

INVESTIGATE AND ANALYSIS OF DEEP
LEARNING AND MACHINE LEARNING
ALGORITHM FOR FACE MASK DETECTION
SYSTEM

MUHAMMAD EZZUDEEN BIN SURATMAN
EA18017

B.ENG (HONS.) ELECTRICAL ENGINEERING
(ELECTRONICS)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : MUHAMMAD EZZUDDEEN BIN SURATMAN

Date of Birth : 27 OCTOBER 1997

Title : INVESTIGATE AND ANALYSIS OF DEEP LEARNING AND
MACHINE LEARNING ALGORITHM FOR FACE MASK
DETECTION SYSTEM

Academic Session : SEMESTER 2 2021/2022

I declare that this thesis is classified as:

- CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*
- RESTRICTED (Contains restricted information as specified by the organization where research was done)*
- OPEN ACCESS I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:



(Student's Signature)



(Supervisor's Signature)

971027017649
New IC/Passport Number
Date:

PROF. MADYA IR. TS. DR
FAHMI BIN SAMSURI
Name of Supervisor
Date: 22-06-2022

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons	(i)
	(ii)
	(iii)

Thank you.

Yours faithfully,

(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the Bachelor of Electrical Engineering (Electronics) with Honours.



(Supervisor's Signature)

Full Name : PROF. MADYA IR. TS. DR FAHMI BIN SAMSURI

Position : SENIOR LECTURER

Date : 22-06-2022

ASSOC. PROF. IR. TS. DR. FAHMI BIN SAMSURI
ASSOCIATE PROFESSOR
DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
UNIVERSITI MALAYSIA PAHANG
LEBUHRAYA TUN RAZAK
26300 GAMBANG, KUANTAN, PAHANG
TEL : +609-549 2338 FAX : +09-424 5055



STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

A handwritten signature in black ink, appearing to be 'Muhammad Ezzuddeen Bin Suratman', is written above a horizontal line.

(Student's Signature)

Full Name : MUHAMMAD EZZUDEEN BIN SURATMAN

ID Number : EA18017

Date : 22-06-2022

INVESTIGATE AND ANALYSIS OF DEEP LEARNING AND MACHINE LEARNING
ALGORITHM FOR FACE MASK DETECTION SYSTEM

MUHAMMAD EZZUDEEN BIN SURATMAN

Thesis submitted in fulfillment of the requirements
for the award of the
Bachelor of Electrical Engineering (Electronics) with Honours

College of Engineering
UNIVERSITI MALAYSIA PAHANG

JUNE 2022

ACKNOWLEDGEMENTS

Alhamdulillah, with his assistance and advice, I was able to complete the semester successfully and the task of the Final Year Project 1 was successfully completed. I'd like to express my gratitude to Prof. Madya Ir. TS. Dr Fahmi Bin Samsuri, my supervisor, for his guidance and expertise in completing the study. I am quite appreciative for his assistance, which he provided from the first week of my Final Year Project until the day I completed it.

In addition, I'd want to convey my heartfelt gratitude to my parents for their unwavering spiritual support and encouragement in helping me complete this final year assignment. Their thoughts and prayers keep me going and make me believe that I can achieve it. I've also recognised that performing this project will polish my skills. Finally, I'd like to express my gratitude to all my friends that assisted me in completing my final year project. I am grateful for all your assistance.

ABSTRAK

Orang ramai pada masa kini cenderung memakai topeng muka pelindung kerana wabak COVID-19 yang melanda dunia kita beberapa tahun lalu dan memakai topeng muka pelindung telah menjadi kebiasaan baharu. Banyak tempat awam yang menyediakan perkhidmatan tertentu mahu orang ramai memakai topeng dengan betul sebelum memasuki tempat tersebut. Oleh itu, dengan membangunkan sistem pengesanan topeng muka, ia cenderung untuk membantu masyarakat global menyedari persekitaran yang dikelilingi oleh virus dan mencegah jangkitan. Walaupun vaksin telah dibangunkan, orang ramai masih perlu sedar kerana segelintir masyarakat yang tidak mahukan vaksin. Untuk membangunkan sistem ini, pembelajaran mesin dan pembelajaran mendalam adalah kaedah terbaik untuk digunakan dengan menggunakan beberapa pakej pembelajaran mesin asas seperti Tensorflow, Keras dan OpenCV. Kaedah ini mengesan imej wajah seseorang daripada imej, video dan pemantauan masa nyata dengan betul dan kemudian mengenal pasti ia mempunyai topeng muka atau tidak dan akan memaklumkan pihak berkuasa jika tidak memakai topeng muka. Sistem ini boleh digunakan di premis sebelum orang ramai memasuki tempat itu dan akan menghapuskan keperluan untuk menempatkan pekerja untuk memantau orang yang masuk di pintu masuk dan meminimumkan jangkitan.

ABSTRACT

People nowadays tend to wear a protective facemask because of the pandemic COVID-19 that strike our world few years ago and wear protective facemask has become a new normal. Many public place that provides a certain service want people to wear mask correctly before entering the place. Therefore, by developing the facemask detection system, it tends to help a global society to aware the environment that surround by the virus and to prevent the infections. Although vaccines have been developed, people still need to be aware because of some society that stick not to wanting a vaccine. For develop this system, machine learning and deep learning is the best method to use by using some basic machine learning package such as Tensorflow, Keras and OpenCV. This method detects the image of someone face from the image, video and real time monitoring correctly and then identifies it has a facemask on it or not and will alert the authority if not wearing a facemask. This system can be use at the premise before people entering the place and would eliminate the need to place a worker to monitor the people coming in at the entrance and minimize the infections.

TABLE OF CONTENT

DECLARATION	
TITLE PAGE	
ACKNOWLEDGEMENTS	ii
ABSTRAK	iii
ABSTRACT	iv
TABLE OF CONTENT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF ABBREVIATIONS	x
CHAPTER 1 INTRODUCTION	1
1.1 Project Background	1
1.2 Problem Statement	2
1.3 Objective of Project	2
1.4 Scopes of Project	2
1.5 Thesis Overview	3
CHAPTER 2 LITERATURE REVIEW	4
2.1 Introduction	4
2.2 Machine Learning	4
2.2.1 Neural Network	5
2.2.2 Supervised Learning	6
2.3 Deep Learning	7

2.3.1	Convolutional Neural Network	9
2.4	Object Detection algorithm model	10
2.4.1	YOLO	10
2.4.2	Single-Shot Detector	11
2.5	Image Pre-processing	12
2.5.1	Architecture of Deep Learning	13
2.5.2	COCO dataset	14
2.6	Face detector	15
CHAPTER 3 METHODOLOGY		18
3.1	Introduction	18
3.2	Flow diagram of the project	18
3.2.1	Dependencies	19
3.2.2	Dataset	20
3.2.3	Data Pre-processing	22
3.2.4	Train Model	24
3.2.5	Calculation of Performance	28
CHAPTER 4 RESULTS AND DISCUSSION		30
4.1	Introduction	30
4.2	Model Accuracy	30
4.3	Face mask detection performance	35
CHAPTER 5 CONCLUSION		39
5.1	Introduction	39
5.2	Recommendation for future development	Error! Bookmark not defined.

REFERENCES	41
APPENDIX A SAMPLE APPENDIX 1	43
APPENDIX B SAMPLE APPENDIX 2	45

LIST OF TABLES

Table 2.1	CNN parameter	14
Table 4.1	Accuracy model of 5 neighbour	31
Table 4.2	Accuracy model of 7 neighbour	32
Table 4.3	Accuracy model of 9 neighbour	33
Table 4.4	Accuracy of SVM model	33
Table 4.5	Accuracy of Decision Tree model	34
Table 4.6	Accuracy of CNN model	34

LIST OF FIGURES

Figure 2.1	Layers in ANNs	6
Figure 2.2	CNNs layer architecture	8
Figure 2.3	Intersection over union	11
Figure 2.4	MobilNet V2 convolutional block diagram	12
Figure 2.5	list of pre-trained object	15
Figure 2.6	Examples of haar feature	16
Figure 3.1.1	Flow diagram of the project to test the model accuracy	18
Figure 3.1.2	Deep learning required package for this system	19
Figure 3.1.3	Without face mask dataset	20
Figure 3.1.4	Wearing face mask dataset	21
Figure 3.1.5	Wearing face mask incorrect dataset	21
Figure 3.1.6	labelling an image for detect a mask	22
Figure 3.1.7	Sample coding	23
Figure 3.1.8	RGB to Grayscale	24
Figure 3.1.9	Classification to define hyperplane using SVM	25
Figure 3.1.10	Impurity of data in Decision Tree	26
Figure 3.1.11	Sample coding	26
Figure 3.1.12	Convolutional Neural Network architecture for the system	27
Figure 3.1.13	Sample coding	27
Figure 3.1.14	Sample coding	28
Figure 3.1.15	Confusion Matrix	29
Figure 4.1	KNN model 5 neighbour confusion matrix	31
Figure 4.2	KNN model 7 neighbour confusion matrix	32
Figure 4.3	KNN model 9 neighbour confusion matrix	33
Figure 4.4	CNN model flow diagram for face mask detection	34
Figure 4.5	File require to use DNN face detector	35
Figure 4.6	Training result for 3 different classes	36
Figure 4.7	Training result for 2 different classes	36
Figure 4.8	Real time camera detection for 3 different class	37
Figure 4.9	Real time camera detection for 2 different class	37
Figure 4.10	More person face mask detection	38
Figure 4.11	Low brightness to make a detection	38

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
KNN	K-Nearest Neighbour
SVM	Support Vector Machine
GUI	Graphic User Interface
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
DNN	Deep Neural Network
SSD	Single Shot Detector

CHAPTER 1

INTRODUCTION

1.1 Project Background

For the past 2 years, the world has been strike by the new type of virus that called COVID-19 and new norm have been implemented. Because of that, people have been forced to wear a face mask and keep sanitize to prevent the virus spread.

Certain respiratory viral infections, such as COVID-19, necessitate the use of a clinical mask. The general public should know if they should wear the mask for source control or avoid COVID-19. Masks have the potential to reduce sensitivity to noxious individuals during the "pre-symptomatic" stage, as well as stigmatize certain persons who use masks to avoid virus spread.

The covid-19 virus has brought about a new normal existence in which social distance and the use of face masks play an important part in restricting the virus's spread. However, most people do not wear face masks in public place, which contributes to the transmission of infections. To avoid such situations, we must sanitize and make people aware about the importance of wearing face masks. Humans are not permitted to participate in this process due to the risk of being affected by virus.

Face mask detection will determining whether or not someone is wearing a mask, comparable to detecting any object in a scene and then determining whether or not it has a mask on it. Many methods for object detection have been proposed, and deep learning give the best result when come to detect the object.

1.2 Problem Statement

Some people are stubborn to wearing a face mask in public place and entering certain premise. This problem will troublesome the premise and will increase the need to place a worker to monitor the people coming in at the entrance to prevent the infections. With this face mask detection system, it will detect if a person wears a face mask or not before entering a building or premise by using real time camera. According to [1], expression recognition, facial tracking, and position estimation are all required for face detection. The aim is to identify the face in a single photograph given a single image. Face identification is challenging since faces alter in size, shape, colour, and other characteristics and are not immutable. When an opaque image is occluded by something other than the camera, for example, the task becomes tough.

1.3 Objective of Project

This final year project has three main objectives that are related to the title. The objectives must be met based on the four objectives listed below in order to develop and ensure that this system runs smoothly and efficiently:

1. Determine the best machine learning and deep learning model for the system.
2. To analyze the performance and accuracy each of the model.
3. Develop computer vision-based system focused on the real time monitoring of people to detect the present of the face mask.

