

DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK

BY

LEONG WAI CHUN

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 2021

REPORT STATUS DECLARATION FORM

Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL
NETWORK

Academic Session: 202105

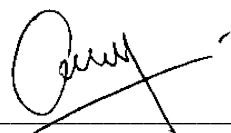
I LEONG WAI CHUN
(CAPITAL LETTER)

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.


(Author's signature)

Verified by,


(Supervisor's signature)

Address:

16, JALAN BAHAGIA MAKMUR 3,
TAMAN BAHAGIA MAKMUR,
28000 TEMERLOH, PAHANG

Ts. Dr. Cheng Wai Khuen

Supervisor's name

Date: 2nd of SEPTEMBER 2021

Date: 3/9/2021

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

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 2nd of September 2021

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Leong Wai Chun (ID No: 18ACB06442) has completed this final year project entitled “DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK” under the supervision of Dr. Cheng Wai Khuen (Supervisor) from the Department of Computer Science, 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,



(*Leong Wai Chun*)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK**” 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 : Leong Wai Chun

Date : 2nd September 2021

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. Cheng Wai Khuen who have given me a great opportunity to involve in the Distributed Computing System field study. Other than that, he has given me a lot of guidance in order to complete the project. When I was facing some problems in the project, Dr. Cheng would always give me some advice which assists me in order to overcome the problem that I'm facing. Once again, a million thanks to my supervisor.

ABSTRACT

This project is regarding to the problem of dog loneliness. Dog can be lonely when the dog's owner left the dogs alone at home without understand what the dogs will feel when they are at home alone. Dog might be depressed due to the feel of loneliness being left alone at home. The major problem that will be focus on this project will be the dog loneliness and the problem of dog does not have any other dog companion to connect with each other when they are lonely. As most of the existing product unable to solve one of the problems which will be the dog's emotion. The existing product unable to detect that what does the dog felt when they are left alone. The only feature that the existing product offers is connected 2 parties with each other which is between dog's owner and the dog. Some researches and literature reviews will be carried out to look deeper into how a machine can able to understand the dog's emotion to solve the dog loneliness problem by using image recognition and sound recognition method. The reviews of existing products including will be Pawbo+, PetChatz, Furbo Dog Camera, Petcube Bites Treat Camera, and Petzi Treat Cam are written in the literature review section of this report. After reviewing the strengths and weakness of the existing products above, a clearer vision will be captured on the challenges will be presented in this report. The proposed solution of this project is to develop a dog social network which able to connect 2 dogs together to communicate with each other and understand how the dog feel by developing a model which will recognize the dog's emotion to tell the dog owner that how does the dog feel when they are left alone. The development technologies and tools involve in this project will be Google Cloud Platform, Android Mobile, Keras, Jupyter Notebook, Audacity, Python, Node.js and Android Kotlin language for program implementation.

TABLE OF CONTENTS

TITLE PAGE.....	i
REPORT STATUS DECLARATION FORM.....	ii
SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS.....	iii
DECLARATION OF ORIGINALITY.....	iv
ACKNOWLEDGEMENT.....	v
ABSTRACT.....	vi
TABLE OF CONTENTS.....	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Statement and motivation	4
1.3 Project Scope	5
1.4 Project Objectives	6
1.5 Impact, Significance and Contributions	7
CHAPTER 2: LITERATURE REVIEW	8
2.1 System Review	8
2.1.1 Pawbo+	8
2.1.2 PetChatz	10
2.1.3 Furbo Dog Camera	12
2.1.4 Petcube Bites Treat Camera	14
2.1.5 Petzi Treat Cam	16
2.2 Comparison between the existing system and proposed system	18
2.3 Recommendations	19
CHAPTER 3: METHODS/TECHNOLOGIES INVOLVED	21
3.1 Methodology.....	21
3.2 Project Workflow in Phased Development.....	22
3.3 Technologies Involved	25
3.4 System Design	27
3.4.1 Google Cloud Architecture Diagram	27
3.4.2 Use Case Diagram	28
3.4.3 Activity Diagram.....	29

3.5 Implementation Issues and Challenges	36
3.6 Project Timeline	38
CHAPTER 4: PRELIMINARY WORK	41
4.1 Dog Image Emotion Recognition Model	41
4.2 Dog Bark Emotion Recognition Model.....	53
4.3 Combining Dog Image Emotion Classifier and Dog Bark Audio Classifier Model	62
4.4 Google Cloud Platform.....	68
4.5 Dog Social Media Mobile Application	76
4.6 Data Visualization on Google Data Studio	90
CHAPTER 5: CONCLUSION.....	93
BIBLIOGRAPHY	95
FINAL YEAR PROJECT WEEKLY REPORT	97
POSTER.....	102
PLAGIARISM CHECK RESULT	103

LIST OF TABLES

Table 2.1 Comparison between existing system and proposed system.....	18
---	----

LIST OF FIGURES

Figure 1.1 The Statistic of Pet Dog and Cat from 2014 till 2018 in Malaysia...	1
Figure 1.2 The Statistic of Hours Worked Per Week from Year 2015 till 2018 in Malaysia.....	2
Figure 1.3 Shows an Article about “Dogs who bite aren’t barking mad... they’re just depressed”.....	3
Figure 2.1 Pawbo+.....	8
Figure 2.2 PetChatz.....	10
Figure 2.3 Furbo Dog Camera.....	12
Figure 2.4 Petcube Bites Treat Camera.....	14
Figure 2.5 Petzi Treat Cam.....	16
Figure 3.1 Phased Development Methodology.....	21
Figure 3.2 Project Workflow in Phased Development.....	22
Figure 3.3 Architecture Diagram of Proposed Solution.....	27
Figure 3.4 Use Case Diagram.....	28
Figure 3.5 Activity Diagram of Create Account.....	29
Figure 3.6 Activity Diagram of Add Friend.....	30
Figure 3.7 Activity Diagram of Start RTMP Streaming.....	31
Figure 3.8 Activity Diagram of Call Friend.....	34
Figure 3.9 Activity Diagram of View Report.....	35
Figure 3.10 Gantt Chart 1.....	38
Figure 3.11 Gantt Chart 2.....	39
Figure 3.12 Gantt Chart 3.....	40
Figure 4.1 Preparing Image Dataset with Python Script.....	41
Figure 4.2 Resizing Dog Images with Python Script.....	42
Figure 4.3 Display 5 Sample Images for Each Label.....	42
Figure 4.4 Loading, Pre-Processing and Splitting the Dataset.....	43
Figure 4.5 Building a ResNet-like Model.....	44
Figure 4.6 Less Data And More Data Dataset Split Into Training, Validating, Testing Data.....	44
Figure 4.7 Training Prediction Result for Less Data.....	45
Figure 4.8 Training Prediction Result for More Data.....	45

Figure 4.9 Testing Dataset Prediction Result for Less Data.....	45
Figure 4.10 Testing Dataset Prediction Result for More Data.....	46
Figure 4.11 Epoch and Learning Rate Graph Result with Batch Size 16.....	47
Figure 4.12 Epoch and Learning Rate Graph Result with Batch Size 32.....	47
Figure 4.13 Epoch and Prediction Graph Result with Batch Size 16.....	48
Figure 4.14 Epoch and Prediction Graph Result with Batch Size 32.....	48
Figure 4.15 Prediction Result on Test Data of the best model with Batch Size 16.....	49
Figure 4.16 Prediction Result on Test Data of the best model with Batch Size 32.....	49
Figure 4.17 Block of Code That Create VGG16 Model.....	49
Figure 4.18 Epoch and Prediction Graph Result with Batch Size 16.....	50
Figure 4.19 Epoch and Prediction Graph Result with Batch Size 32.....	50
Figure 4.20 Prediction Result on Test Data of the best model with Batch Size 16.....	51
Figure 4.21 Prediction Result on Test Data of the best model with Batch Size 32.....	51
Figure 4.22 Prediction Result on Ricky the Dog Image with the Chosen Model.....	52
Figure 4.23 Dog Bark Audio Dataset in Google AudioSet.....	53
Figure 4.24 Dog Bark Audio Dataset From Google AudioSet.....	54
Figure 4.25 Dog Bark Audio Dataset Class Label Indices.....	54
Figure 4.26 Using Grep to Extract Rows Containing Growling Class Label Indices.....	55
Figure 4.27 Python Script to Gather Dog Audio Dataset.....	55
Figure 4.28 Python Script to Convert Audio File into WAV Format.....	56
Figure 4.29 Using Audacity to Extract Specific Dog Bark Audio Spectrum..	57
Figure 4.30 Different Audio Spectrum for each class label.....	57
Figure 4.31 Code to Extract Audio Features.....	58
Figure 4.32 Load Into Pandas and Separate into 2 Independent Dataset.....	58
Figure 4.33 Code to Convert Class Labels into Numeric Form Using Label Encoder.....	59
Figure 4.34 Code to Build Sequential Model.....	59

Figure 4.35 Finding Best Epoch for Dog Bark Classification model.....	60
Figure 4.36 Test Accuracy Result for Dog Bark Classifier Model.....	60
Figure 4.37 Prediction Result on Test Data using Trained Dog Bark Classifier Model.....	61
Figure 4.38 Loading 2 Models.....	62
Figure 4.39 Functions Used to Do Prediction.....	62
Figure 4.40 Functions Used to Do Weighted Average Calculation.....	63
Figure 4.41 Prediction Result Before and After Weighted Average.....	63
Figure 4.42 Prediction Accuracy and Class Label Graphs Before Weighted Average.....	64
Figure 4.43 Prediction Accuracy and Class Label Graphs After Weighted Average.....	64
Figure 4.44 Prediction Accuracy For Dog Image Emotion Classifier.....	65
Figure 4.45 Prediction Accuracy For Dog Bark Audio Classifier.....	65
Figure 4.46 Prediction Accuracy After Weighted Average Both Models.....	65
Figure 4.47 Prediction Accuracy Graph Before and After Weighted Average...	66
Figure 4.48 Predicted Label Using Weighted Average Technique.....	67
Figure 4.49 Linux VM Instance Profile.....	68
Figure 4.50 Dog Emotion Recognition Project Directory in Linux VM GCP Instance.....	68
Figure 4.51 Dog Emotion Recognition Model Directory in Linux VM GCP Instance.....	68
Figure 4.52 Dog Emotion Recognition Python Script in Linux VM GCP Instance.....	69
Figure 4.53 Dog Emotion Recognition Python Script Output Detecting Ricky Dog Image and Barking Audio.....	69
Figure 4.54 REST API Directory in Linux VM GCP Instance.....	70
Figure 4.55 Audios Directory in Linux VM GCP Instance.....	70
Figure 4.56 Images Directory in Linux VM GCP Instance.....	70
Figure 4.57 Block of Codes for Basic Dog Social Network in Node.js REST API JavaScript.....	71
Figure 4.58 Block of Codes for Upload Audio File in Node.js REST API JavaScript.....	71

Figure 4.59 Block of Codes for Upload Image File in Node.js REST API JavaScript.....	72
Figure 4.60 Block of Codes for Executing DogEmotionClassifier Python Script in Node.js REST API JavaScript.....	72
Figure 4.61 Block of Codes for Inserting Prediction Label into Database in Node.js REST API JavaScript.....	73
Figure 4.62 Block of Codes for Checking Sick Label in Node.js REST API JavaScript.....	74
Figure 4.63 Block of Codes for Getting Reports in Node.js REST API JavaScript.....	75
Figure 4.64 Login Interface of Dog Social Media.....	76
Figure 4.65 SignUp Interface of Dog Social Media.....	77
Figure 4.66 Main Menu Interface of Dog Social Media.....	77
Figure 4.67 User Profile Interface of Dog Social Media.....	78
Figure 4.68 Edit Profile Details Interface of Dog Social Media.....	79
Figure 4.69 Add Friend Interface of Dog Social Media.....	80
Figure 4.70 Start Server Interface of Dog Social Media.....	80
Figure 4.71 Demo Starting Server of Dog Social Media.....	81
Figure 4.72 Demo Swapping Camera of Dog Social Media.....	82
Figure 4.73 Block of Code Creating Floating Window.....	82
Figure 4.74 Block of Code Making Floating Window Draggable.....	83
Figure 4.75 Block of Code Starting/Stopping RTMP Server.....	83
Figure 4.76 Block of Code Swapping Camera.....	84
Figure 4.77 Friend Listing Interface in Dog Social Media.....	84
Figure 4.78 View Friend Details Interface in Dog Social Media	85
Figure 4.79 Calling Interface in Dog Social Media	86
Figure 4.80 Demo Dog Predicted Sick in Calling Interface	87
Figure 4.81 Demo Email Received by Micky’s Owner	87
Figure 4.82 Report Listing Interface in Dog Social Media	88
Figure 4.83 View Report Details Interface in Dog Social Media	89
Figure 4.84 View Number of Users in Data Studio	90
Figure 4.85 Dog Emotion Predicted By Friend Analysis in Data Studio	91
Figure 4.86 Demo Applying Filter During Analysis in Data Studio	92

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>RAD</i>	Rapid Application Development
<i>IoT</i>	Internet of Thing
<i>VM</i>	Virtual Machine
<i>USB</i>	Universal Serial Bus
<i>API</i>	Application Programming Interface
<i>CNN</i>	Convolutional Neural Network
<i>GCP</i>	Google Cloud Platform

CHAPTER 1: INTRODUCTION

1.1 Background Information

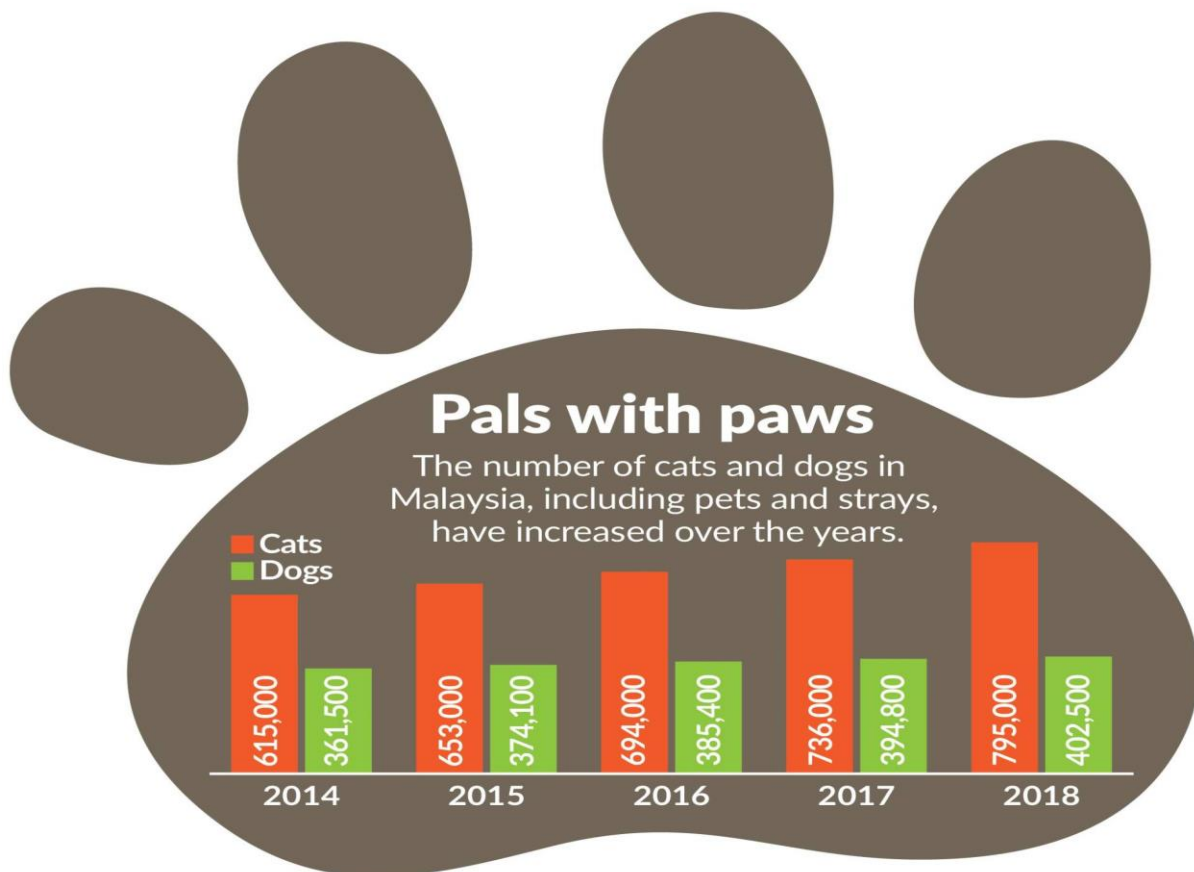


Figure 1.1 The Statistic of Pet Dog and Cat from 2014 till 2018 in Malaysia (Pals with paws, 2020)

In Malaysia based on figure 1.1 statistic, we can see that more people starting to adopt a pet dog because of their cuteness or as a companion to the people. Dogs can be a companion of human too where every day the dog owner come back from work, their pet dog will welcome their owner as this will decrease the loneliness of humans. But, there is a problem where the dog owner did not realize that the dog might felt lonely too when the owner has gone for work. (Pet peeves in animal welfare, 2020)

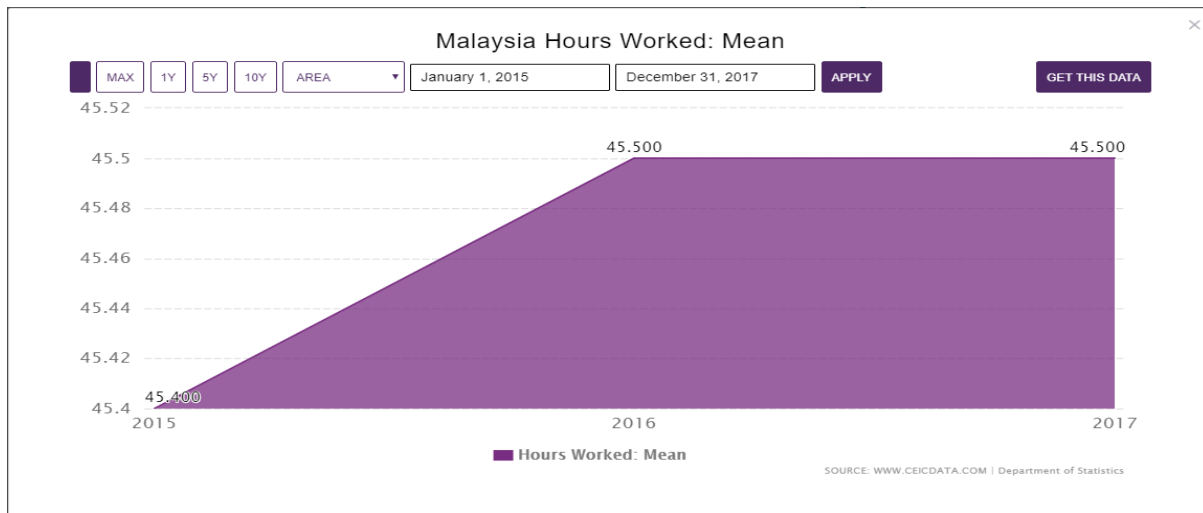


Figure 1.2 The Statistic of Hours Worked Per Week from Year 2015 till 2018 in Malaysia (Malaysia Hours Worked Mean, 2020)

Based on figure 1.2 shows that Malaysian working hours per weeks are increasing from year 2015, it shows that the time that they pet owner spent with their pet dog had become shorten too. By average the owner will spend 9 hours in work which roughly from 9am till 6pm and from travelling back from the work place till the owner’s accommodation it might takes roughly 2 hours because of heavy traffic jam. The time the owner spent with the pet dog each day will be average 2 hours only per day as the owner of the dog also need to enjoy their private lifestyle too. So the dogs will be left roughly 11hours alone where the dog do not have any companions play with it. This will cause the dog feel lonely without the owner and even a dog companion. A lonely dog will start to feel unhappy and they would not engage with their owners too. The dog will start to show these symptoms when they are unhappy which they will lick or bite their own fur and causing their hair loss or even damaged. The dog will sleep more than normal during the daytime because most of the daytime they are left alone due to owner working that time and caused the dog to feel bored and lonely. Dog appetite will reduce that might cause the dog to felt ill due to lack of foods intake. The dog will start to urine everywhere around the house too when they are unhappy to get their owner’s attention. A lonely dog might fall into depression that might really affects the dog health. The dog will become inactive and their eating and sleeping habits often change too. The dog would not participate in the things they once enjoyed too. A dog lover would not want this happen to their dog too right. (Malaysia | Hours Worked: Mean | Economic Indicators, 2020)

The screenshot shows the MailOnline website interface. At the top, the logo 'MailOnline' is on the left, and 'Science & Tech' is on the right. Below the logo is a navigation bar with links: Home | News | U.S. | Sport | TV&Showbiz | Australia | Femail | Health | **Science** | Money | Video | Travel | DailyMailTV | Discounts. Below this is a 'Latest Headlines' bar with a 'Login' link on the right. The main content area features an advertisement placeholder on the left. The article title is 'Dogs who bite aren't barking mad... they're just depressed'. Below the title, it says 'By DAVID DERBYSHIRE FOR MAILONLINE' and 'UPDATED: 07:56 GMT, 19 July 2010'. There are social media sharing buttons for Facebook, Twitter, Pinterest, and others, along with a 'View comments' link showing 23 comments. To the right of the article is another advertisement placeholder and a search bar. Below the search bar are more social media links for 'Like Daily Mail', 'Follow Daily Mail', 'Follow @DailyMail', 'Follow @dailymailtech', and 'Follow Daily Mail'. At the bottom right, there are links to 'Download our iPhone app' and 'Download our Android app'. A 'Today's headlines' section is also visible, featuring a headline about 'Incredible moment archaeologists discover 200 ancient Roman amphorae 'used to store wine' in underwater cave...'.

Figure 1.3 Shows an Article about “Dogs who bite aren’t barking mad... they’re just depressed”

Based on an article that posted by David Derbyshire for MailOnline he mentioned that dogs that bite are not actually barking mad but actually the dog are just depressed. Due to the depression of dogs the dog will become aggressive towards human which is one of the most frequent behavior problem in dogs. There are 3,800 patients that bitten by dogs had been treated by the NHS every year. During the research conducted, the researcher found that from 19 samples of dog’s blood snappy, angry animals had lower concentrations of serotonin. The lowest concentration of serotonin reading came from dogs whose anti-social behavior appeared to be an attempt at self-defending themselves. Snappiest dogs also had a higher levels of stress hormone cortisol as the researcher’s report in journal Applied Animal Behavior Science mentioned. The researchers hope the findings will make it be easier to diagnose the dog’s depression and treat them with anti-depressants. The vets believe that dogs are vulnerable to depression due to lack of time spending with owner and had been left alone for hours each day. So rather than letting the dogs get depressed and use anti-depressants to reduce the stress level in dogs, it is better that we prevent the dog fell into depression in the first place. (Derbyshire, 2020)

1.2 Problem Statement and motivation

According to recent research, we found out that there are something that lacks in the existing products in the market.

- **Unable to determine the emotion of dog**

The dog will feel lonely when the owner left the dog alone at home, but the problem is that the owner unable to understand that is their dog feel lonely when they are left alone when the owner went to work during the day time. One of the problem is that the dog's owner unable to identify that the dog is lonely since the dog's owner cannot determine that what does the emotion is , like example is the dog happy right now or is the dog sad right now.

- **Dog loneliness is another problem that caused dog to depress**

Loneliness in dog is a huge problem that will caused the dog to be depress too. One of the main problem of dog being lonely is that it's due to the dog lack of a companion to play with them. The dog owner is one of the dog's companion as the owner will play some game with their dogs during the leisure time. So if the dog's owner left their dog alone at home, this will caused the dog loss a precious companion to play with and caused the dog become lonely.

In order to solve the problems that mentioned in the problem statement, we will be doing some research on how to understand the dog's emotion based on the face expression that shown by the dog. In existing products that offer in the market, they did not offer a feature which able to tell the dog's owner that how does the dog feel when they are left alone at home. So, this is the main cause that the dog's owner unable to solve the dog's loneliness and caused the dog to feel depressed because they do not even know that their dog is feeling depressed due to loneliness. The dog's owner will spent most of the time in their work, so the dog will be left at home alone. The only things that the dog's owner know is that their dog greet them when they are home and their home are safe from any intruder since the dog are left at home guarding the house security. But here's the problem came, the dog owner unable to clarify that why their dog does not have the appetite to eat their meal, or even does not have any interest to the things

CHAPTER 1 INTRODUCTION

that the dog interested on previously. The dog monitoring system that offers in the market only let the dog's owner to monitor the dog without telling the dog's owner how the dog felt when they are left alone. Dog's owner might not able to understand the dog's feeling since the dog's owner does not watch their dog 24 hours all the time. So by developing a dog emotion recognition, dog owner will be able to understand their dog more since they will be able to know when the dog will be happy or even sad when they are not around.

Another problem to solve is the dog's loneliness. Dogs felt lonely when they are left alone without able to do anything. In most of the family these days, they only have the time to take care of 1 dog in their home. Having many dogs might caused a lot of trouble to the dog's owner too, as the dog's owner need to buy more foods for the dogs. This might increase the expenses of the dog's owner too. In order to solve this problem, a dog social network platform will be developed to solve the problem. The dog will be able to communicate to another dog or even their owner to eliminate the feel of loneliness as they will have another companion to talk when they are bored. This will solve the depression issue of lonely dogs.

1.3 Project Scope

From the problem statement stated above, it seems like the dog's owner does not have enough time to spent with their pet dog which caused them to falls into depression. So the scope of this project is to develop a smart environment for dog social network. With this smart environment, dogs are able to communicate with each other remotely through the dog social network. The dog's owner does not need to worry about their dog might falls to depression state due to loneliness. This project will be cover more in using Artificial Intelligence technique to study the behavior of dog through visualization and sound recognition. In visualization, it will recognize the face expression the dog emitted and for the sound recognition, it will be used to improve dog emotion recognition results to get a better prediction result. the A distributed system will be built in the project which let the dogs to communicate with each other remotely through the dog social network platform. This project will cover the samples of dog population within Malaysia because the behavior of the dog will be affected by the Malaysian culture. The main objective of this project is to provide a social network for the dogs to communicate with each other, to use visualization tools and sound recognition to identify the emotion of the dog.

1.4 Project Objectives

- **Provide a social network for the dogs**

The main objective is to provide a social network for the dogs to communicate with each other remotely by using the distributed system architecture. The dogs will use the provided platform to communicate with another dogs. In this social network only the initial part which is starting the program and connect to the remote RTMP Server of another dog, then the dog will be able to talk with the other dog directly. This will let the dogs to talk to another dog freely when they want to chit chat with another dog when they are bored.

- **Using visualization tools and sound recognition to identify the emotion of the dog**

Another objective to use visualization tools like image recognition tools with the help of Artificial Intelligence in order to identify the emotion of the dog. In this objective it will be divided to several sub-objectives which will includes on studying on the dog behavior by collecting images of dogs with different emotion and separate them into different classes based on the dog's emotion, train a model to fit the sample dataset and able to identify the behavior of dog from the trained model. First sub-objective is to find a dataset that have a lot of samples of dog behaviors that will be useful to the project and try to study on the dataset like the relationship between the attributes. Second sub-objective is to find the best model and tweak the model to fit the sample dataset. The third sub-objective is to use the trained model to identify the dog behavior and perform some actions based on the behavior of the dog. The behavior of dogs can be the dog expression. If the dog has a happy expression when chit chatting with another partner dog, the system will show that the other dog emotion like they are happy, angry, or sick. As for sound recognition, this will also perform dog emotion recognition, but the result will be combined with the image recognition result to get a better prediction on the dog's emotion.

1.5 Impact, Significance and Contributions

Based on the social perspective, this project will be contributed in the social to prevent the dog falls into depression. By preventing the dogs from falling into depression, we can solve several cases like depressed dog biting innocent people, causing the owner of the dog more trouble due to the depression of the dog and prevent the society from hating and hurting the dogs due to the dog bite people due to depression. By having this social network platform, we can create a healthy environment for the dogs to spend their leisure time with others dog by communicating through our smart environment dog social network while the owner of the dog is not around.

Based on the system perspective, this project will use the Artificial Intelligence installed on the cloud to learn the dog behavior. In this project it will store the trained model in the cloud, then use camera to capture the dog and using microphone to record the barking of the dog emitted and sent to the cloud, then in the cloud it will used the image of the dog and perform analyzation to analyze the behavior of the dog then the barking voice of the dog will be analyzing to improve the prediction result of by combining with the image recognition prediction result. After analyze, the cloud will send the result back to the devices and perform the following actions.

CHAPTER 2: LITERATURE REVIEW

2.1 System Review

2.1.1 Pawbo+



Figure 2.1 Pawbo+

Pawbo+ is a device which it let the pet owner to stay connected with their pet all the time. The device has Wi-Fi Interactive Pet Camera which the user can use it to interact with their pet when there is internet connection available and the treat dispenser which used to dump

CHAPTER 2 LITERATURE REVIEW

treat to out from the device when the user presses a button on their smartphone. The camera that Pawbo+ used can record 720p HD live video with 130° wide-angle lens that let the user to capture their beloved pet precious moment anytime anywhere. The Pawbo Life App supports Android and IOS devices so the user can download the apps without worrying about compatibility issues if they updated their smartphone operating system to the latest version. The Pawbo+ can permit up to 8 members to connect to the camera at once which is a perfect method to let the family members to check their pet in the same time throughout the day. Pawbo+ have a special feature which is the built-in light-pointer function that let the pet to chase over the red dot when the user presses a button in the Pawbo Life App with their smartphone. If the user wants to treat the pet, the user can just press a button to dump treat to their pet through the Pawbo+ device. (Pawbo+ Review , 2020)

Strengths

Pawbo+ have two-way communication which lets the pet owner to communicate with the pet with Pawbo Life App and at the same time can hear the pet talk to them with their pet language directly from Pawbo+ built-in microphone. This is a great feature not only talking to your pet in one way but let you hear what your pet trying to response you too. Pawbo+ has another great feature which is the instant social sharing which allow the pet owner to quickly snap a photo of the pet with the Snapshot Function and share the image of the pet owner's pet directly to Facebook, Twitter and even Instagram.

Weaknesses

Although Pawbo+ is a great device but there is some problem with it too. Pawbo+ has a problem with the video quality which is little underwhelming when it compares with other pet devices. A lot of the details has been lost in the shadows so if the pet owner owned a dark color pet or even hide somewhere around dark furniture, it going to be difficult to spot the pet. Another problem will be the light performance which is very poor when it compares to other pet devices. The camera sensor for Pawbo+ also have some problem with LED lighting too which tend to cause some area to become dark. If the pet falls into that dark area zone, it would be a problem for the pet owner to spot their pet too. Pawbo+ only can connect the device through Wi-Fi connection only and it does not have any Ethernet port to connect the devices

with Ethernet cable. In order to prevent any internet connection problem, Pawbo+ must be place somewhere near from the access point. User need to access the Pawbo+ either from the QR code on the Pawbo+ device or on the same Wi-Fi network with the Pawbo+ device. There is no user account that let the user to access Pawbo+ device camera. So if the user has a new smartphone when it went to holiday and forgot to install Pawbo Life app, the user would not be able to check up their pet.

