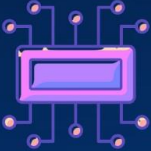
The current IoT light requires manual control through App or Voice Control. However, imagine a scene you are busying with your work untill you don't have a 3rd hand to open the app and activate Alexa.

## So, Why don't we just let the light control on its own?


**Methodology:**




Camera capture human activity



Microcontroller upload the images and control the light



Server receive the image and predict user action using HAR model



IoT light adjust based on the result

**Objective:**

- To train a multi-classes indoor activity recognition model using CNN-LSTM architecture
- To develop an automatic light calibrating technique on Yeelight framework based on human activity.

**Results:**

- Trained a DenseNet201-LSTM HAR model using Kinetic\_700\_2020 dataset, achieved accuracy at 74%
- Deployed a program on Jetson Nano to capture user action, stream images to server and configure the IoT light over LAN based using Yeelight library.
- Deployed a composite docker container for model prediction, user configuration website and database

# Activity Aware

# IoT Light

using Human Activity Recognition

Presented by: Peh Hong Bo  
Supervised by: Dr. Aun Yichiet



## PLAGIARISM CHECK RESULT

### Appendix E: PLAGIARISM CHECK RESULT

#### DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHT USING HUMAN ACTIVITY RECOGNITION

##### ORIGINALITY REPORT

<b>7</b> %	<b>5</b> %	<b>4</b> %	<b>1</b> %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

##### PRIMARY SOURCES

<b>1</b>	<a href="http://eprints.utar.edu.my">eprints.utar.edu.my</a> Internet Source	<b>1</b> %
<b>2</b>	Submitted to Universiti Tunku Abdul Rahman Student Paper	<b>&lt;1</b> %
<b>3</b>	<a href="http://mdpi-res.com">mdpi-res.com</a> Internet Source	<b>&lt;1</b> %
<b>4</b>	<a href="http://repositorio.pucrs.br">repositorio.pucrs.br</a> Internet Source	<b>&lt;1</b> %
<b>5</b>	Nursyazwani Adnan, Noorfazila Kamal, Kalaivani Chellappan. "An IoT Based Smart Lighting System Based on Human Activity", 2019 IEEE 14th Malaysia International Conference on Communication (MICC), 2019 Publication	<b>&lt;1</b> %
<b>6</b>	Tuhin Kumar Bera, Pinaki Pratim Acharjya. "chapter 8 An Investigation Into the Use of Deep Learning to Recognize Human Activity", IGI Global, 2023 Publication	<b>&lt;1</b> %

## PLAGIARISM CHECK RESULT

7	Keerthana Rangasamy, Muhammad Amir As'ari, Nur Azmina Rahmad, Nurul Fathiah Ghazali. "Hockey activity recognition using pre-trained deep learning model", ICT Express, 2020 Publication	<1 %
8	<a href="http://www.arxiv-vanity.com">www.arxiv-vanity.com</a> Internet Source	<1 %
9	<a href="http://www.mdpi.com">www.mdpi.com</a> Internet Source	<1 %
10	<a href="http://hdl.handle.net">hdl.handle.net</a> Internet Source	<1 %
11	Juarez Monteiro, Roger Granada, Rodrigo C. Barros, Felipe Meneguzzi. "Deep neural networks for kitchen activity recognition", 2017 International Joint Conference on Neural Networks (IJCNN), 2017 Publication	<1 %
12	<a href="http://link.springer.com">link.springer.com</a> Internet Source	<1 %
13	<a href="http://www.annauniv.edu">www.annauniv.edu</a> Internet Source	<1 %
14	Submitted to University of Lancaster Student Paper	<1 %
15	<a href="http://blog.csdn.net">blog.csdn.net</a> Internet Source	<1 %

## PLAGIARISM CHECK RESULT

16	Linghui Li, Sheng Tang, Junbo Guo, Rui Wang, Bo Lyu, Qi Tian, Yongdong Zhang. "Image Captioning Based on Adaptive Balancing Loss", 2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM), 2018 Publication	<1%
17	Submitted to University of Nottingham Student Paper	<1%
18	card-file.onaft.edu.ua Internet Source	<1%
19	dokumen.pub Internet Source	<1%
20	www.frontiersin.org Internet Source	<1%
21	Submitted to University of Bedfordshire Student Paper	<1%
22	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
23	Submitted to Heriot-Watt University Student Paper	<1%
24	Juarez Monteiro, Roger Granada, Jotao Paulo Aires, Rodrigo C. Barros. "Evaluating the Feasibility of Deep Learning for Action Recognition in Small Datasets", 2018	<1%

## PLAGIARISM CHECK RESULT

	<b>International Joint Conference on Neural Networks (IJCNN), 2018</b> Publication	
25	<b>community.native-instruments.com</b> Internet Source	<1 %
26	<b>scholar.google.com</b> Internet Source	<1 %
27	<b>Submitted to Harrisburg University of Science and Technology</b> Student Paper	<1 %
28	<b>Submitted to The Robert Gordon University</b> Student Paper	<1 %
29	<b>vslab.ia.ac.cn</b> Internet Source	<1 %
30	<b>web.archive.org</b> Internet Source	<1 %
31	<b>www.csl.uni-bremen.de</b> Internet Source	<1 %
32	<b>www.malrep.uum.edu.my</b> Internet Source	<1 %
33	<b>"Advances in Soft Computing and Machine Learning in Image Processing", Springer Science and Business Media LLC, 2018</b> Publication	<1 %