1.4 Scopes of Project

There are several project scopes that has been listed down. Most of these scopes do recover the project objective as well as a project planning to accomplish this project:

1. To investigate and perform analysis to determine the appropriate deep learning or machine learning algorithm that suits the system.
2. To investigate the different between machine learning and deep learning algorithm.
3. The process will be done in Pycharm Software using python language.

1.5 Thesis Overview

The rest of this thesis is structured as follows:

Chapter 2 presents the literature review of research done by others field of face mask detection, type of machine learning that suit for object detection and the architecture of neural network. The object detection algorithm used for the system and how image processing using architecture of CNN.

Chapter 3 discusses the research technique and algorithm utilised in this project's face mask detection system.

Chapter 4 discuss about the performance of the project by carrying out analysis on the results performance for each algorithm used.

Chapter 5 concludes thesis with summary of the contributions and suggestions of the future research direction with regards to the issue

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The literature review is an important part of this chapter because it helps to do some research on journals and articles as well as determine the nature of the research. Research is carried out by reading or researching journals on the topic of investigating and analysing deep learning and machine learning algorithms for face mask detection systems. Many journals were discovered that can be used as a reference and guide for future work in the final year project. The purpose of this chapter is to review current parameters in order to learn from earlier attempts on a discovery and to propose our strategy or concept for improving the system under development. This section focuses on how others analyse raw data or get excellent results. The different methods used by others will be assessed to acquire information for a precise use in the development of the system. This chapter characterizes ideas, key terms, and formulae for examination.

2.2 Machine Learning

Machine learning is an artificial intelligence discipline that is widely described as a machine's capacity to emulate intelligent human behaviour. Artificial intelligence systems are used to tackle complicated issues in the same way that people do. The objective of artificial intelligence is to construct computer models that demonstrate "intelligent behaviours" similar to humans. This refers to machines that can recognise a visual picture, understand a natural language text, or perform a physical activity. Writing a programme for the machine to follow, such as training a computer to recognise photographs of various objects, takes time or is impossible. While humans may readily

do this activity, teaching a computer to do so is tough. Machine learning takes the idea of letting computers to learn on their own.

Data, such as financial transactions, images of people or things, documents, and so on, is the starting point for machine learning. The data is gathered and prepared for use as training data, or information used to train the machine learning model. The more data there is, the more effective the programme will be. The programmers then choose a machine learning model to employ, supply the data, and let the computer model to train itself to detect patterns or make predictions. The model may also be tweaked by a human programmer over time, including modifying its parameters, to give more accurate results. As a result of training on examples again and over, it can discover patterns and make predictions about the future, developing the ability to reason. When a machine is fed a vast quantity of data, it uses Machine Learning Algorithms to learn how to understand, process, and analyse it. Because it is defined using labelled datasets to train algorithms that reliably categorise data or predict outcomes, supervised learning is the appropriate sort of machine learning for the system.

2.2.1 Neural Network

Neural networks are a type of machine learning algorithm that is widely used. Artificial neural networks (ANNs) are based on the human brain and consist of thousands, or millions of interconnected processing nodes organised into layers. Cells, or nodes, are connected in an artificial neural network, with each cell processing inputs and producing output that is sent to other neurons. Data that has been labelled moves through the nodes, or cells, with each cell performing a different function. The different nodes in a neural network trained to identify whether a picture contains a cat or not would assess the information and arrive at an output that indicates whether a picture contains a cat or not.

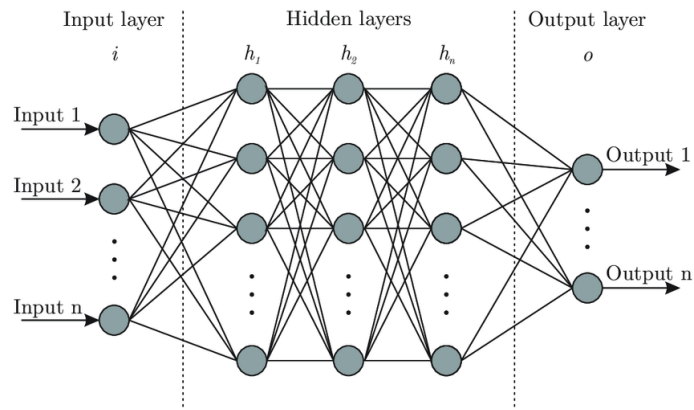


Figure 2.1 Layers in ANNs

2.2.2 Supervised Learning

Supervised learning, often known as supervised machine learning, is a machine learning and artificial intelligence subcategory. It is distinguished by the use of labelled datasets to train algorithms that properly categorise data or predict outcomes. As input data is fed into the model, the weights are adjusted until the model is well fitted, which occurs as part of the cross-validation process. Supervised learning assists enterprises in solving a wide range of real-world issues on a large scale, such as categorising spam in a distinct folder from your email. In Supervised Learning, the dataset on which we train our model is labelled. There is a clear and distinct mapping of input and output. When it comes to data mining, supervised learning may be divided into two sorts of challenges.

1. *Classification*

An algorithm is used in classification to properly allocate test results to certain groups. It detects certain entities in the dataset and attempts to form judgments about how those things should be labelled or described. Linear classifiers, support vector machines (SVM), decision trees, k-nearest neighbour, and random forest are examples of common classification techniques.

2. *Regression*

Regression is a statistical method for determining the connection between dependent and independent variables. It is widely used to produce forecasts, such as those

for a company's sales revenue. Popular regression techniques include linear regression, logistic regression, and polynomial regression.

K-Nearest Neighbour (KNN)

The KNN algorithm assumes that similar things exist in close proximity . In other words, similar things are near to each and hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) calculating the distance between points on a graph.

KNN is a computationally intensive approach since it calculates the distance between each data point and every point in the training set and KNN algorithm requires the entire dataset to make a prediction. This means that every time a prediction is produced, the algorithm must wait for the provided data to be compared to each point. To make a prediction, KNN also needs that the complete data set be stored into memory. It is feasible to load the data into RAM in batches, but this is incredibly time intensive.

Support Vector Machine (SVM)

Support Vector Machines (SVM) are commonly employed for high-dimensional data classification jobs and allow us to categorise data that does not have a linear connection. When compared to KNN, SVM performs better with large dimensionality features by producing a hyperplane to split the data with a straight line. The decision boundary is a hyperplane that separates the classes of data on each side of the plane.

Decision Tree

Feature tests represent each internal node to be like a flowchart structure. Class label is like each leaf node and feature conjunction that led to class label is the branch. This decision tree is used to predictive modelling in statistic and mining data.

2.3 Deep Learning

Deep Learning is a machine learning area[7] focused like a brain functionality which is artificial neural networks algorithm. Raw input will be extract higher

information level from Multiple. layers in image processing, for example, may recognise edges, whereas higher levels may identify human-relevant concepts like as numerals, characters, or faces. The great majority of current deep learning models are constructed on artificial neural networks, notably convolutional neural networks (CNNs)[8]. Deep learning levels learn to turn incoming data into an abstract. The raw input could be a pixel matrix when use recognition image the first layer could abstract the pixels and recognize the edges. The second layer could compose the edge arrangements. The third layer could recognize a nose and eyes and image contain person face will recognize in forth layer.

CNNs are used in Deep Learning research for Computer Vision applications like as Image Classification and Object Recognition.[2]. Dataset of an images must be train to predict the output from its learned patterns in order to recognise and classify the object. This method is known as "Supervised Learning," and it requires an external dataset of labelled images to predict the label of an unseen image. The primary distinction between a typical Artificial Neural Network (ANN) and a CNN is that a CNN has just one completely linked layer, as illustrated in Figure 2, whereas an ANN connects each neuron to all other neurons. [2]. Because the size of the pictures often leads to over-fitting, ANNs are not ideal for image processing. Because the size of the pictures often leads to over-fitting, ANNs are not ideal for image processing, it will cause of the noise and inaccurate value to the data. Consider whether the picture size is appropriate (32x32x3). This image must be flattened into a vector with 3072 rows if it is to be fed into an ANN (32x32x3). The ANN's first layer needs have 3072 weights in order to receive this input vector. It generates a complicated vector (270,000 weights) for bigger photos, such as (300x300x3), which needs the usage of a more capable CPU.[2].

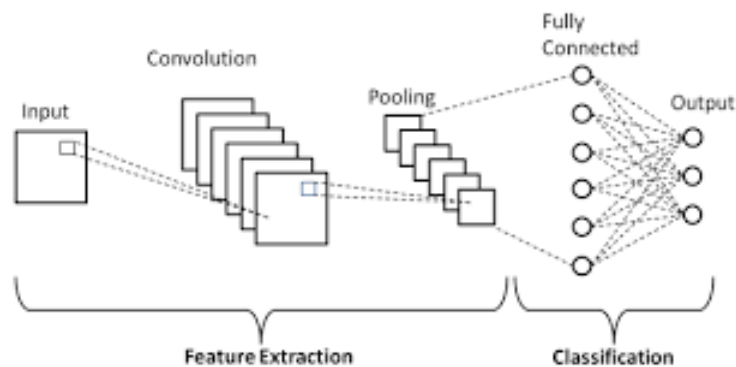


Figure 2.2 CNNs layer architecture

2.3.1 Convolutional Neural Network

CNNs are a set of layers that combine to execute a mathematical operation on an image. It takes in the raw pixel intensity rather than utilising typical feature extraction approaches. A [30x30] colour picture, for example, will be sent to CNN's input layer as a three-dimensional matrix. CNN learns complicated picture attributes automatically by integrating the outputs of several layers with "learnable" filters to forecast the input image's class or label probability.[2]

There are three types of layers in a Convolutional Neural Network (CNN), which is:

- Convolutional (CONV)

This layer is where CNN applies filters to learn features from the input image, it is the most important layer in any CNN architecture. Filters and feature maps make up this layer[2].

- Pooling (POOL)

This layer serves as an intermediary in the network, compressing or down sampling the incoming volume along spatial dimensions[2].

- Fully Connected (FC)

Similar to ANNs, FC layer has neuron that are fully connected to the neurons in the previous layer. For multi-class classification problems, activation function "softmax" have been used each of the layer to produce the output for another layer. The FC layer is in charge of predicting the input image's final class or label. As a result, the output dimension is [1x1xN], where N is the number of classes or labels that are considered for classification[2].

2.4 Object Detection algorithm model

A real-time object detection model designed to run on non-GPU computers and may benefit users of low-configuration computers. There are numerous improved object detection algorithms available, including YOLO, Faster R-CNN, Fast R-CNN, R-CNN, Mask R-CNN, R-FCN, SSD, RetinaNet, and others. YOLO is a Deep Neural Network algorithm for object detection that is faster and more accurate than most others. YOLO is intended for GPU-based computers with a graphics card that is larger than 12GB.

Object detection model are often divided into two type of architecture which is single stage object detectors like YOLO and SSD and dual stage object detectors like R-CNN etc. The primary distinction between this two is region of interest (ROI) is defined first then subsequently conducted just on the region of interest. So, two-stage object detection models are more accurate in compared to one-stage models, but they slower and demand more higher computer specification.