2.1.2 PetChatz

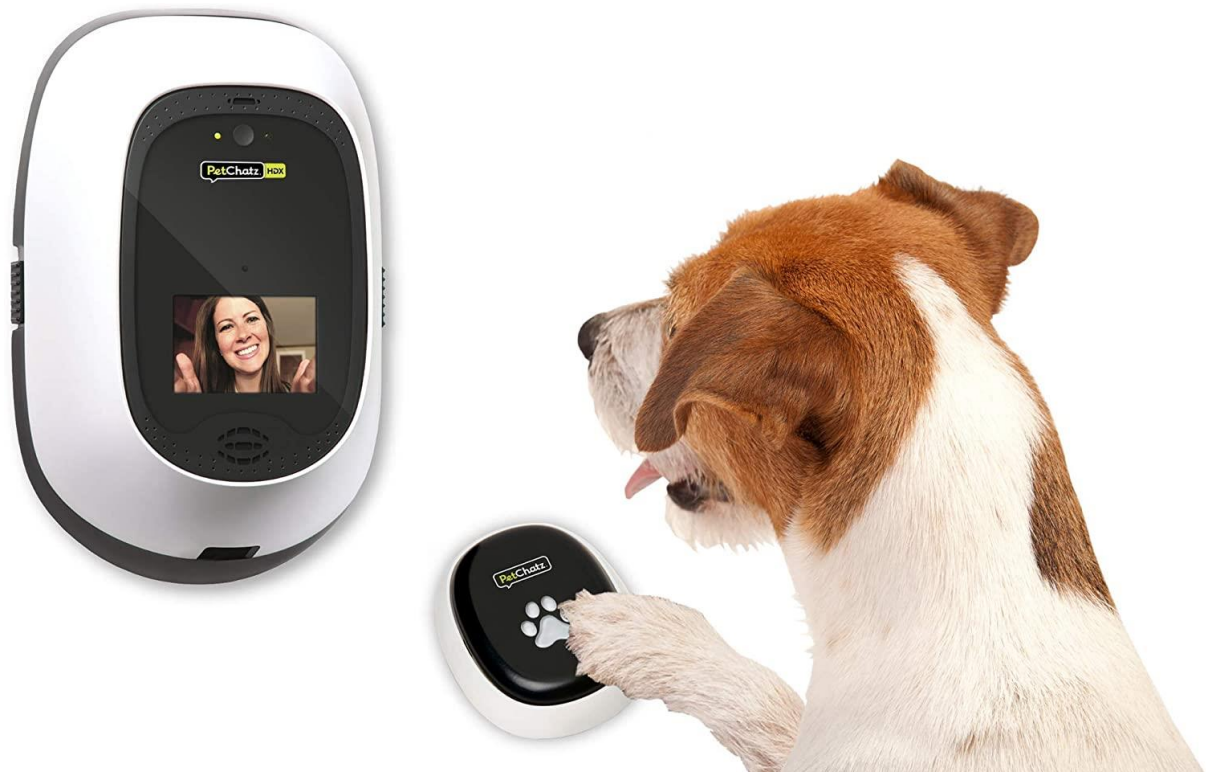


Figure 2.2 PetChatz

PetChatz is a pet interaction device that let the pet owner to interact with their lonely pet full day without any monthly fees. PetChatz do support two-way video chats which let the pet owner to communicate with their pet and in the same time able to listen what does the pet wanted to tell the owner though the built-in microphone on PetChatz device. PetChatz also do support treat dispense that will dump treat to the pet when the owner wanted to treat their beloved pet when they are bored, brain games which will treat the pet when they win the game, calming aromatherapy that used to make the pet relax. PetChatz do have another optional accessory that is sold separately which is the PawCall accessory. Pet can use their paw to press on the button on PawCall to call their owner. PawCall are placed on the floor or wall near to the PetChatz device. (A Home Alone Pet Needs This Practical Solution Petchatz®, 2020)

CHAPTER 2 LITERATURE REVIEW

Strengths

PetChatz support two-way communication which let the pet able to hear the voice of their owner in the same time the owner can hear the voice of their beloved pet too. This can solve the pet loneliness when the pet owner is not around. If the pet need to call their owner, they can use their paw and press on the PawCall. It will call the owner directly when it has been pressed. The owner just need to train the pet to press on the PawCall button only in order to contact the owner. PetChatz allow the owner to stay connect with their pet too and try to provide comfort to their pet when they are working or going out for vacation and leaving their pet alone at home. The design of PetChatz is pet friendly too that allow the pet owner to capture memories of the owner's pet by recording, taking videos or even pictures. PetChatz do allow the pet owners to contact with their pet without worrying any limitation on chat times or even hidden monthly fees.

Weaknesses

Although PetChatz have a lot of benefits but it is not waterproof. So the device can be place indoor only. PetChatz do lack of a features which supports night vision. If the pet wanted to contact with their owner but it is night time and the lights is off. Pet would not be able to switch on the light right, so the owner would not be able to see their beloved pet, they only able to hear their pet voice only which is a disadvantage to both pet and pet owner. The camera that PetChatz use is low resolution too, so the owner might see their pet in blur resolution condition which would not let the pet owner enjoy the interaction with their pet. Another problem is that the sound quality is a bit of lacking too. Sometimes the dog sound will be a little bit blur, this might due to poor microphone quality has been used. Another problem is that PetChatz unable to let the pet owner to directly share the picture or videos that they capture directly to social media. PetChatz and PawCall is sold separately which does not come within the same box. The owner might need to spent another few bucks to buy a PawCall just to let the pet to be able to call their owner.

2.1.3 Furbo Dog Camera



Figure 2.3 Furbo Dog Camera

Furbo Dog Camera is specially designed for dog owners in mind, which give the dog owner the ability to check on their dog and interact with their dog when they are away from home. This device equipped with two-way audio which let the dog owner to hear and talk to their beloved dog. This device is equipped with 1080p HD video capability which can record the video much clearer. This device does support infrared night vision too which let the dog owner to checkup with their dog when the room it is dark. Furbo Dog Camera do compactible with Alexa too which can add voice control by combining with a screen Alexa device. Furbo Dog Camera do support a feature which can toss treat to the dog when the dog owner press on a button on their smartphone app. Furbo Dog Camera have a real-time updates features that will update the status of the dog on the dog owner smartphone. If Furbo Dog Camera detected that the dog is barking, it will alert the dog owner's smartphone which the dog is barking and do the dog owner wanted to checkup with why the dog is barking or not. Furbo Dog Camera support an interesting feature that will alert the dog when it is activity time, the owner wanted to take a selfie of the dog, or even a person alerts too. Furbo Dog Camera have a dog-friendly color signal too which will switch between the two color that the dog can see which is yellow

CHAPTER 2 LITERATURE REVIEW

and blue to get the dog attention. Furbo Dog Camera will produce a clicking sound to alert the dog that the owner is tossing treats to the dog. (Barrington, 2020)

Strengths

Furbo Dog Camera is specially designed for dog, so it is more dog-friendly. The color signal that it used are also dog user friendly too which they know that dog able to recognize two color only which is blue and yellow. Furbo Dog Camera are easy to set up, what the dog owner need to do is just connect the smartphone app with Bluetooth or even Wi-Fi and it will start sending real-time alerts and push notification about the status of the dog. Another interesting feature that Furbo Dog Camera has which is the infrared night vision which let the pet owner to check and see with their dog even when the room is dark. So day and night is not going to be a problem to the pet owner. They can just interact with their dog anytime. Furbo Dog Camera have another interesting feature which will update the pet owner the real-life dog status through push notification on the owner smartphone app. When the dog is barking, Furbo Dog Camera will detect it and send a push notification to the pet owner and ask them whether they want to check with their pet or not. Furbo Dog Camera has a clicking sound feature when the device toss treat to the dog. This feature can be used as a training method to train the dog when will the device be tossing the treat to it.

Weaknesses

Although Furbo Dog Camera does have a lot of interesting features with it but there is some weakness with it too. The audio capability for Furbo Dog Camera is fairly weak. When the pet owner interacting with the dog, the sound quality that came out from the speaker are not clear. The dog might not be able to hear clearly what the owner want to say to it. The microphone that use not a good quality microphone too. When the dog barks, the owner does not hear it clearly too what the dog trying to say. The bark alert feature is a great feature but the sensitivity of the bark alert is way too sensitive. The pet owner sometimes might need to disable or even turning it off the push notification when they receive it way too frequent. The Furbo Dog Camera cannot withstand heavy impact too due to the material that it used to build the device, so if the dog knocked the device down on to the floor, the device might just break apart from the heavy impact.

2.1.4 Petcube Bites Treat Camera



Figure 2.4 Petcube Bites Treat Camera

Petcube Bites Treat Camera is a pet interaction camera that the owner can use to interact with their beloved pet when the owner is not at home. Petcube Bites Treat Camera has a wide-angle view camera lens that supports a 160° view. So the pet owner can check and see their pet in a wider angle of view in their smartphone. The device can record 1080p HD video which is a high definition quality video which can display their pet in the pet owner's smartphone much clearer. The device does support night vision too which can let the pet owner check and see with their pet when the room is dark. The device does support 4x digital zoom too to let the pet owner zoom in to check and see every corner within the 160° angle view. The Petcube Bites Treat Camera supports two-way audio to let the pet owner hear and speak with their pet through the smartphone. It has a full duplex sound with a 4-microphone array and a speaker bar which makes you like speaking on your pet through a phone. This device has a built-in treat dispenser too. The pet owner can use their smartphone to toss treats to their pet remotely. The amount of treat and distance of treat that need to be tossed can be controlled from the smartphone. It can be set to autoplay mode or scheduled times to automatically toss the treat to the pet even when the pet owner is busy when they do not have the time with their pet. The tank that holds up the treats is large too; it can hold up to 100 treats. The pet owner can share the pictures or even recorded videos easily to any popular social networks from their smartphone app when they capture any

CHAPTER 2 LITERATURE REVIEW

funny or interesting moment of their pet. This Petcube Bites Treat Camera support cloud-based video recording and even timeline history which let the pet owner the ability to check back the records on what their pet doing throughout the day. (McQuillan, 2020)

Strengths

Petcube Bites Treat Camera is easy to set up. So the pet owner does not need to spent the whole day reading the long user manual to understand how the installation works. What the owner need to do is plug in the power cable, download the Petcube app into their smartphone and follow a short online instruction on the setup and that's all the Petcube is ready to use. The Petcube Bites Treat Camera do come with an interesting feature which is the night vision mode, so if the dog is sleeping during the night time, the pet owner can just connect to Petcube and checkup with their pet condition without worrying the room is dark. Petcube has the ability to be able to mount on the wall, so the pet owner does not need to worry about the device getting knock over by the pet or something. Petcube has another feature which allow the others to watch other pet owners cameras too. Just like the San Diego Cat Café, the owner had allowed the public to access to their cameras which is a great idea for those cat loves to checkup what the cats doing from time to time.

Weaknesses

Although Petcube Bites Treat Camera do have a lot of interesting features that helps the pet owner interacting with the pet but there is a security issue with the features which let the public from accessing to the device too. If someone has the intention to break into the house, they can just connect into the Petcube device and check around what time might the pet owner gone to work, and when will the house be empty. People can use this features to spy the pet owner and even the pet too so it is not a good idea for this feature since it concerns about the security problem for the pet owner.

2.1.5 Petzi Treat Cam



Figure 2.5 Petzi Treat Cam

Petzi Treat Cam is a pet cam that allows the pet owner to checkup what does their pet doing when the pet owner is not around the house. Petzi Treat Cam is called the treat dispenser too because the owner can dump treat to their pet remotely from their smartphone. The treat dispenser can work with any treat that is smaller than 2.54cm. Petzi Treat Cam do support night vision that is massively clear. So the pet owner does not need to worry about unable to check up what is the pet doing in a dark room. The audio that Petzi Treat Cam produce is clear and high quality too which let the pet owner listen clearly the sound of his pet and even their pet can listen clearly what their owner said. Petzi allows the pet owner to capture what their pet doing too using the smartphone app while checking up on their pet. (Petzi Treat Cam Review, 2020)

CHAPTER 2 LITERATURE REVIEW

Strengths

Petzi Treat Cam do have a lot of interesting features like the night vision. What the owner need to do is just leave a small light or even without using any light to trigger the night vision to be activate. So the pet owner can see more clearly what their pet is doing during that time. The treat dispenser on Petzi Treat Cam is very easy to refill too, all the pet owner need to do is removing the cover and refill the treat from the top hole. Petzi Treat Cam has the ability to speak to the dog clearly too. So the pet owner can interact with their lonely pet at home when the pet owner is not around in the house. The pet owner can capture any interesting picture of their pet too when they check up what their pet doing that time.

Weaknesses

There are some weaknesses with Petzi Treat Cam which is larger treat might be stuck from the treat dispenser output hole. So the pet owner only limited on buying smaller treat to prevent the treat dispenser from getting stuck by larger treat. Petzi Treat Cam dispense multiple treat at once too which might waste a lot of treat if they were dispenses so much in the same time. The pet might get bored from the treat too if the treat is dispensing so many in the same time and this would not be a good idea to use back the same treat to train the dog. Petzi Treat Cam does not alert the pet owner when the pet bypass the device, so the owner always need to connect to the device camera in order to check and see does the pet is around nearby the camera or not. The pet cannot see their pet owner too but only can hear their pet owner voice. The pet would not enjoy the interaction session too if they unable to see their pet owner but only can hear their voice only. The video recorded has 3 to 4 second of delay too. So let say if the pet stay in front of the device for 2 seconds and walk away, the pet owner might just be talking to himself only and does not realize that the dog just went away pass 4 seconds ago. Another drawback is that the pet owner unable to record the video when interacting with the pet. The pet owner might not be able to record down any interesting or special moment of their pet during the pet interaction session.

2.2 Comparison between the existing system and proposed system

	Pawbo+	PetChatz	Furbo Dog Camera	Petcube Bites Treat Camera	Petzi Treat Cam	Proposed System
2-way communication	Yes	Yes	Yes	Yes	No	Yes
Notify dog detected depressed	No	No	No	No	No	Yes
Encourage human and pet communication	Yes	Yes	Yes	Yes	Yes	Yes
Encourage pet and pet communication	No	No	No	No	No	Yes
Support wireless connection	Yes	Yes	Yes	Yes	Yes	Yes
Support Ethernet port	No	No	Yes	No	Yes	Yes
Quick and easy to setup for non-tech-savvy user	Yes	Yes	Yes	Yes	Yes	Yes

Table 2.1 Comparison between existing system and proposed system

Table 2.1 shows the comparison between existing systems with the proposed system. The existing system used for comparison Pawbo+, PetChatz, Furbo Dog Camera, Petcube Bites Treat Camera, and Petzi Treat Cam.

2.3 Recommendations

After reviewing several existing products, each of them has some limitations in terms of letting the pet and pet to communicate with each other. Their products only do let the pet owner to communicate with their pet only but it does not let the pet and another pet to communicate. Another problem is that, products like Pawbo+ and PetChatz does not support night vision too. This going to be a drawback to the users because if the room is getting dark, the user that in the dark room would not be able to see by another user from the device. This might cause some problem to the users during the communication. Multi-channel communication is very important in this part, product like Petzi Treat Cam does not have this feature which let the owner does not have any idea what the pet trying to tell them since they cannot hear their pet barking sound without the microphone. Another thing in the product is that the pet cannot see their owner face too, so they do not have any idea who is the person that talking to them as they might think that probably it is the device that are talking to them. From the review above, I found out that there are several products like Pawbo+, PetChatz, and Petcube Bites Treat Camera does not have an Ethernet port too as they only support wireless connection to the internet. If the pet owner setup the device far away from the access point, this will cause the internet connection to become poor. In order to prevent video or voice lag problem during the conversation, they should use Ethernet cable connection connecting to Ethernet port on the pet device. Last but not least, some of the products also lack of notification or alert to the pet. They tend to directly connect to the pet device when the pet owner call the pet without notifying the pet. Especially treat dispensing, the pet would only notify the pet device dispense the treat if they are right in front of the pet device, but if the pet is not right in front of the pet device and the user dispense the treat, the pet would just ignore it since they do not even know that the treat has been dispense. So alert or notification are very important to alert the pet that the treat has been dispense.

Hence, the proposed system will develop a smart environment to encourage pet and pet to communicate with each other. In the proposed system, the pet owner would just need to help the pet to create their pet profile account and connect the dog to the another dog to have a chitchat, and leave the device for the pet to play around by themselves. The proposed system will be supporting multi-channel communication to let both dog to communicate with each other. Both dog are able to see another dog face and heard their barking sound through the pet

CHAPTER 2 LITERATURE REVIEW

device. This device let the user to connect the proposed system with using wireless connection or Ethernet connection (need to get a USB type C to Ethernet adapter). If the user does not have a stable wireless connection, the user can just plug in the Ethernet cable as the second options onto the proposed system. In the proposed system, the dog emotion will be predicted and sent to the proposed system. If the dog is depressed or looks ill, the proposed system will sent an email with an image of his dog current situation to the dog's owner to notify him/her that the dog currently is detected depressed or sick.

CHAPTER 3: METHODS/TECHNOLOGIES INVOLVED

3.1 Methodology

The smart environment for dog social network project will select RAD phased development model as the methodology. In this phased development methodology, there will be 4 phases which is planning, analysis, design, implementation. After the implementation, it will push the part of the system implemented out phases by phases. After the 1st phase completed the next iteration will continue from analysis phase until implementation phase and push it out to the client as the next system version until all the end of the project life cycle. Each phase in this system will be a part of the system.

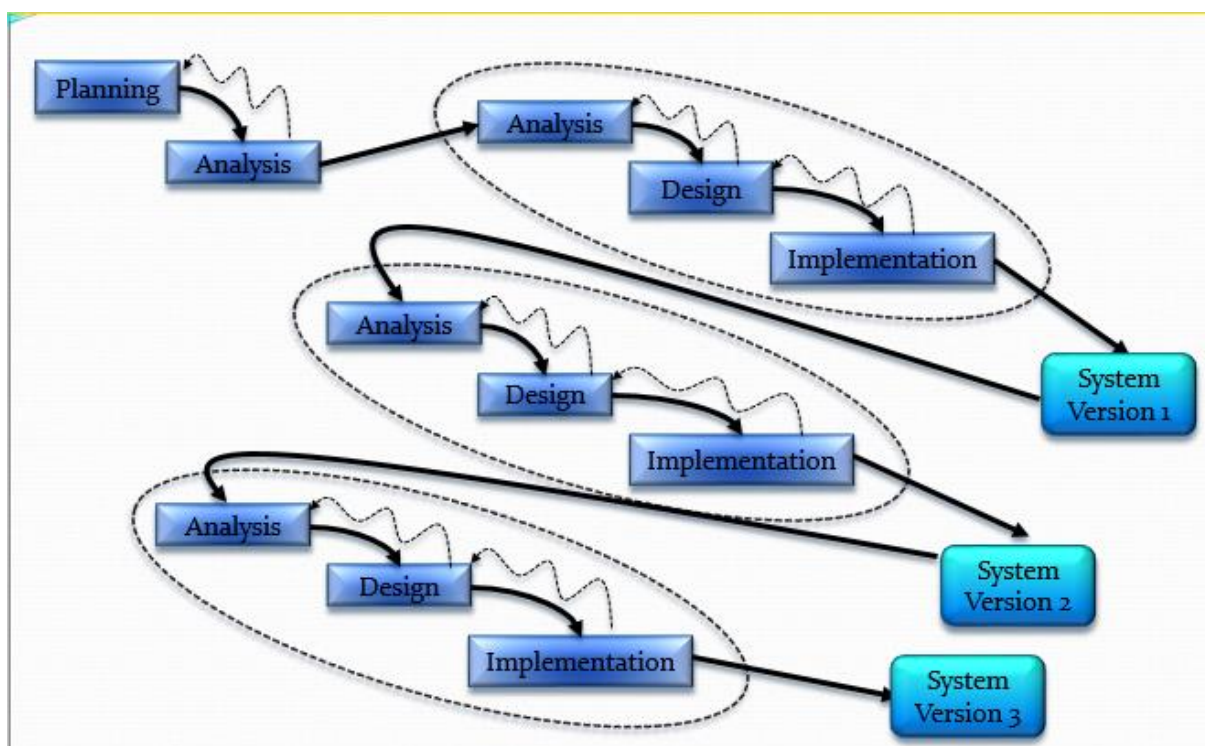


Figure 3.1 Phased Development Methodology

3.2 Project Workflow in Phased Development

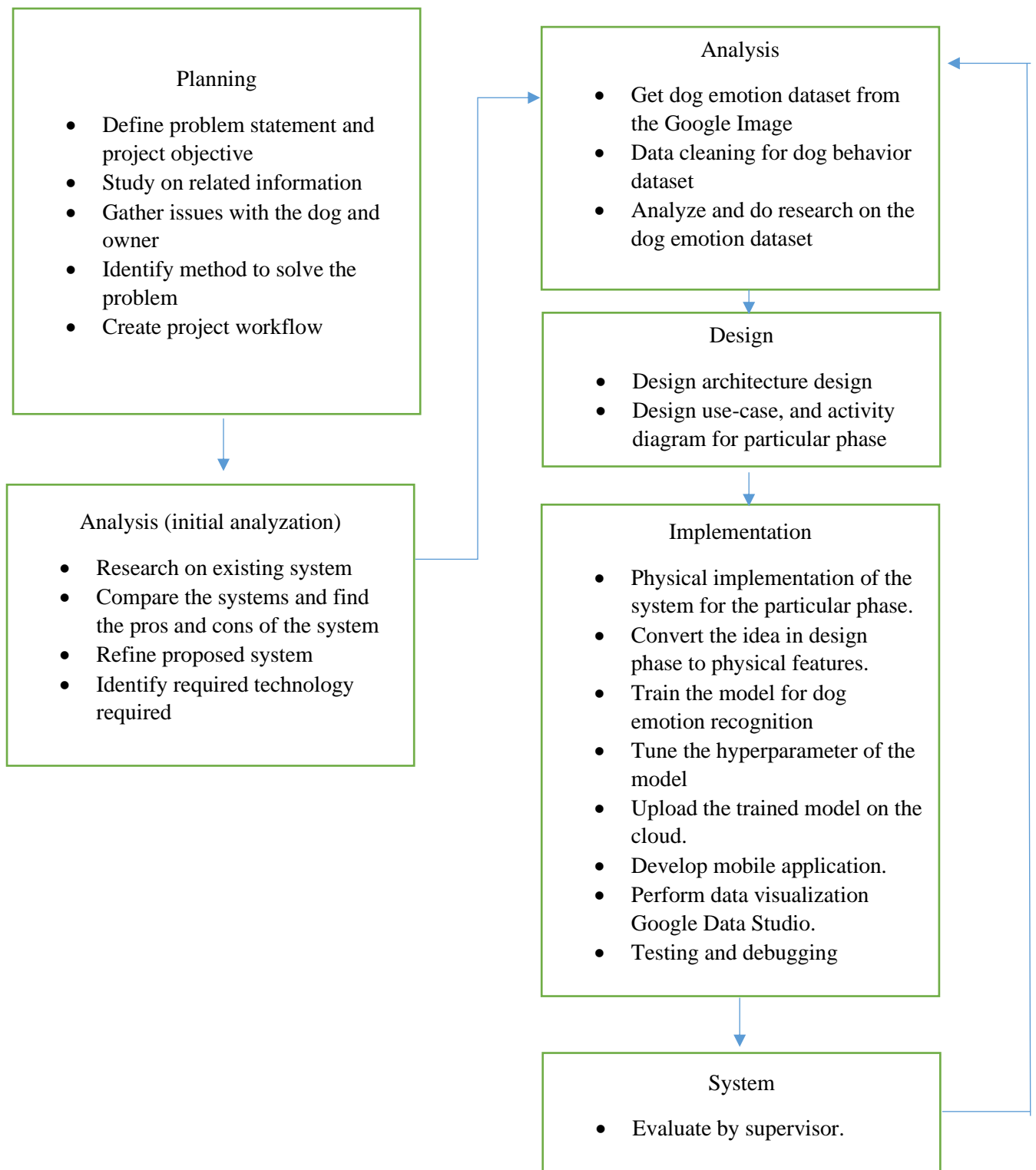


Figure 3.2 Project Workflow in Phased Development

CHAPTER 3 METHODS/TECHNOLOGIES INVOLVED

- Planning Phase

In this phase, we need to define the problem statement, project objective, study on related information. Information will be gathered including the problem occurs with the dog and owner. So after gathering the problem of the dogs and owner, we need to identify a method to solve the problem, by listing out the objective that needed to solve the problem.

- Analysis Phase (initial analyzation)

This phase, we do research on existed system and compare with each of the system the pros and cons of the systems and our proposed system. Then we will refine the proposed system by solving some problems that the existing system lack of. Later on, we identify what are the require tools and technologies that are needed in the project, who will be the user using our system, where and when will the system be used. As mentioned before, the target user will be the dogs, the location on implementing the system will be indoor area, and the dog will use the system when they are bored or want to find their companion dog to chat with each other.

- Analysis Phase

This phase, we will gather some images and audio of different dog emotion class as our dataset for the training part, and study on different type of dog emotion. Then, we will perform data cleaning to remove the images and audio that is not relevant to the particular class in the dataset. In this part, we need to start to analyze the dataset to have a better understanding of the dataset and perform data cleaning to clean up any bias from the dataset.

- Design Phase

In this phase, the architecture design, use-case diagram, activity diagram will be planned in this stage too to shows how the system works.

- Implementation Phase

In this phase, it will involve physical implementation of the system according to the features that need to develop in each phase. We will need to train a model and compare which model is the most suitable to get a more accurate prediction. The next step will be tuning the hyperparameter to improve the accuracy of our model. Then set-up the Google Cloud Platform by uploading the AI model onto the cloud. Then we will continue implementing on the client side. At the client side, we will set-up the environment by creating a mobile application. In the application the dog will be able to create an account, search friend, add friend, call friend and even let the dog's owner to visualize on the predicted emotion of another dog when communicating with his/her dog. The camera and microphone will capture the frame and host the captured frame on a RTMP server to stream the captured camera and audio frames. Then the application will read the camera and audio frame from the RTMP server. The frame that show on the client side will be uploaded to the cloud for dog emotion recognition with the trained model and return the result to the client, if the dog is ill, the system will sent an email to the dog's owner. Then, we will perform data visualization on the Google Data Studio to analyze the dog's emotions predicted. After done the physical implementation, test cases will be created to perform testing on the system before the deployment of the system.

3.3 Technologies Involved

The technologies that will be use in this project will be Google Cloud Platform, Android Mobile, Keras, Jupyter Notebook, Audacity, Python, Node.js and Android Kotlin language for program implementation.

- Android Mobile

At the client side, we will use Android mobile device that have camera and microphone. The Android mobile device will contain an application that get the captured frame from the RTMP server and sent the frame to the cloud and perform prediction on the dog emotion using the trained dog emotion recognition model.

- Android Kotlin Language

Android Kotlin language will be used to implement an Android application that capture the dog behavior using the camera and capture the dog barking using microphone embedded on the device. This program will have call interface that let the user to communicate with another user.

- Google Cloud Platform

In Google Cloud Platform, we will be using the Ubuntu vm instance. All the captured data including the dog barking sound and dog behaviour images will be upload to the Google Cloud Platform Ubuntu vm instance. At the Google Cloud Platform, it will be implemented with trained dog behaviour AI model. The captured dog behaviour images will be analyzing though the AI model to identify the behaviour of the dog, and the dog barking sound will be analyse to improve the image recognition prediction result of the dog emotion. The result will then send to the client device to show the result of the prediction. The predicted result will be saved in the database that hosted in the cloud for data visualization. If the predicted result shows that the dog is sick, it will notify the dog's owner by sending an email to the dog's owner with an image attached in the email.

CHAPTER 3 METHODS/TECHNOLOGIES INVOLVED

- Keras

Keras will be used to train on the dog emotion recognition model with several hyperparameters and find out which hyperparameter is the best for the dog emotion recognition model to provide the best accuracy.

- Jupyter Notebook

Jupyter Notebook will be used to perform data cleaning, analysis on the collected dataset, perform data visualization on the collected dataset and finding the best hyperparameter when tuning the model.

- Node.js

Node.js will be used to create an API route for the image and audio file that send from the client side. The image and audio that received will be save on the cloud Ubuntu VM and it will execute the command to perform the prediction of the following image and sound, then return the prediction result back to the client. The Node.js will also have other routes which provides dog social media features like creating an account, search friend, add friend, getting the friend RTSP IP and port URL and etc.

- Python Language

Python language will be used in creating scripts to helps in scraping images from the internet, downloading videos from YouTube and converts the downloaded video into wav audio format. Python language will be used in creating a script to load the trained dog emotion model to predict the dog's image and barking audio file and return the result back to Node.js.

- Audacity

Audacity will be used in extracting the particular part of dog barks which suitable for the certain dog emotion label. Like for example, the growling of a dog will emit a certain audio frequency, so we will crop the part of audio and saved into the directory which belongs to the growling label.

3.4 System Design

3.4.1 Google Cloud Architecture Diagram

First, The Android Mobile will upload the image and audio to the Google Cloud Platform Ubuntu VM Instance. This image and audio will be uploaded through POST Request to the Node.js REST API. Then, the Node.js will save the image to the '/images' directory and audio to the '/audios' directory in the Ubuntu VM instance. The Node.js will then execute the python script (dogEmotionClassifier.py) which will grab the image and audio from the respective directory. After done predicted the dog emotion, the python script (dogEmotionClassifier.py) will return the predicted result back to the Node.js. The Node.js will store the predicted result in the MySQL Database. Then, Node.js will return the predicted result to the Android Mobile and display on the calling interface. If the predicted is predicted as 'sick', the Node.js will get the user's email from the MySQL database and sent an email to the user's email to notify the user that the dog is sick with an image attached to the email.