PLAGIARISM CHECK RESULT

34	Cheng-Hong Yang, Po-Yin Chang. "Forecasting the Demand for Container Throughput Using a Mixed-Precision Neural Architecture Based on CNN-LSTM", Mathematics, 2020 Publication	<1 %
35	Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Trevor Darrell, Kate Saenko. "Long-term recurrent convolutional networks for visual recognition and description", 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015 Publication	<1 %
36	Submitted to University of Sydney Student Paper	<1 %
37	catalog.lib.kyushu-u.ac.jp Internet Source	<1 %
38	jurnal.iaii.or.id Internet Source	<1 %
39	researchr.org Internet Source	<1 %
40	"Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. AI, Product and Service",	<1 %

PLAGIARISM CHECK RESULT

	Springer Science and Business Media LLC, 2021 Publication	
41	Jothi Letchumy Mahendra Kumar, Mamunur Rashid, Rabi Muazu Musa, Mohd Azraai Mohd Razman et al. "The classification of EEG-based wink signals: A CWT-Transfer Learning pipeline", ICT Express, 2021 Publication	<1 %
42	edepositireland.ie Internet Source	<1 %
43	export.arxiv.org Internet Source	<1 %
44	web-tools.uts.edu.au Internet Source	<1 %
45	Chirag Kumar, Ketan Rajawat. "Network Dissensus via Distributed ADMM", IEEE Transactions on Signal Processing, 2020 Publication	<1 %
46	Devansh Srivastav, Akansha Bajpai, Abhishek Singhal. "A Temporal Convolutional Neural Network Based Activity Recognition Model using a Real-Time Two-Dimensional Single Pose Estimation Framework", 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2022 Publication	<1 %

47	Elham Mottaghi, Mohammad-R. Akbarzadeh-T.. "Automatic Evaluation of Motor Rehabilitation Exercises Based on Deep Mixture Density Neural Networks", Journal of Biomedical Informatics, 2022 Publication	<1 %
48	Hailin Wang, Ke Qin, Guiduo Duan, Guangchun Luo. "Denoising Graph Inference Network for Document-Level Relation Extraction", Big Data Mining and Analytics, 2023 Publication	<1 %
49	M Abu, N A H Zahri, A Amir, M I Ismail, L M Kamarudin, H Nishizaki. "Classification of Multiple Visual Field Defects using Deep Learning", Journal of Physics: Conference Series, 2021 Publication	<1 %
50	Muhammad Ehatisham-Ul-Haq, Muhammad Awais Azam, Yasar Amin, Usman Naeem. "C2FHAR: Coarse-to-Fine Human Activity Recognition With Behavioral Context Modeling Using Smart Inertial Sensors", IEEE Access, 2020 Publication	<1 %
51	Sung Yong Chun, Chan-Su Lee. "Applications of Human Motion Tracking: Smart Lighting Control", 2013 IEEE Conference on Computer	<1 %

## PLAGIARISM CHECK RESULT

	<b>Vision and Pattern Recognition Workshops, 2013</b> Publication	
52	Submitted to University of Newcastle upon Tyne Student Paper	<1 %
53	Yasuo Yamao, Takehiko Oami, Jun Yamabe, Nozomi Takahashi, Taka-aki Nakada. "Machine learning model for predicting oliguria in critically ill patients", Research Square Platform LLC, 2023 Publication	<1 %
54	aquila.usm.edu Internet Source	<1 %
55	archive.org Internet Source	<1 %
56	arxiv.org Internet Source	<1 %
57	dokumen.tips Internet Source	<1 %
58	downloads.hindawi.com Internet Source	<1 %
59	eprints.utm.my Internet Source	<1 %
60	www.asashi.com.my Internet Source	<1 %



# PLAGIARISM CHECK RESULT

61 [www.nature.com](http://www.nature.com)  
Internet Source

<1%

---

Exclude quotes  On

Exclude matches  < 8 words

Exclude bibliography  On

PLAGIARISM CHECK RESULT

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



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	PEH HONG BO
<b>ID Number(s)</b>	19ACB06176
<b>Programme / Course</b>	Bachelor of Computer Science (Honours)
<b>Title of Final Year Project</b>	DYSUN - AN ACTIVITY-AWARE SMART IOT LIGHTS USING HUMAN ACTIVITY RECOGNITION

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)</b>
<b>Overall similarity index: <u>7</u> %</b>  <b>Similarity by source</b>  Internet Sources: <u>5</u> % Publications: <u>4</u> % Student Papers: <u>1</u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>0</u></b>	
<b>Parameters of originality required, and limits approved by UTAR are as Follows:</b> (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 <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

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

***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

\_\_\_\_\_  
Signature of Co-Supervisor

Name: Dr. Aun Yichiet

Name: \_\_\_\_\_

Date: 15 SEP 2023

Date: \_\_\_\_\_

**Appendix F: FYP 1 CHECKLIST**



**UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF INFORMATION & COMMUNICATION  
TECHNOLOGY (KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student ID	19ACB06176
Student Name	PEH HONG BO
Supervisor Name	Dr. Aun Yichiet

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.
/	Title Page
/	Signed Report Status Declaration Form
/	Signed FYP Thesis Submission Form
/	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)
/	Weekly Log
/	Poster
/	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
/	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

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

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 14 SEP 2023