2.4.1 YOLO

You Only Look Once (YOLO) was developed to provide a one-step approach that included item detection and categorization. On GPU systems, YOLO achieves around 45 FPS, and a light version known as Tiny-YOLO manages roughly 244 FPS.[3]. YOLO can predict the class and the appears of bounding boxes. The supplied picture is first separate into $S \times S$ grids[3]. Second, each grid cell has B bounding boxes, each with an associated confidence score. The chance that an object exists in each enclosing box is defined as confidence.

$$C = \Pr(\text{Object}) * IOU_{Pred}^{truth}$$

Intersection over union (IOU) is an acronym for a measure of overlap between two bounding boxes it must be between 0 and 1. As shown in Figure 3, the intersection is the overlapped region between the predicted bounding box and the ground truth (Object), and the union is the total area of the predicted and ground truth boxes. For accurate detection, ground truth must closer to projected box bounding and the value of IOU must near to value 1.

$$IOU = \frac{\textit{Area of Overlap}}{\textit{Area of Union}}$$

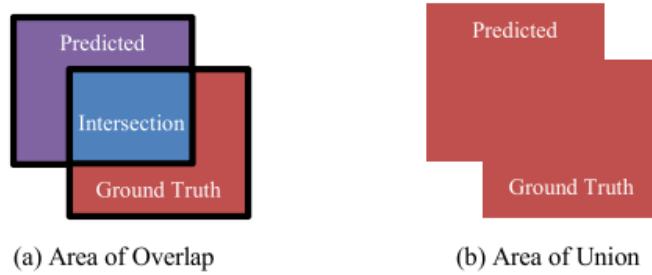


Figure 2.3 Intersection over union

2.4.2 Single-Shot Detector

There are two parts when using SSD which is backbone model and an SSD head. As a feature extractor, the backbone model is often an images classification that have been pre-trained. This is often a ResNet-trained network that has had the last fully linked classification layer removed. As a result, a deep neural network that can extract semantic meaning from an input image while keeping its spatial structure, although at a lesser resolution. The backbone for ResNet34 produces 256 7x7 feature maps for an input picture. More convolutional layer added to backbone is called SSD head with the outputs read as bounding boxes and classifications of objects in the spatial position of the final layer activations.

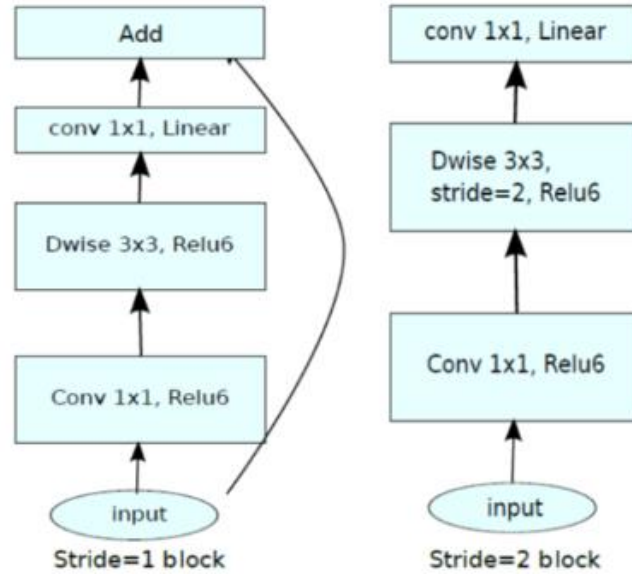


Figure 2.4 MobilNet V2 convolutional block diagram

There are two kinds of blocks in MobileNetV2. The first is stride 1 block that refer to residual block and another option for shrinking is a block with a stride 2 block. Each block contains 3 layers. Convolution 1x1 using ReLU6 is on 1st layer, depth wise convolution is on 2nd layer and lastly convolution 1x1 with linearity on 3rd layer.

To develop the model, there is several frameworks or dependencies that need to be install before running the algorithm such as:

- TensorFlow: It is a Google-developed deep learning framework that can be used to design, build, and train models. However, it necessitates a significant amount of GPU processing power and is Linux-friendly[4].
- OpenCV: It has a deep learning framework and was created by Intel. It only works on CPUs, and it's simple to set up on Windows.

2.5 Image Pre-processing

The image is converted to grayscale during the pre-processing step because the RGB colour image contains too much redundant information for face mask detection. Each pixel in an RGB colour image is stored with a 24-bit value. The grayscale images included adequate information for categorization and saved 8 bits for each pixel. To retain the regularity of the input pictures to the architecture, the images were moulded into (64x64) forms. The images are then normalised, and a pixel's value after normalisation

ranges from 0 to 1. To get the value, the data must be divided by 255 because it the rescale image from 0-255. Normalization helps the learning algorithm in learning faster and more efficiently.[5].

2.5.1 Architecture of Deep Learning

The deep learning architecture learns a variety of important nonlinear features. The learned architecture is then used to predict previously unseen samples. Images from various sources are collected to train deep learning architecture. The architecture of the learning technique is heavily reliant on CNN[5].

2.5.1.1 Dataset Collection

The dataset can be obtained at the GitHub or Kaggle website because this classification someone already made to reduce the time to produce dataset. But dataset also can be own create if the classification is a new classification that not already made.

2.5.1.2 Architecture of CNN

The learning model is based on CNN, a pattern recognition algorithm that is very useful for images[6]. An input layer, several hidden layers, and an output layer make up the network. Multiple convolution layers make up the hidden layers, which learn appropriate filters for extracting important features from the given samples. Multiple dense neural networks use the features extracted by CNN for classification purposes. Table 2.1 shown the architecture of develop network.

Table 2.1 CNN parameter

Layer	Type	Kernel	Kernel Size	Output Size
1	Convolution2D	32	(3×3)	(62×62×32)
2	Convolution2D	32	(3×3)	(60×60×32)
3	MaxPooling2D	-	(2×2)	(30×30×32)
4	Convolution2D	32	(3×3)	(28×28×32)
5	Convolution2D	32	(3×3)	(26×26×32)
6	MaxPooling2D	-	(2×2)	(13×13×32)
7	Convolution2D	32	(3×3)	(11×11×32)
8	Convolution2D	32	(3×3)	(9×9×32)
9	MaxPooling2D	-	(2×2)	(4×4×32)
10	Flatten	-	-	512
11	Dense	-	-	100
12	Dropout	-	-	100
13	Dense	-	-	30
14	Dropout	-	-	30
15	Dense	-	-	10
16	Dropout	-	-	10
17	Dense	-	-	2

2.5.2 COCO dataset

Many objects detection, face detection and more have been using computer vision application and COCO dataset. The COCO dataset by Microsoft is a large-scale object recognition and segmentation tool. It is commonly used by Computer Vision and Machine Learning engineers. Understanding visual scenes is a goal of computer vision. This process involves identifying objects that are present and determining their attributes. Figure 2.5 shown the pre-trained 80 objects are included in the COCO dataset classes for object detection and tracking.

```
'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck',  
'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench',  
'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra',  
'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',  
'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove',  
'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork',  
'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli',  
'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant',  
'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',  
'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book',  
'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
```

Figure 2.5 list of pre-trained object

2.6 Feature Extraction

Extraction of features is a method of removing extraneous data information, lowering computing costs while retaining vital and relevant data. Additionally, the decreased data aids in increasing the model's learning rate. Furthermore, for feature extraction, real-time face mask identification employs machine learning and deep learning algorithms. Deep learning relies on neural networks to extract characteristics without the need for human involvement. Several backbones such as MobileNetv2 and Xception will do the feature extraction of the input data[7]. Following that, the output is sent to the classifier network, which categorises a person with or without a mask. Machine Learning model that are useful to get the extraction are the algorithms such as histogram of oriented gradients(HOG) and Principal Component Analysis(PAC).

2.7 Face detector

Other name of face detection is face recognition, function as computer technique that searches for and recognises human faces in digital pictures using artificial intelligence (AI). Also utilised in a multitude of industries such as personal safety, biometrics and security, to offer real-time monitoring and tracking of individuals. There is some face detector that commonly use such as:

a. Haar cascade classifier

This Haar feature is used to assess whether there is face present on the image. Each haar characteristic has a value, which may be computed by summing the each of rectangle area. The integral image notion made it simple to calculate the area of a rectangle.

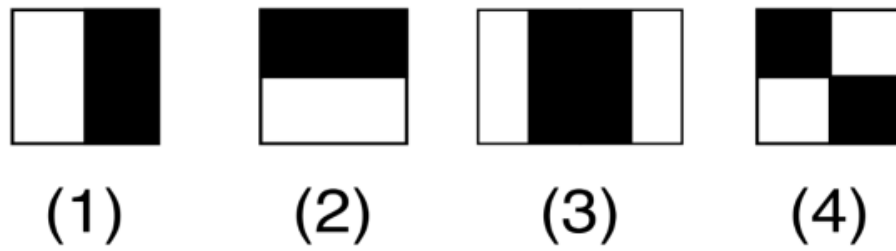


Figure 2.6 Examples of haar feature

This classifier computes the value of a feature using the rectangular integral. The haar cascade classifier increases the weight of each rectangle by its area, and the results are put together.

b. DNN Face Detector in OpenCV

OpenCV's most current version has a Deep Neural Network (DNN) module with an excellent pre-trained face identification convolutional neural network (CNN). The new model outperforms existing methods such as Haar in terms of face detection. Caffe is the framework that will be utilised to train the new model. It contains out-of-the-box Haar cascades, but a pre-trained deep learning face detector has been included from version 3.3. The SSD framework are include in DNN face detector which leverages a base network like ResNet-10

c. HOG classifier

The Histogram of Oriented Gradients, or HOG, is a feature descriptor that is frequently used to extract features from image data. It is widely used for object detection in computer vision tasks. The HOG descriptor focuses on an object's structure or shape. HOG is also capable of providing edge direction. The gradient and orientation (magnitude and direction) of the edges are extracted in this way. Furthermore, these orientations are computed in 'localised' portions. This means that the entire image is divided into smaller regions, and the gradients and orientation for each region are calculated before generating a histogram for each of these regions separately. The gradients and orientations of the pixel values are used to generate the histograms.

All this face detector has their pros and cons because certain face detector need best CPU to run although it will take a long time to complete the running. By comparing with other face classifier in term of accuracy, DNN face detector have a better accuracy but low FPS and also easy to use because the require file has been made that contain the weight and the network architecture. HOG classifier is the most popular nowadays and have higher FPS but unable to detect face at odd angle and Haar Cascade classifier is outdated classifier and give not a best result. So after make a research about best face detector to use, this system used the DNN Face detector.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explains the methods and strategies that were used, as well as the steps that must be considered when developing the system by gaining the information from doing literature review. Firstly, the language that suitable for this project are by using python language because it is the most popular programming language and have a huge community support. Pycharm is a software used to create algorithm of the system. The algorithm is based on the deep learning architecture that is commonly use in face and object detection system.

3.2 Flow diagram of the project

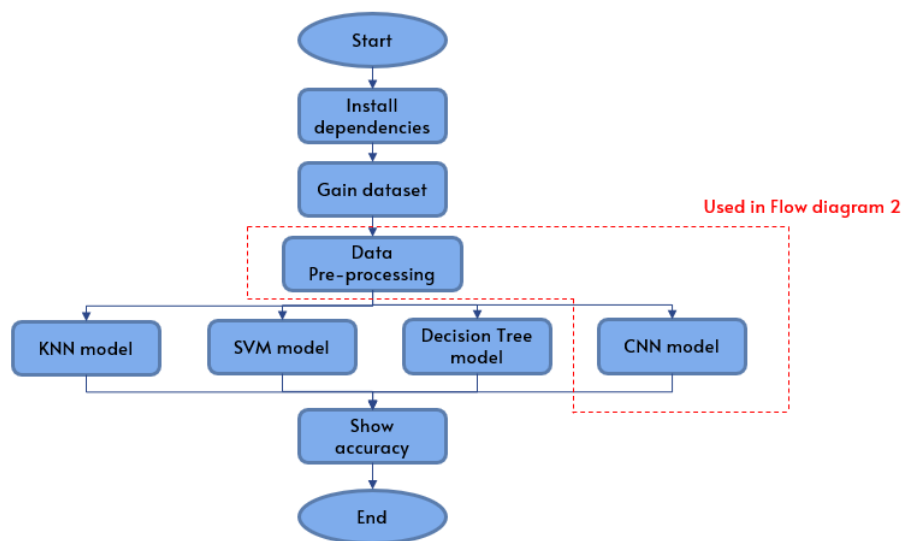


Figure 3.1.1 Flow diagram of the project to test the model accuracy

Figure 3.1.1 show the flow diagram to determine the best model to used in face mask detection system by showing the accuracy of each model. Certain dependencies need to be install first to perform certain operation. Then to gain the dataset, because the python community are bigger, the dataset can be get into certain website like github, kaggle and etc to reduce the time gaining the dataset or create our own dataset if data is a new data to be classify.