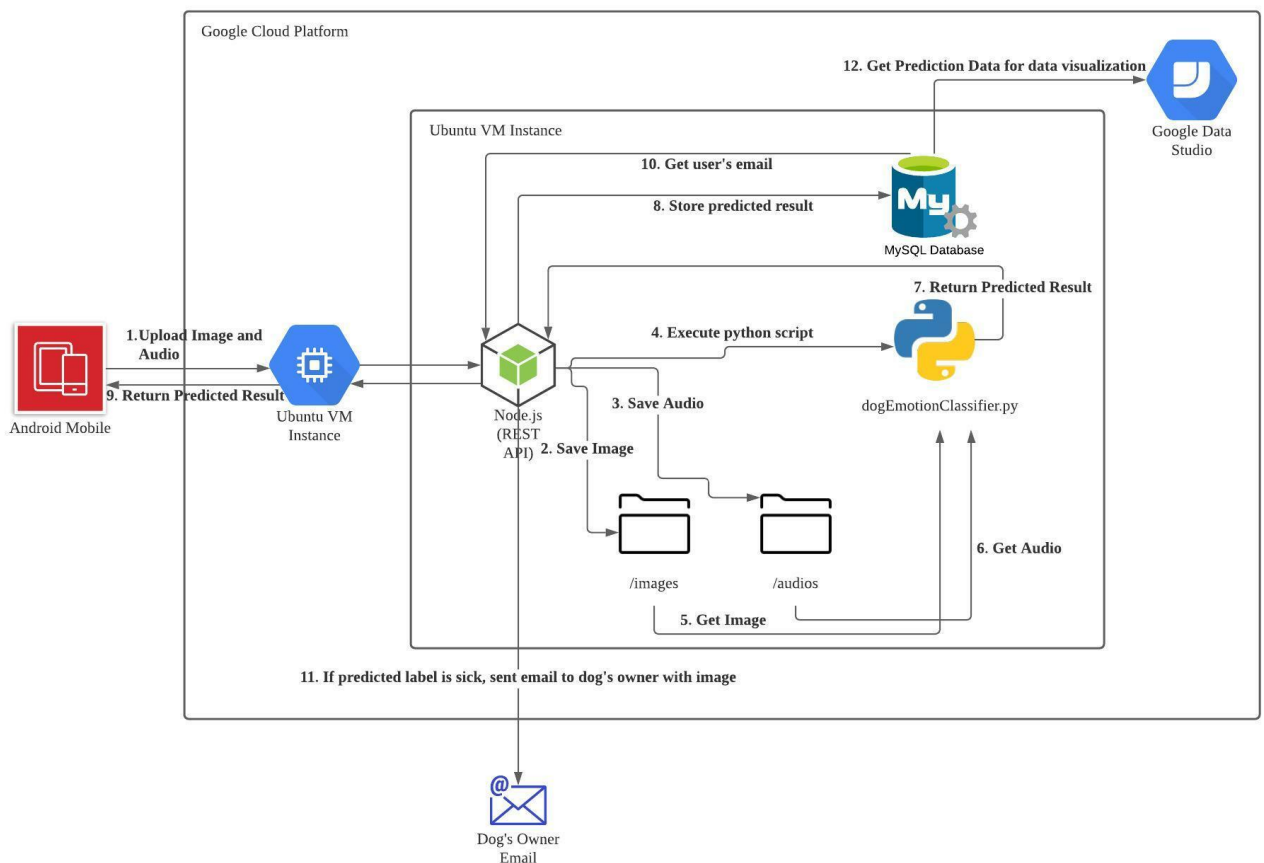


Figure 3.3 Architecture Diagram of Proposed Solution

3.4.2 Use Case Diagram

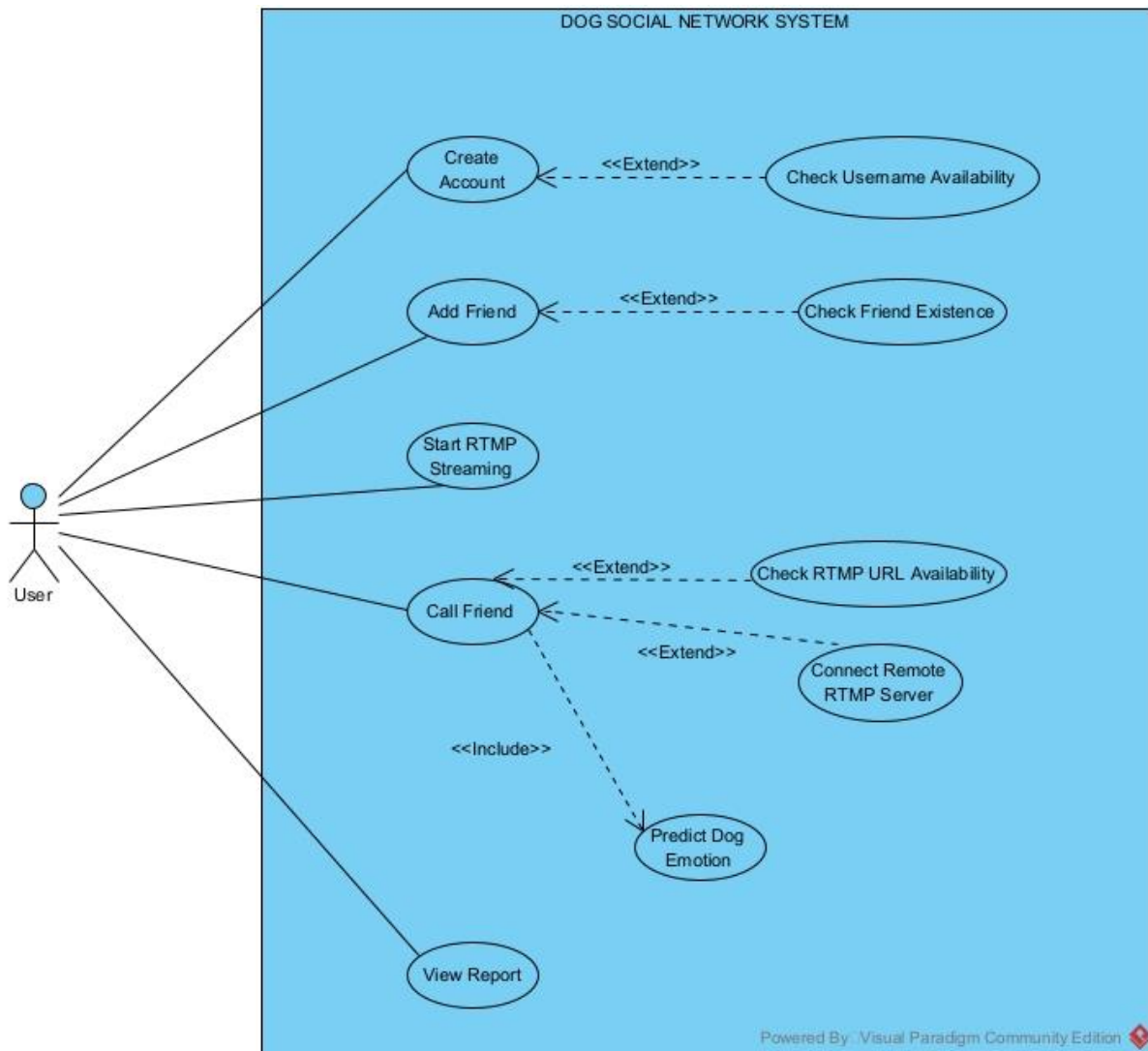


Figure 3.4 Use Case Diagram

3.4.3 Activity Diagram

3.4.3.1 Create Account

When the user wants to create an account, the user need to input all the required information like username, password, email, and RTMP IP and port. After than the system will check whether the username available or not. If it's not available, the system will prompt an error message, else it will create an account for the user. After done creating an account for the user, it will redirect back the user to login page.

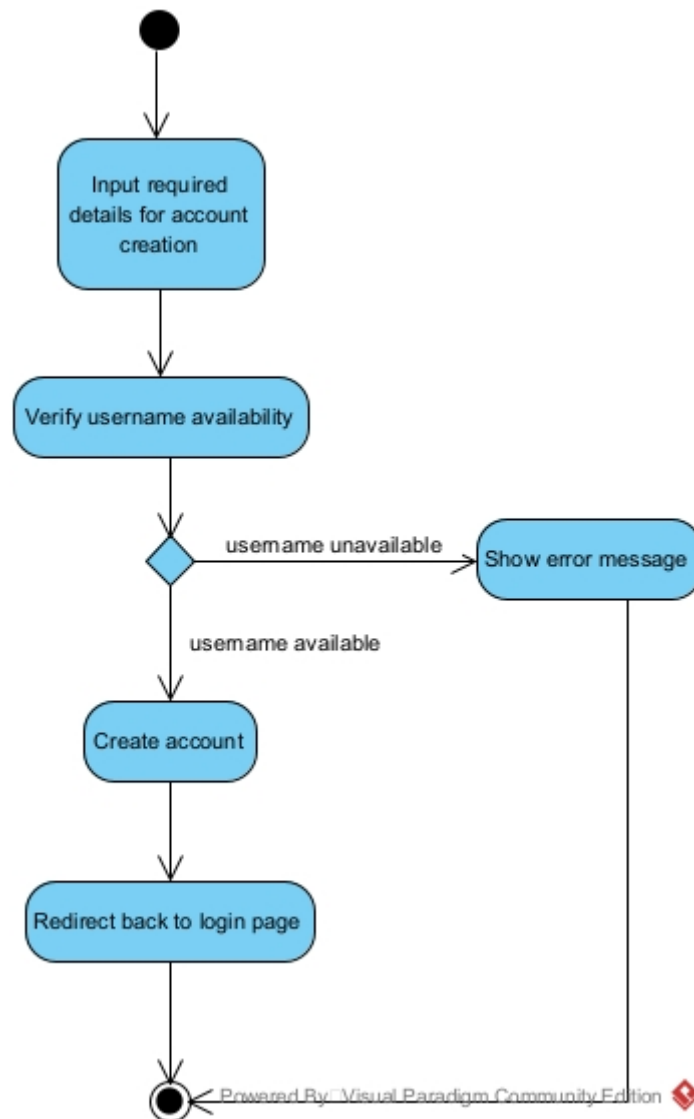


Figure 3.5 Activity Diagram of Create Account

3.4.3.2 Add Friend

When the user wants to add a friend, the user needs to input the username into the respective text field. Then the user needs to click on the button to search for the username for the friend. The system will then check and see if there is any possible username in the database, if no possible username matches the search query, the system will show an error message, else the system will list all the possible usernames on the interface. The user can click on the username that matched his friend username and view his/her friend profile. Then the user can add the friend. If the username has been added, the system will show an error message, else the system will show a success message which means that the user has just added the friend.

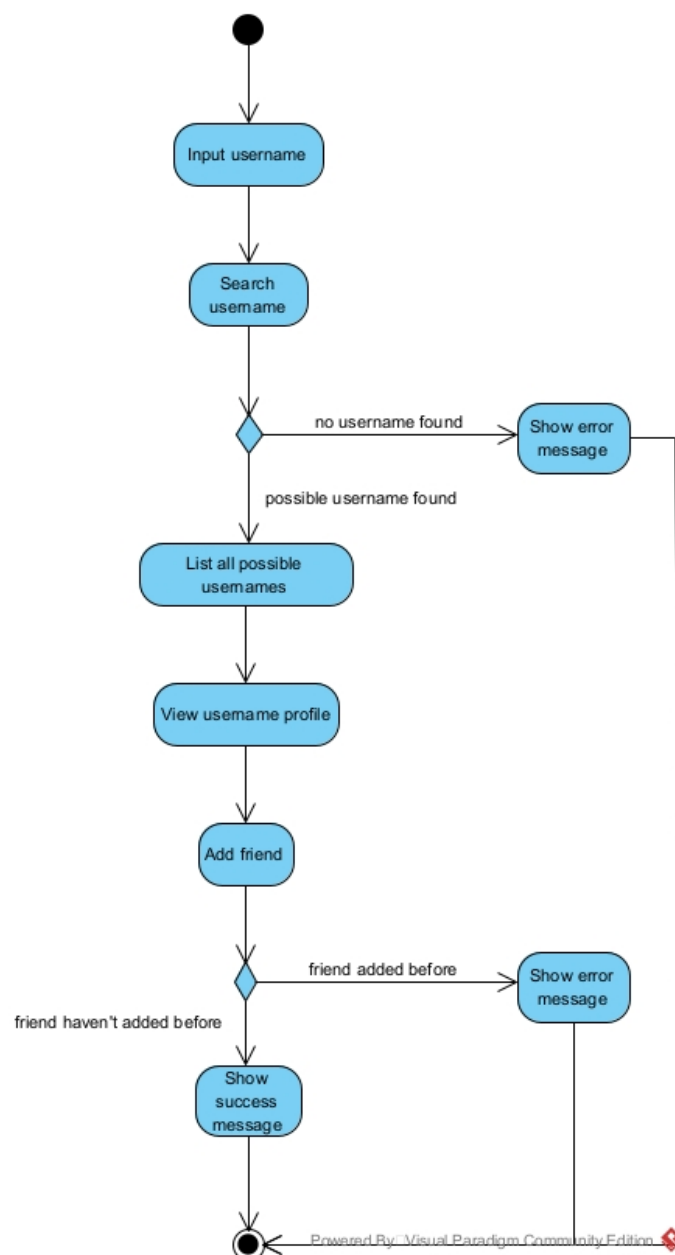


Figure 3.6 Activity Diagram of Add Friend

3.4.3.3 Start RTMP Streaming

When the user want to start the RTMP server, the user need to click on the “Start Server” button on the interface. Then the system will start the RTMP_Toggle android service. The system will pop up a floating window on the interface too. The system will the prepare video for RTMP server. If the system fail to prepare the video, the system will then show an error message, else the system will continue preparing the audio for RTMP server. If the system unable to prepare the audio, the system will then show an error message, else it will start RTMP streaming on port 1935. The system will display the video frame captured by camera on the floating window.

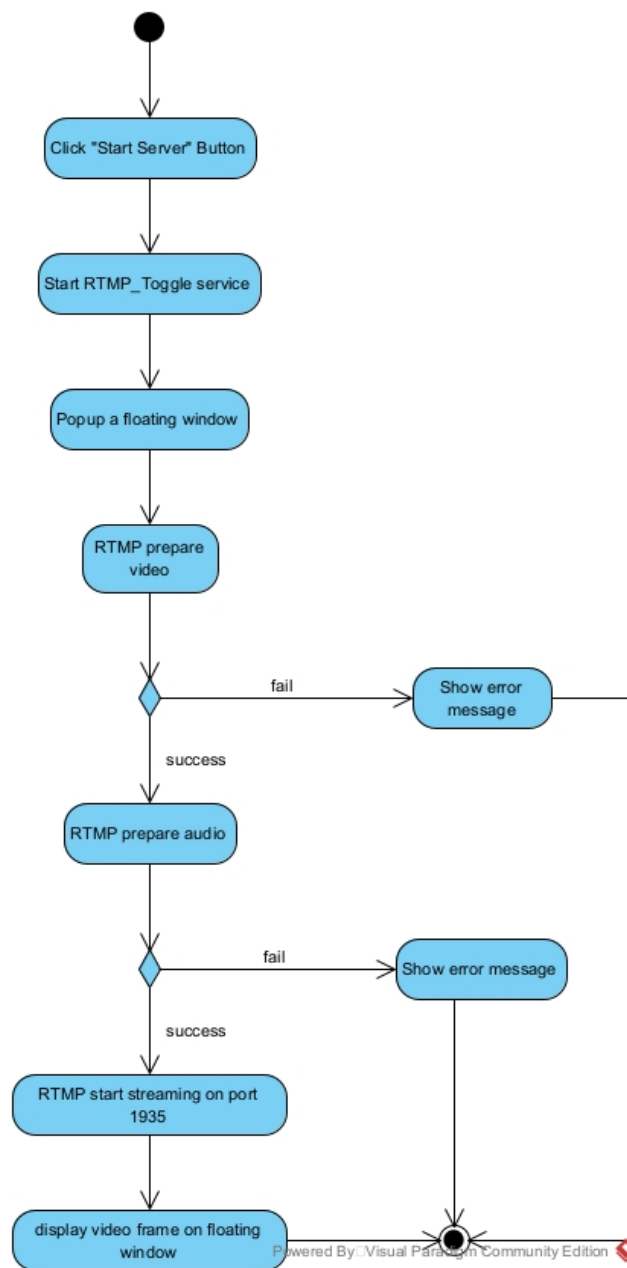
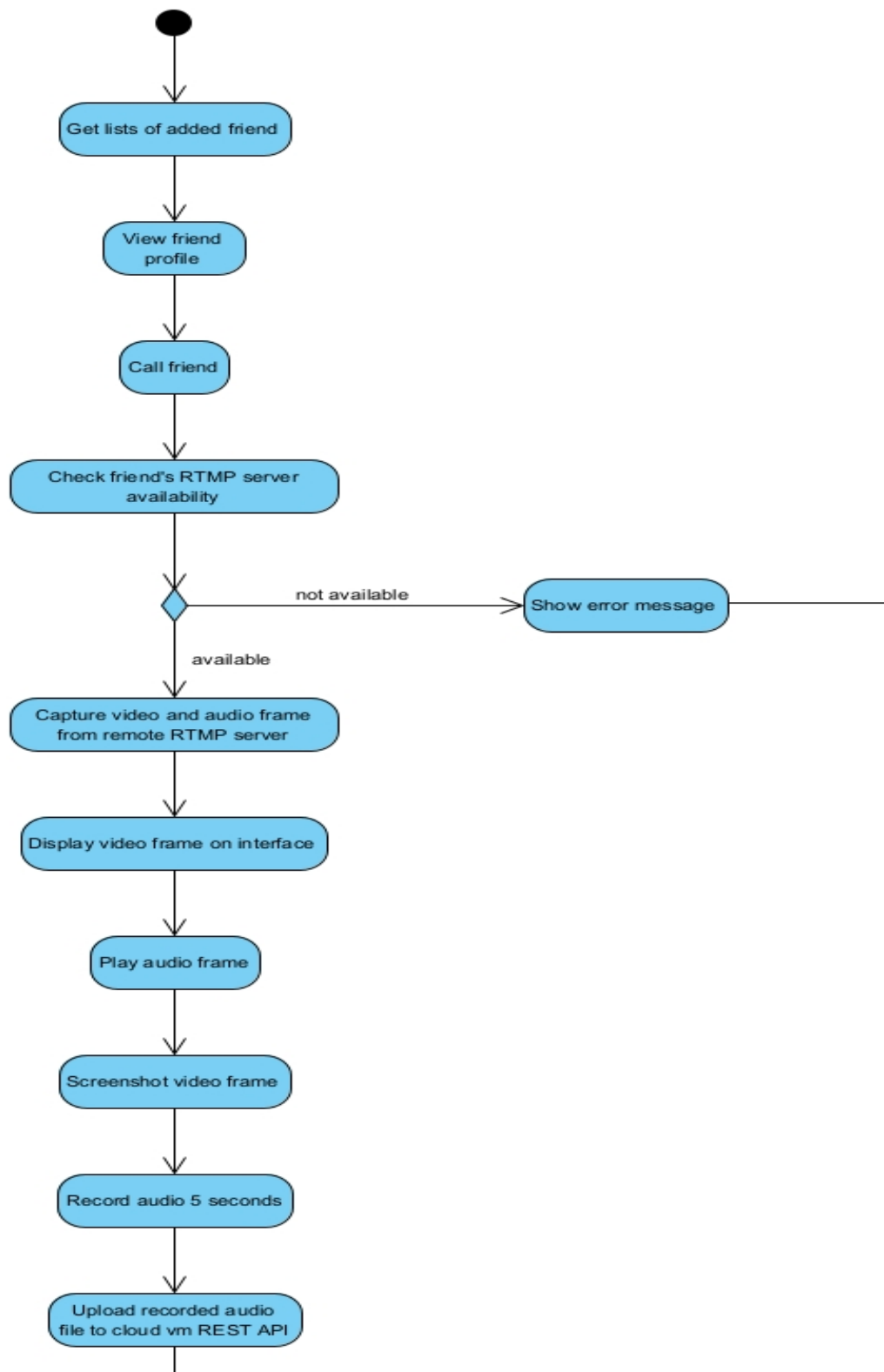


Figure 3.7 Activity Diagram of Start RTMP Streaming

3.4.3.4 Call Friend

When the user wants to call its friend, the user will get the lists of added friends first. Then the user selects the friend that want to call to view the friend's profile. The user will then click on the call friend button to call the friend. The system will check for the friend's RTMP server availability first. If the remote RTMP server is not available, the system will shows an error message, else the system will proceed to capture the video and audio frame from the remote RTMP server. The system will display the video frame on the interface. Then the system will play the audio frame. The system will take a screenshot of the video frame of its friend, then the microphone will record the audio for 5 seconds. After done, the system will upload the recorded audio to the cloud vm REST API. Then the system will upload the image file to the cloud vm REST API. The system will save the image and audio file in the cloud vm. The system will then start predicting the image first. Then the system will start predicting the audio file. The system will combine both the prediction result and start calculating the best prediction label. After that the predicted label will be added into the database. The system will check the predicted label too. If the predicted label is sick, the system will get the dog's owner email from the database, then it will sent an email to the dog's owner email with an image attached in it. Then the system will display the predicted result on the call interface. For the prediction label is not sick, the system will directly display the predicted result on the call interface.



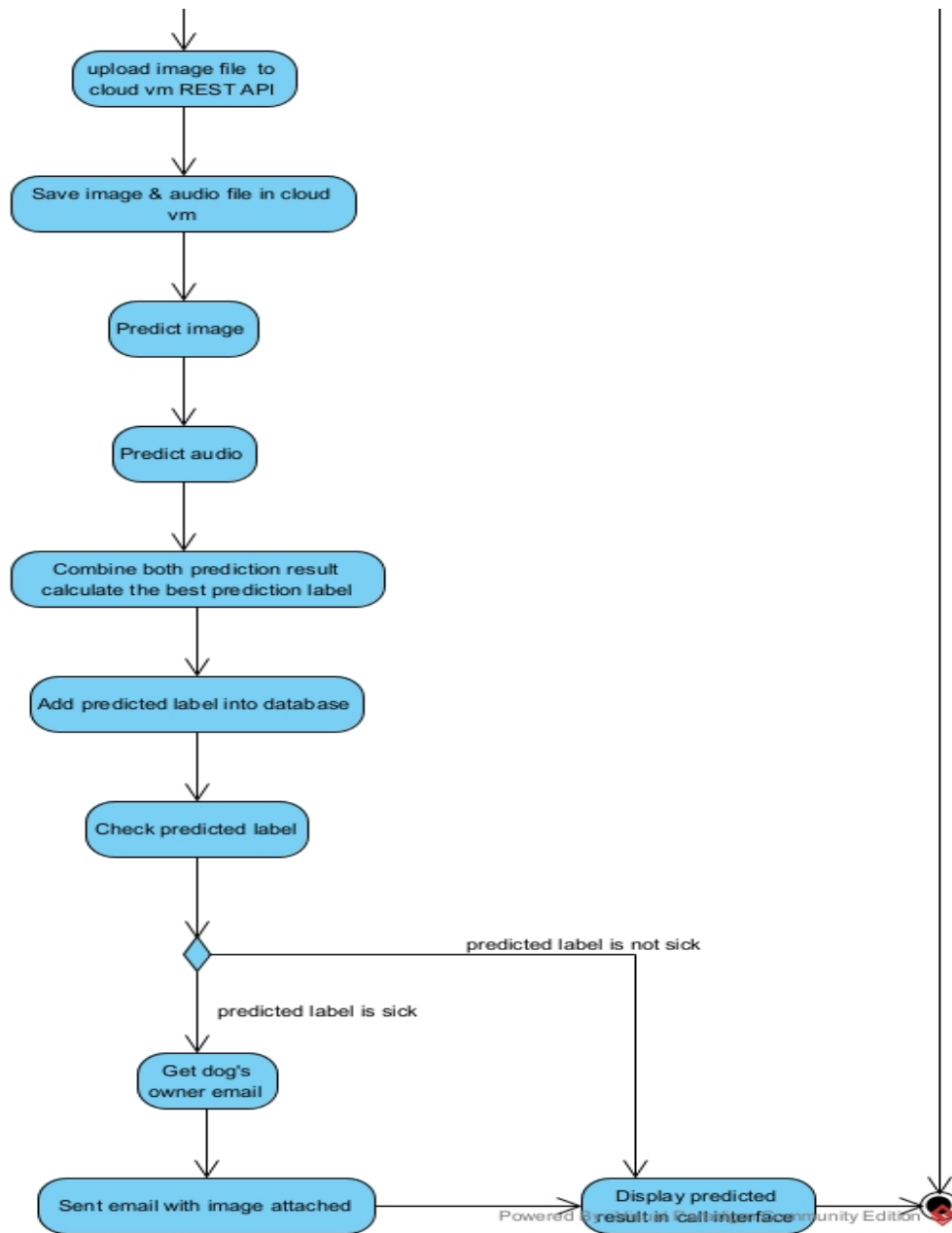


Figure 3.8 Activity Diagram of Call Friend

3.4.3.5 View Report

When the user wants to view reports of previous call predicted labels, the user can first get all the user's report lists. Then the user can click on the report name to view the report. The system will get the report data from the database. Then the system will start calculating the predicted label. The system will plot a bar chart with the calculated predicted label. Then, the system will plot a line chart. The system will also display a description message for the report to explain what the graph means.

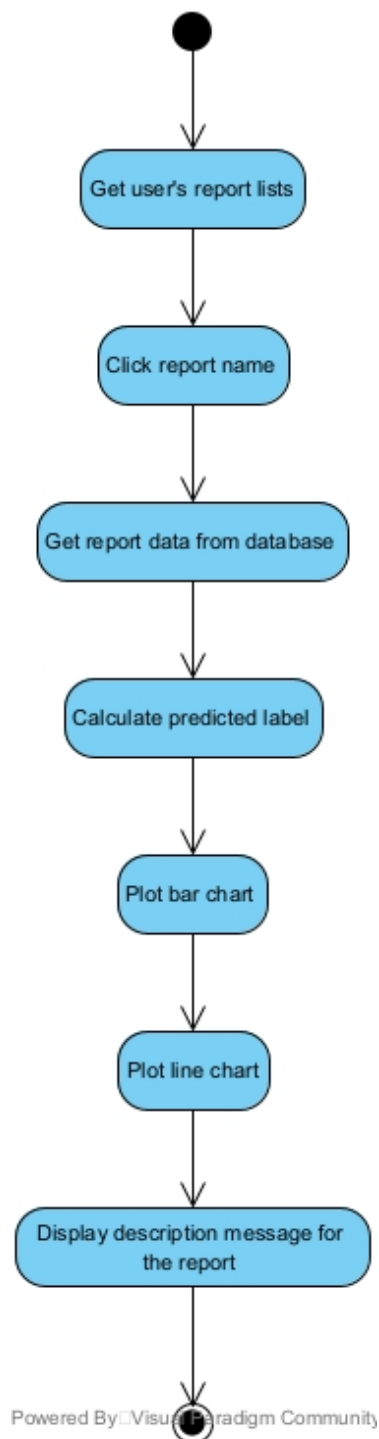


Figure 3.9 Activity Diagram of View Report

3.5 Implementation Issues and Challenges

During the implementation phase, there were some problem that faced when doing training on the dog barking recognition model. The first problem that faced in this dog barking recognition model analysis is when I tried to use a raw audio file to train the model. My testing accuracy for the model are very low. So, I tried to check and see what is the problem that caused the testing accuracy low but the training accuracy are high. So, after doing some analysis I found out that in the raw audio file right, there are a lot of noises that included together for a single label. Like for example, I want to train my model to recognize this type of noise are belongs to growling label but in that raw audio file it includes other noises like human talking sound after the dog growling. To solve this problem, first I used Audacity to read the audio file first, then I analyze which part of the audio spectrum belongs to the dog growling sound. After found the part, I will copy the part of audio spectrum and save it into an audio file. After repeating this step for all the other audio file, I will start the dog barking recognition model again and compare the testing accuracy before and after the data cleaning on the raw audio file. Surprisingly that it solved the problem, and my testing accuracy increase compare to the one before I clean the raw audio file.

The challenge that faced in the project was the starting RTMP server part in the Android application. In this RTMP server starting feature, when I tried to start the RTMP server, it does works well as I can use my VLC player to test connecting to my RTMP URL hosting from my Android application. But there is one problem after I have press the back button the video frame does not continue capturing but the audio still continues playing in my VLC player. So, after some time in finding what caused this weird problem that caused the video feed not capturing. It seems like the android activity for starting RTMP server kills the SurfaceView that used to play the video frame. In order to keep this SurfaceView alive and available all the time when surfing around to the other android activities, I need to make a floating window for the SurfaceView so that after I have gone to another activity it will still works fine. Then for the RTMP server hosting to keep on continue in the background, I need to makes the RTMP server streaming in the Android service. So the finalize starting RTMP server feature will be first the user enter into the interface and it prepared the floating window with SurfaceView in it. Then, when the user click on the start RTMP server button, the system will start the Android service which will starts the RTMP server streaming the captured frame from camera and microphone. So the problem that video feed stopped after going to another Android activity solved after

CHAPTER 3 METHODS/TECHNOLOGIES INVOLVED

making the SurfaceView into floating window and runs the RTMP server streaming in Android service.

3.6 Project Timeline

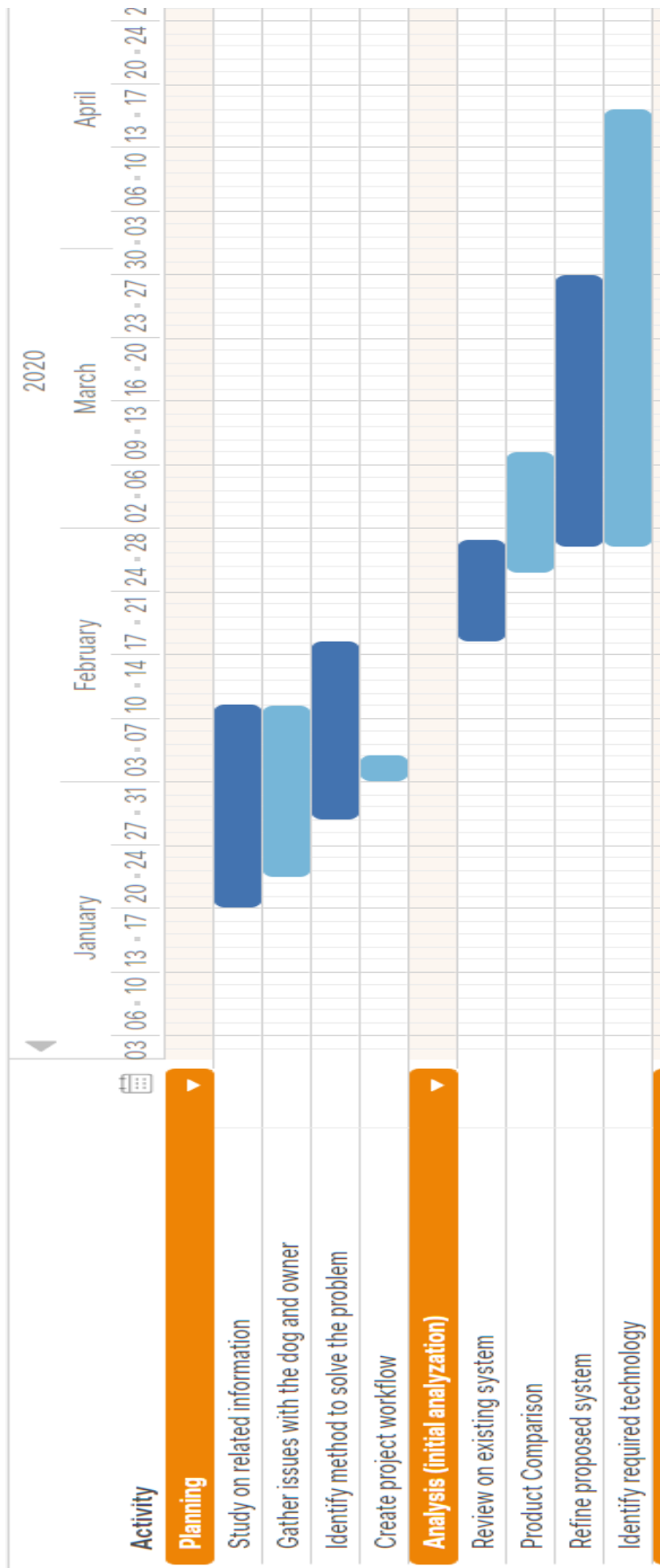


Figure 3.10 Gantt Chart 1

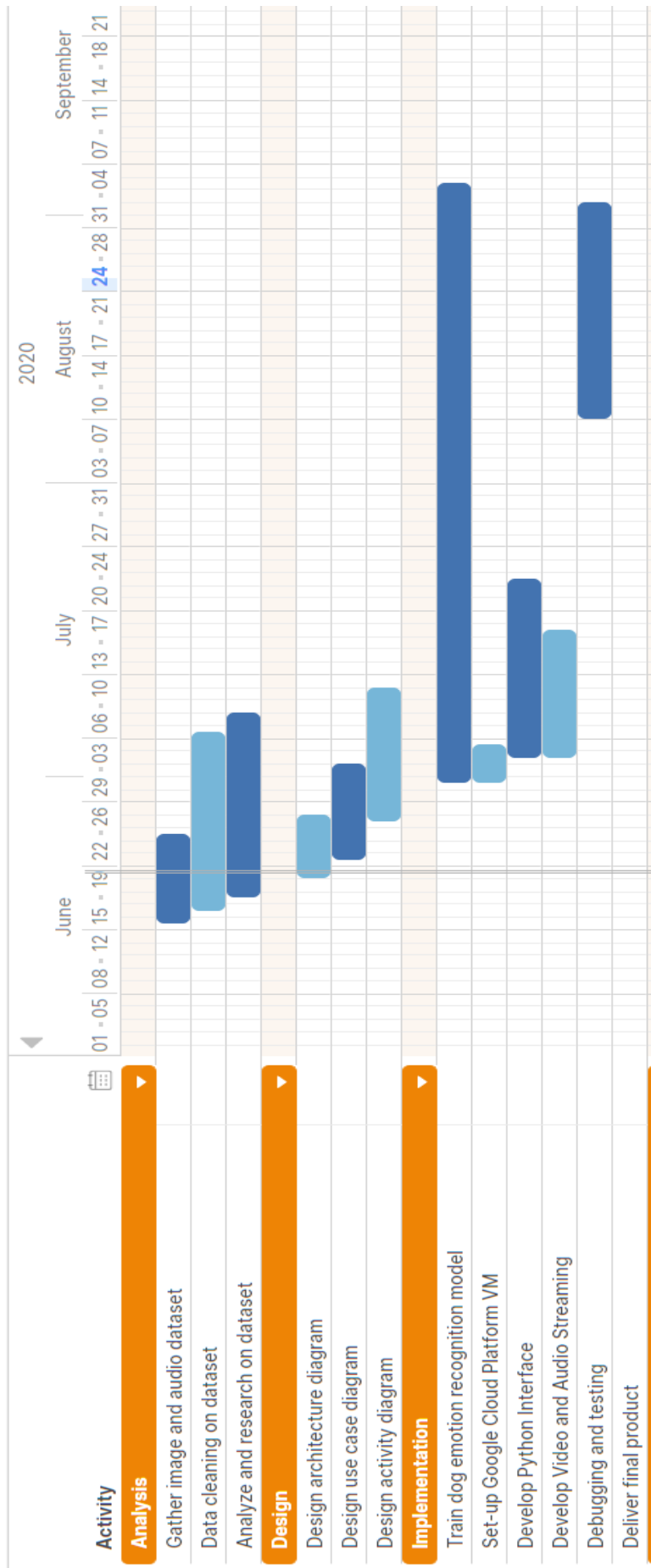


Figure 3.11 Gantt Chart 2

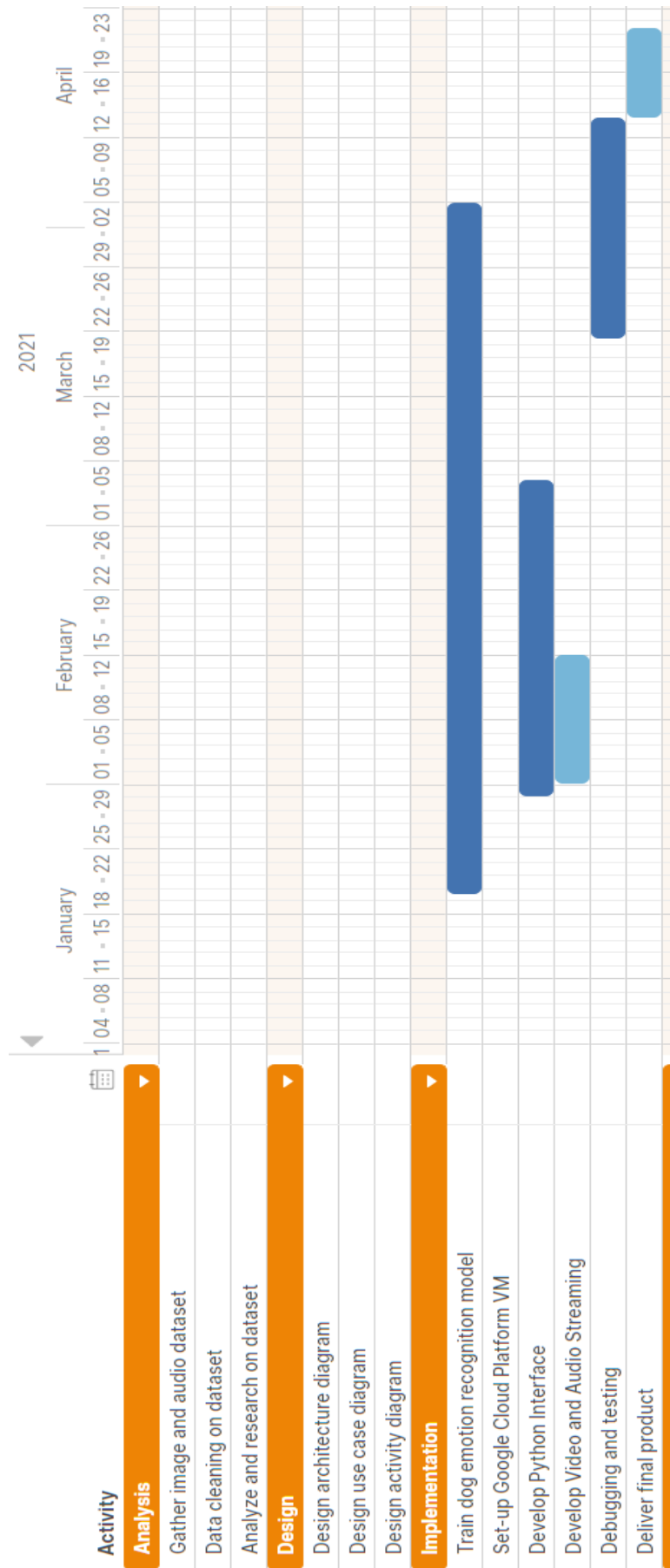


Figure 3.12 Gantt Chart 3

CHAPTER 4: PRELIMINARY WORK

4.1 Dog Image Emotion Recognition Model

Before starting to train the dog image emotion recognition model, we need to gather our image dataset and separate them into different class which in our case we will be separating them into different directories. Figure 4.1 shows that we used the python script to scrap dog images from Google Image by using an automated bot to save images to the local storage.

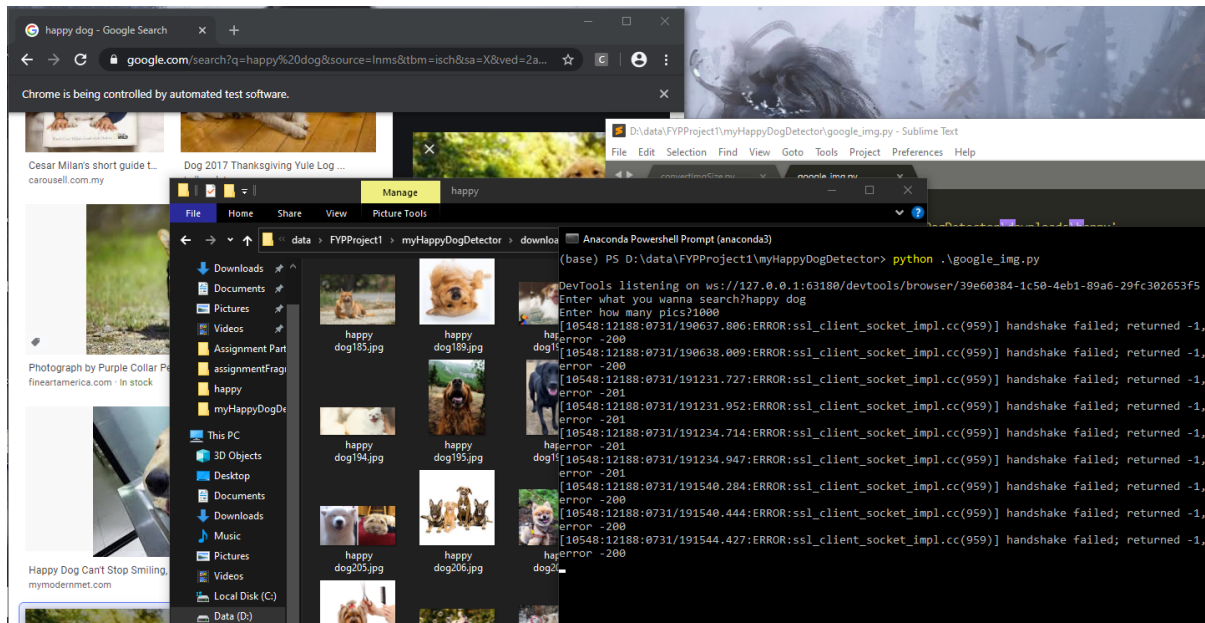


Figure 4.1 Preparing Image Dataset with Python Script

After complete scrapping images from Google Image, we started to perform data cleaning on the dog image dataset. We will identify which dog image that is not relevant to the class and remove the image from the directory. Example, in our “happy” class directory, it should be filled with all happy dog’s images. So we try to identify which images that the dog does not show happy emotion and remove it from the directory.

All the images that scrap from Google Image, there will be different type of image resolution. So to standardize it, in figure 4.2 we use the python script to perform image resizing to resize all the image in the dataset to the specific image resolution which is 224x224 resolution.

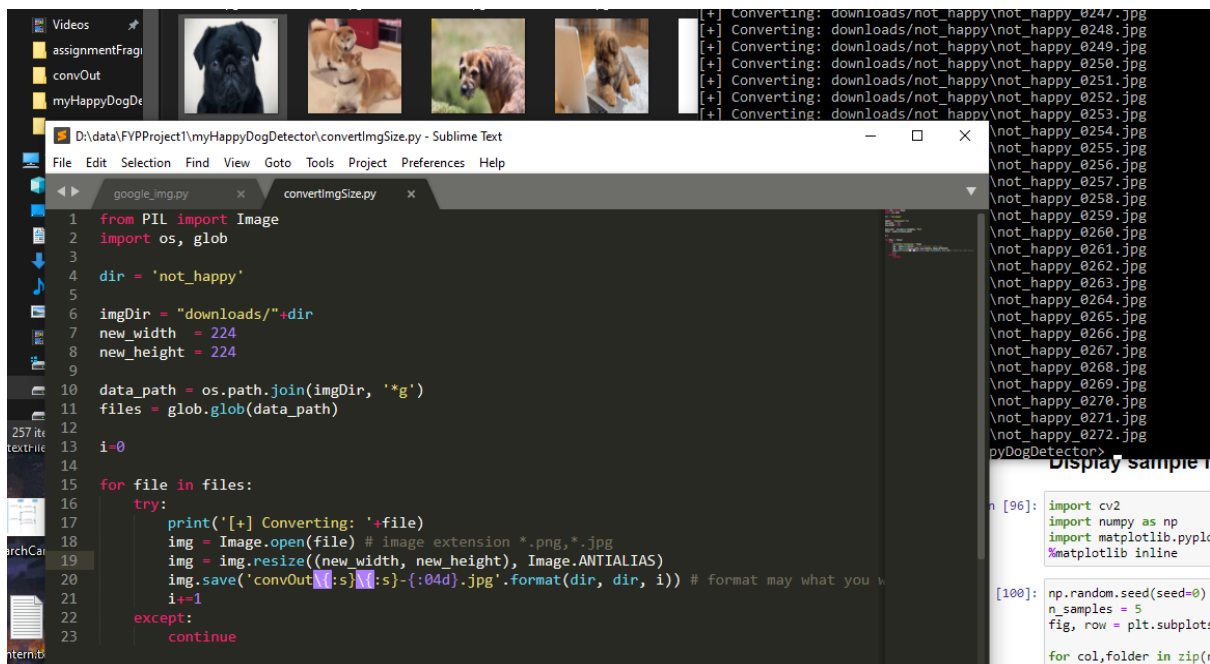


Figure 4.2 Resizing Dog Images with Python Script

In Figure 4.3, before starting to load the dog image dataset. We will display 5 random images for each of the label just to test and see whether we can load the image successfully or not. In our dataset we will have 3 classes which will be “happy”, “angry” and “sick”.

Display sample images

```

In [6]: import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

In [7]: np.random.seed(seed=0)
n_samples = 5
fig, row = plt.subplots(3, n_samples, figsize = (4*n_samples, 3*3))

for col, folder in zip(row, data_folders):
    col[int(np.floor(n_samples/2))].set_title(folder, fontsize=25)
    working_dir = os.path.join(data_path, folder)
    os.chdir(working_dir)
    for col_ax, img in zip(col, np.random.choice(os.listdir(os.getcwd()), n_samples, replace=False)):
        rand_img = cv2.imread(img)
        rand_img = cv2.cvtColor(rand_img, cv2.COLOR_BGR2RGB)
        col_ax.imshow(rand_img)
        col_ax.axis('off')
plt.subplots_adjust(left=0.2, wspace=0.02)
os.chdir(current_path)
    
```

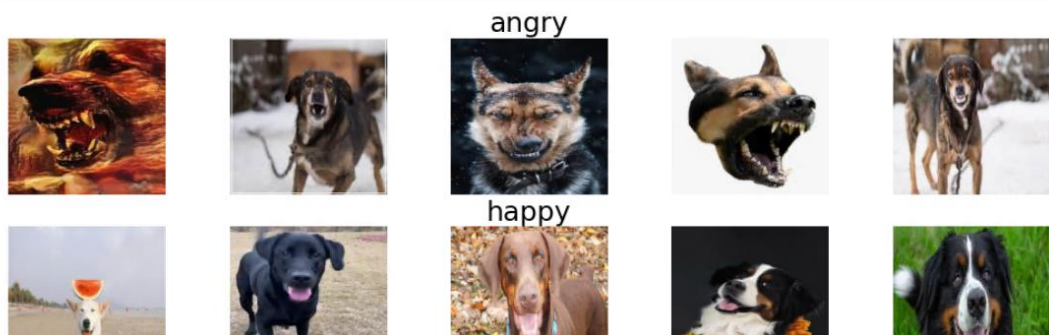


Figure 4.3 Display 5 Sample Images for Each Label

After done checking the images loading, we will start importing all the images from the 3 directory which representing their prediction label. In figure 4.4, we will start the loading and pre-processing phase. We will normalize the imported images and then split the imported images into 3 set of data, training, validation and testing data.

The screenshot shows a Jupyter Notebook with three code cells. The first cell, titled 'Load & pre-processing', contains Python code to load images from a directory, convert them to BGR2RGB, and resize them. The second cell, titled 'normalization', contains code to convert the image data to float32 and normalize it. The third cell, titled 'Splitting', contains code to shuffle the data and split it into training, validation, and testing sets using sklearn's train_test_split function.

```

Load & pre-processing

In [8]: %%time
img_rows = 224
img_cols = 224
img_list = []
label_list = []
labels = dict([('happy',0),('angry',1),('sick',2)])
for folder in data_folders:
    working_dir = os.path.join(data_path, folder)
    os.chdir(working_dir)
    current_list = os.listdir(os.getcwd())
    for img in current_list:
        img_in = cv2.imread(img)
        img_in = cv2.cvtColor(img_in, cv2.COLOR_BGR2RGB)
        img_in = cv2.resize(img_in, (img_rows, img_cols), cv2.INTER_AREA)
        img_list.append(img_in)
        label_list.append(labels[folder])
    os.chdir(current_path)

Wall time: 2.93 s

normalization

In [9]: img_data = np.array(img_list).astype(np.float32)
img_label = np.array(label_list)
img_data /= np.max(img_data)

Splitting

splitting dataset into training, validation & testing

In [10]: from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
X_shuffled, y_shuffled = shuffle(img_data, img_label, random_state=0)

X_train, X_valid_test, y_train, y_valid_test = train_test_split(X_shuffled, y_shuffled, test_size=0.2, random_state=0, stratify=y_shuffled)
X_train, X_valid, X_test, y_train, y_valid, y_test = train_test_split(X_valid_test, y_valid_test, test_size=0.5, random_state=0, stratify=y_valid_test)
print('X_train shape: {}'.format(X_train.shape))
print('y_train shape: {}'.format(y_train.shape))
print('X_valid shape: {}'.format(X_valid.shape))
print('y_valid shape: {}'.format(y_valid.shape))
print('X_test shape: {}'.format(X_test.shape))
print('y_test shape: {}'.format(y_test.shape))

```

Figure 4.4 Loading, Pre-Processing and Splitting the Dataset

We will start building our 1st model which will be ResNet-like model. In figure 4.4, the first block of code shows the code to build a ResNet-like network. As our dataset is small, so we will be using ImageDataGenerator to perform data augmentation to replace our original batch with the randomly transformed batch. So when doing training on the CNN, we will be using the randomly transformed batch not the original batch.

```

M def identity_block(n_f,x):
    shortcut = x
    x = Conv2D(n_f,(3,3),strides=(1,1),padding='same',kernel_initializer='he_normal',activation='relu')(x)
    x = BatchNormalization()(x)
    x = Conv2D(n_f,(3,3),strides=(1,1),padding='same',kernel_initializer='he_normal')(x)
    x = BatchNormalization()(x)
    x = Add()([shortcut,x])
    x = Activation('relu')(x)
    return x

def conv_block(n_f,x):
    x = Conv2D(n_f,(3,3),strides=(2,2),padding='same',kernel_initializer='he_normal',activation='relu')(x)
    x = BatchNormalization()(x)
    x = Conv2D(n_f,(3,3),strides=(2,2),padding='same',kernel_initializer='he_normal',activation='relu')(x)
    x = BatchNormalization()(x)
    return x

M datagen = ImageDataGenerator(width_shift_range=0.2,
                              height_shift_range=0.2,
                              horizontal_flip=True,
                              zoom_range=0.2, ##### modified
                              rotation_range=45)

M inputs = Input(shape=img_data[0].shape)

```

Figure 4.5 Building a ResNet-like Model

The next step we will find whether less data and more data will affect the prediction result for training and testing prediction. In figure 4.6, we will first take a look how many data will it be for less and more data dataset.

<pre> M from sklearn.model_selection import train_test_split from sklearn.utils import shuffle X_shuffled, y_shuffled = shuffle(img_data,img_label,random_state= X_train,X_valid_test,y_train,y_valid_test = train_test_split(X_s X_valid,X_test,y_valid,y_test = train_test_split(X_valid_test,y_ print('X_train shape: {}'.format(X_train.shape)) print('y_train shape: {}'.format(y_train.shape)) print('X_valid shape: {}'.format(X_valid.shape)) print('y_valid shape: {}'.format(y_valid.shape)) print('X_test shape: {}'.format(X_test.shape)) print('y_test shape: {}'.format(y_test.shape)) </pre> <pre> X_train shape: (384, 224, 224, 3) y_train shape: (384,) X_valid shape: (48, 224, 224, 3) y_valid shape: (48,) X_test shape: (48, 224, 224, 3) y_test shape: (48,) </pre>	<pre> from sklearn.model_selection import train_test_split from sklearn.utils import shuffle X_shuffled, y_shuffled = shuffle(img_data, X_train,X_valid_test,y_train,y_valid_test = train_test_split(X_s X_valid,X_test,y_valid,y_test = train_test_split(X_valid_test,y_ print('X_train shape: {}'.format(X_train.shape)) print('y_train shape: {}'.format(y_train.shape)) print('X_valid shape: {}'.format(X_valid.shape)) print('y_valid shape: {}'.format(y_valid.shape)) print('X_test shape: {}'.format(X_test.shape)) print('y_test shape: {}'.format(y_test.shape)) </pre> <pre> X_train shape: (636, 224, 224, 3) y_train shape: (636,) X_valid shape: (80, 224, 224, 3) y_valid shape: (80,) X_test shape: (80, 224, 224, 3) y_test shape: (80,) </pre>
---	--

Figure 4.6 Less Data And More Data Dataset Split Into Training, Validating, Testing Data

Now we will try using the hyperparameter of 200 epochs, 16 batch size and learning rate of 0.0005. We can see that for less data training prediction result that shows in figure 4.7 the accuracy for validation and testing in training prediction are 70.83% and 66.67% and for figure 4.8 that is for more data training prediction result, the validation and testing in training

prediction are 73.75% and 72.5% which is higher than the less data training prediction. The testing loss result in training prediction for more data are much lesser than less data.

```

Epoch 00120: ReduceLROnPlateau reducing learning rate to 1e-05.

In [20]: ▶ hm.print_valid_test_score(model_1_1,X_valid,y_valid,X_test,y_test)
          hm.training_plot(hist_1,model_name)

2/2 [=====] - 0s 78ms/step - loss: 0.8192 - accuracy: 0.7083
2/2 [=====] - 0s 82ms/step - loss: 0.8482 - accuracy: 0.6667
Valid: accuracy = 0.708333 ; loss = 0.819201
Test: accuracy = 0.666667 ; loss = 0.848221
    
```

Figure 4.7 Training Prediction Result for Less Data

```

Epoch 00123: ReduceLROnPlateau reducing learning rate to 1e-05.

In [18]: ▶ hm.print_valid_test_score(model_1_1,X_valid,y_valid,X_test,y_test)
          hm.training_plot(hist_1,model_name)

3/3 [=====] - 0s 160ms/step - loss: 0.8289 - accuracy: 0.7375
3/3 [=====] - 1s 169ms/step - loss: 0.6038 - accuracy: 0.7250
Valid: accuracy = 0.737500 ; loss = 0.828930
Test: accuracy = 0.725000 ; loss = 0.603796
    
```

Figure 4.8 Training Prediction Result for More Data

Now we will proceed in doing testing with test dataset on both more data trained model and less data trained model and compare the result. We can see a huge different in the testing accuracy that returned to us. In figure 4.9, we tested that for less data it will makes the model to be overfitted with the dataset, that is why we can see that in training prediction the accuracy is 70% and above, but in testing prediction it had dropped till 33.33% which is very critical. In figure 4.10, we can see some improvement on the model, as after we feed more data to the dataset, the model had lesser overfitting issue with the dataset, and it had improved the testing accuracy to 53.75%. This means that we need more data collected in order to makes the model prediction accuracy to be higher.

```

▶ model_file_name = 'Model_1_1-200.hdf5'
  chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
  print('Choosing the model via early stopping')
  test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
  print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
2/2 [=====] - 0s 86ms/step - loss: 0.8482 - accuracy: 0.3333
Test: accuracy = 0.333333 ; loss = 0.848221
    
```

Figure 4.9 Testing Dataset Prediction Result for Less Data

```

▶ model_file_name = 'Model_1_1-200.hdf5'
  chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
  print('Choosing the model via early stopping')
  test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
  print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
3/3 [=====] - 1s 172ms/step - loss: 0.6038 - accuracy: 0.5375
Test: accuracy = 0.537500 ; loss = 0.603796

```

Figure 4.10 Testing Dataset Prediction Result for More Data

Now after we know that we need more data in order to make the model to have higher prediction accuracy. So, we will stick with the more data dataset for now on. Now we need to find the suitable hyperparameter for the model. First, we will find the best learning rate range that is the most suitable for the model with the option of 2 different batch size 16 and 32 and both will be running with 50 epochs. In figure 4.11 shows the result for 4 different learning rate category which is 0.1, 0.01, 0.001, 0.0001 tested with 0 to 50 epochs in iterations with the batch size 16 and figure 4.12 shows the same 4 learning rate category tested with the iterations of 50 epochs with the batch size 32. From the observation, we found out that the learning rate of 0.1, 0.01, 0.001 turned out to be too large for the dataset to handle it. The loss and accuracy barely improve over the epochs as we can see that the model was bouncing up and down. The learning rate of 0.0001 were learning and it shows that the model accuracy and loss is improving over the epochs although some parts of the model are bouncing up and down but it still shows that it is improving little by little. So, the models with learning rate around 0.0001 would be ideal. So, 0.0005 is chosen as the learning for the models to explore in the following sections. As for the batch size, the difference between 16 and 32 for 0.0001 learning rate it shows that the accuracy is bouncing up and down for 16 batch size but it some part the accuracy is high and the loss is improved, unlike 32 batch size that shows the accuracy improved little bit throughout the 50 epochs but the bouncing is much higher comparing to batch size 16. So, for the experiment, we will be comparing both batch size of 16 and 32 with 200 epochs test and we will test it with our model.

CHAPTER 4 PRELIMINARY WORK

Please use model.fit, which supports generators.

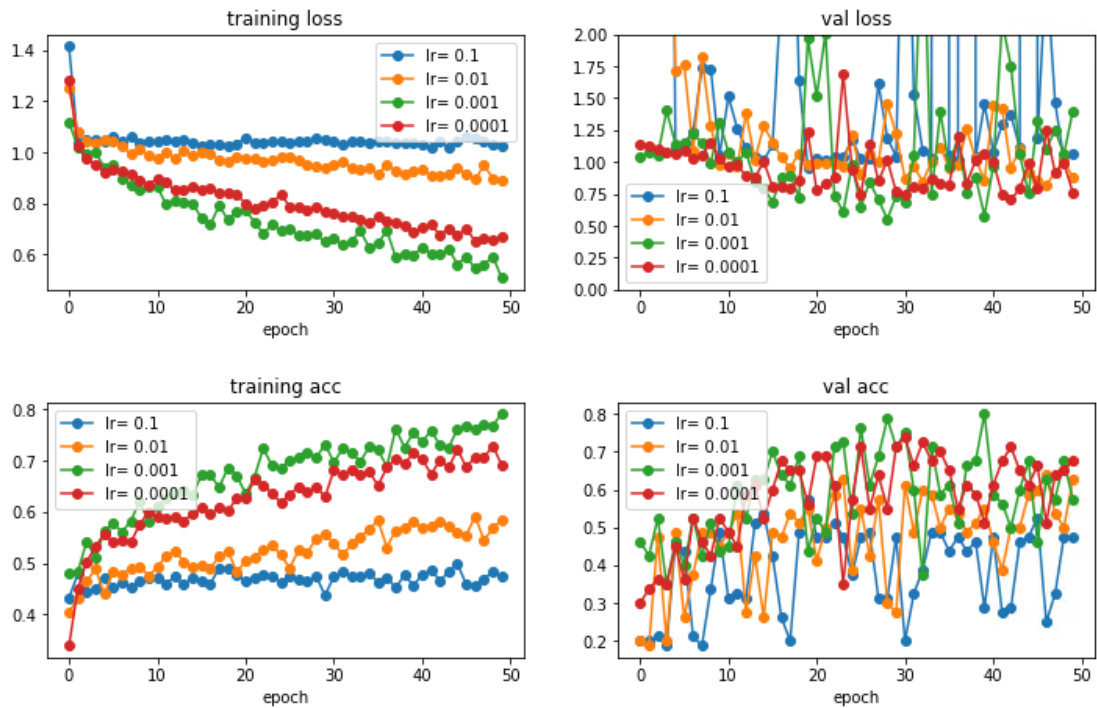


Figure 4.11 Epoch and Learning Rate Graph Result with Batch Size 16

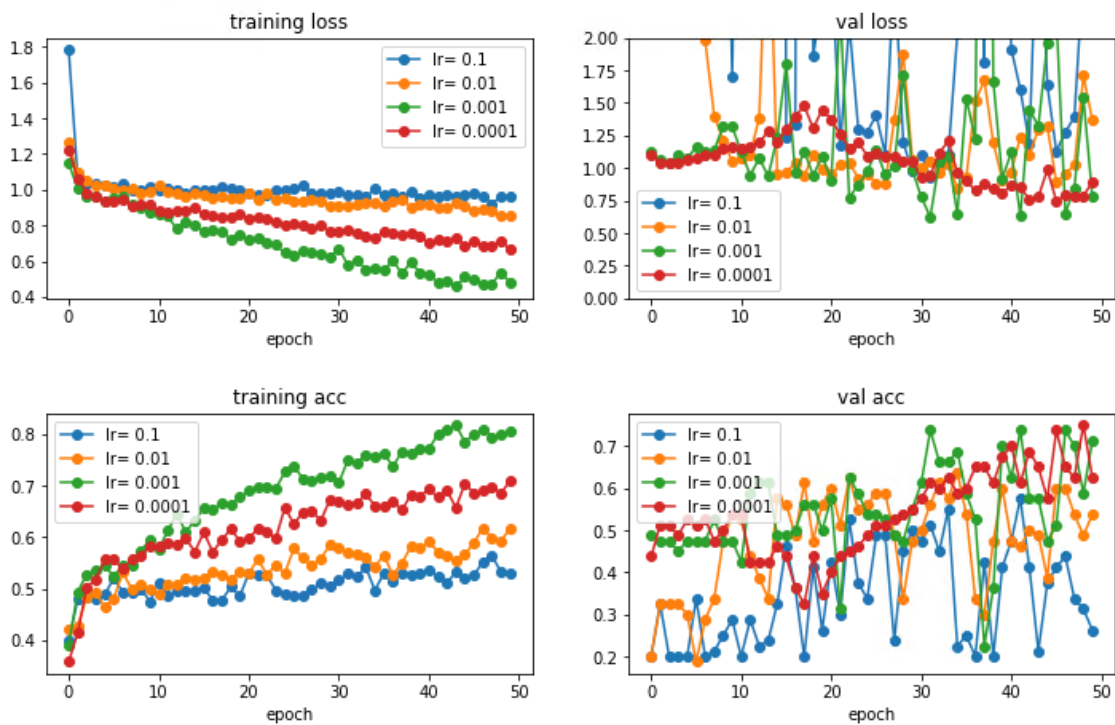


Figure 4.12 Epoch and Learning Rate Graph Result with Batch Size 32

CHAPTER 4 PRELIMINARY WORK

So continue the previous experiment, we will starting to test which epoch is the best for our model with the learning rate starting from 0.0005 to the minimum 0.00001 for the 2 batch size 16 and 32. Figure 4.13 will be the prediction result for our model 1_1 with batch size 16 and figure 4.14 will be the prediction results for our model 1_2 with batch size 32. From the experiment results on ResNet CNN, for the both batch size 16 and 32 the validation loss and accuracy is bouncing up and down before 75 epochs, and it start to get stable after 75 epoch. We can see that the accuracy is improving for both batch size 16 and 32 when the epoch is increasing unlike before 75 epoch the model is not stable.