3.2.1 Dependencies

The dependencies (packages) or a framework must be installed. TensorFlow, Keras, which covers TensorFlow's numerical calculation libraries and allows you to design and train only a few lines of code, and OpenCV are the required dependencies.



Figure 3.1.2 Deep learning required package for this system

a. TensorFlow

TensorFlow, a programming interface for expressing machine learning algorithms, is used to fabricate Machine Learning in a many science of computer. TensorFlow is a complete open-source machine learning platform. It includes a broad, huge number of tools, libraries, and community resources for developing and deploying ML-powered applications, including data reshaping.

b. Keras

Keras provides critical thoughts and building blocks for quickly developing and implementing machine learning arrangements. The scalability and cross-platform capabilities of TensorFlow are fully utilised. Keras' core data structures are layers and models. Keras is used to implement CNN all layers.

c. OpenCV

OpenCV (Open-Source Computer Vision Library) is a computer vision and machine learning software library that can be used to distinguish and recognise faces, recognise objects, group movements in recordings, trace progressive modules, follow eye gestures, track camera actions, expel red eyes from flash photos, find comparative pictures from an image database, perceive landscape and overlay it with enhanced reality, and so on. In the scaling and colour conversion of data pictures, the suggested technique makes advantage of these OpenCV capabilities.

3.2.2 Dataset

Dataset from the GitHub consist of 6000 images and has been classify into three different classes which is image of people wearing facemask, without facemask and with mask incorrect. All the image has many variations of size and resolution. Because of that the image will be resize into 64x64 in data pre-proessing.

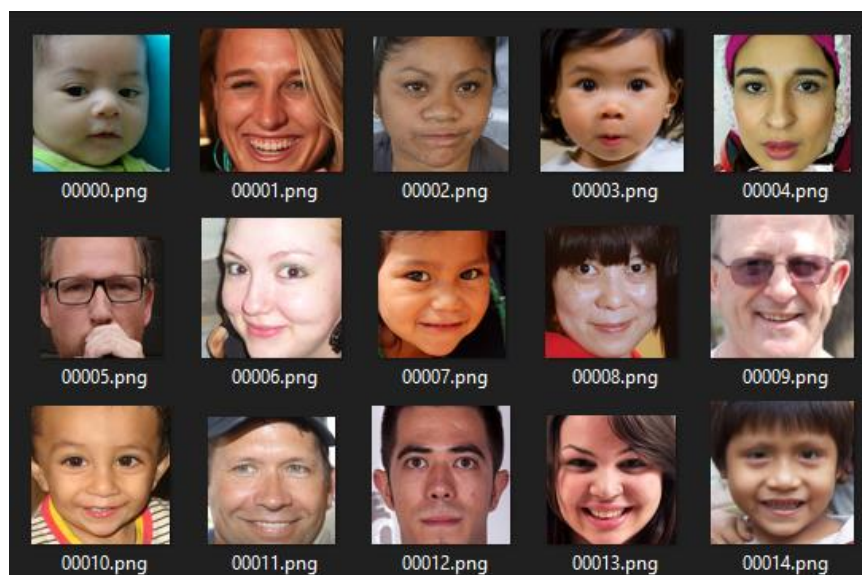


Figure 3.1.3 Without face mask dataset



Figure 3.1.4 Wearing face mask dataset

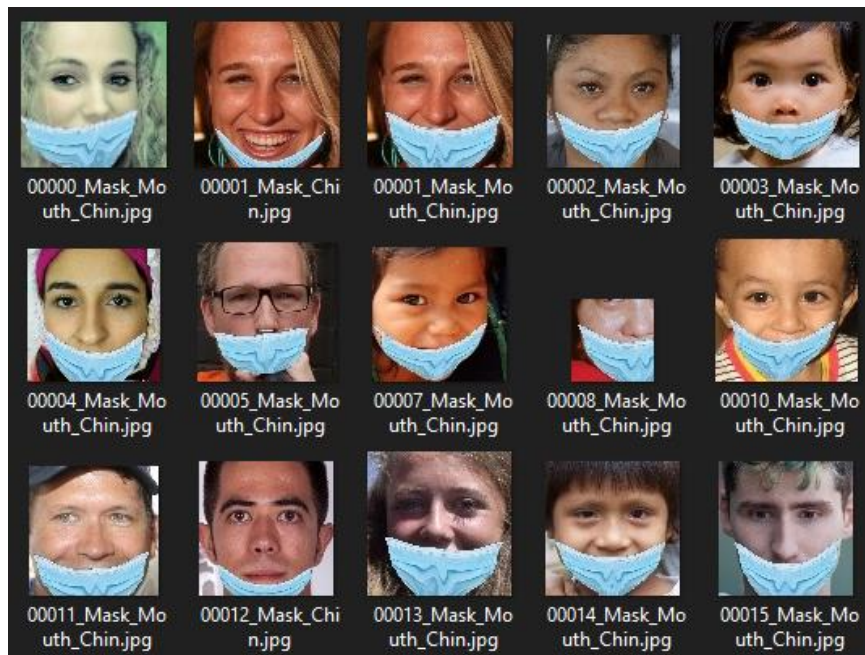


Figure 3.1.5 Wearing face mask incorrect dataset

When to create our own dataset, the data and image that have been capture need to be labelling first before train because it's used to identify the raw data. This labelling images are called bounding box. The training data is then used to construct a computer vision model that may be used to automatically classify pictures, determine item position, identify important spots in an image, or image segment. Figure 3.1.6 show the label image of person wearing the face mask. Huge number of label image needed before train to

make the model learning the image that have been label. Because the more data to obtain, the more time consume needed. Another alternative to obtain the data are by download the dataset at the open-source website like GitHub and Kaggle.



Figure 3.1.6 labelling an image for detect a mask

3.2.3 Data Pre-processing

Pre-processing is the stage in which the image is resized. The image would be enormous once captured hence it would be easy. Transforming the data and utilising it into another process is called data pre-processing. These structured data may be used with an information model or composition and record the relationships between multiple entities. Pre-processing is the part where the resizing of the image is involved. The image would be large once captured therefore it would be easy. Data pre-processing is the process of converting the data and used it into another process. These organised data are compatible with model information[7]. If the images were smaller than 80x80 pixels, and the faces were even smaller, the image is scaled up by a factor of 2 or more. The image is resized into (224x224) pixels to get the best result.

a. Visualised the data

Process to translate abstract data into meaningful representations for knowledge exchange and insight finding. It is beneficial to investigate a certain trend in the dataset[8].

```
DIRECTORY = "C:/data/" # Windows/PC
CATEGORIES = ['face_no_mask', 'face_with_mask_correct', 'face_with_mask_incorrect']
IMG_SIZE = 64

def create_data():
    for category in CATEGORIES:
        path = os.path.join(DIRECTORY, category)
        class_num_label = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                img_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                X.append(img_array)
                y.append(class_num_label)
            except:
                pass
```

Figure 3.1.7 Sample coding

The total number of images in the dataset is visualised in 'with mask', 'with mask incorrect' and 'without mask' categories. The *os.listdir(path)* statement categorises the list of directories in the specified data path. Then it will proceed with convert image to grayscale and resizing the image. In this project, image size have been set to 64x64.

b. Conversion RGB to Grayscale

Modern descriptor-based image recognition systems routinely work on grayscale images without delving into the method used to convert from RGB image to grayscale. This is due to the fact that when using robust descriptors, the color-to-grayscale technique has little effect. Non-essential data may increase the quantity of training data required for exceptional performance. Grayscale is used to extract descriptors rather than working on colour photographs straight immediately since it streamlines the process and reduces computer requirements[9].

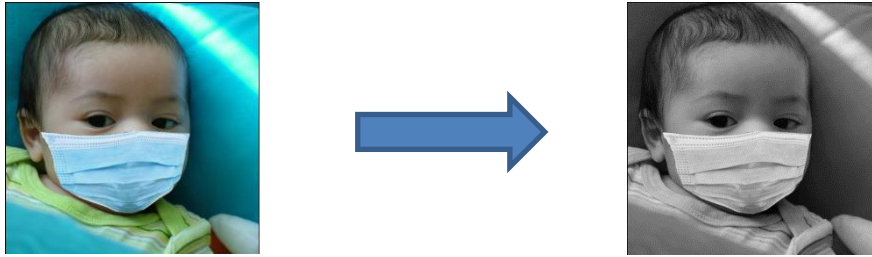


Figure 3.1.8 RGB to Grayscale

c. Reshaping Image

During image relevation, the input is a three-dimensional tensor with a prominent unique pixel in each channel. All the images must be the same size and correspond to the same 3D feature tensor[10]. However, neither images nor their corresponding feature tensors are normally coextensive. Most CNNs can only accept images that have been fine-tuned. This causes a number of issues during data collection and model implementation. However, reconfiguring the input images before augmenting them in the network can assist in overcoming this limitation.

3.2.4 Train Model

In this stage, certain parameter will be configured in KNN, SVM, Decision Tree and CNN model to determine the accuracy best model suit for the system.

a. KNN

The number of neighbours and the kind of distance are the parameter settings utilised in the KNN method. The number of neighbours must be an odd number utilised as 5, 7 and 9 and the distance types are Minkowski. Minkowski will measure designed for vector spaces with real values. Normed vector space will give the result calculate for distance which refer to the space where distances length must be positive.

$$\left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (1)$$

Equation (1) show the general form for Minkowski distance and different distance can be get by manipulated the value of p. When set the value of p = 1

will give Manhattan distance and set $p = 2$ will give Euclidean distance. Because using Python library SKlearn so the default setting is 5 neighbour and Euclidean distance.

b. SVM

Optimum hyperplane can be determined by increase the distance between the classes. The maximum distance between data items in each class is defined as the distance between classes[11]. Figure 3.1.9 shown the illustration to define hyperplane.

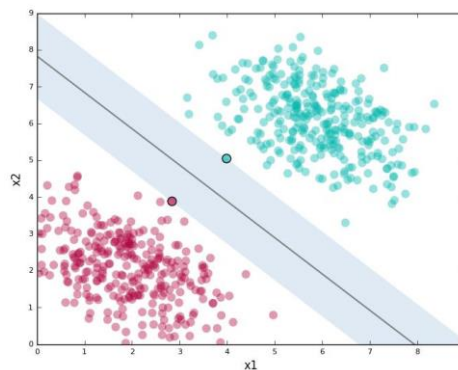


Figure 3.1.9 Classification to define hyperplane using SVM

When compared to KNN, SVM performs better with large dimensionality features by producing a hyperplane to split data with a straight line. Red and Blue dot indicate the classes of the data then SVM will strive to determine the optimal dividing line boundary between the two sets of data and this is the process of learning for SVM[11].

c. Decision Tree

For this model, decision tree is used to calculate the entropy and information that have been gain. Computing the entropy yields the amount of uncertainty in the data, whereas calculating the difference in the entropy yields the information gain.