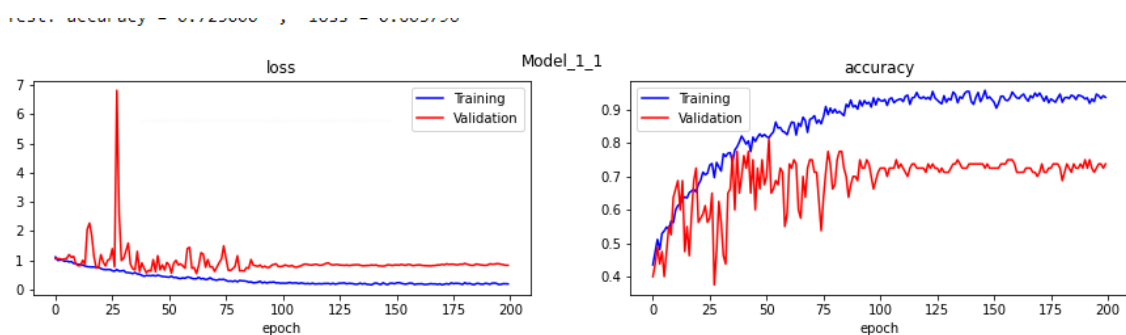


Figure 4.13 Epoch and Prediction Graph Result with Batch Size 16

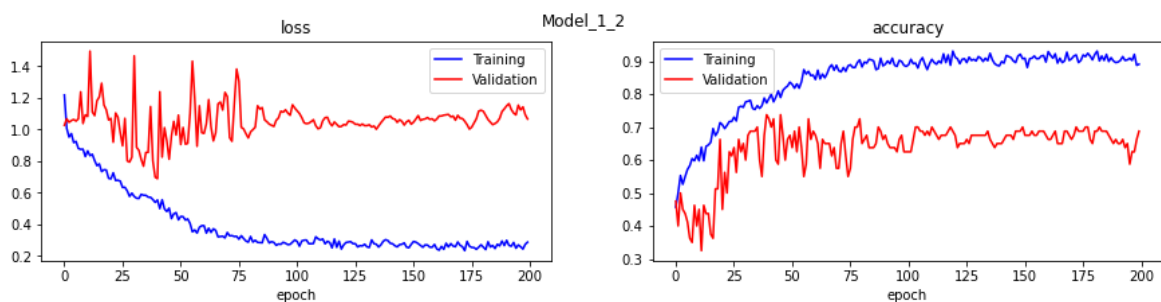


Figure 4.14 Epoch and Prediction Graph Result with Batch Size 32

From these two batch 16 and 32, we take out the best hyperparameter for the model and did some comparison testing on the test data which the result shows in figure 4.15 and 4.16. We found out that when the batch size is 16, the model can achieve 53.75% of accuracy and 60.38% of loss while for the batch size is 32, the model can only achieve 43.75% of accuracy and 66.29% of loss. So, to conclude that from the experiment results, we choose batch size 16 with the best hyperparameter as our model as it achieve better accuracy and low loss than the batch size is 32.

```

▶ model_file_name = 'Model_1_1-200.hdf5'
chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
print('Choosing the model via early stopping')
test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
3/3 [=====] - 1s 172ms/step - loss: 0.6038 - accuracy: 0.5375
Test: accuracy = 0.537500 ; loss = 0.603796

```

Figure 4.15 Prediction Result on Test Data of the best model with Batch Size 16

```

▶ model_file_name = 'Model_1_2-200.hdf5'
chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
print('Choosing the model via early stopping')
test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
3/3 [=====] - 0s 165ms/step - loss: 0.6629 - accuracy: 0.4375
Test: accuracy = 0.437500 ; loss = 0.662889

```

Figure 4.16 Prediction Result on Test Data of the best model with Batch Size 32

Now we have found the best hyperparameter for ResNet-like model, we need to do some comparison with another network which is the VGG16 and see does ResNet-like model or VGG16 is the most suitable for the dog image emotion recognition. We will first initialize the VGG16 by creating it. The figure 4.17 will be the code to create the VGG16 model.

```

#model creation
x = Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same", activation="relu")(inputs)
x = Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu")(x)
x = MaxPool2D(pool_size=(2,2),strides=(2,2))(x)
x = Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu")(x)
x = MaxPool2D(pool_size=(2,2),strides=(2,2))(x)
x = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu")(x)
x = MaxPool2D(pool_size=(2,2),strides=(2,2))(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = MaxPool2D(pool_size=(2,2),strides=(2,2))(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu")(x)
x = MaxPool2D(pool_size=(2,2),strides=(2,2),name='vgg16')(x)
x = Flatten(name='flatten')(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
outputs = Dense(3, activation='softmax')(x)

model_2_1 = Model(inputs=inputs,outputs=outputs)
model_2_1.compile(optimizer=optimizer,loss='sparse_categorical_crossentropy',metrics=['accuracy'])

```

Figure 4.17 Block of Code That Create VGG16 Model

CHAPTER 4 PRELIMINARY WORK

We have done creating the VGG16 model, now we will test running on both different batch size which is 16 and 32. The 2 tests will run with the learning rate starting from 0.0005 to the minimum 0.00001. Figure 4.18 will be the prediction result for our model 1_1 with batch size 16 and figure 4.19 will be the prediction results for our model 1_2 with batch size 32. From the experiment results on VGG16 CNN, the loss for batch 16 remains constant around 1.0 but for the batch 32 the loss keep on bouncing up and down for training and for validation it remains constant around 1.05. As for the accuracy, for batch size 16 the bouncing up and down rate are very high and the validation accuracy, it remains constant throughout the epochs iteration. For the batch 32 accuracy, the bouncing up and down rate are not as high as batch size 16 but it still are not stable. As for its validation accuracy, it remains constant with the accuracy of 47.5% like batch size 16. The training accuracy are not improving throughout the epochs iteration for both of batch size 16 and 32.

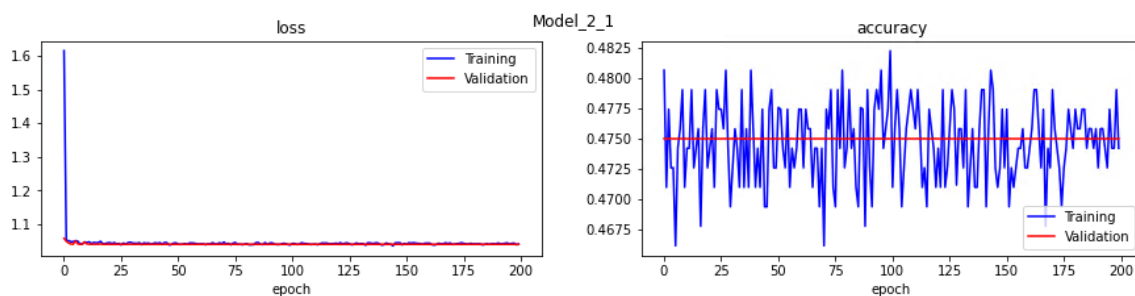


Figure 4.18 Epoch and Prediction Graph Result with Batch Size 16

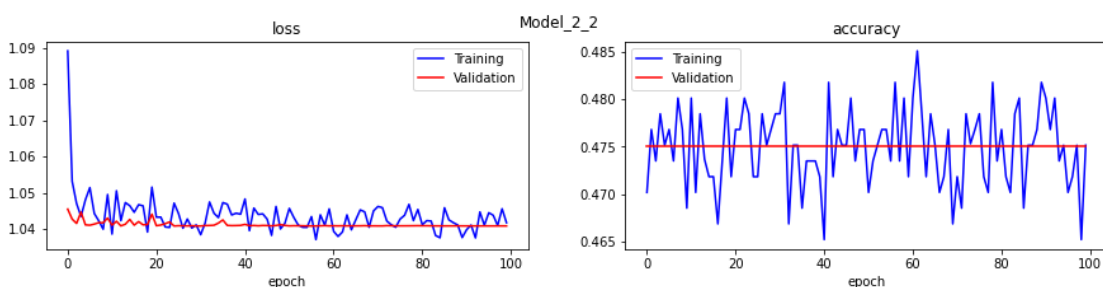


Figure 4.19 Epoch and Prediction Graph Result with Batch Size 32

From these two batch 16 and 32, we take out the best hyperparameter for the model and did some comparison testing on the test data which the result shows in figure 4.20 and 4.21. We found out that the accuracy for both batch 16 and 32 model they are the same which is 100% and for the loss rate also very high which is more than 1.0. As the testing accuracy should not be higher than the training accuracy. By combining the result from the training accuracy and

the testing accuracy we can conclude that the dataset are not suitable for VGG16 model. So we will choose ResNet-like model for our dog image emotion recognition.

```
▶ model_file_name = 'Model_2_1-200.hdf5'
chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
print('Choosing the model via early stopping')
test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
3/3 [=====] - 28s 9s/step - loss: 1.0408 - accuracy: 1.0000
Test: accuracy = 1.000000 ; loss = 1.040804
```

Figure 4.20 Prediction Result on Test Data of the best model with Batch Size 16

```
▶ model_file_name = 'Model_2_2-100.hdf5'
chosen_model = load_model(os.path.join('best_models_hdc',model_file_name))
print('Choosing the model via early stopping')
test_loss,test_accuracy = chosen_model.evaluate(X_test, y_test)
print("Test: accuracy = %f ; loss = %f" % (test_accuracy, test_loss))

Choosing the model via early stopping
3/3 [=====] - 29s 10s/step - loss: 1.0408 - accuracy: 1.0000
Test: accuracy = 1.000000 ; loss = 1.040798
```

Figure 4.21 Prediction Result on Test Data of the best model with Batch Size 32

Now its time to test our chosen ResNet-like model which is our best model right now. We will test on some sample dog image to predict the dog emotion. The result is great as it able to predict the dog emotion correctly. The first row, first Ricky the dog image shows that the prediction result predicts it correctly as Ricky is happy when he is in the park. The next image shows Ricky is angry as he cannot go to the park for a walk. The third image shows that Ricky is happy on the grass when he is playing on the field of flower watching his owner. Starting from the second row. It shows that Ricky is happy when the owner bring him to the park again. The next image shows that Ricky is sick but actually this should be happy as when he saw his owner taking picture of him and the last one is that Ricky is sick when the owner left him alone at home this might be a sign of depress. So from all these result, all the 6 results there is only 1 image that is predicted incorrectly and the others predicted the data correctly.

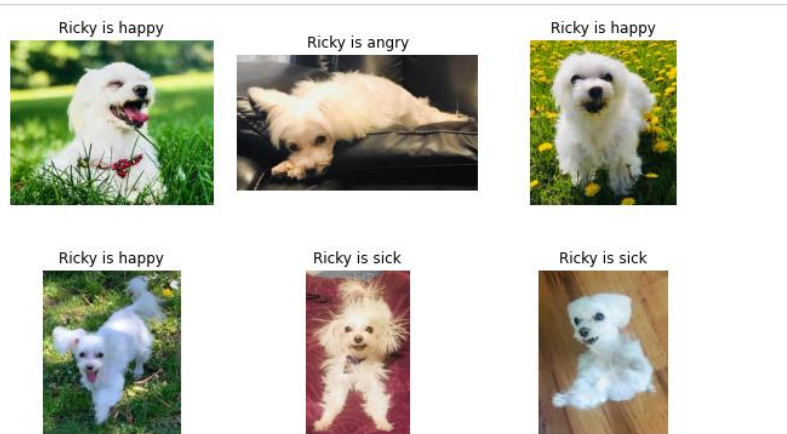


Figure 4.22 Prediction Result on Ricky the Dog Image with the Chosen Model

4.2 Dog Bark Emotion Recognition Model

Before we start training a dog bark emotion recognition mode, first we need to gather some dog bark dataset from the internet. In this case we will get the dog bark audio files from Google AudioSet. Figure 4.23 shows the part of YouTube video that have the following barking sound and the barking sound had been separated into different label.

Dog

Any sounds coming from the familiar domesticated canid which has been selectively bred over millennia for companionship, protection, as well as for superior sensory capabilities, and other useful behaviors.

[13,705 annotations in dataset](#)



Bark

Principal communication sound produced by dogs. Often transliterated as woof, especially for large dogs.

[2,632 annotations in dataset](#)



Yip

A sharp high-pitched bark or cry, typically from a miniature dog.

[2,363 annotations in dataset](#)



Figure 4.23 Dog Bark Audio Dataset in Google AudioSet

Now we need to download the following dataset that contains the YouTube video URL and the particular time that will contains the dog bark sound. Figure 4.24 shows that how the dataset looks like in there. 1st column will be the YouTube video ID, 2nd row will be the starting second, 3rd row will be the ending second, and the last column will be the dog barking label that predicted.

CHAPTER 4 PRELIMINARY WORK

A	B	C	D	E	F	G	H	I	J	K	L
# Segments csv created Sun Mar 5 10:54:31 2017											
# num_ytic	num_segs	num_uniq	num_positive_labels=52882								
# YTID	start_seco	end_seco	positive_labels								
#NAME?	30	40	"/m/09x0i/t/dd00088"								
#NAME?	50	60	"/m/012xff"								
#NAME?	0	10	"/m/03fw/m/04rlf /m/09x0r"								
#NAME?	30	40	"/t/dd000 /t/dd00005"								
#NAME?	200	210	"/m/032sl/m/073cg4"								
#NAME?	30	40	"/m/01y3hg"								
#NAME?	30	40	"/m/015lz/m/07pws3f"								
-ODLPzsiXX	30	40	"/m/04rlf /m/07qwdck"								
-ODdlOulFI	50	60	"/m/0130 /m/02jz0l /m/0838f"								
-OFHUc78C	30	40	"/m/02w4/m/04rlf"								
-003e95y4	100	110	"/m/07r4\ /t/dd00125"								
-0SdAVK79	30	40	"/m/0155 /m/01lyv /m/0342h /m/042v_ε /m/04rlf /m/04szw /m/07s0s5 /m/0fx80y /m/0gg8l"								
-0Vl4HyWl	410	420	"/m/085jv /m/0114l2"								
-0mG4W5l	270	280	"/m/04rlf /m/05fw6 /m/07r4k7 /m/09x0r /m/0ygtg"								
-0mjrMpos	80	90	"/m/04zmvq"								
-11LhdJgBl	30	40	"/m/04rlf /m/07qn4z3"								
-1LrH01Eid	30	40	"/m/02p0 /m/04rlf"								
-1TLtjPtnm	10	20	"/m/03lty /m/04rlf /m/07szfh9"								
-1iKlvsRBt	80	90	"/m/015lz /m/07pws3f"								
-24dqQM_	30	40	"/m/04rlf /t/dd00003"								
-275_wTLr	7	17	"/m/07q6cd_"								

Figure 4.24 Dog Bark Audio Dataset From Google AudioSet

To extract the particular dog barking label first we need to know the exact codename for our required label. First, we need to check out the class labels csv file and record down the dog class label indices that we need. In this case we will use dog's growling as an example.

```

C:\Users\Nameless\Downloads\class_labels_indices.csv - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
balanced_train_segments.csv x class_labels_indices.csv x
57 55,/m/0/pdhp0,"Biting"
58 56,/m/0939n_,"Gargling"
59 57,/m/01g90h,"Stomach rumble"
60 58,/m/03q5_w,"Burping, eructation"
61 59,/m/02p3nc,"Hiccup"
62 60,/m/02_nn,"Fart"
63 61,/m/0k65p,"Hands"
64 62,/m/025_jnm,"Finger snapping"
65 63,/m/0115bq,"Clapping"
66 64,/m/01jg02,"Heart sounds, heartbeat"
67 65,/m/01jg1z,"Heart murmur"
68 66,/m/053hz1,"Cheering"
69 67,/m/028ght,"Applause"
70 68,/m/07rkbfbh,"Chatter"
71 69,/m/03qtwd,"Crowd"
72 70,/m/07qfr4h,"Hubbub, speech noise, speech babble"
73 71,/t/dd00013,"Children playing"
74 72,/m/0jkb,"Animal"
75 73,/m/068hy,"Domestic animals, pets"
76 74,/m/0bt9lr,"Dog"
77 75,/m/05tny_,"Bark"
78 76,/m/07r_k2n,"Yip"
79 77,/m/07qf0zm,"Howl"
80 78,/m/07rc7d9,"Bow-wow"
81 79,/m/0ghcn6,"Growling"
82 80,/t/dd00136,"Whimper (dog)"

```

Figure 4.25 Dog Bark Audio Dataset Class Label Indices

CHAPTER 4 PRELIMINARY WORK

Then, we will use grep command to get all the rows that contain the growling class label indices and save it into a file.

```
nameless@DESKTOP-702L6AC:/mnt/c/Users/Nameless/Downloads$ cat balanced_train_segments.csv | grep "/m/0ghcn6" | wc -l
61
nameless@DESKTOP-702L6AC:/mnt/c/Users/Nameless/Downloads$ cat balanced_train_segments.csv | grep "/m/0ghcn6" | head
-ODzHayGHvk, 50.000, 60.000, "/m/0ghcn6"
-qJ-NY6wY3s, 30.000, 40.000, "/m/0bt91r,/m/0ghcn6,/m/0j bk"
-skhG36uWM0, 20.000, 30.000, "/m/01z5f,/m/068hy,/m/09x0r,/m/0bt91r,/m/0ghcn6"
107poxGGFws, 110.000, 120.000, "/m/09x0r,/m/0ghcn6"
4sFG0VcvM8, 20.000, 30.000, "/m/068hy,/m/09x0r,/m/0bt91r,/m/0ghcn6,/m/0j bk"
5aC9_oX8dr0, 460.000, 470.000, "/m/0ghcn6"
6qmGCfCrU6w, 20.000, 30.000, "/m/01z5f,/m/068hy,/m/0bt91r,/m/0ghcn6,/m/0j bk"
72wtqWYGkGA, 10.000, 20.000, "/m/0ghcn6,/t/dd00125"
9qtSLOFY1Aw, 40.000, 50.000, "/m/068hy,/m/07qf0zm,/m/0bt91r,/m/0ghcn6,/m/0j bk"
9yaibWt53S0, 60.000, 70.000, "/m/0bt91r,/m/0ghcn6,/m/0j bk"
nameless@DESKTOP-702L6AC:/mnt/c/Users/Nameless/Downloads$
```

Figure 4.26 Using Grep to Extract Rows Containing Growling Class Label Indices

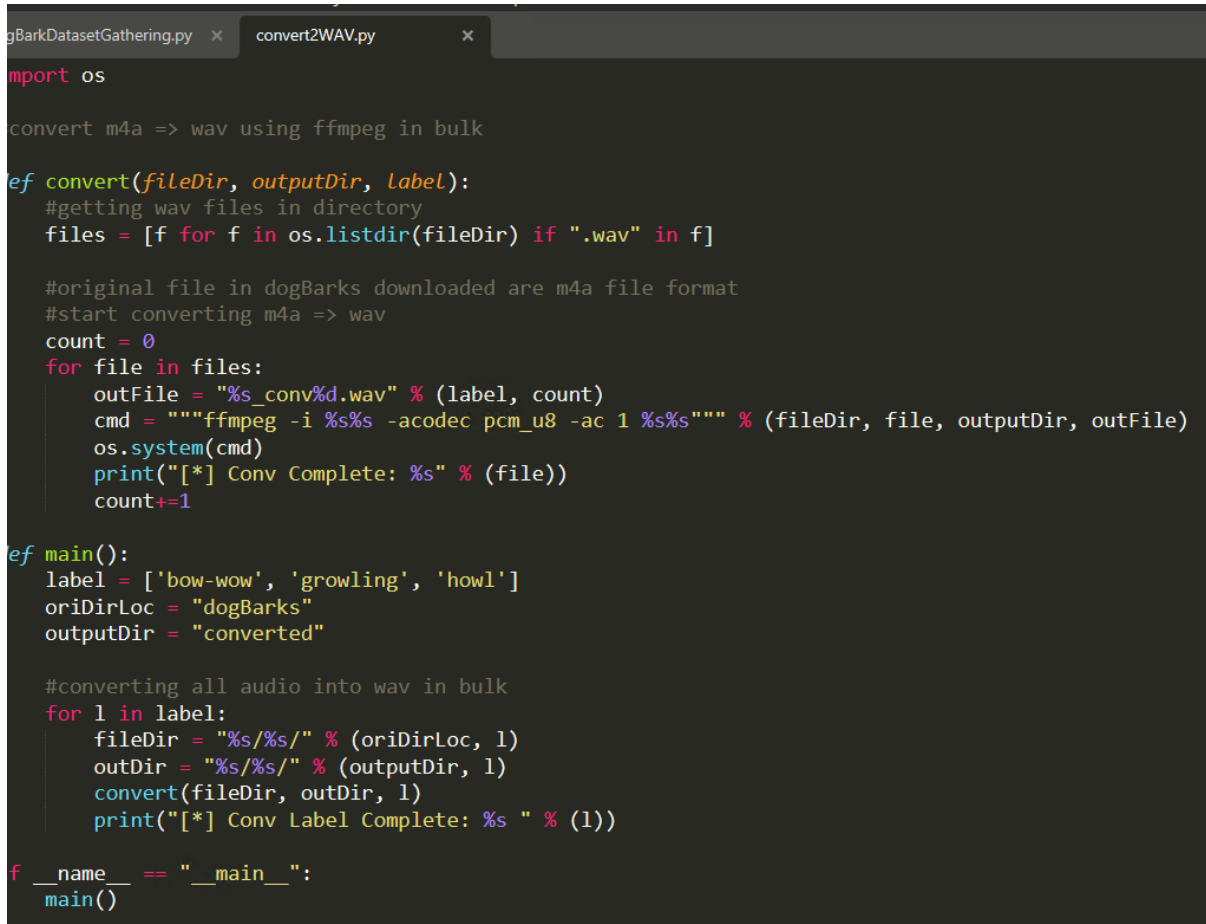
Now we will create a Python script to help us automate downloading all the required YouTube videos. This Python script will download the video based on the start time and end time provided. After downloading the video, it will convert the video into audio file format.

```
dogBarkDatasetGathering.py x
3 =====
4 /m/0ghcn6 => growling #angry dog
5 /m/07qf0zm => howl #get ur dog to the vet (sick dog)
6 /m/07rc7d9 => Bow-wow #happy/communicating dog (happy dog)
7 ...
8
9 #download youtube videos
10 def downloadAudio(YTID, startTime, endTime, clabel, dlCount):
11     #init param
12     #youtube video URL
13     videoID = YTID
14     videoURL = "https://www.youtube.com/embed/" + videoID
15
16     #output directory
17     dataDir = "dogBarks"
18     filename = "%s_%d.wav" % (clabel,dlCount)
19
20     #convert time range (second) => hh:mm:ss format (for ffmpeg postprocessing)
21     startTime = str(datetime.timedelta(seconds=startTime))
22     endTime = str(datetime.timedelta(seconds=endTime))
23
24     #youtube-dl param setting
25     #reference: youtube-dl options => https://github.com/yt-dl-org/youtube-dl/blob/master/youtube_dl/You
26     ydl_opts = {
27         'format': 'bestaudio/best',
28         'outtmpl': """"%s/%s/%s"""" % (dataDir,clabel,filename), #save output file name in specific label
29         'postprocessors': [{
30             'key': 'FFmpegExtractAudio',
31             'preferredcodec': 'wav',
32             'preferredquality': '192'
33         }],
34         'postprocessor_args': [ #FFMPEG arg: getting the dog bark audio based on the dataset startTime
```

Figure 4.27 Python Script to Gather Dog Audio Dataset

CHAPTER 4 PRELIMINARY WORK

After done downloaded all the audio files, we need to convert the audio file into wav file format. To do this we create a Python script that will helps us in auto converting all the audio files. This Python script will use FFMPEG command to convert the audio file into wav audio file format.



```
import os

convert m4a => wav using ffmpeg in bulk

def convert(fileDir, outputDir, label):
    #getting wav files in directory
    files = [f for f in os.listdir(fileDir) if ".wav" in f]

    #original file in dogBarks downloaded are m4a file format
    #start converting m4a => wav
    count = 0
    for file in files:
        outFile = "%s_conv%d.wav" % (label, count)
        cmd = "ffmpeg -i %s%s -acodec pcm_u8 -ac 1 %s%s" % (fileDir, file, outputDir, outFile)
        os.system(cmd)
        print("[*] Conv Complete: %s" % (file))
        count+=1

def main():
    label = ['bow-wow', 'growling', 'howl']
    oriDirLoc = "dogBarks"
    outputDir = "converted"

    #converting all audio into wav in bulk
    for l in label:
        fileDir = "%s/%s/" % (oriDirLoc, l)
        outDir = "%s/%s/" % (outputDir, l)
        convert(fileDir, outDir, l)
        print("[*] Conv Label Complete: %s " % (l))

if __name__ == "__main__":
    main()
```

Figure 4.28 Python Script to Convert Audio File into WAV Format

Now we need to extract the exact part of audio spectrum that containing the required dog bark sound that we need. In our case we need to gather 3 class labels which will be bow-wow, growling, and howl. First, we try to use Audacity to study that which type of audio spectrum pattern that match the class labels that we need after we found the pattern, we will make a copy of the audio spectrum part and save it into a new audio file.

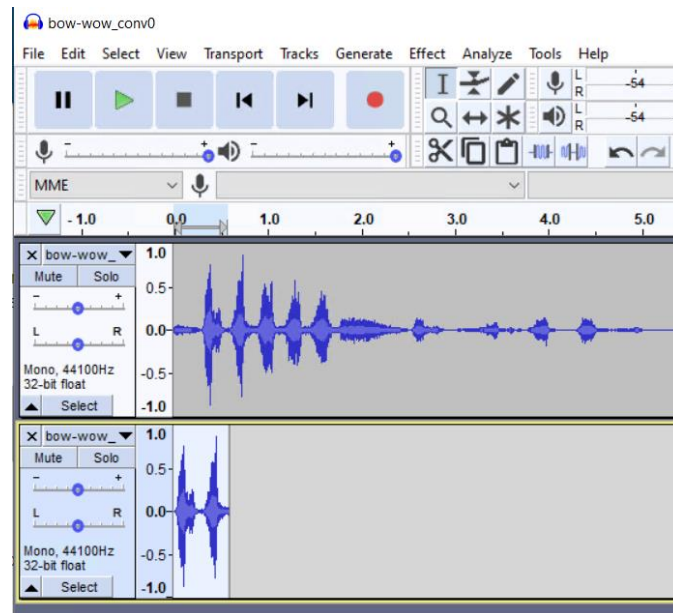


Figure 4.29 Using Audacity to Extract Specific Dog Bark Audio Spectrum

After done editing all the audio files using Audacity, now we will be loading some audio files into the Jupyter Notebook. First let's take a look on how the audio spectrum looks like for each class label. For bow-wow class label, we can see that there are 2 audio spectrums with a gap between the barking sound. For growling, the audio spectrum will be bouncing up and down due to the vibration sound that made by the dog. For howling class label, the audio spectrum will remain constant when the dog howl.

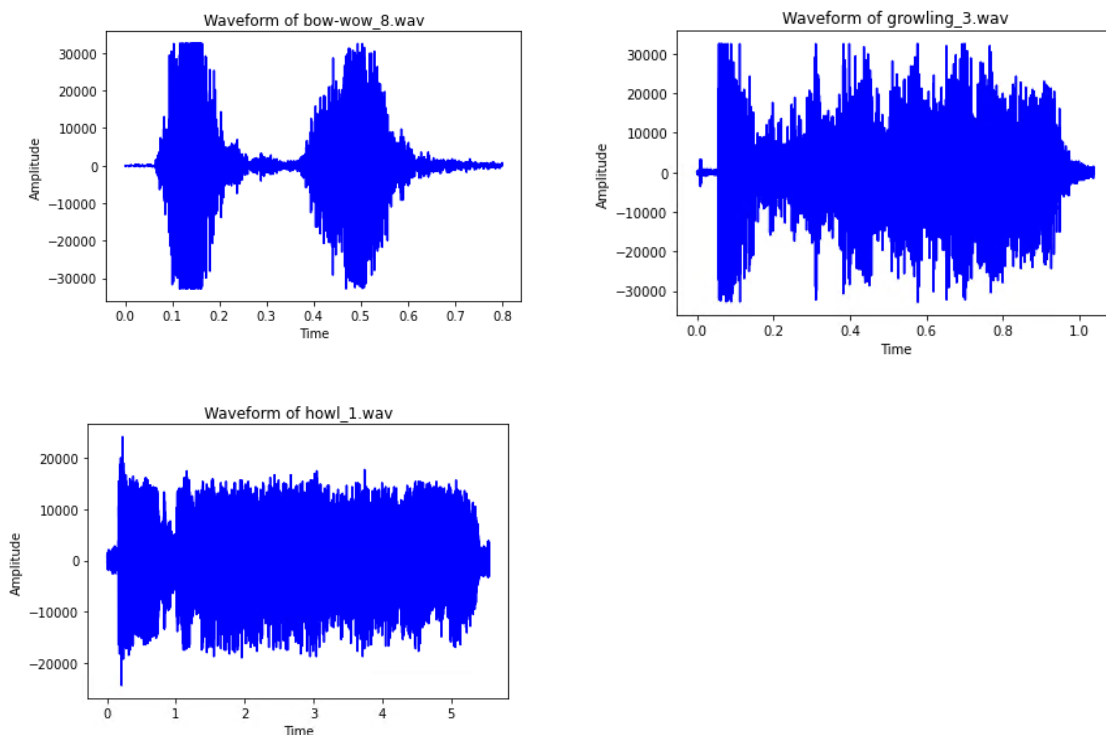


Figure 4.30 Different Audio Spectrum for each class label

CHAPTER 4 PRELIMINARY WORK

After done visualizing the data, we will start our loading and pre-processing data phase. The first block of code will be the function use to extract the audio feature from the dog bark audio file. Second block of code will iterate all the directories load the audio files, and call the audio feature extractor function to extract the audio feature.

```
In [82]: ▶ def audioFeatureExtractor(filePath):
          audio, sample_rate = librosa.load(filePath, res_type='kaiser_fast')
          mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
          mfccs_scaled_features = np.mean(mfccs_features.T,axis=0) #scaling

          return mfccs_scaled_features
```

Start iteration extracting audio features

```
In [83]: ▶ dogBarksAudioDir = "editedBarks"
          labels = ['bow-wow', 'growling', 'howl']
          extracted_features = []

          #extracting audio features from each labelled directory with audio
          for label in labels:
              fileDir = "%s/%s/" % (dogBarksAudioDir, label)
              #getting wav files in directory
              files = [f for f in os.listdir(fileDir) if ".wav" in f]

              for file in files:
                  filePath = '%s%s' % (fileDir, file)
                  data = audioFeatureExtractor(filePath)
                  extracted_features.append([data, label])
```

Figure 4.31 Code to Extract Audio Features

After done, we will create a Pandas dataframe and insert the extracted audio features and class label for each rows. Then we will split the dataset in Pandas dataframe into 2 individual dataset for X and y. X will be the audio features, y will be the class labels.

```
▶ extracted_features_df = pd.DataFrame(extracted_features, columns=['feature', 'class'])
#peak 5 results
extracted_features_df.head()
```

```
84]:
```

	feature	class
0	[-172.25468, -17.27532, -24.850897, -23.064583...	bow-wow
1	[-70.43721, -12.256557, -40.769714, -29.876968...	bow-wow
2	[-138.35191, 160.90584, -82.907875, -21.042604...	bow-wow
3	[-168.46394, 92.96472, -107.08077, -9.013321, ...	bow-wow
4	[-132.16565, 147.04985, -96.683266, -12.094917...	bow-wow

Split the dataset into independent and dependent dataset

```
▶ X=np.array(extracted_features_df['feature'].tolist())
  y=np.array(extracted_features_df['class'].tolist())
```

Figure 4.32 Load Into Pandas and Separate into 2 Independent Dataset

Now, we will use label encoder to convert the y class label into numeric form.

Label Encoder: converting the labels into numeric form

```
In [89]: ▶ from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
y = to_categorical(labelencoder.fit_transform(y))
```

Figure 4.33 Code to Convert Class Labels into Numeric Form Using Label Encoder

After we have done the pre-processing phase, now we need to build our model. In this case we will create a sequential model for our dog bark audio recognition. In figure 4.34 will be the code on how to create the sequential model.

```
-----
: ▶ num_labels=y.shape[1] #no. of classes
: ▶ model=Sequential()
    ###first layer
    model.add(Dense(100,input_shape=(40,)))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###second layer
    model.add(Dense(200))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))
    ###third layer
    model.add(Dense(100))
    model.add(Activation('relu'))
    model.add(Dropout(0.5))

    ###final layer
    model.add(Dense(num_labels))
    model.add(Activation('softmax'))
```

Figure 4.34 Code to Build Sequential Model

Now, we have our Sequential model created, we will start training our dog bark classification model. In this case we will find the best epochs for our dog bark classification model. We will set the highest epochs number to be 100, and the batch size will be 32. Based on the observation in figure 4.35, we can see that the epoch of 1 validation loss had been improved but the epochs starting from 2 to 100 it does not improve anymore. So, this means that the best epoch number for our model will be 1. The model file for the best epoch will be created and save in hdf5 format.

```

|: ▶ num_epochs = 100
      num_batch_size = 32

      model.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimizer='adam')

      checkpointer = ModelCheckpoint(filepath='best_models_hdc/dog_barks_classification.hdf5',
                                   verbose=1, save_best_only=True)

      hist = model.fit(X_train, y_train,
                      batch_size=num_batch_size,
                      epochs=num_epochs,
                      validation_data=(X_test, y_test),
                      callbacks=[checkerpointer],
                      verbose=1)

```

```

Epoch 1/100
1/2 [=====>.....] - ETA: 0s - loss: 6.6376e-05 - accuracy: 1.0000
Epoch 00001: val_loss improved from inf to 0.81593, saving model to best_models_hdc\dog_barks_classification.hdf5
2/2 [=====] - 1s 363ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.8159 - val_accuracy:
0.8333
Epoch 2/100
1/2 [=====>.....] - ETA: 0s - loss: 1.1176e-08 - accuracy: 1.0000
Epoch 00002: val_loss did not improve from 0.81593
2/2 [=====] - 0s 17ms/step - loss: 1.2691e-06 - accuracy: 1.0000 - val_loss: 1.0914 - val_accura
cy: 0.8333
Epoch 3/100
1/2 [=====>.....] - ETA: 0s - loss: 2.2352e-08 - accuracy: 1.0000
Epoch 00003: val_loss did not improve from 0.81593
2/2 [=====] - 0s 11ms/step - loss: 1.4653e-07 - accuracy: 1.0000 - val_loss: 1.3792 - val_accura
cy: 0.8333
Epoch 4/100
1/2 [=====>.....] - ETA: 0s - loss: 0.0353 - accuracy: 0.9688
Epoch 00004: val_loss did not improve from 0.81593
2/2 [=====] - 0s 13ms/step - loss: 0.0235 - accuracy: 0.9792 - val_loss: 1.4406 - val_accuracy:
0.8333

```

Figure 4.35 Finding Best Epoch for Dog Bark Classification model

Now we will test on our model accuracy. Based on the observation it seems that our test accuracy for dog bark classifier are 75% which is good.

Test accuracy for the model

```

[124]: ▶ test_accuracy=model.evaluate(X_test,y_test,verbose=0)
        print("[*] Test Accuracy: " + str(test_accuracy[1]))

```

```

[*] Test Accuracy: 0.75

```

Figure 4.36 Test Accuracy Result for Dog Bark Classifier Model

The last part we will try using the model to predict on the dog barking audio test data. In figure 4.37 1st image, we use a bow-wow audio to test with our dog bark classifier model, and it predicted correctly. 2nd image, we use growling audio to test with our dog bark classifier model and it predicted correctly again. The last image we will test with howl audio with our dog bark classifier model and it predicted correctly. So for these 3, it had predicted the test dog barking audio data correctly.

CHAPTER 4 PRELIMINARY WORK

```
labels = [ 'bow-wow', 'growling', 'howl' ]

for i in range(3):
    filePath = "%s%s%d%s" % (dogBarksAudioDir, bow_wow, i, ".wav")

    audio, sample_rate = librosa.load(filePath, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0) #scaling
    data = mfccs_scaled_features.reshape(1,-1)

    ipd.display(ipd.Audio("%s" % (filePath)))

    predicted_label = chosen_model.predict_classes(data)

    print("[*] Predicted Label: " + labels[predicted_label[0]])
```

▶ 0:00 / 0:00

[*] Predicted Label: bow-wow

```
for i in range(3):
    filePath = "%s%s%d%s" % (dogBarksAudioDir, growling, i, ".wav")

    audio, sample_rate = librosa.load(filePath, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0) #scaling
    data = mfccs_scaled_features.reshape(1,-1)

    ipd.display(ipd.Audio("%s" % (filePath)))

    predicted_label = chosen_model.predict_classes(data)

    print("[*] Predicted Label: " + labels[predicted_label[0]])
```

▶ 0:00 / 0:00

[*] Predicted Label: growling

```
for i in range(3):
    filePath = "%s%s%d%s" % (dogBarksAudioDir, howl, i, ".wav")

    audio, sample_rate = librosa.load(filePath, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T,axis=0) #scaling
    data = mfccs_scaled_features.reshape(1,-1)

    ipd.display(ipd.Audio("%s" % (filePath)))

    predicted_label = chosen_model.predict_classes(data)

    print("[*] Predicted Label: " + labels[predicted_label[0]])
```

▶ 0:00 / 0:00

[*] Predicted Label: howl

Figure 4.37 Prediction Result on Test Data using Trained Dog Bark Classifier Model

4.3 Combining Dog Image Emotion Classifier and Dog Bark Audio Classifier Model

To improve the dog emotion classifier, we will use weighted average technique to combine the 2 models prediction result which is dog image emotion classifier and dog bark audio classifier model. First in figure 4.38, we need to load the dog image emotion classifier and dog bark audio classifier model.

```

: | dogBarkModel = 'dog_barks_classification.hdf5'
  dogEmotionModel = 'dog_emotion_recognition.hdf5'

  dogBarkLoadedmodel = load_model(os.path.join('model',dogBarkModel))
  dogEmotionLoadedmodel = load_model(os.path.join('model',dogEmotionModel))

```

Figure 4.38 Loading 2 Models

Then, we will define the functions to do prediction on image and audio file.

Defining Functions

```

|: | def dogEmotionPred(filepath):
  img_rows = 224
  img_cols = 224

  img = cv2.imread(filepath)
  img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)

  img = cv2.resize(img,(img_rows,img_cols)).astype('float32')
  img /= 255
  img = np.expand_dims(img,axis=0)
  predict_label = dogEmotionLoadedmodel.predict(img)

  return predict_label

|: | def dogBarkPred(filepath):
  audio, sample_rate = librosa.load(filepath, res_type='kaiser_fast')
  mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
  mfccs_scaled_features = np.mean(mfccs_features.T,axis=0) #scaling
  data = mfccs_scaled_features.reshape(1,-1)

  #predicted_label = chosen_model.predict_classes(data)
  predicted_label = dogBarkLoadedmodel.predict(data)

  return predicted_label

```

Figure 4.39 Functions Used to Do Prediction

Now we need to create a weighted average calculation function. We set higher weight for the dog image recognition model because the image recognition will have higher accuracy comparing to the dog bark recognition model.

```

def average(a, b):
    combinedList = []
    combinedList.append(a.tolist())
    combinedList.append(b.tolist())

    dogPredArr = np.array(combinedList)
    predAvg = np.average(dogPredArr, axis=0, weights=[7,3])

    return predAvg

```

Figure 4.40 Functions Used to Do Weighted Average Calculation

After done initializing the functions, we will do some testing on a test image and audio. Based on the figure 4.41 shows, the 1st row will explain on how does the data in 3 column means here, 1st column will be happy label, 2nd will be angry label, 3rd will be sick label. As for audio recognition previously we set the label into bow-wow, growling, and howling category, actually bow-wow is equivalent to happy, growling means angry, and howling means the dog is sick. So it means that both classification model have the same label output. Let's study the before average result, as you can see that over here it will calculate the prediction result for all the class labels for both model prediction. The highest value will be the prediction result. To merge the 2-prediction result, we use weighted average technique to combine them. The after average row shows the calculation after combining both prediction result from dog image emotion classifier and dog bark emotion classifier. The last row will shows the class label that is predicted from the weighted average calculated prediction result.

```

output meaning => [happy      angry    sick]

Before Average
=====
Dog Emotion Prediction: [0.997417, 0.0004495901, 0.0021334458]
Dog Bark Prediction: [1.0, 3.5093627e-33, 1.1245187e-23]

After Average
=====
Prediction (image + audio model): [0.9981918811798096, 0.000314713065745309, 0.0014934120699763298]

[*] Dog is happy

```

Figure 4.41 Prediction Result Before and After Weighted Average

Now we will test and see the accuracy of the prediction result before and after weighted average. In figure 4.42 will be the predicted accuracy graphs before weighted average and figure 4.43 will be the predicted accuracy graphs after weighted average. Based on the observation, after weighted average we can see that the incorrect class label the accuracy it will reduce their prediction accuracy which will remains the correct class label accuracy higher. This means that

CHAPTER 4 PRELIMINARY WORK

with weighted average technique combining 2 models it will improve the accuracy. We will proceed another experiment to proof that this technique works.

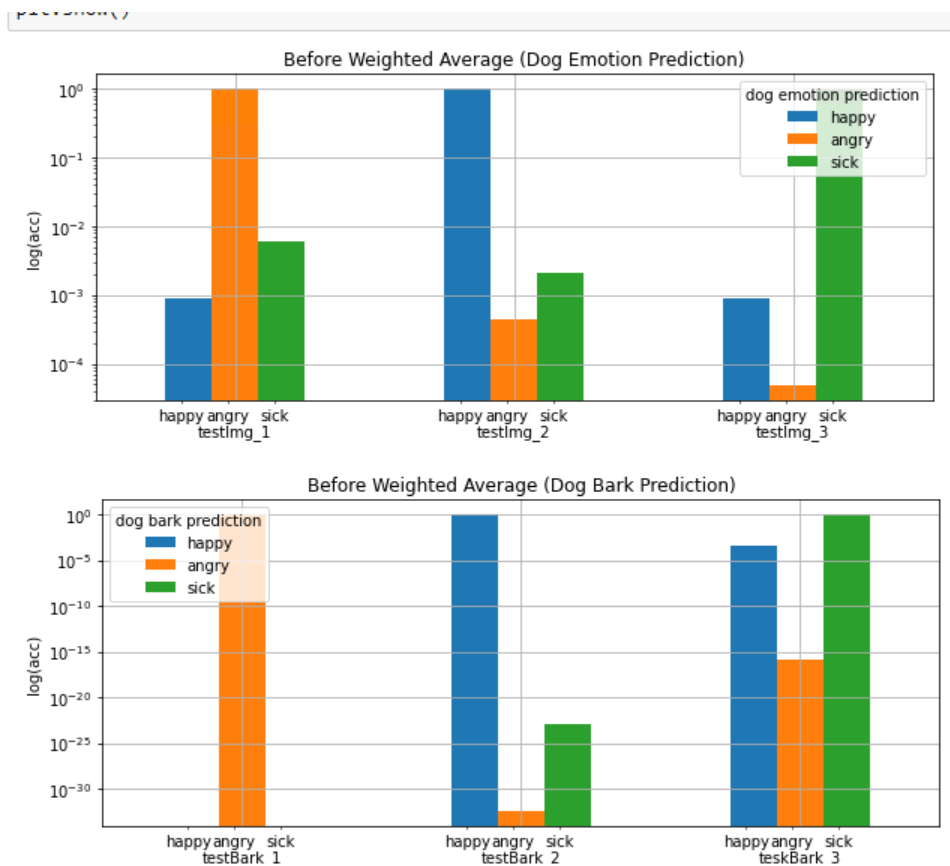


Figure 4.42 Prediction Accuracy and Class Label Graphs Before Weighted Average

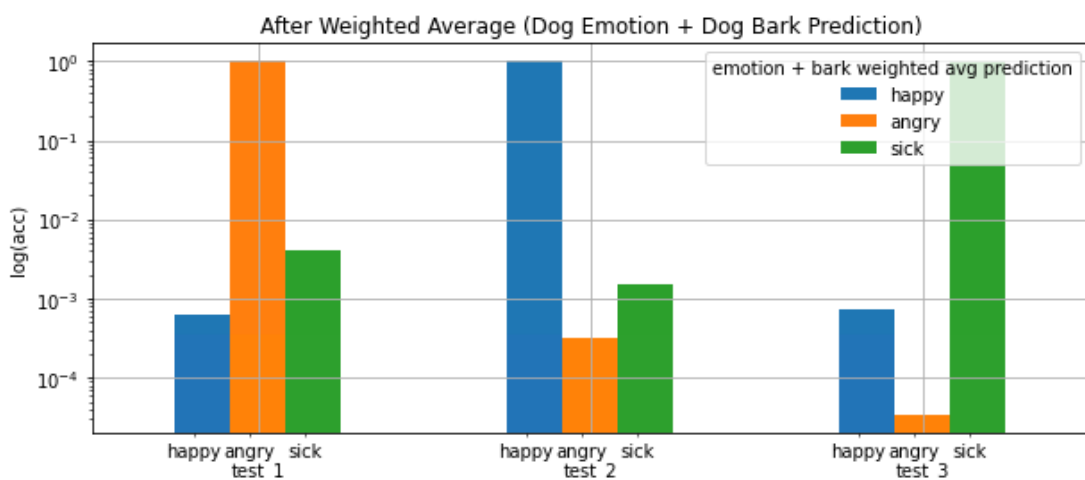


Figure 4.43 Prediction Accuracy and Class Label Graphs After Weighted Average

Now, we will do some benchmarking accuracy before and after weighted average. We will use 210 sample data with 70 sample data for each class labels. In figure 4.44, we can see that for the dog image emotion classifier out of 210 sample data it able to predict 187 sample data correctly. In figure 4.45, the dog bark audio classifier out of 210 sample data it able to predict 192 sample data correctly. So after weighted average both the model prediction value, we will be able to achieve 201 sample data predict correctly which means that we can conclude using this weighted average to combine 2 models prediction result, we will get higher correct prediction accuracy.

```
: ▶ dogEmotionAcc = accCalc(correctPred, len(extracted_featuresImg))
print("[*] Correct Predicted: " + str(correctPred))
print(dogEmotionAcc)

[*] Correct Predicted: 187
0.8904761904761904
```

Figure 4.44 Prediction Accuracy For Dog Image Emotion Classifier

```
▶ dogBarkAcc = accCalc(correctPred, len(extracted_featuresAudio))
print("[*] Correct Predicted: " + str(correctPred))
print(dogBarkAcc)

[*] Correct Predicted: 192
0.9142857142857143
```

Figure 4.45 Prediction Accuracy For Dog Bark Audio Classifier

```
▶ avgDogPredAcc = accCalc(correctPred, 210)
print("[*] Correct Predicted: " + str(correctPred))
print(avgDogPredAcc)

[*] Correct Predicted: 201
0.9571428571428572
```

Figure 4.46 Prediction Accuracy After Weighted Average Both Models

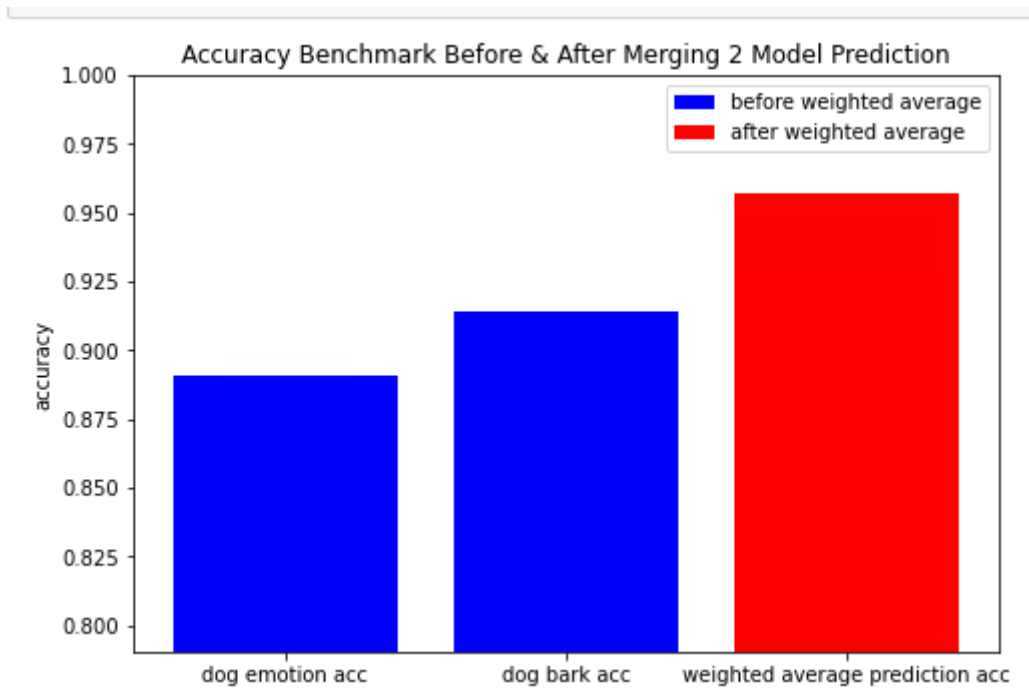


Figure 4.47 Prediction Accuracy Graph Before and After Weighted Average

CHAPTER 4 PRELIMINARY WORK

Now let's test using weighted average technique to predict on our 3 sample data. In figure 4.48, we can see that it successfully predicted our sample data correctly. So, which means that we have proof that using weighted average technique to combine 2 models will improve our prediction accuracy.

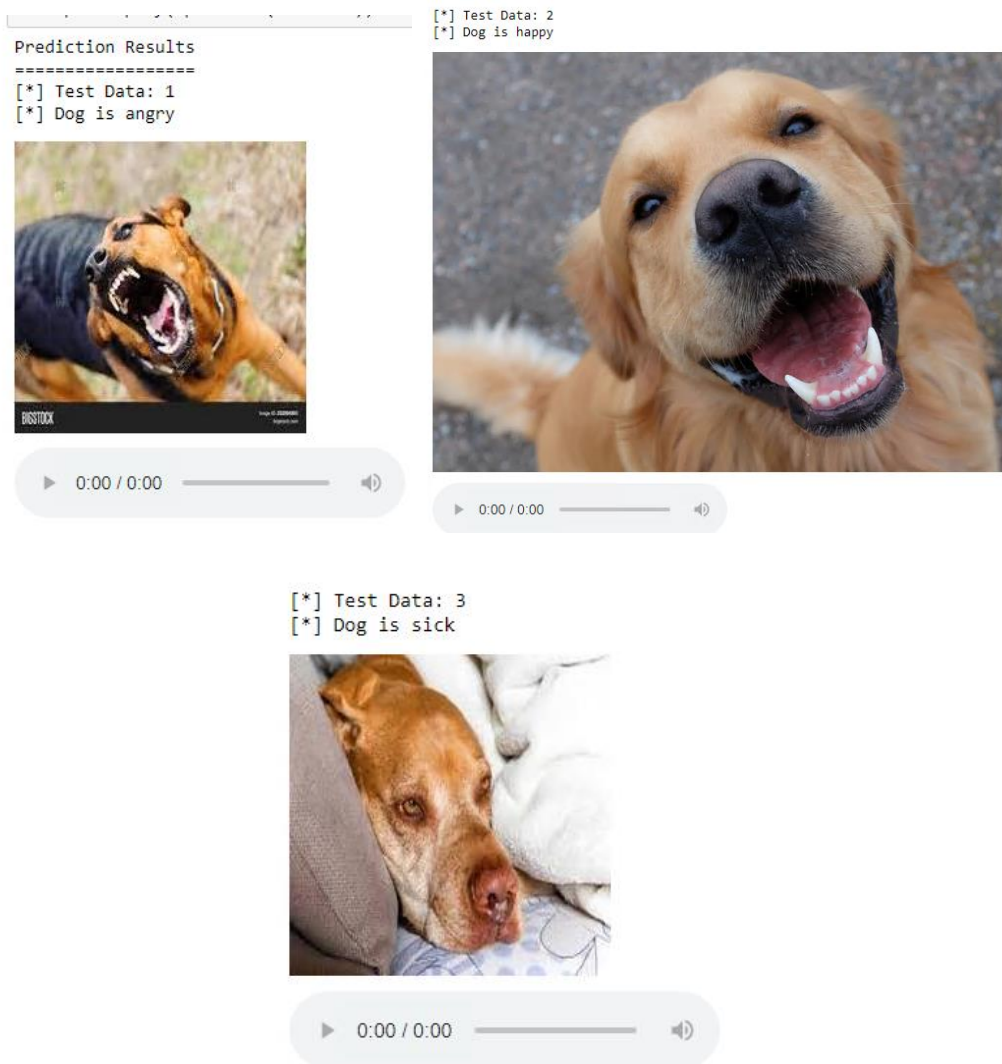


Figure 4.48 Predicted Label Using Weighted Average Technique

4.4 Google Cloud Platform

In the Google Cloud Platform, we have created a Linux VM instance in us-central1-a zone, then the external IP “35.193.203.35” has been set as a static IP for the VM instance. Figure 4.49 shows the Linux VM profile with the Internal IP and the External IP in Google Cloud Platform.

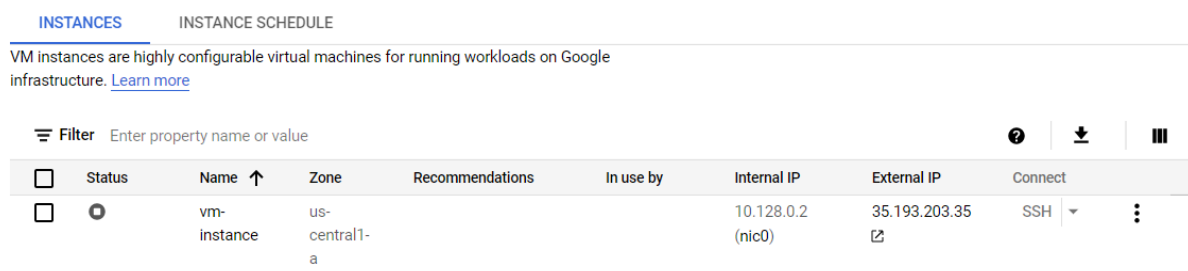


Figure 4.49 Linux VM Instance Profile

In the Linux VM instance, there will be 1 directory name “cloudFiles” which includes Node.js REST API script, Python dogEmotionClassifier.py and others files that shows in figure 4.50. In the model directory that shows in figure 4.51 there will be the models for dog image emotion classifier and the dog bark audio classifier that will be used for dog emotion recognition later.

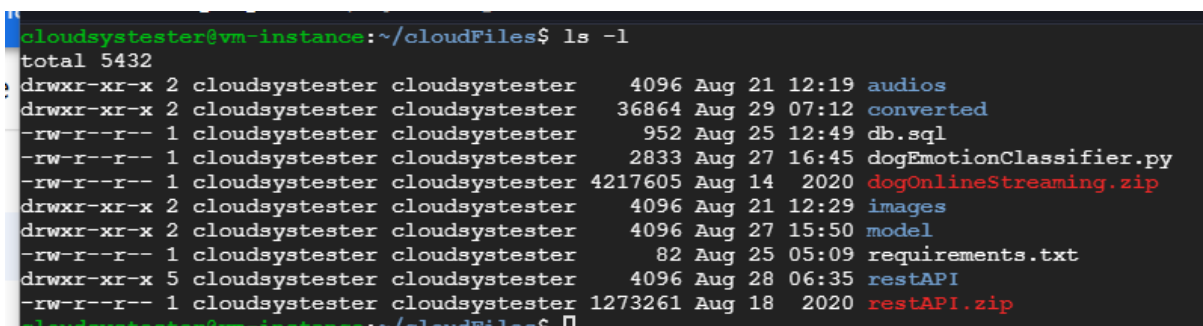


Figure 4.50 Dog Emotion Recognition Project Directory in Linux VM GCP Instance

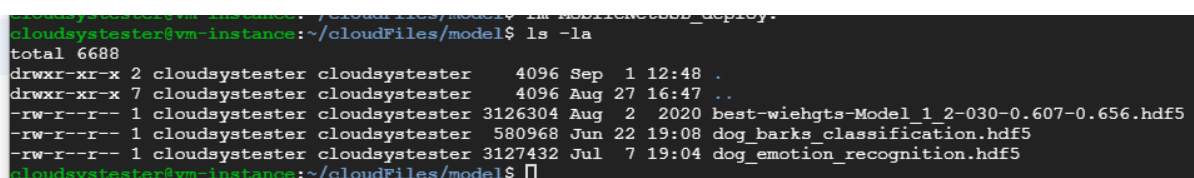
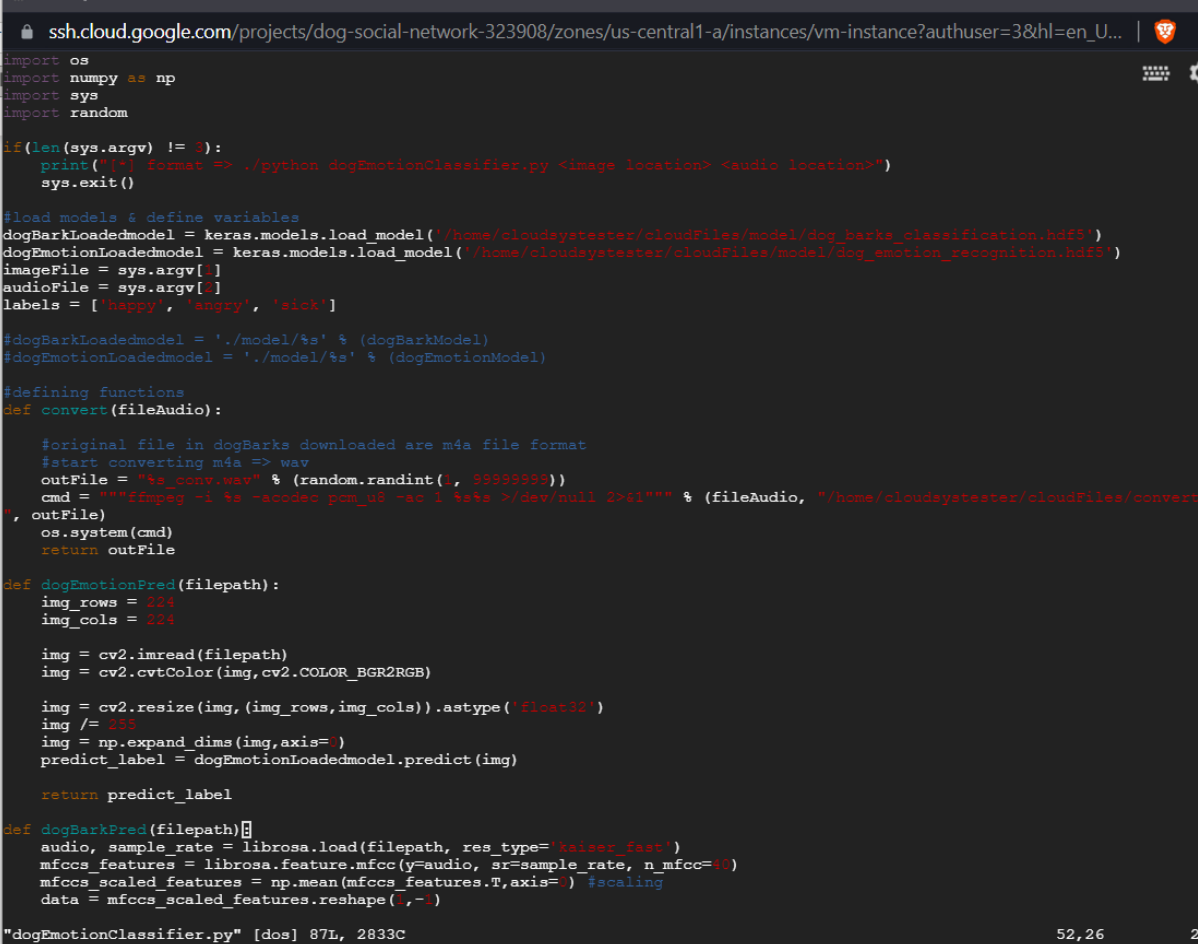


Figure 4.51 Dog Emotion Recognition Model Directory in Linux VM GCP Instance

Now let’s take a look at the dogEmotionClassifier.py Python script. This script will be used to predict the dog emotion recognition. In figure 4.52, this is the script to perform the dog emotion recognition. The script will be loading the model from the model directory, and then perform

CHAPTER 4 PRELIMINARY WORK

prediction for the dog image. In figure 4.53, that will be the output of the predicted result when executing the script to predict Ricky Dog image and barking sound.



```
ssh.cloud.google.com/projects/dog-social-network-323908/zones/us-central1-a/instances/vm-instance?authuser=3&hl=en_U...
import os
import numpy as np
import sys
import random

if(len(sys.argv) != 3):
    print("[*] format => ./python dogEmotionClassifier.py <image location> <audio location>")
    sys.exit()

#load models & define variables
dogBarkLoadedmodel = keras.models.load_model('/home/cloudsystester/cloudFiles/model/dog_barks_classification.hdf5')
dogEmotionLoadedmodel = keras.models.load_model('/home/cloudsystester/cloudFiles/model/dog_emotion_recognition.hdf5')
imageFile = sys.argv[1]
audioFile = sys.argv[2]
labels = ['happy', 'angry', 'sick']

#dogBarkLoadedmodel = './model/%s' % (dogBarkModel)
#dogEmotionLoadedmodel = './model/%s' % (dogEmotionModel)

#defining functions
def convert(fileAudio):
    #original file in dogBarks downloaded are m4a file format
    #start converting m4a => wav
    outFile = "%s_conv.wav" % (random.randint(1, 99999999))
    cmd = """ffmpeg -i %s -acodec pcm_u8 -ac 1 %s >/dev/null 2>&1""" % (fileAudio, "/home/cloudsystester/cloudFiles/convert", outFile)
    os.system(cmd)
    return outFile

def dogEmotionPred(filepath):
    img_rows = 224
    img_cols = 224

    img = cv2.imread(filepath)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

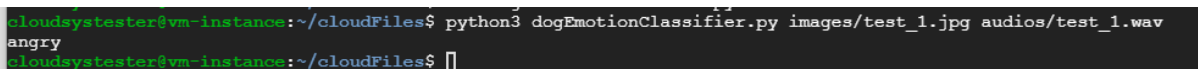
    img = cv2.resize(img, (img_rows, img_cols)).astype('float32')
    img /= 255
    img = np.expand_dims(img, axis=0)
    predict_label = dogEmotionLoadedmodel.predict(img)

    return predict_label

def dogBarkPred(filepath):
    audio, sample_rate = librosa.load(filepath, res_type='kaiser_fast')
    mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    mfccs_scaled_features = np.mean(mfccs_features.T, axis=0) #scaling
    data = mfccs_scaled_features.reshape(1,-1)

"dogEmotionClassifier.py" [dos] 87L, 2833C 52, 26 2
```

Figure 4.52 Dog Emotion Recognition Python Script in Linux VM GCP Instance



```
cloudsystester@vm-instance:~/cloudFiles$ python3 dogEmotionClassifier.py images/test_1.jpg audios/test_1.wav
angry
cloudsystester@vm-instance:~/cloudFiles$
```

Figure 4.53 Dog Emotion Recognition Python Script Output Detecting Ricky Dog Image and Barking Audio

As shown in figure 4.54, this would be the REST API directory. There will be the Node.js ‘restapi.js’ JavaScript that will be our connection between our client host and the Linux VM. In figure 4.55 and figure 4.56, these directories will be storing the images and audio that uploaded by the client. These images would be kept for future model training to improve model prediction accuracy.

```
cloudsystester@vm-instance:~/cloudFiles/restAPI$ ls -la
total 192
drwxr-xr-x  5 cloudsystester cloudsystester 4096 Aug 28 06:35 .
drwxr-xr-x  7 cloudsystester cloudsystester 4096 Sep  1 12:51 ..
drwxr-xr-x  2 cloudsystester cloudsystester 57344 Aug 29 07:12 audios
drwxr-xr-x  2 cloudsystester cloudsystester 65536 Aug 29 07:12 images
drwxr-xr-x 98 cloudsystester cloudsystester 4096 Aug 23 16:09 node_modules
-rw-r--r--  1 cloudsystester cloudsystester 28697 Aug 23 16:09 package-lock.json
-rw-r--r--  1 cloudsystester cloudsystester  384 Aug 23 16:09 package.json
-rw-r--r--  1 cloudsystester cloudsystester 12427 Aug 28 06:35 restapi.js
cloudsystester@vm-instance:~/cloudFiles/restAPI$
```

Figure 4.54 REST API Directory in Linux VM GCP Instance

```
cloudsystester@vm-instance:~/cloudFiles/restAPI$ cd audios
cloudsystester@vm-instance:~/cloudFiles/restAPI/audios$ ls -l
total 10992
-rw-r--r-- 1 cloudsystester cloudsystester 10649 Aug 27 16:05 001ef335efeef7356802fd2f301213f1
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 28 04:43 004be8e224b0dc8c4b31c87678417387
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 28 10:12 00f2b2e619095fc842c3fd46fde403aa
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 26 07:07 01184157dde36b3577639a4d26f50153
-rw-r--r-- 1 cloudsystester cloudsystester 15334 Aug 26 04:23 01220f5c95ab790db9d17ff819619ff1
-rw-r--r-- 1 cloudsystester cloudsystester 13772 Aug 26 04:53 013fe48f4570268c6c6fedaf59fb0a90a
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 27 16:19 0172b88b7c924a95d72e382f2245464d
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 26 12:39 01cd9bd03961d03c206c19bbb20cc925
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 26 06:43 01d585227936d8105fe364bccaf62db6b
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 27 16:20 02e5b211090fa44271c2ca56a3463558
-rw-r--r-- 1 cloudsystester cloudsystester 10454 Aug 26 06:59 032647b2735c2cd00d831fc549bde295
-rw-r--r-- 1 cloudsystester cloudsystester 9868 Aug 26 04:22 033818466db5898e11cceef6237138c5f
-rw-r--r-- 1 cloudsystester cloudsystester 10484 Aug 27 16:50 03397cad6e10cf97a0556dc0b6150440
-rw-r--r-- 1 cloudsystester cloudsystester 12577 Aug 26 04:22 037da78e59d27da6d452888788669da
```