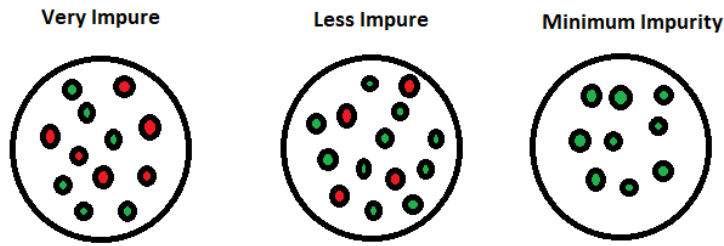


Figure 3.1.10 Impurity of data in Decision Tree

Entropy is a metric used in information theory that evaluates the impurity or uncertainty in a set of data. It controls how a decision tree divides data.

$$H(X, Y) = - \sum_{x_i \in X} \sum_{y_i} p(x_i, y_i) \log p(x_i, y_i) \quad (2)$$

The value of x_i and y_i is the probability of $p(x_i, y_i)$ shown in equation (2).

d. CNN

SSD mobilNetV2 have been load into CNN which is a pre-trained model for object detection. ReLU activation function and MaxPooling layers are added after the first Convolution layer. The Convolution layer is taught by hundreds of filters. Each of the filter is to determine the edge of the image object. The size of the kernel can be set into certain size based on height and width convolution of two dimensional. It should be aware of the predicted shape of the input, input shape information must be sent to the first layer of the model. Following layers are capable of conducting shape reckoning instinctively.

```
base_model = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input
(shape=(IMG_SIZE, IMG_SIZE, 3)))

# Construct the head of the model that will be placed on top of the base model
head_model = base_model.output
head_model = AveragePooling2D(pool_size=(7, 7))(head_model)
head_model = Flatten(name="flatten")(head_model)
head_model = Dense(128, activation="relu")(head_model)
head_model = Dropout(0.5)(head_model)
head_model = Dense(NUM_CLASS, activation="softmax")(head_model)
```

Figure 3.1.11 Sample coding

The Conv2D class's activation parameter is set to "relu." It depicts a nearly linear function with all of the benefits of linear models, including the ease with which gradient-descent methods may be used to optimise it. By choosing the Max Pooling size of 7 x 7, the spatial dimensions can be lowered. The next convolution layer has 100 filters, each of which is activated using the ReLU activation function and is followed by another MaxPooling layer as illustrated in Figure 3.1.12.

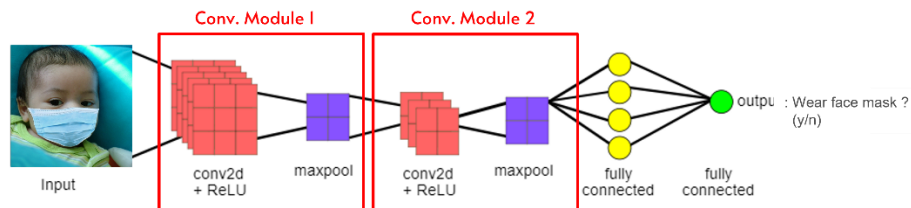


Figure 3.1.12 Convolutional Neural Network architecture for the system

To avoid overfitting, a Dropout layer with a 50% probability of setting inputs to zero is added to the model. A Dense layer of 64 neurons with a ReLU activation function is then added. In the last layer (Dense), which has two outputs for two categories, the Softmax activation function is applied.

The compile method must first be used to configure the learning process. The "adam" optimizer is used here. As a loss function, categorical crossentropy, also known as multiclass log loss, is employed (the objective that the model tries to minimize). Because this is a classification problem, the metrics are set to "accuracy" as shown in Figure 3.1.13.

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
              metrics=["accuracy"])
```

Figure 3.1.13 Sample coding

- e. Split the data into train and test

When generating a prediction, having a good model and a well-designed train test split can help to get accurate results. The test size is set at 0.1, implying that 90% of the data in the dataset is utilised for training and 10% for testing. The sequential is then fitted to the pictures in the training and test sets. In this project, 20% of the training data is used for validation. The model is trained for 20 epochs (iterations) in order to strike a compromise between accuracy and overfitting risk. This will consist of what it thinks that the image is either people wearing facemask, wearing mask incorrectly or not wearing a mask. It will actually consist of probabilities, such as 70% for wearing facemask, 20% for not wearing and 10% for wearing mask incorretly. This is where the error need to be reduced.

```
# initialize the initial learning rate, number of epochs to train for,  
# and batch size  
INIT_LR = 1e-4  
EPOCHS = 20  
BS = 32
```

```
# train the head of the network  
print("[INFO] training head...")  
H = model.fit(  
    aug.flow(trainX, trainY, batch_size=BS),  
    steps_per_epoch=len(trainX) // BS,  
    validation_data=(testX, testY),  
    validation_steps=len(testX) // BS,  
    epochs=EPOCHS)
```

Figure 3.1.14 Sample coding

3.2.5 Calculation of Performance

In each model algorithm, performance testing has been calculate using performance criteria employed include accuracy, precision, recall, and F1 score by referring to the confusion matrix in Figure 3.1.15. The best performing parameter in one algorithm will be chosen and compared to another model.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 3.1.15 Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad (3)$$

Accuracy values is calculated using Equation (3). TP stand for true positive, and FP stand for false positive which mean this is a positive class population while TN and FN stand for true negative and false negative respectively which include them in negative class population[11]. Then to calculate the precision is by using Equation (4).

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

After that, next evaluation is Recall. The proportion of truth of the positive class prediction relative to the entire data with an actual positive label is used to compute Recall and the calculation can be compute using Equation (5).

$$Recall = \frac{TP}{TP + FN} \quad (5)$$

And lastly to calculate the F1-score is by using Equation (6)

$$F1 - score = \frac{2 * (Precision * Recall)}{(Precision + Recall)} \quad (4)$$

All this equation are included in the SKlearn library package Python. Just use the function of classification report and the result will appear. But certain model will require certain time to train and testing.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

This chapter discussed the results obtained during the process of completing the final year project. The performance result of certain model can be obtained by using the Python library SKlearn and then the best model will be used to the system.. In order to achieve the best detection in real-time video, the camera resolution must be high, and the brightness must be standardized.

4.2 Model Accuracy

This is the performance value of each of the model after train and testing the dataset of three different class which is wear face mask, wear face mask incorrect and not wearing face mask by also showing confusion matrix and the value of accuracy, precision, recall and F1-score by using the calculation method that have been proposed before.

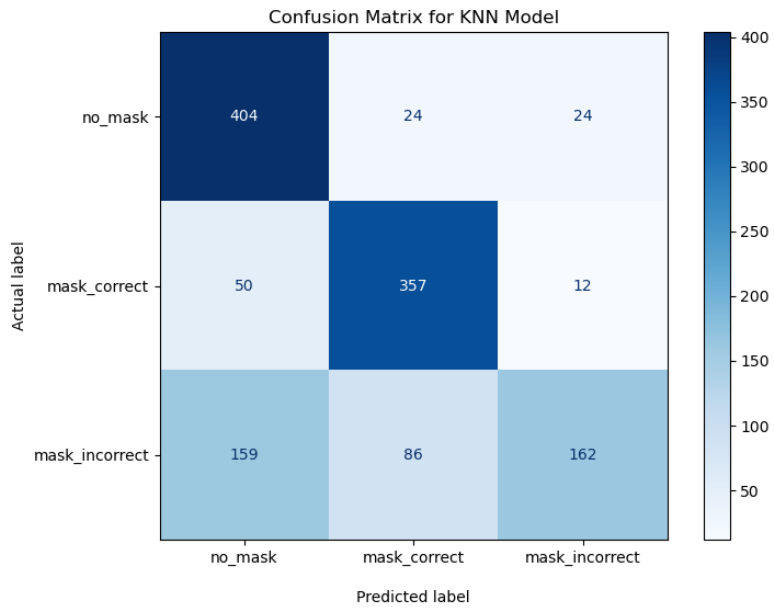


Figure 4.1 KNN model 5 neighbour confusion matrix

Table 4.1 Accuracy model of 5 neighbour

	Precision	Recall	F1 score
wear mask incorrect	0.82	0.4	0.54
wear mask	0.76	0.85	0.81
not wear mask	0.66	0.89	0.67
Accuracy	0.72		

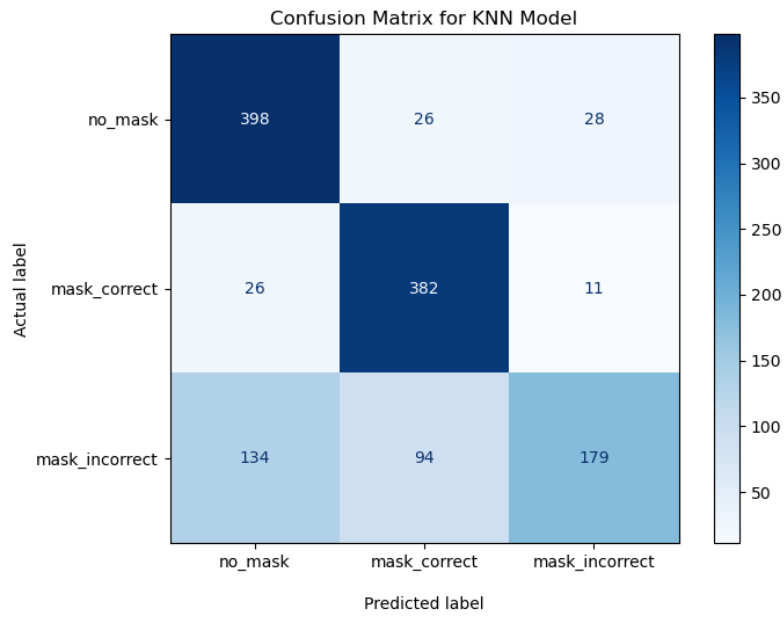


Figure 4.2 KNN model 7 neighbour confusion matrix

Table 4.2 Accuracy model of 7 neighbour

	Precision	Recall	F1 score
wear mask incorrect	0.82	0.44	0.57
wear mask	0.76	0.91	0.83
not wear mask	0.71	0.88	0.79
Accuracy	0.75		

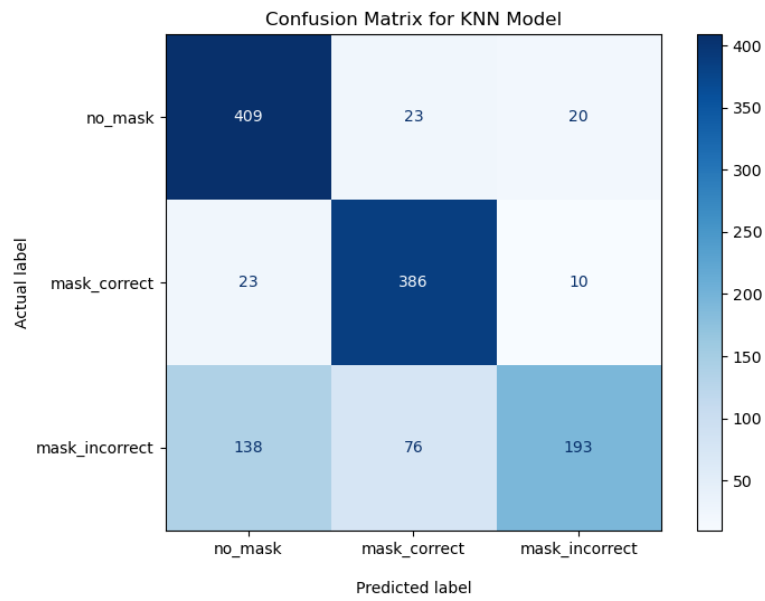


Figure 4.3 KNN model 9 neighbour confusion matrix

Table 4.3 Accuracy model of 9 neighbour

	Precision	Recall	F1 score
wear mask incorrect	0.87	0.47	0.61
wear mask	0.8	0.92	0.85
not wear mask	0.72	0.9	0.8
Accuracy	0.77		

Table 4.4 Accuracy of SVM model

	Precision	Recall	F1 score
wear mask incorrect	0.68	0.68	0.68
wear mask	0.93	0.96	0.95
not wear mask	0.5	0.11	0.68
Accuracy	0.9		

Table 4.5 Accuracy of Decision Tree model

	Precision	Recall	F1 score
wear mask incorrect	0.83	0.85	0.83
wear mask	0.92	0.94	0.93
not wear mask	0.87	0.85	0.86
Accuracy	0.87		

Table 4.6 Accuracy of CNN model

	Precision	Recall	F1 score
wear mask incorrect	0.94	0.97	0.95
wear mask	0.99	0.98	0.99
not wear mask	0.98	0.95	0.97
Accuracy	0.97		

By refer to the result, CNN have the higher model accuracy compared to others with 97% accuracy while KNN give the lowest accuracy with 72% but with the value of neighbours increase the accuracy also increase. So, CNN are the best model to this system. Figure 24 shown the flow diagram of the system using CNN model.

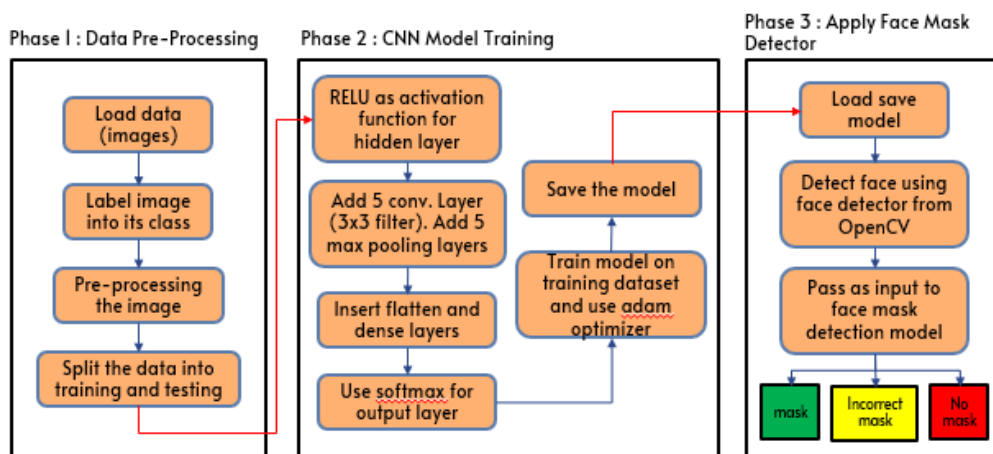
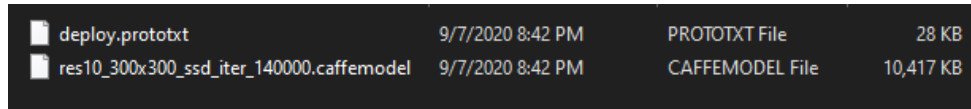


Figure 4.4 CNN model flow diagram for face mask detection

After train the CNN model, the data need to be loaded into face detector model which is DNN face detector model using OpenCV package. To utilise the DNN face detector with OpenCV, download the Caffe file which contain two type of file which is “deploy.prototxt” and “res10_300x300_ssd_iter_140000.caffemodel”



deploy.prototxt	9/7/2020 8:42 PM	PROTOTXT File	28 KB
res10_300x300_ssd_iter_140000.caffemodel	9/7/2020 8:42 PM	CAFFEMODEL File	10,417 KB

Figure 4.5 File require to use DNN face detector

4.3 Face mask detection performance

Figure 4.6 illustrates that after training, validating, and testing the model on the datasets, the technique achieves up to 99 percent accuracy for three separate classes. One of the key reasons for obtaining this degree of precision is MaxPooling. It provides basic internal representation while minimising the number of parameters that the model must learn. By down sampling the input picture representation, this sample-based discretization approach decreases its dimensionality. The initial epoch set for this training is 20 but with the error reduced close to zero value, it stops at epoch 13. The main purpose for training the model is to reduce the error close to zero as possible. By increase the number of epoch iteration, the error will be reduced, and the accuracy of training will increase.

When cover a face with mask or a hand, the system can recognise partially obscured faces with excellent accuracy. To eliminate between annotated mask and hand-covered face, it examines the degree of occlusion of four places – nose, mouth, chin, and eye. As a result, the model will only regard a mask that totally covers the face, including the nose and chin, as "with mask." The method's key issues include shifting perspectives and a lack of clarity. The unclear shifting faces in the live feed make it much more challenging.

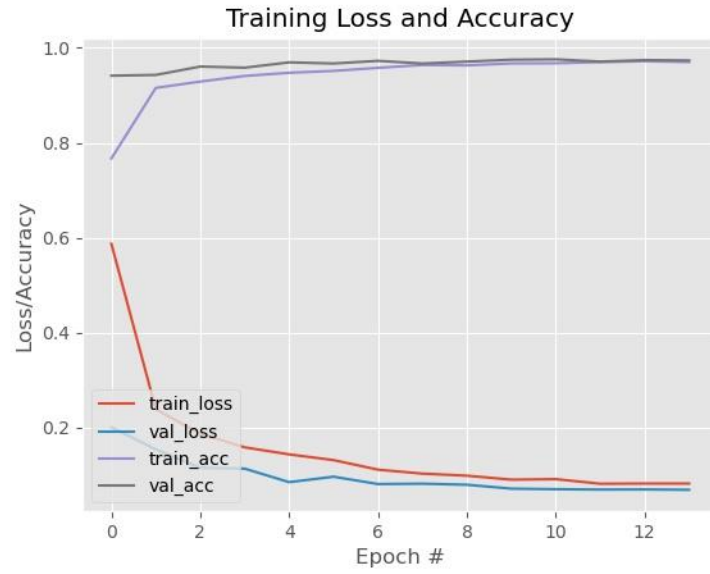


Figure 4.6 Training result for 3 different classes

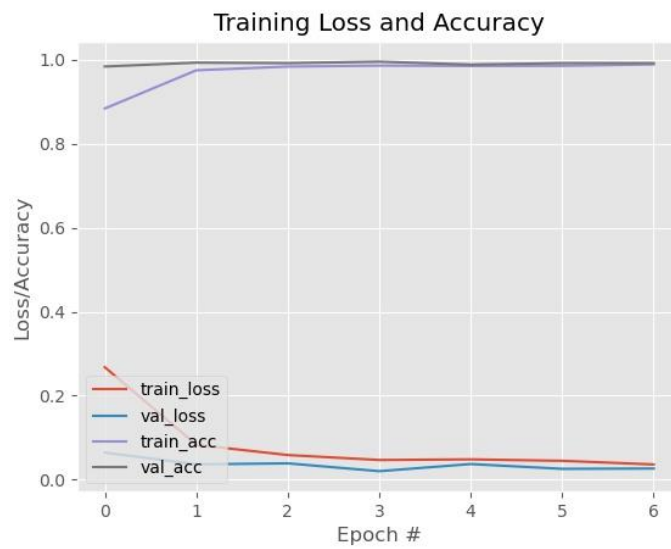


Figure 4.7 Training result for 2 different classes

For Figure 4.7, same initial epoch set with 3 different classes (20 epoch). The iteration stops at epoch 6 because the error has been reduced to very minimum value. Figure 4.8 show the output camera for detection of 3 different class.

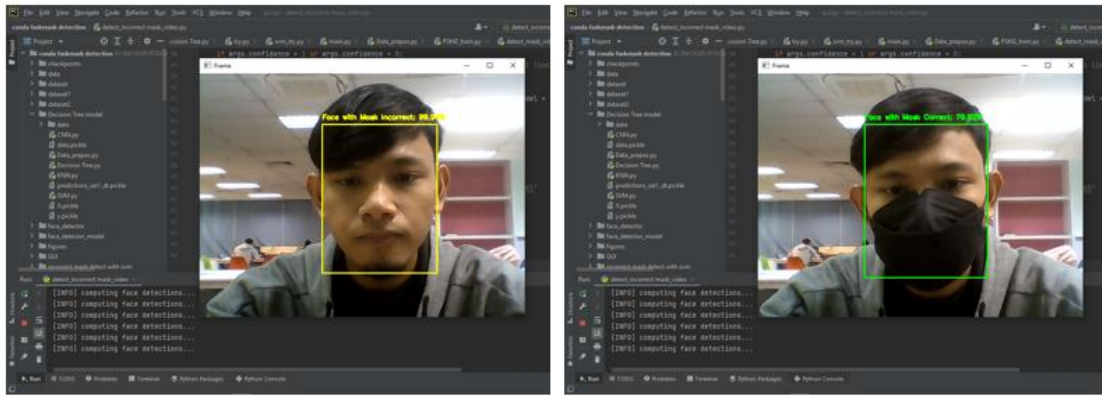


Figure 4.8 Real time camera detection for 3 different class

The output shows not as expected because the detection to detect the mask are not accurate such as the person not wearing a face mask, but the detection show “face with mask incorrect” and the accuracy of person with wearing a face mask are lower which is around 70%-80%. This expected output and accuracy are very different with the CNN model that have been train before.