Figure 4.55 Audios Directory in Linux VM GCP Instance

```
cloudsystester@vm-instance:~/cloudFiles/restAPI/images$ ls -l | head -n 10
total 170604
-rw-r--r-- 1 cloudsystester cloudsystester 323575 Aug 29 04:36 004a3056a1cad13fbba72b04a5df4bea
-rw-r--r-- 1 cloudsystester cloudsystester 349681 Aug 28 04:51 00ffb226e4b81750c568af80379eeaaaf
-rw-r--r-- 1 cloudsystester cloudsystester 461734 Aug 29 04:51 011f11b9a78e7cc103a3e6d663f0449c
-rw-r--r-- 1 cloudsystester cloudsystester 258008 Aug 26 06:27 0196a2d186d17f0e8b7cfdafe4067542
-rw-r--r-- 1 cloudsystester cloudsystester 276625 Aug 27 16:23 01bce0687bfa244a5291a1bdabde8b10
-rw-r--r-- 1 cloudsystester cloudsystester 267161 Aug 29 04:46 028de42e4b4694a8309d80df0d37cbde
-rw-r--r-- 1 cloudsystester cloudsystester 297342 Aug 28 10:25 02e496ec7bcd06171059bfae66ffff2c
-rw-r--r-- 1 cloudsystester cloudsystester 326064 Aug 26 07:13 0302012f3082c240706d30851dbfd311
-rw-r--r-- 1 cloudsystester cloudsystester 5396 Aug 28 04:44 034cefaafd3374d435d753de2564e030
```

Figure 4.56 Images Directory in Linux VM GCP Instance

Now let's walk through the code in REST API JavaScript. In figure 4.57, these blocks of code will be the basic functions for the dog social network mobile app to communicate with the cloud vm server database.

```

> app.post('/signup', (req, res) =>{ ...
})
> app.post('/login', (req, res) => { ...
})
> app.post('/getProfileDet', (req, res) => { ...
})
> app.post('/changePw', (req, res) =>{ ...
})
> app.post('/updateConnVal', (req, res) =>{ ...
})
> app.get('/searchFriends', (req, res) => { ...
})
> app.post('/addFriend', (req, res) => { ...
})
> app.get('/friendList', (req, res) => { ...
})

```

Figure 4.57 Block of Codes for Basic Dog Social Network in Node.js REST API JavaScript

In figure 4.58, this block of code will be the route for the client to upload the dog barking audio file to the cloud vm.

```

app.post('/uploadAudio', uploadAudio.single('file'), function(req, res){
  if(req.file){
    var audioname = req.file.filename
    //console.log(audioname)
    prevUploadAudioFN = audioname

    res.send("audio upload sucessful\n")
  }else{
    res.send("audio upload unsuccessful\n")
  }
})

```

Figure 4.58 Block of Codes for Upload Audio File in Node.js REST API JavaScript

In figure 4.59, this block of code will be the route for the client to upload the image file to the cloud vm. Over here, it will initialize the variables for dogEmotionClassifier.py script location, the image and audio file location and create the CLI command to execute. Then, it will extract the login user and friend's name from the original image file name. After that it will go to the exec() function that shows in figure 4.60 to execute the Python script. If the command executed successfully, it will call the checkSick() function.

```

app.post('/predict', uploadImg.single('file'), function(req, res){
  if(req.file){
    var imgname = req.file.filename

    classifierScript = "/home/cloudsystemester/cloudFiles/dogEmotionClassifier.py"
    imgDir = "/home/cloudsystemester/cloudFiles/restAPI/images"
    audioDir = "/home/cloudsystemester/cloudFiles/restAPI/audios"

    const cmd = `python3 ${classifierScript} "${imgDir}/${imgname}" "${audioDir}/${prevUploadAudioFN}`

    // extracting username, friend's name & email
    var oriImgName = req.file.originalname
    var tmp = oriImgName.split('_')
    var loginuser = tmp[1]
    var friendname = tmp[2]

    var infoArr = [friendname, imgDir, imgname, loginuser]

    //console.log(infoArr)
    exec(cmd,infoArr, res)
  }else{
    res.send("image upload unsuccessful\n")
  }
}

```

Figure 4.59 Block of Codes for Upload Image File in Node.js REST API JavaScript

```

//exec command
function exec(cmd, infoArr, res) {
  shell.exec(cmd, (code, stdout, stderr) => {
    if (stderr) {
      res.send("[!] Error in execution")
    } else {
      checkSick(stdout,infoArr, res)
    }
  })
}

```

Figure 4.60 Block of Codes for Executing DogEmotionClassifier Python Script in Node.js REST API JavaScript

In figure 4.61, the first part it will insert the predicted label into the database for future references. Then in figure 4.62, it will check whether the dog is sick or not. If it is sick, it will get the image file and attached into the email body. Then it will get the dog's owner email address. After getting that it will sent an email to the dog's owner to notify the owner that the dog is predicted sick.

```

//check if the predicted dog is sick, sent email to notify dog's owner
function checkSick(predictrsl,infoArr, res){
  var friendname = infoArr[0]
  var imgDir = infoArr[1]
  var imgname = infoArr[2]
  var currentuser = infoArr[3]

  res.send(predictrsl)

  //insert prediction into db
  var sql = "INSERT INTO PredictionData (report_name, username, friend, predicted_label) VAL

  let ts = Date.now();
  let date_ob = new Date(ts);
  let date = date_ob.getDate();
  let month = date_ob.getMonth() + 1;
  let year = date_ob.getFullYear();

  var report_date = year + "-" + month + "-" + date
  var reportname = report_date + "_" + currentuser + "_" + friendname

  pool.query(sql, [reportname, currentuser, friendname, predictrsl], (err, result, field) =>
    if(err){
      return console.log("[!] ERROR in INSERT PredictionData: " + err.message)
    }
  }

```

Figure 4.61 Block of Codes for Inserting Prediction Label into Database in Node.js REST API JavaScript

```

// if dog is sick (get predict result from exec func)
if(predictrsl.includes("sick")){
  var sql = "SELECT email FROM Profiles WHERE username=?"

  pool.query(sql, [friendname], (err, result, field) =>{
    if(err){
      return console.log("[!] ERROR in /predict: " + err.message)
    }

    if(result.length > 0){
      var friendemail = result[0]['email']

      //send email
      var mailOptions = {
        from: 'dogsocialmedia118@gmail.com',
        to: friendemail,
        subject: '[*] Check Your Dog Right Now',
        text: 'Your dog looks ill. Please take it to the vet ASAP!>',
        attachments: [{
          filename: 'dogImage.png',
          path: imgDir + "/" + imgname,
          cid: 'dogimage'
        }]
      };

      transporter.sendMail(mailOptions, function(error, info){
        if (error) {
          console.log(error);
        } else {
          console.log('Email sent: ' + info.response);
        }
      });
    }
  });
}

```

Figure 4.62 Block of Codes for Checking Sick Label in Node.js REST API JavaScript

In figure 4.63, these blocks of code would be getting the reports name and the reports detail for the specific user. These data that extracted will be sent back to the client mobile app for data visualization.


```

app.get('/getReportLists', (req, res) => {
  var username = req.query.username

  var sql = 'select distinct report_name from PredictionData where username=?'

  pool.query(sql, [username], (err, result, field) => {
    if(err){
      res.send("[!] ERROR in /getReportLists: " + err.message)
      return console.log("[!] ERROR in /getReportLists: " + err.message)
    }

    if(result.length > 0){
      var jsonObject = {}
      jsonObject['reports'] = result
      res.send(jsonObject)
    }else{
      res.send("[*] You've no report.")
    }
  })
})

app.get('/getReport', (req, res) => {
  var reportname = req.query.reportname

  var sql = 'SELECT predict_date, predicted_label FROM PredictionData WHERE report_name=?'

  pool.query(sql, [reportname], (err, result, field) => {
    if(err){
      res.send("[!] ERROR in /getReport: " + err.message)
      return console.log("[!] ERROR in /getReport: " + err.message)
    }

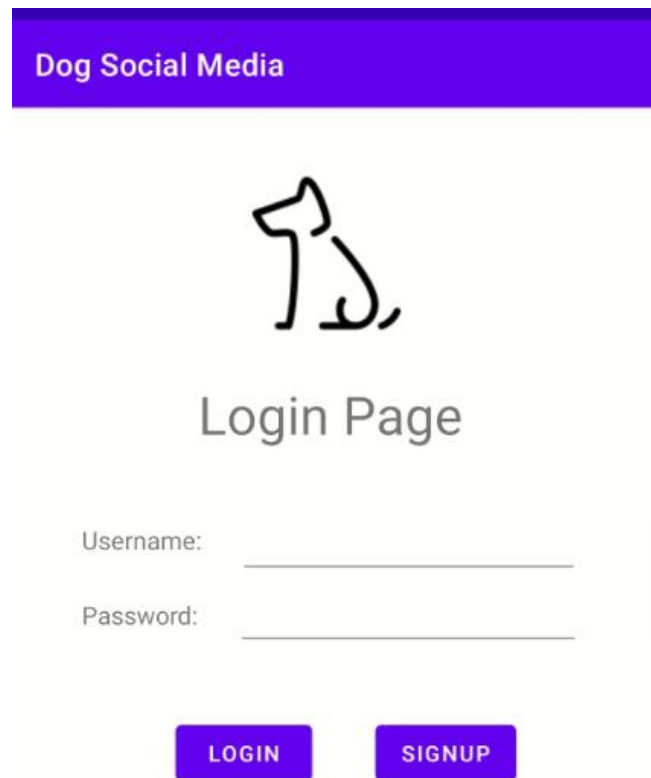
    if(result.length > 0){
      var jsonObject = {}
      jsonObject['data'] = result
      res.send(jsonObject)
    }
  })
})

```

Figure 4.63 Block of Codes for Getting Reports in Node.js REST API JavaScript

4.5 Dog Social Media Mobile Application

This part we will discuss about the mobile application for the dog social media. In figure 4.64, this would be the login interface for the user to login into their own account. If the user does not have an account, the user can click the signup button to go to the signup page.



The image shows a mobile application login interface. At the top, there is a purple header bar with the text "Dog Social Media" in white. Below the header is a simple line-art icon of a dog sitting. Underneath the icon, the text "Login Page" is displayed in a light gray font. The main content area contains two input fields: "Username:" followed by a white text box with a gray border, and "Password:" followed by a white text box with a gray border. Below these fields are two purple buttons with white text: "LOGIN" on the left and "SIGNUP" on the right. The entire interface is centered on a white background.

Figure 4.64 Login Interface of Dog Social Media

The user can just type in their details in order to create an account. In figure 4.65, we will create an account for abubakar user. After done creating an account we will be redirected to the login page automatically.

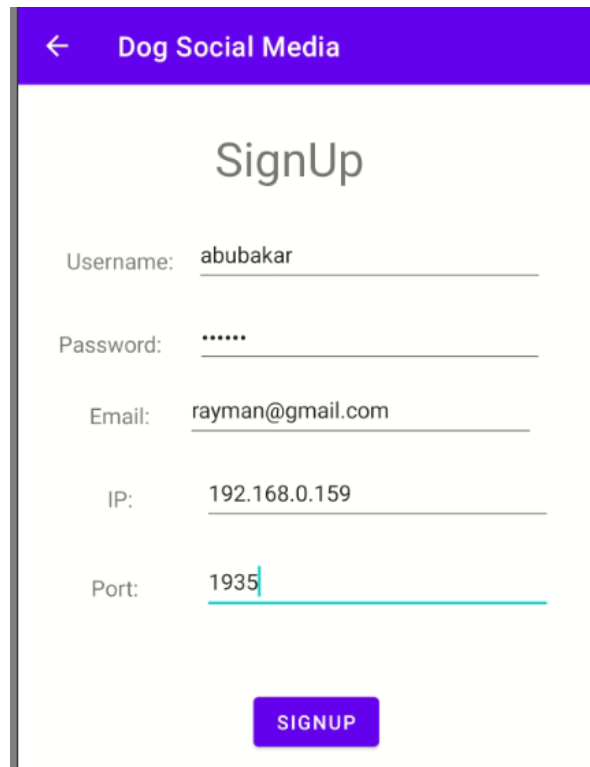


Figure 4.65 SignUp Interface of Dog Social Media

Now let's login into abubakar user account, in figure 4.66 there will be 5 buttons. Each buttons will have their own features. Let's check out the 1st button which will be the profile interface.

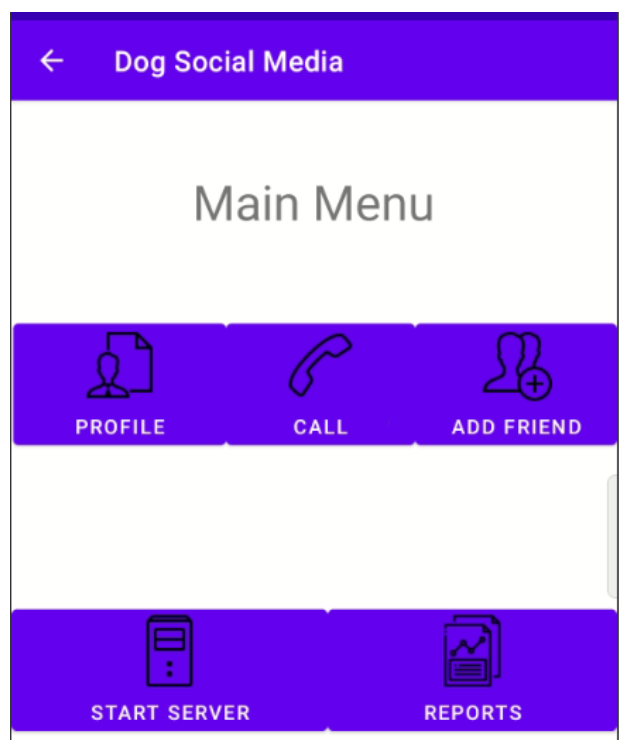


Figure 4.66 Main Menu Interface of Dog Social Media

As you can see that, in figure 4.67 we have just landed on the user's profile interface. In this page, the user will be able to check and see their profile details. If the user wants to change the login credential or wants to update the RTMP connection IP and port, the user can click on the edit profile button to go to the interface.

← Dog Social Media

Profile

Username:

Email:

Connection

IP:

Port:

[EDIT PROFILE](#)

Figure 4.67 User Profile Interface of Dog Social Media

Over here, in figure 4.68 we have a profile details edit interface. If the user wants to change his login credential, the user can type in the old password first, then only type in the new password and click on the change password button to change the password. If the user wants to update the RTMP connection IP and port, the user can just type in the IP and port number into the following textedit field, then click on the update connection button to update the details.

← Dog Social Media

Profile Edit

Old Password: _____

New Password: _____

CHANGE PASSWORD

Connection

IP: _____

Port: _____

UPDATE CONNECTION

Figure 4.68 Edit Profile Details Interface of Dog Social Media

Now, we will check out the add friend feature first. In figure 4.69, the 1st image will be the search friend interface. In this part, the user can type in the username of its friend. Then the system will check the database and return all the possible usernames to the user. In our case, we want to add snowdog user as our friend, so we click on snowdog username. The system will redirect us to an interface that will let us view snowdog user profile details. If we want to add snowdog user as our friend, we can just click on the add friend button.

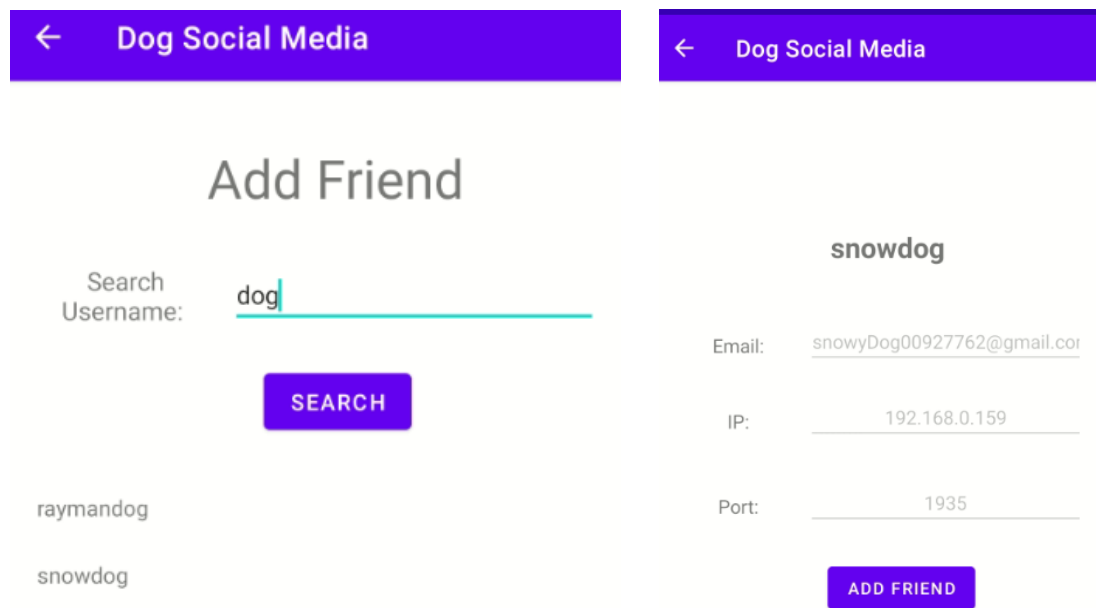


Figure 4.69 Add Friend Interface of Dog Social Media

Now we will discuss about the start server feature, this feature will allow the user to start the RTMP server in their own mobile device. Another user would need to connect back to this RTMP server in order to view and listen to the user speaking and user's action. In figure 4.70, we can see that there would be a black floating window on our screen. This floating window would let us to view ourselves during the call.

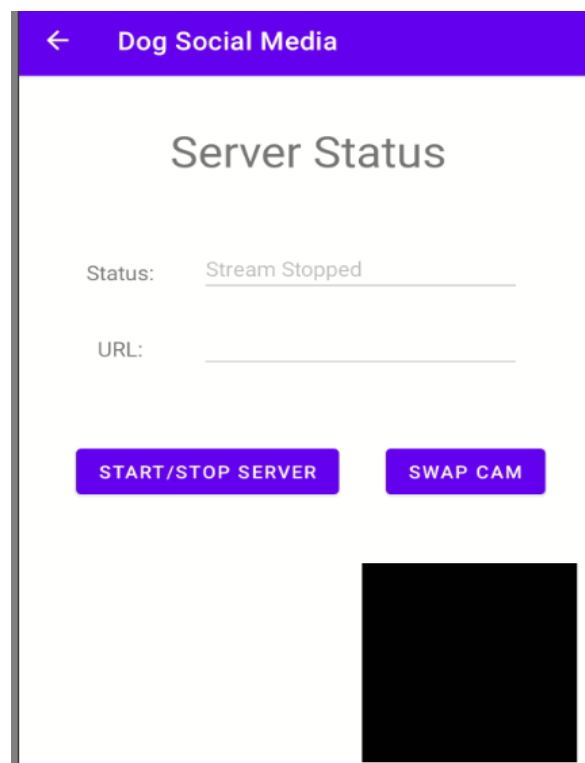


Figure 4.70 Start Server Interface of Dog Social Media

CHAPTER 4 PRELIMINARY WORK

Now let's start the RTMP server, as we can see that the server status changed into Stream on Live which means that the other friends would be able to connect back to our RTMP stream to view and listen to our dog actions. The URL part would show the currently listening RTMP IP and port. If the IP and port are different, the user can just go back to the interface shown in figure 4.68 to edit the IP and port. If the user wants to swap the camera from back camera to front camera, in figure 4.72 the user can just click on the swap cam button. This would change the camera from back camera to front camera.

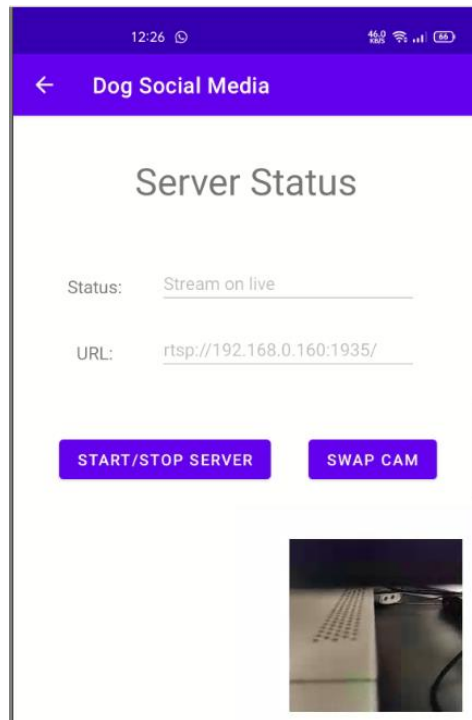


Figure 4.71 Demo Starting Server of Dog Social Media

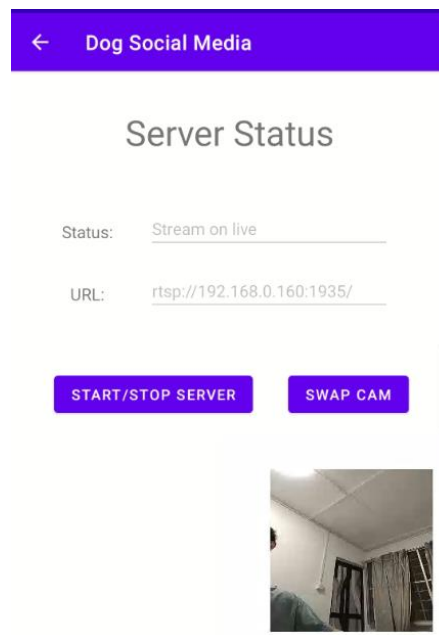


Figure 4.72 Demo Swapping Camera of Dog Social Media

Now let's discuss the backend code on how we start the RTMP server. In figure 4.73, we can see that this block of code will create a floating window on the dog social media interface. The reason I used floating window is that, when I tried to start the RTMP server in the same Android activity, I can connect to the RTMP server URL using my VLC. But, after I press the back button to go to another activity interface, my RTMP video stream will stop working but the audio frame still receiving from the Android mobile device. The main problem is the SurfaceView over here that had been killed when I have changed to another Android activity interface. So creating a floating window with a SurfaceView in it solve the video stream stop receiving problem.

```

wm = getSystemService(WINDOW_SERVICE) as WindowManager
ll = LinearLayout(context: this)
var llParameters: LinearLayout.LayoutParams = LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT, LinearLayout.LayoutParams.WRAP_CONTENT)
ll!!.setBackgroundColor(Color.argb(alpha: 66, red: 255, green: 0, blue: 0))
ll!!.setLayoutParams(llParameters)

var parameters: WindowManager.LayoutParams = WindowManager.LayoutParams(w: 400, h: 400, WindowManager.LayoutParams.TYPE_APPLICATION_OPENABLE, WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE, Gravity.CENTER)
parameters.gravity = Gravity.CENTER or Gravity.CENTER
parameters.x = 0
parameters.y = 0

//add surfaceview into floating windows
val surfaceView = SurfaceView(context: this)
val surfaceViewParams = ViewGroup.LayoutParams(
    ViewGroup.LayoutParams.MATCH_PARENT,
    ViewGroup.LayoutParams.WRAP_CONTENT
)
surfaceView.setLayoutParams(surfaceViewParams)

ll!!.addView(surfaceView)
wm!!.addView(ll, parameters)

```

Figure 4.73 Block of Code Creating Floating Window

CHAPTER 4 PRELIMINARY WORK

In figure 4.74, this block of code would allow us to drag the floating window around our dog social media interface. So we can find a sweet spot that we want the floating window to stay.

```
l1!!.setOnTouchListener(object : View.OnTouchListener {
    var updatedParameters = parameters
    var x = 0.0
    var y = 0.0
    var pressedX = 0.0
    var pressedY = 0.0

    override fun onTouch(v: View?, event: MotionEvent): Boolean {
        when (event.action) {
            MotionEvent.ACTION_DOWN -> {
                x = updatedParameters.x.toDouble()
                y = updatedParameters.y.toDouble()
                pressedX = event.rawX.toDouble()
                pressedY = event.rawY.toDouble()
            }
            MotionEvent.ACTION_MOVE -> {
                updatedParameters.x = (x + (event.rawX - pressedX)).toInt()
                updatedParameters.y = (y + (event.rawY - pressedY)).toInt()
                wm!!.updateViewLayout(l1, updatedParameters)
            }
            else -> {
            }
        }
        return false
    }
})
```

Figure 4.74 Block of Code Making Floating Window Draggable

In figure 4.75, this block of code would allow the user to start and stop the RTMP server. When starting the RTMP Server, it will prepare the audio and video frame captured by the microphone and camera and stream it on RTMP port 1935.

```
val serverToggleBtn: Button = findViewById(R.id.serverToggleBtn_serverStatus)
serverToggleBtn.setOnClickListener { it: View?

    val statusTF: TextView = findViewById(R.id.statusTF_serverStatus)
    val urLTF: EditText = findViewById(R.id.urLTF_serverStatus)

    if (rtspCamera.isStreaming) {
        statusTF.text = Editable.Factory.getInstance().newEditable(source: "Stream Stopped")
        urLTF.text = Editable.Factory.getInstance().newEditable(source: "")

        rtspCamera.stopStream()
        rtspCamera.stopPreview()
    } else {
        if (rtspCamera.isRecording || rtspCamera.prepareAudio(bitrate: 64 * 1024, sampleRate: 32000, isStereo: true,
            statusTF.text = Editable.Factory.getInstance().newEditable(source: "Stream on live")
            rtspCamera.startStream()

            urLTF.text = Editable.Factory.getInstance().newEditable(rtspCamera.getEndPointConnection())
        } else {
            Toast.makeText(
                context: this, text: "Error preparing stream, This device cant do it",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```

Figure 4.75 Block of Code Starting/Stopping RTMP Server

In figure 4.76, this block of code would allow the user to change the camera either to the back of camera or the front camera.

```
val swapcamBtn: Button = findViewById(R.id.swapCamBtn_serverStatus)
swapcamBtn.setOnClickListener{ it: View!