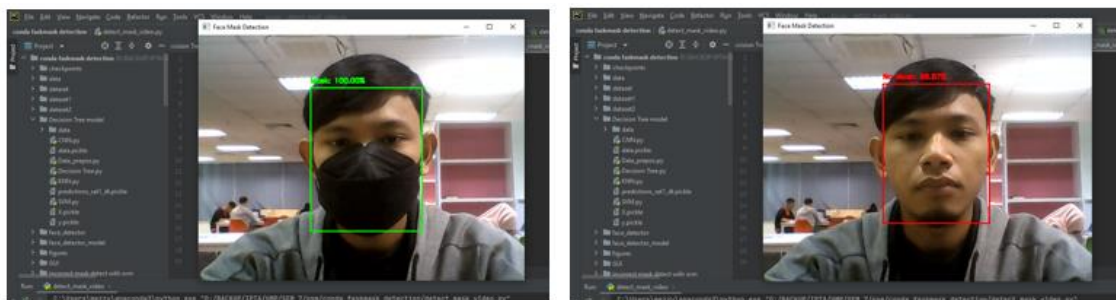


Figure 4.9 Real time camera detection for 2 different class

As shown in Figure 4.9, detection of 2 different class give the best output for detection because the accuracy for detect a face mask and not wearing a face mask is higher and the detection also can detect two or more Mask Detection in one frame like shown in Figure 4.10.

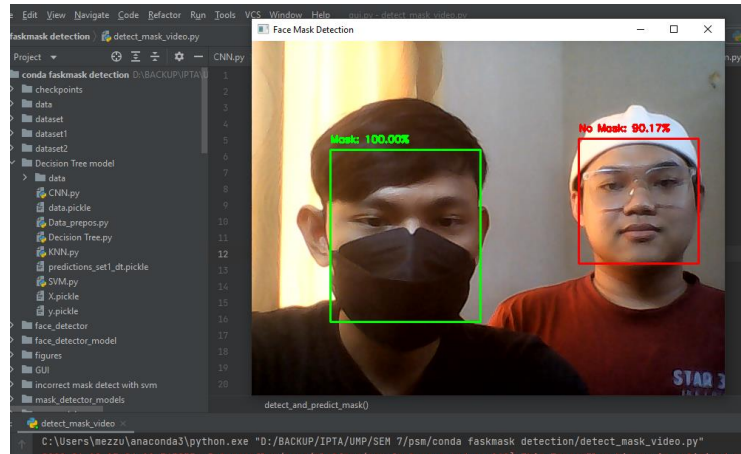


Figure 4.10 More person face mask detection

Brightness also needs to be considered for detection, because the system cannot detect the object if the light is too low as shown in Figure 4.11.

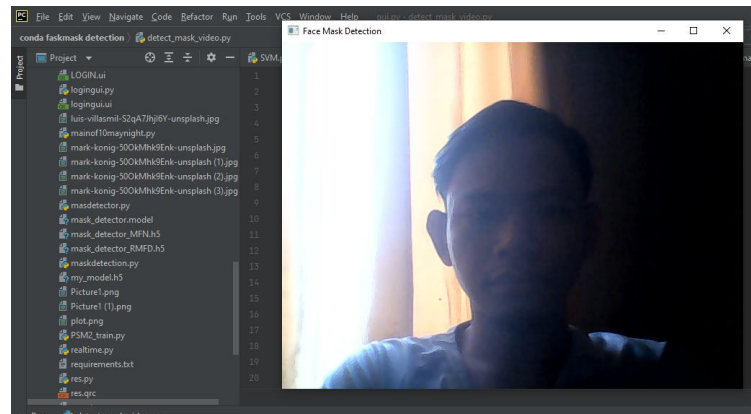


Figure 4.11 Low brightness to make a detection

For KNN model, the value of neighbour can be manipulated to increase the accuracy of model, the time compute for the model to compute is 1.78 seconds for 5 neighbour, 2 seconds for 7 neighbour and lastly 1.85 seconds for 9 neighbours. Here to remember that KNN cannot read large amount of data because it will lead to noise of the data. That comes with SVM model to overcome the drawback of KNN. Face detector are used to detect the face of people within the camera frame. The system could detect faced based on the brightness of the video.

CHAPTER 5

CONCLUSION

5.1 Introduction

As a conclusion, the artificial intelligence techniques known as the techniques that were used increasingly to model environmental systems. The techniques covered are machine learning, deep learning, artificial neural networks, convolutional neural network and also supervised learning. This algorithm is well-suited to serve as a face detector in a face mask detection system before displaying the final result of whether or not people are wearing masks. The Python language, in conjunction with the Pycharm software, proved to be extremely useful, as it handled all of the process and because of it have a huge community support, to make an algorithm is easy by referring to many open sources site. Artificial intelligence is excellent for a wide range of system development, particularly image detection and recognition, but it requires additional training and troubleshooting to achieve high accuracy in the final result. The application created with the proper model will undoubtedly make all users accessible and everyone's life more convenient.

5.2 Recommendation for future development

The present built programme can recognise people faces as well as face masks. This application may be improved further by:

1. Troubleshooting the algorithm of detection for 3 different classes which is wearing face mask, not wearing face mask and wearing face mask incorrect. With this, more detection can be made.

2. Make the system of door automatically open for a certain premise when someone wearing a face mask or not wearing face mask properly. This requires camera and certain board such as Raspberry Pi and Arduino to operate. Camera will automated detect the present of face mask. If someone wear mask properly, the door will open and vice versa.

3. Make a Graphic User Interface (GUI) for more interactive system function

5.3 Impact to Society

Because of the intriguing feature of artificial intelligence, this application will undoubtedly have a large influence on each individual, where it is able to detect a face mask present on the face because wearing face mask have become a new norm to our society. Although the restriction to wear a face mask is now loose in public, the application can be use inside certain premise and industry place for safety user. This will make sure the safety environment when people wearing a face mask to entering certain place.

REFERENCES

- [1] A. Malge, H. M. Dhaduk, and P. M. Mallikarjuna Shastry, "An approach to face detection and recognition using viola jones," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 5 Special Issue, pp. 52–56, 2019.
- [2] I. Gogul and V. S. Kumar, "Flower species recognition system using convolution neural networks and transfer learning," *2017 4th Int. Conf. Signal Process. Commun. Networking, ICSCN 2017*, no. March, 2017, doi: 10.1109/ICSCN.2017.8085675.
- [3] G. Plastiras, C. Kyrkou, and T. Theocharides, "Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing," *ACM Int. Conf. Proceeding Ser.*, 2018, doi: 10.1145/3243394.3243692.
- [4] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2016, [Online]. Available: <http://arxiv.org/abs/1603.04467>.
- [5] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud, and J. H. Kim, "An automated system to limit COVID-19 using facial mask detection in smart city network," *IEMTRONICS 2020 - Int. IOT, Electron. Mechatronics Conf. Proc.*, 2020, doi: 10.1109/IEMTRONICS51293.2020.9216386.
- [6] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, *A survey of the recent architectures of deep convolutional neural networks*, vol. 53, no. 8. Springer Netherlands, 2020.
- [7] B. Suvarnamukhi and M. Seshashayee, "Big Data Concepts and Techniques in Data Processing," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 10, pp. 712–714, 2018, doi: 10.26438/ijcse/v6i10.712714.
- [8] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 8, pp. 2674–2693, 2019, doi: 10.1109/TVCG.2018.2843369.
- [9] C. Kanan and G. W. Cottrell, "Color-to-grayscale: Does the method matter in image recognition?," *PLoS One*, vol. 7, no. 1, 2012, doi: 10.1371/journal.pone.0029740.
- [10] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation," *J. Big Data*, vol. 6, no. 1, 2019,

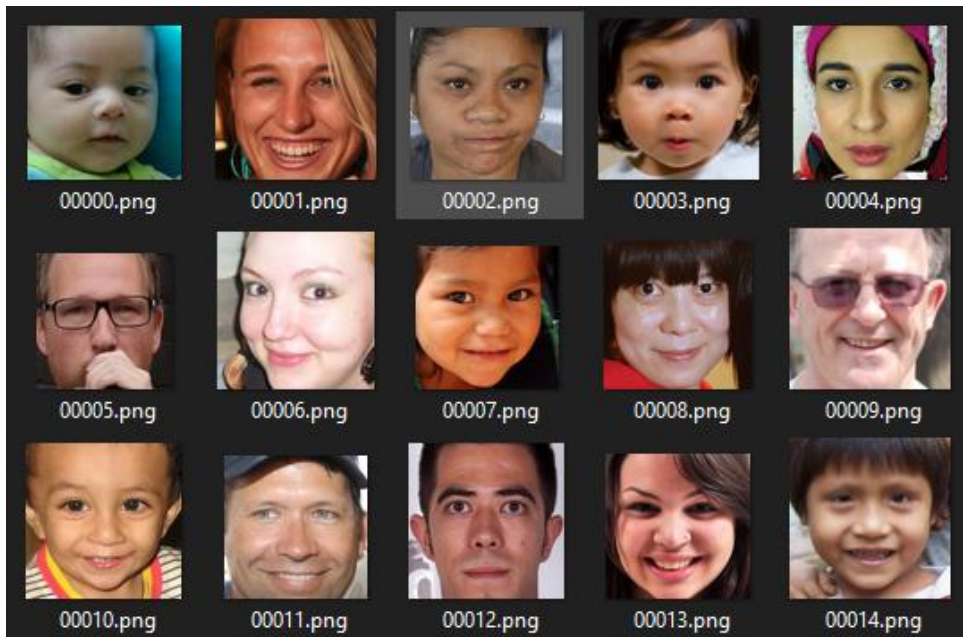
doi: 10.1186/s40537-019-0263-7.

- [11] M. Nur and Y. Utomo, “Face Mask-e Wearing Detection Using Soft-Margin Support Vector Machine (SVM),” vol. 10, no. 2, pp. 72–81, 2021, doi: 10.14421/ijid.2021.3038.
- [16] P. Viola and Michael Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017*, pp. 1193–1197, 2001, doi: 10.1109/ICECDS.2017.8389630.
- [17] M. Mohanty and R. Sikka, “Android application to detect drowsiness during driving vehicle,” *Mater. Today Proc.*, no. xxxx, pp. 2–4, 2021, doi: 10.1016/j.matpr.2021.03.202.
- [18] L. Cai, J. Zhu, H. Zeng, J. Chen, C. Cai, and K. K. Ma, “HOG-assisted deep feature learning for pedestrian gender recognition,” *J. Franklin Inst.*, vol. 355, no. 4, pp. 1991–2008, 2018, doi: 10.1016/j.jfranklin.2017.09.003.
- [19] Y. Wei, Q. Tian, J. Guo, W. Huang, and J. Cao, “Multi-vehicle detection algorithm through combining Harr and HOG features,” *Math. Comput. Simul.*, vol. 155, pp. 130–145, 2019, doi: 10.1016/j.matcom.2017.12.011.
- [20] A. Raghunandan, Mohana, P. Raghav, and H. V. R. Aradhya, “Object Detection Algorithms for Video Surveillance Applications,” *Proc. 2018 IEEE Int. Conf. Commun. Signal Process. ICCSP 2018*, pp. 563–568, 2018, doi: 10.1109/ICCSP.2018.8524461.

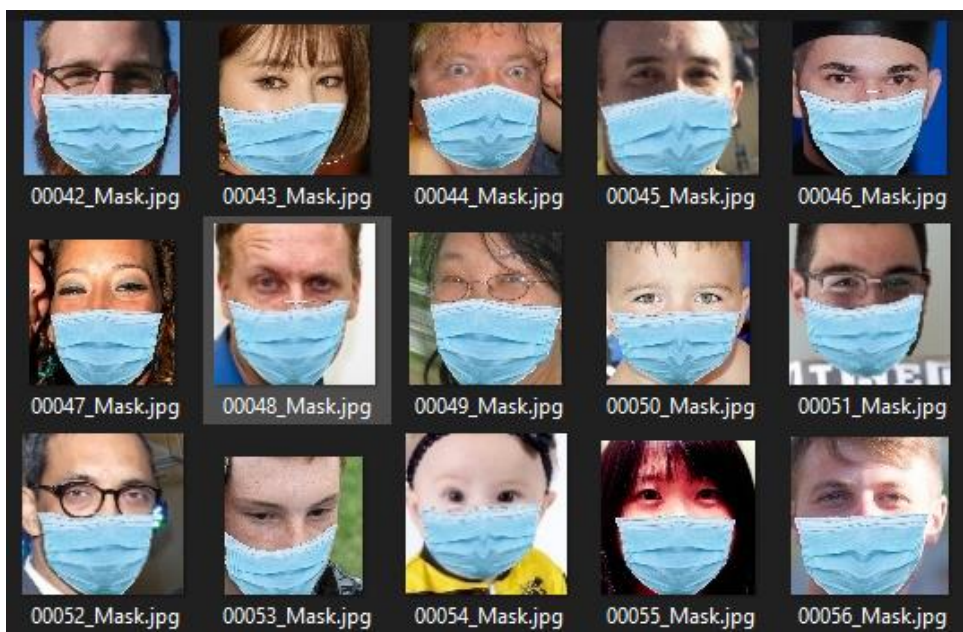
APPENDIX A SAMPLE APPENDIX 1

Datasets of 3 different classes that obtain from the GitHub and Kaggle.

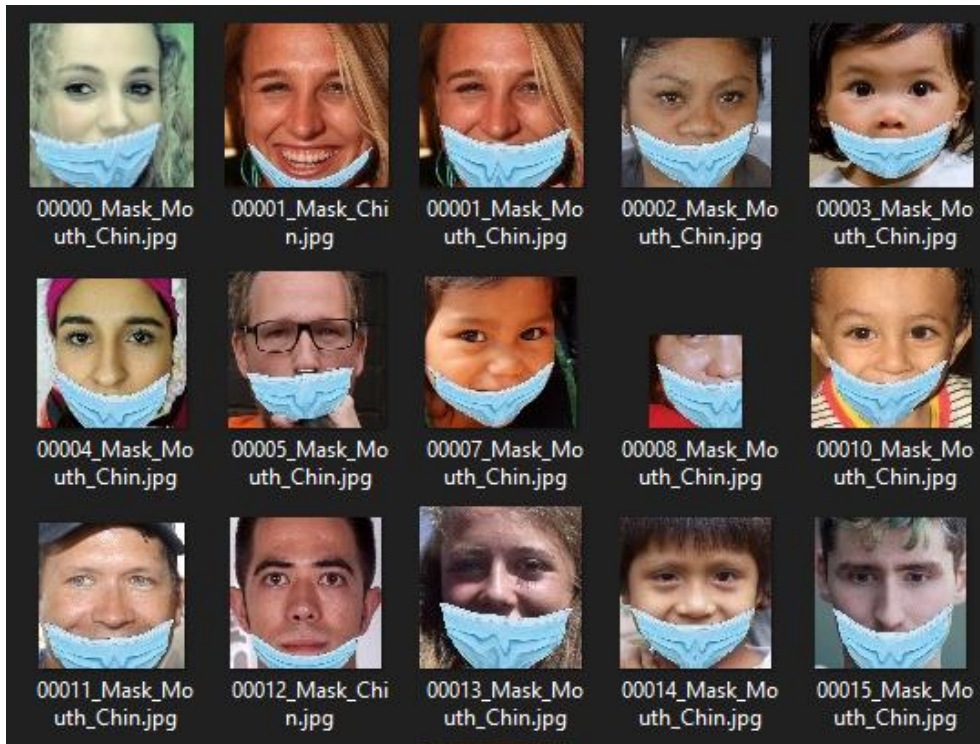
1. Not wear a mask



2. Wear a face mask



3. Wear an incorrect face mask



APPENDIX B

SAMPLE APPENDIX 2

1. Code for Data Pre-processing

```
#import libraries
from sklearn.model_selection import train_test_split
import numpy as np
import os
import PIL
import cv2
import pickle

DIRECTORY = "C:data/" # Windows/PC
CATEGORIES = ['face_no_mask', 'face_with_mask_correct',
'face_with_mask_incorrect']
IMG_SIZE = 64 # IMG_SIZE = 224 alternative size

# data
X = []
# labels(0,1,2)
y = []

def create_data():
    for category in CATEGORIES:
        path = os.path.join(DIRECTORY, category)
        class_num_label = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path, img),
cv2.IMREAD_GRAYSCALE)
                img_array = cv2.resize(img_array, (IMG_SIZE,
IMG_SIZE))
                X.append(img_array)
                y.append(class_num_label)
            except:
                pass

create_data()
SAMPLE_SIZE = len(y)
data = np.array(X).flatten().reshape(SAMPLE_SIZE, IMG_SIZE*IMG_SIZE) #
pixel-features

# Turn X and y into numpy arrays
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE) # images
y = np.array(y) # target

print("Features, X shape: ", X.shape)
print("Target, y shape: ", y.shape)
print("Data shape: ", data.shape)

#Saves us from having to regenerate our data by saving our data
pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
```

```

pickle.dump(y, pickle_out)
pickle_out.close()

pickle_out = open("data.pickle", "wb")
pickle.dump(data, pickle_out)
pickle_out.close()

pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)
pickle_in = open("data.pickle", "rb")
data = pickle.load(pickle_in)

print('# of Samples:', len(y))
print('# of face_no_mask:', (y == 0).sum())
print('# of face_with_mask_correct:', (y == 1).sum())
print('# of face_with_mask_incorrect:', (y == 2).sum())

# Split our data into testing and training.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=45)

# Print the length and width of our testing data.
print('Length of our Training data: ', len(X_train), '\nLength of our
Testing data: ', len(X_test))

```

2. code for KNN model

```

# import packages
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns # for confusion matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # to plot image, graph
import pickle
import time # for computation time assessment
from time import time

start = time()

pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)

pickle_in = open("data.pickle", "rb")
data = pickle.load(pickle_in) # Data Matrix will serve as X

print('# of Samples:', len(y))
print('# of face_no_mask:', (y == 0).sum())
print('# of face_with_mask_correct:', (y == 1).sum())
print('# of face_with_mask_incorrect:', (y == 2).sum())

```

```

# Get Column Names
cols = []
for i in range(0, len(data[0])):
    cols.append("P" + str(i))

# Convert to Dataframe
numpy_data = data
X = pd.DataFrame(data=numpy_data, columns=cols)
print(X.head())

y = pd.DataFrame(data=y, columns=["Mask_Target"])
print(y.head())

# Shape
print('\nImage Data Shape:', X.shape)
print('Image Data Shape Features:', data.shape)
print('Image Data Shape Target:', y.shape)

# Normalize the pixel values (0-255)
X = X / 255.0

# Split our data into testing and training.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=45)

# Print the length and width of our testing data.
print('Length of our Training data: ', len(X_train), '\nLength of our
Testing data: ', len(X_test))

# Initialize KNN model
knn = KNeighborsClassifier(n_neighbors=7)
# Use training data to fit KNN model
knn.fit(X_train, y_train.values.ravel())
# make prediction on entire test data
predictions_set1 = knn.predict(X_test)
# Calculate Confusion Matrix
cm = confusion_matrix(y_test, predictions_set1)
# Calculate accuracy
accuracy = accuracy_score(y_test, predictions_set1)
print('Model accuracy is: ', accuracy)
print("Time compute: ", time()-start)
print("\nClassification Report KNN\n", classification_report(y_test,
predictions_set1))

#plot confusion matrix knn
from sklearn.metrics import plot_confusion_matrix
class_name = ['no_mask', 'mask_correct', 'mask_incorrect']
disp = plot_confusion_matrix(knn, X_test, y_test,
display_labels=class_name, cmap=plt.cm.Blues)
plt.ylabel('Actual label')
plt.xlabel('\nPredicted label')
disp.ax_.set_title('Confusion Matrix for KNN Model')
plt.show()

```

3. Code for SVM model

```
# For data management
import pickle
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import train_test_split

# For plotting
import matplotlib.pyplot as plt

pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)
pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)
pickle_in = open("data.pickle", "rb")
data = pickle.load(pickle_in)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)

svm = svm.SVC()
svm.fit(X_train, y_train)

y_pred = svm.predict(X_test)
accuracy = svm.score(X_test, y_test)
print("Accuracy %f" % accuracy)
metrics.accuracy_score(y_true=y_test, y_pred=y_pred)

print(metrics.classification_report(y_test, y_pred))
```

4. Code for Decision Tree

```
# Import Metrics, Classifier and Graphing Packages
from sklearn.metrics import accuracy_score, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
import seaborn as sns # for confusion matrix
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # to plot image, graph
import pickle
import time # for computation time assessment
from time import time

pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in) # 3D Feature set

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in) # 1D Target set

pickle_in = open("data.pickle", "rb")
data = pickle.load(pickle_in) # 2-D Feature Set, Data matrix will
serve as X
```

```

print('# of Samples:', len(y))
print('# of face_no_mask:', (y == 0).sum())
print('# of face_with_mask_correct:', (y == 1).sum())
print('# of face_with_mask_incorrect:', (y == 2).sum())

# Get Column Names
cols = []
for i in range(0, len(data[0])):
    cols.append("P" + str(i))

# Convert to Dataframe
numpy_data = data
X = pd.DataFrame(data=numpy_data, columns=cols)
print(X.head())

y = pd.DataFrame(data=y, columns=["Mask_Target"])
print(y.head())

print('\nImage Data Shape:', X.shape)
print('Image Data Shape Features:', data.shape)
print('Image Data Shape Target:', y.shape)

# Normalize the pixel values
X = X / 255.0

# Split our data into testing and training.
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=45)

# Print the length and width of our testing data.
print('Length of our Training data: ', len(X_train), '\nLength of our
Testing data: ', len(X_test))

# Initialize Decision Trees model
decision_trees = DecisionTreeClassifier()
# Use training data to fit Decision Trees model
decision_trees.fit(X_train, y_train.values.ravel())
# Predict Train Data Labels
predictions_set = decision_trees.predict(X_test)
pickle_out = open("predictions_set1_dt.pickle", "wb")
pickle.dump(predictions_set, pickle_out)
pickle_out.close()
# Calculate accuracy
accuracy = accuracy_score(y_test, predictions_set)
print('Model accuracy is: ', accuracy)
print("\nClassification Report DT\n", classification_report(y_test,
predictions_set))

```


5. Code for CNN model

```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import tensorflow
import sklearn
from sklearn.model_selection import train_test_split
import cv2
import seaborn as sns
import pickle
import os

filename = r"D:\BACKUP\IPTA\UMP\SEM 7\psm\conda faskmask
detection\Decision Tree model\X.pickle"
pickle_in = open(filename, 'rb')
X = pickle.load(pickle_in)

filename = r"D:\BACKUP\IPTA\UMP\SEM 7\psm\conda faskmask
detection\Decision Tree model\y.pickle"
pickle_in = open(filename, 'rb')
y = pickle.load(pickle_in)

CATEGORIES = ['face_no_mask', 'face_with_mask_correct',
'face_with_mask_incorrect']

# double check to see the types of the loaded files
print('Type of X:', type(X))
print('Type of y:', type(y))
print('Shape of X:', X.shape)
print('Shape of y:', y.shape)

resized_X = []
for img in X:
    resized_X.append(cv2.resize(img, (64, 64)))

X = np.asarray(resized_X)
X = X.reshape(-1, 64, 64, 1)
print(X.shape)

# normalize the pixel values
X = X / 255.0

IMG_DIM = X.shape[1]
print('IMG_DIM:', IMG_DIM)

# Split dataset into Training and Testing
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)

print('Training Size:', len(X_train))
print('Testing Size:', len(X_test))

cnn_model = tensorflow.keras.models.Sequential()
# Start of Convolution Layers & Maxpooling
cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu',
input_shape=(IMG_DIM,
```

```

IMG_DIM, 1))
cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu'))
cnn_model.add(tensorflow.keras.layers.MaxPool2D())

cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu'))
cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu'))
cnn_model.add(tensorflow.keras.layers.MaxPool2D())

cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu',
input_shape=(IMG_DIM,
IMG_DIM, 1)))
cnn_model.add(tensorflow.keras.layers.Conv2D(filters=64,
kernel_size=3, activation='relu'))
cnn_model.add(tensorflow.keras.layers.MaxPool2D())

# Start of Neural Nets
cnn_model.add(tensorflow.keras.layers.Flatten())
cnn_model.add(tensorflow.keras.layers.Dense(512, activation='relu'))
cnn_model.add(tensorflow.keras.layers.Dropout(0.3))
cnn_model.add(tensorflow.keras.layers.Dense(512, activation='relu'))
cnn_model.add(tensorflow.keras.layers.Dropout(0.3))
cnn_model.add(tensorflow.keras.layers.Dense(256, activation='relu'))
cnn_model.add(tensorflow.keras.layers.Dense(128, activation='relu'))
cnn_model.add(tensorflow.keras.layers.Dense(3, activation='softmax'))
cnn_model.summary()

# Compile the Model
cnn_model.compile(optimizer=tensorflow.keras.optimizers.Adam(),
loss='sparse_categorical_crossentropy',
metrics=['acc'])
# Train Model
epochs = 13
H = cnn_model.fit(X_train, y_train, epochs=epochs,
validation_split=0.2)

# Evaluate performance
cnn_model.evaluate(X_test, y_test)

y_pred = np.argmax(cnn_model.predict(X_test), axis=-1)

# Calculate Confusion Matrix
cm = confusion_matrix(y_test, y_pred)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Model accuracy is: ', accuracy)
print("\nClassification Report CNN\n", classification_report(y_test,
y_pred))

cm = sklearn.metrics.confusion_matrix(y_test, y_pred)

plt.figure(figsize=(9, 9))
sns.heatmap(cm, annot=True, fmt='.0f', square=True, linewidths=.5,
cmap='Blues_r')
plt.ylabel('Actual Label')
plt.xlabel('\nPredicted Label')
print(sklearn.metrics.classification_report(y_test, y_pred))

```