    rtspCamera.switchCamera()
}
```

Figure 4.76 Block of Code Swapping Camera

Now we will check out the calling friend feature. In figure 4.77, this interface will list out all the friends that the user had added before. Over here we can see that the following user had just added 2 friends only so far.

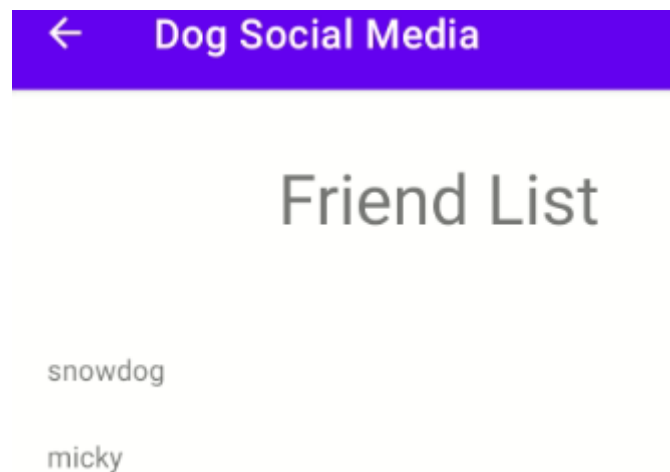


Figure 4.77 Friend Listing Interface in Dog Social Media

Let say in this case we want to call micky, we can just click on snowdog user and it will redirect us to the interface shows in figure 4.78 that will list the friend micky profile details. If the user wants to call micky, the user can click on the call button.

micky

Email:

IP:

Port:

Figure 4.78 View Friend Details Interface in Dog Social Media

In figure 4.79, we can see that this is how the calling interface looks like when it successfully connected to the friend's RTMP server. At the prediction part, the 1st part will be screenshotting, this will take a screenshot of the VideoView that plays the friend's RTMP video frame. Then the 2nd part will be audio recording, this part will record the microphone for 5 seconds to record the barking audio. After done, it will uploads the audio and image to the cloud vm Node.js. After done predicting it will return the predicted label. As shown in figure 4.79, the predicted label is happy as the dog is happily playing with his owner.

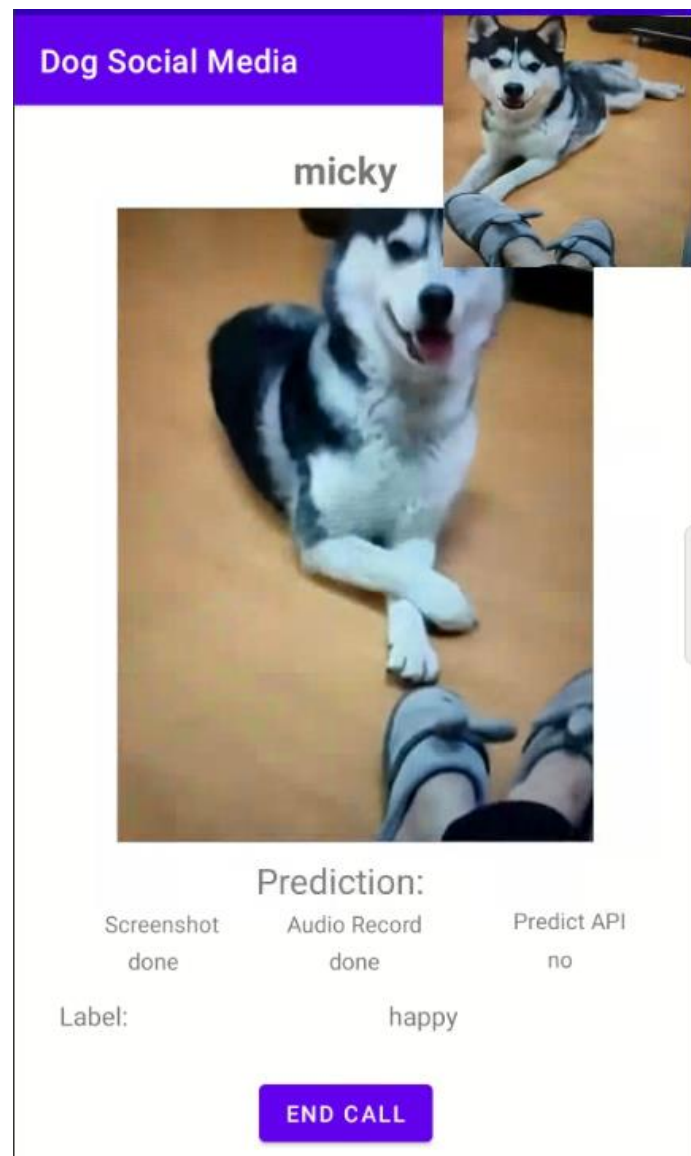


Figure 4.79 Calling Interface in Dog Social Media

Now let's test what would happen when the dog is detected sick. In figure 4.80, here the calling interface predicted the dog looks sick. Then in figure 4.81, micky's owner would receive an email with the image attached in the body to notify micky's owner that the owner need to take action with the dog as the dog looks lonely and depress.

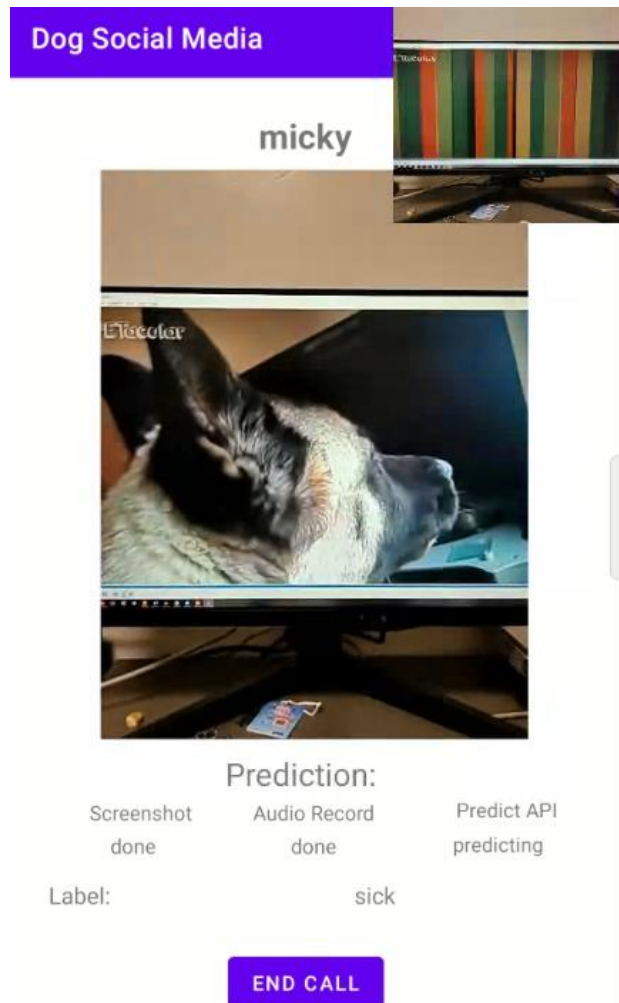


Figure 4.80 Demo Dog Predicted Sick in Calling Interface

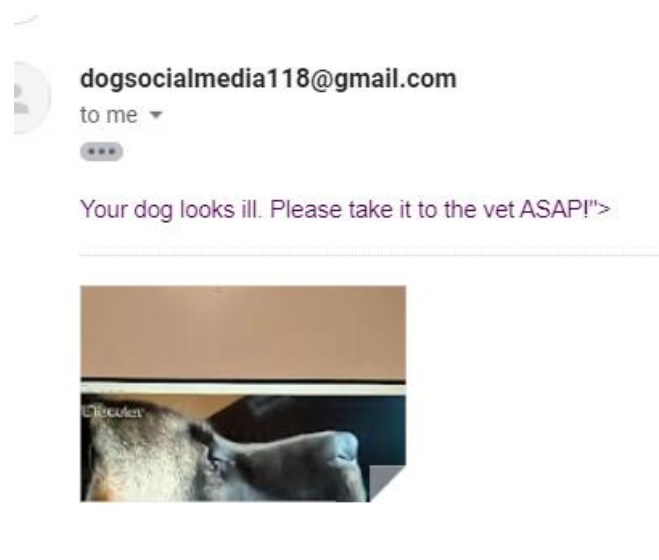


Figure 4.81 Demo Email Received by Micky's Owner

Now we will be checking the last feature which will be the view reports feature. In figure 4.82, it will shows a list of report that captured predicted data for the specific day and then who is the friend that the dog calling to. Let's break down how can we read the report name, the format for the report name would be `<data_captured_date>_<current_login_user>_<friend>`. Now let's check out the 1st report as we want to check and see what happened to our dog when we left it alone at home for the whole day.

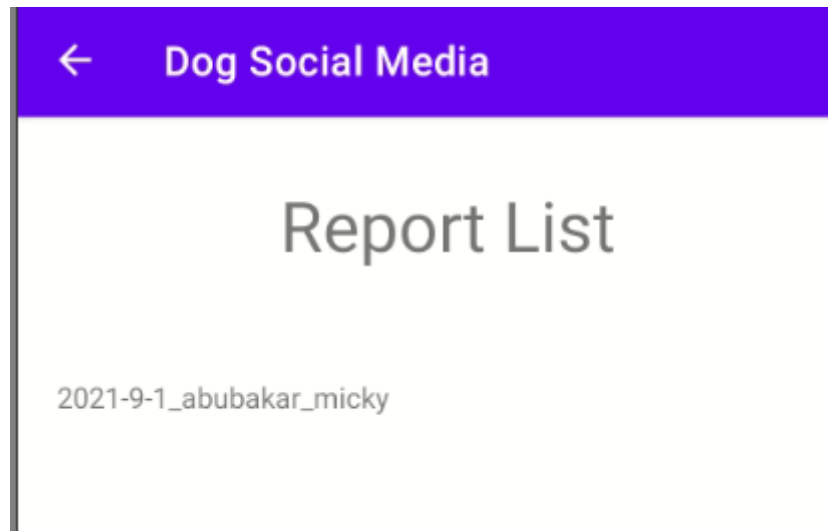


Figure 4.82 Report Listing Interface in Dog Social Media

In figure 4.82, we will have a bar chart and a line chart. Each of this chart would have their own meaning let's take a look. For the bar chart. This would calculate how many the predicted label predicted for each class label for the whole day. Based on our observation, we can see that our dog conversation with another dog it makes the another dog feel sick all the time as sick class label are the highest prediction among the other class label. Now let's take a look at the line chart, from 5:09pm till 5:12pm the friend dog emotion had changed from angry to sick and it remains constant in sick emotion. So we know that something is wrong with the dogs, so the dog's owner need to take a look at their dog and see what happened to them that makes them sick. The below part will have a description to explain to the user what the graph means, as stated there the dog is unhealthy and depress as the dog is predicted sick all the time. So the dog's owner of micky or abubakar need to take the dog to the nearest vet.

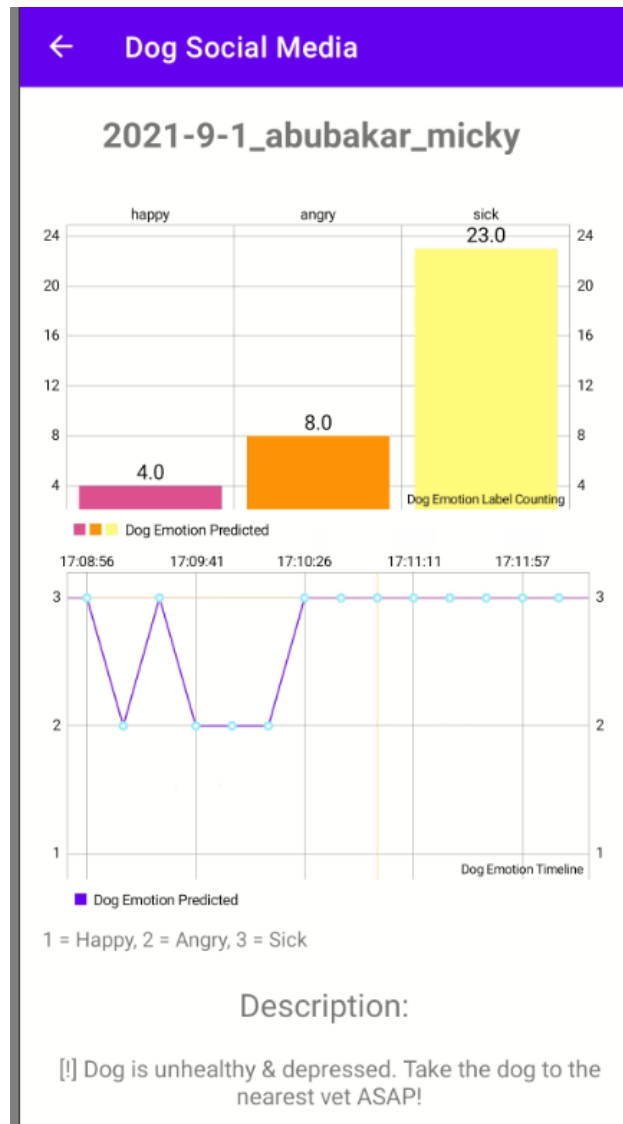


Figure 4.83 View Report Details Interface in Dog Social Media

4.6 Data Visualization on Google Data Studio

We will be using Google Data Studio for our data visualization. Here let’s take a look at our 1st graph. In figure 4.84, we will be able to see how many users that had registered for our dog social media. The table below will show the detail of the users.

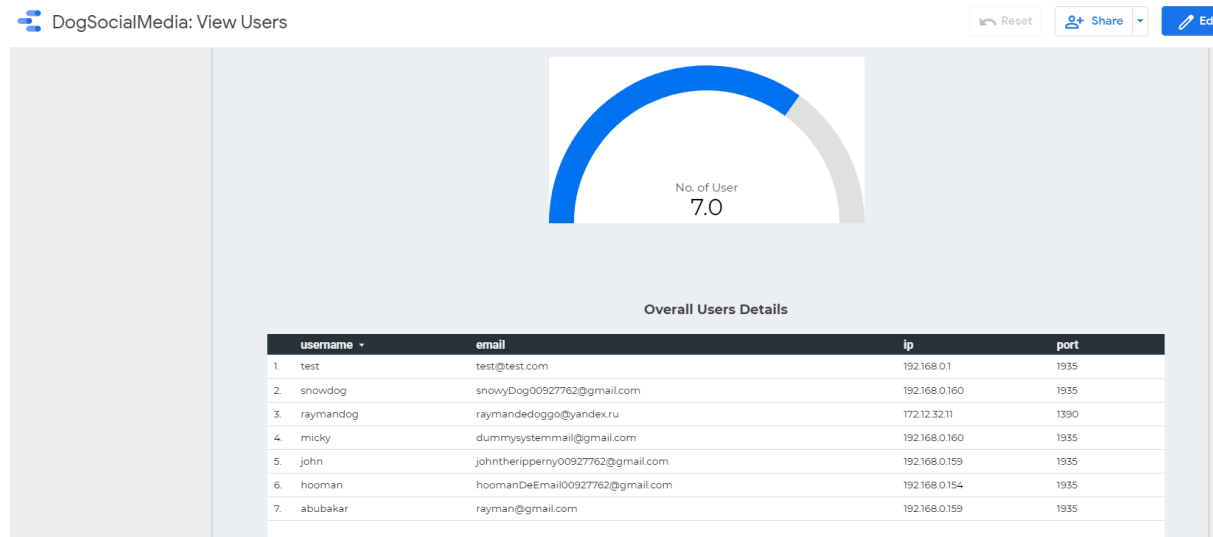


Figure 4.84 View Number of Users in Data Studio

In the report that shows in figure 4.85, here we will have 2 graphs the 1st bar chart graph will be showing the overall predicted class label for each of the dog from all the time. The 2nd bar chart graph will shows the individual report which means that in the particular date, who is the caller that call another friend report. Then the table below will shows the report name, who is the caller and who does the caller called. How many is the predicted class label predicted during the call for the whole day. To do some analysis on the dog behavior, we can utilize the filter feature like report name, date range, caller and friend.

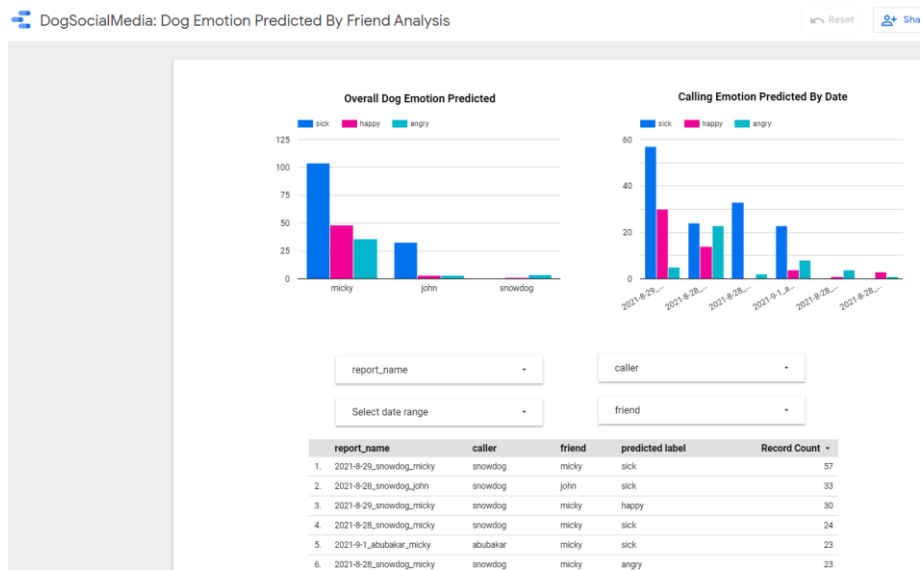


Figure 4.85 Dog Emotion Predicted By Friend Analysis in Data Studio

Now let's test with some filter applied, we now will do some analysis on snowdog user from the prediction data collected from the date 28th of August till 29th of August. Snowdog user had called micky and john friend during that date period. The interesting part is that all the dogs friend that snowdog called they had 1 issue which is very high sick label predicted. The sick class label that predicted when snowdog having a calling conversation with other 2 dogs are the highest among the other 2 class label. We can do some conclusion that most probably this snowdog user does something that the other 2 friend micky and john does not like which makes them feel depress and sick.

CHAPTER 4 PRELIMINARY WORK

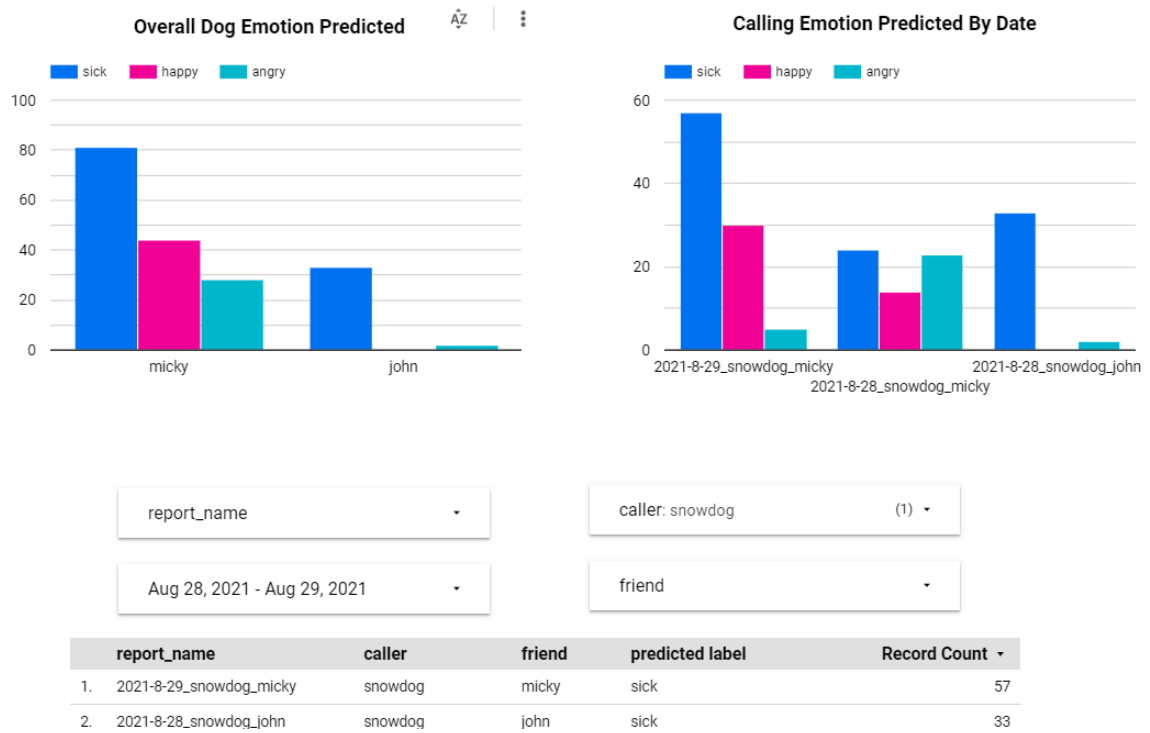


Figure 4.86 Demo Applying Filter During Analysis in Data Studio

CHAPTER 5: CONCLUSION

There are so many people that like to adopt dog as dogs can be a great companion for human being. Dog can be a part of the human life as when the dog's owner is lonely, the dog's owner can just play with their dog to overcome their loneliness. When the dog's owner came back from work with all the accumulated stress from the work, the stress can just be reduced when the dog's owner saw the dog waiting for him in front of the gate.

Here comes the problem that the dog's owner did not understand how the dog feel. As the dog have been left alone when the dog's owner goes to work. During the whole period, the dog will felt lonely as they do not have any dog companion to play with as they only have their own owner to play with only all the time. So due to this problem dog loneliness will be accumulate and develop depression in the dog. The dog owner would not notice that until when the dog's owner found out that the things that the dog like to do or the food the dog like to eat, suddenly the dog do not have the appetite to eat it. These is a huge problem as the dog's owner does not know what is happening to the dog when the dog's owner left the dog alone at home when he/she is working.

In the market nowadays, actually there are some existing products that could let the dog's owner to monitor their dog but there is a problem in those product as they could not understand what does the dog feel like is the dog is happy right now or the dog is not happy right now. The product that sell in the market could not do this kind of stuff which is a drawback to the dog's owner too. Another one is that the product in the market could not let the dog to communicate with another dog through a platform. The dog's owner does not have the money to own another dog for the dog as a companion as the food will be double up and it would cause trouble for the owner too.

So to solve the problem, we have come out the proposed solution to develop a smart dog social network which to let the dog and another dog to communicate with each other remotely by using our distributed system architecture. The dogs will be using our platform to communicate with another dog. In this social network only the initial part which is starting the program and connect to the remote RTMP Server of another dog. The dog can just directly talk with another dog directly when they are bored. Another feature which will be the image recognition model trained to detect the dog emotion. With this feature we can identify the dog emotion by training a model to detect the dog emotion. With the model, we will be able to detect the dog is happy or not happy. If the dog is lonely it will show the dog is not happy. So

CHAPTER 5 CONCLUSION

the dog's owner will be able to take further action to prevent the dog from becoming lonely and depressed.

As for the future work, we would be using the collected dog barking audio and dog images that saved in the cloud vm to train our dog emotion classifier in order to enhance the prediction accuracy to have a higher accuracy during the prediction using our dog emotion classifier model.

BIBLIOGRAPHY

Barrington, K., 2020. We Tried The Furbo Dog Camera That Lets You Monitor And Play With Your Pet From Afar — Here'S What It'S Like To Use. [online] Business Insider Malaysia. Available at: <<https://www.businessinsider.my/furbo-dog-camera-review?r=US&IR=T>> [Accessed 3 March 2020].

Ceicdata.com. 2020. Malaysia | Hours Worked: Mean | Economic Indicators. [online] Available at: <<https://www.ceicdata.com/en/malaysia/labour-force-survey-hours-worked-by-sex--age/hours-worked-mean>> [Accessed 12 February 2020].

Derbyshire, D., 2020. Dogs Who Bite Aren't Barking Mad... They're Just Depressed. [online] Mail Online. Available at: <<https://www.dailymail.co.uk/sciencetech/article-1295809/Dogs-bite-arent-barking-mad--theyre-just-depressed.html>> [Accessed 13 February 2020].

Malaysia Hours Worked Mean. 2020 Available at: <https://www.ceicdata.com/datapage/charts/o_malaysia_hours-worked-mean/?type=area&from=2015-01-01&to=2017-12-31&lang=en> [Accessed 12 February 2020].

McQuillan, K., 2020. Review Of The Petcube Bites Treat Camera (Features, Pros, Cons & Costs). [online] Pet Sitters Ireland | Pet Sitter | Dog Walker | Cat Sitter | Dog Minder. Available at: <<https://petsittersireland.com/petcube-bites-treat-camera/>> [Accessed 5 March 2020].

Pals with paws, 2020. Available at: <<https://apicms.thestar.com.my/uploads/images/2019/12/01/413837.jpg>> [Accessed 11 February 2020].

Petsho. 2020. Petzi Treat Cam Review. [online] Available at: <<https://petsho.com/petzi-review/>> [Accessed 12 March 2020].

The Star Online. 2020. Pet Peeves In Animal Welfare. [online] Available at: <<https://www.thestar.com.my/news/nation/2019/12/01/pet-peeves-in-animal-welfare>> [Accessed 11 February 2020].

Trusted Reviews. 2020. Pawbo+ Review . [online] Available at: <<https://www.trustedreviews.com/reviews/pawbo>> [Accessed 1 March 2020].

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 3
Student Name & ID: LEONG WAI CHUN 18ACB06442	
Supervisor: Dr. Cheng Wai Khuen	
Project Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK	

1. WORK DONE

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

Complete trained the dog barking classifier model and did some data visualization on the model.

2. WORK TO BE DONE

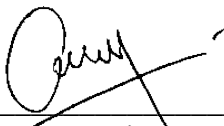
Need to add 1 more class label and retrain the dog emotion image classifier model

3. PROBLEMS ENCOUNTERED

Using raw dog barking audio files containing noises will cause the model to have poor prediction performance. To solve this problem, need to do some cleaning with Audacity program to extract the exact part that containing the audio spectrum which belongs to the particular dog barking class label.

4. SELF EVALUATION OF THE PROGRESS

The Dog emotion image classifier need to be trained to detect 3 class labels. So, the following weeks will be doing training on dog emotion image classifier.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	Study week no.: 6
Student Name & ID: LEONG WAI CHUN 18ACB06442	
Supervisor: Dr. Cheng Wai Khuen	
Project Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK	

1. WORK DONE

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

Complete trained dog emotion image classifier model and does some data visualization on some models with different hyperparameter tuning options. Comparing how does different hyperparameter affects the model prediction and compare training the model with ResNet and VGG16 network which one is more effective.

2. WORK TO BE DONE

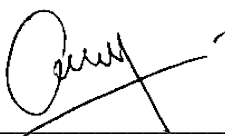
Combine the 2-model dog barking classifier model and dog emotion image classifier model and do some benchmarking.

3. PROBLEMS ENCOUNTERED

When training the dog emotion image classifier with 300+ images the testing prediction score is very low, to solve this problem I have added another 300 images which makes the total of images 600+ and the testing prediction score improved.

4. SELF EVALUATION OF THE PROGRESS

In order to improve the dog emotion classifier prediction model accuracy, the following weeks will be focusing on doing some experiment in combining 2 models and did some benchmarking on the 2 combined models prediction.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	Study week no.: 8
Student Name & ID: LEONG WAI CHUN 18ACB06442	
Supervisor: Dr. Cheng Wai Khuen	
Project Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK	

1. WORK DONE

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

Complete combining dog barking audio classifier and dog emotion image classifier models using weighted average technique and did some benchmarking on the new function.

2. WORK TO BE DONE

Create a dog social media mobile application to utilize the created models that host on the Google Cloud Platform vm instance.

3. PROBLEMS ENCOUNTERED

When doing the combination of 2 models, there is some problem encountered. First, I have no idea how to combine the 2 models prediction result and make it into one, so after some research I found that I can use weighted average technique to combine the 2 model prediction result and make it into 1 which is the enhance version of prediction result.

4. SELF EVALUATION OF THE PROGRESS

To utilize the models that trained, need to deploy the models in the cloud vm instance and create a mobile application that will let the dogs to be able to see each other.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	Study week no.: 12
Student Name & ID: LEONG WAI CHUN 18ACB06442	
Supervisor: Dr. Cheng Wai Khuen	
Project Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK	

1. WORK DONE

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

Complete creating a dog social media mobile application.

2. WORK TO BE DONE

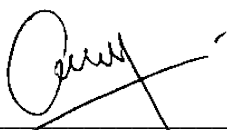
Utilizing the collected data for data visualization

3. PROBLEMS ENCOUNTERED

When I want to start the RTMP server hosting on the client device, it works normally as my VLC player able to connects the remote RTMP server URL successfully. But, after clicking on the back button to go to another Android activity, the video feed started to stop receiving but the audio frame still receiving from the remote device. After some debugging on it, I found the problem that caused the RTMP server does not continue sending the video frame as due to the SurfaceView on the activity has been destroyed. So to solve the problem, need to make the SurfaceView embedded in a floating window which allows accessing from any other Android activities.

4. SELF EVALUATION OF THE PROGRESS

As we have collected so many data along the journey, we need to start utilizing the data and make some graphs for the user and the admin to view it. As using graphs are easier to let the users to determine that how well does their dog predicted when the dog's owner are not at home.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: Y3S3	Study week no.: 13
Student Name & ID: LEONG WAI CHUN 18ACB06442	
Supervisor: Dr. Cheng Wai Khuen	
Project Title: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK	

1. WORK DONE

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

Complete generating reports for data visualization on the mobile application and Google Data Studio

2. WORK TO BE DONE


Complete the documentations.

3. PROBLEMS ENCOUNTERED

When we need to plot a graphs on the mobile application, we need to gather the specific data out from the database. In order to solve this problem, I have created Date, report name containing who is the caller and the friend calling, predicted class label columns in the database PredictionData table. Then at the Node.js REST API side, I have created a function which allows me to insert each of the predicted class label into the PredictionData table in the database.

4. SELF EVALUATION OF THE PROGRESS

The collected data can be fully utilize in creating stories after we have data visualize it.



Supervisor's signature



Student's signature

DOG CHAZ

A Smart Social Network for Dog to Chit Chat

By Leong Wai Chun



INTRODUCTION

Nowadays number of people adopting dog as their companion are increasing as dogs can fill up their loneliness. But, the dog's owner does not know that their dog might be in depression due to loneliness when dog's owner went to work.

Problem:

- Dog become lonely as dog do not have any companion to play with when the dog's owner went to work.
- Dog's owner does not understand how does their dog feel. As dog's owner does not know how the dog feel when the dog was left alone at home.

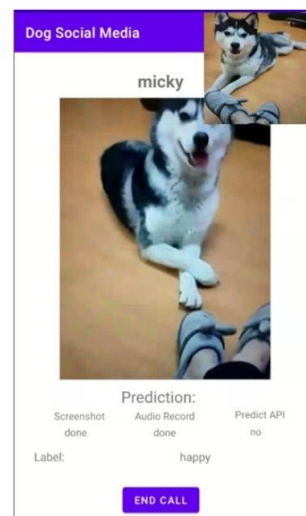
PROPOSED METHOD

- Smart dog social network to let the dog and another dog to communicate
 - This feature will let one dog communicate with another dog in a smart social network with dog emotion recognition feature.
- Dog Emotion Recognition to identify dog emotion
 - This will be the identifying dog emotion feature that used in the smart dog social network to identify the dog emotion.

RESULT



Prediction Results on Ricky the Dog with Trained Dog Emotion Recognition Model



Dog Calling Interface That Used for Chit Chatting with Smart Feature to Recognise Dog Emotion

CONCLUSION

Dog will feel lonely if they do not have a companion to play with but with Smart Social Network, we can provide dog a social network for them to chit chat with another dog to enjoy their leisure time when the dog's owner is not around.

Dog owner able to identify what does the dog feel too, as with the dog emotion recognition feature, the dog's will be able to identify whether the dog is happy, angry, or sick through the Smart Social Network.

PLAGIARISM CHECK RESULT

Project 2: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK

ORIGINALITY REPORT

2 %	1 %	0 %	0 %
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.pawprintsthemagazine.com Internet Source	<1 %
2	www.dailymail.co.uk Internet Source	<1 %
3	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1 %
4	Sridhar Alla, Suman Kalyan Adari. "Beginning Anomaly Detection Using Python-Based Deep Learning", Springer Science and Business Media LLC, 2019 Publication	<1 %
5	Submitted to University of East London Student Paper	<1 %
6	Submitted to University of Westminster Student Paper	<1 %
7	petcube.com Internet Source	<1 %
8	spcwebapps.sonoco.com Internet Source	<1 %

PLAGIARISM CHECK RESULT

Document Viewer

Turnitin Originality Report

Processed on: 02-Sep-2021 16:58 +08
 ID: 1639992211
 Word Count: 15714
 Submitted: 1

Similarity Index	Similarity by Source	
2%	Internet Sources:	1%
	Publications:	0%
	Student Papers:	0%

Project 2: DEVELOP A SMART ENVIRONMENT FOR DO... By Wai Chun Leong

exclude quoted	include bibliography	exclude small matches	mode: quickview (classic) report	Change mode	print	download
<1% match (Internet from 17-Nov-2010) http://www.pawprintsthemagazine.com						
<1% match (Internet from 01-Jun-2020) https://www.dailymail.co.uk/sciencetech/article-1295809/Dogs-bite-arent-barking-mad--theyre-just-depressed.html?ito=feeds-newsxml						
<1% match (student papers from 03-Apr-2019) Submitted to Universiti Tunku Abdul Rahman on 2019-04-03						
<1% match (student papers from 24-Aug-2013) Submitted to Universiti Tunku Abdul Rahman on 2013-08-24						
<1% match (publications) Sridhar Alla, Suman Kalyan Adari, "Beginning Anomaly Detection Using Python-Based Deep Learning", Springer Science and Business Media LLC, 2019						
<1% match (student papers from 22-Dec-2015) Submitted to University of East London on 2015-12-22						
<1% match (student papers from 04-May-2020) Submitted to University of Westminster on 2020-05-04						
<1% match (Internet from 21-Sep-2020) https://petcube.com/bites-2/						
<1% match (Internet from 14-Mar-2020) https://spcwebapps.sonoco.com/b2b/appdocs/						
<1% match (Internet from 06-Dec-2020) https://www.amazon.com/Wireless-Security-Camera-Two-way-Audio/dp/B06XG2QYX9						
<1% match (Internet from 12-Dec-2020) https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/						
<1% match (student papers from 09-May-2020) Submitted to Universiti of Addis Ababa on 2020-05-09						

feedback studio
Wai Chun Leong | Project 2: DEVELOP A SMART ENVIRONMENT FOR DOG SOCIAL NETWORK

CHAPTER 1: INTRODUCTION

1.1 Background Information

Year	Cats	Dogs
2014	615,000	381,000
2015	632,000	374,000
2016	694,000	354,000
2017	726,000	342,000
2018	750,000	402,000

Figure 1.1 The Statistic of Pet Dog and Cat from 2014 till 2018 in Malaysia (Pals with paws, 2020)

In Malaysia based on figure 1.1 statistic, we can see that more people starting to adopt a pet dog because of their cuteness or as a companion to the people. Dogs can be a companion of human too where every day the dog owner come back from work, their pet dog will welcome their owner as this will decrease the loneliness of humans. But, there is a problem where the dog owner did not realize that the dog might felt lonely too when the owner has gone for work. (Pet peevies in animal welfare, 2020)

Match Overview

2%

- 1 [www.pawprintsthemag...](#) <1% >
Internet Source
- 2 [www.dailymail.co.uk](#) <1% >
Internet Source
- 3 [Submitted to Universiti...](#) <1% >
Student Paper
- 4 [Sridhar Alla, Suman Kal...](#) <1% >
Publication
- 5 [Submitted to University...](#) <1% >
Student Paper
- 6 [Submitted to University...](#) <1% >
Student Paper
- 7 [petcube.com](#) <1% >
Internet Source
- 8 [spcwebapps.sonoco.c...](#) <1% >
Internet Source
- 9 [www.amazon.com](#) <1% >
Internet Source
- 10 [www.pyimagesearch.c...](#) <1% >
Internet Source
- 11 [Submitted to University...](#) <1% >

Page: 1 of 94 | Word Count: 15714
Text-Only Report | High Resolution

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



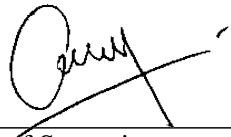
FACULTY OF INFORMATION COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Leong Wai Chun
ID Number(s)	18ACB06442
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	Develop a Smart Environment for Dog Social Network

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 2 </u> % Similarity by source Internet Sources: <u> 1 </u> % Publications: <u> 0 </u> % Student Papers: <u> 0 </u> %	No issue. OK.
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
Parameters of originality required and limits approved by UTAR are as follows: (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

 Name:
 Date: 3//9/2021

 Signature of Co-Supervisor

 Name:
 Date:



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	18ACB06442
Student Name	LEONG WAI CHUN
Supervisor Name	Dr. Cheng Wai Khuen

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
✓	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)

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

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;"> </div> <p>(Signature of Student) Date: 2nd September 2021</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <div style="text-align: center;"> </div> <p>(Signature of Supervisor) Date: 3/9/2021</p>
--	---