EVALUATING DEEP TRANSFER LEARNING MODELS FOR FACE MASK DETECTION

GOH PEI JIN

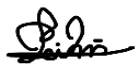
A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : 

Name : Goh Pei Jin

ID No. : 1802410

Date : 21/4/2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"EVALUATING DEEP TRANSFER LEARNING MODELS FOR FACE MASK DETECTION"** was prepared by **GOH PEI JIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Hons) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Khor Kok Chin

Date : 21/4/2022

Signature :

Co-Supervisor :

Date :

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, Dr. Khor Kok Chin for his invaluable advice, guidance and enormous patience throughout the development of the research. His enthusiasm, vision, sincerity and motivation have greatly influenced me. He has taught me the methodology for conducting the research and presenting the findings as clearly as possible. Working and learning under his guidance was a tremendous pleasure and honour. I am grateful for everything he has provided me. I would also want to thank him for his companionship, understanding and wonderful sense of humour, which made the research experience more enjoyable.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and encouraged me. With the generous support and assistance of many individuals, this project will be completed successfully. I would like to express my heartfelt gratitude to every one of them.

ABSTRACT

Due to the fast transmission of coronavirus and the severe sequela of COVID-19, which has no specific cure, the world is facing a massive health crisis. According to the World Health Organization (WHO), wearing a mask in public locations and crowded locations is the most effective prevention of COVID-19. In Malaysia, wearing a face mask is mandatory in public areas. However, it is impossible to detect all passers-by manually as it requires much manpower. This research proposes an automation approach to mask-wearing detection by identifying people who are (i) not wearing a mask, (ii) wearing a mask, (ii) incorrect mask-wearing, and (ii) wearing double masks. Transfer learning methods were adopted by using five pre-trained models: (i) VGG, (ii) MobileNet, (iii) ResNet, (iv) Inception and (v) Xception models. These models were trained based on 2000 real-life data sets collected from various sources with a data augmentation technique. The research results show that the pre-trained ResNet152 model outperformed the other models by achieving 0.8667 accuracy on the testing data set (120 images from the other distribution) and 0.8447 accuracy on the videos captured using a smartphone.

.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
AP	Average Precision
CNNs	Convolutional Neural Networks
COVID-19	Corona Virus Disease 2019
FPS	Frame per second
LR	Learning rate
MCO	Movement Control Order
RNNs	Recurrent Neural Networks
SGD	Stochastic gradient descent
SSD	Single-Shot Detector
VGG	Visual Geometry Group
WHO	World Health Organization

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

COVID-19 pandemic has made an unprecedented change in our day-to-day life and has disturbed the world economy and society. According to the World Health Organisation's (WHO) official Dashboard, on 18 June 2021, COVID-19 has infected approximately 176.95 million people and caused approximately 3.84 million deaths in the world (World Health Organization, 2021). People infected with the COVID-19 may have fever, joint and chest pain, headache, difficulty breathing and depression. The infection may also cause one's life (Mayo Foundation for Medical Education and Research, 2021). The whole world is now combating COVID-19, including Malaysia.

In Malaysia, the first wave of pandemics occurred from 25 January to 16 February 2020. The second wave that happened from 27 February to 30 June 2020 forced the Malaysian government to take various Movement Control Order (MCO) phases to handle the spreading of COVID-19. The third wave began on 8 September 2020 due to the Sabah state election, which has caused the confirmed cases of COVID-19 to soar up until today, and thousands of cases are reported daily (Rampal and Liew,2021). Therefore, precautions like wearing a mask, social distancing, washing hands, avoiding the crowd and disinfection are important to protect us against COVID-19 (Centers for Disease Control and Prevention, 2021).

Wearing a mask is one important approach to prevent COVID-19. Therefore, we need to ensure everyone is wearing a mask throughout the whole pandemic when going outside their home. To ensure everyone is masked needs much manpower from enforcement agencies. Autodetect mask-wearing needs to be implemented to solve the cumbersome workload of manual detection. To detect a face mask, a deep learning technique shall be used to incrementally learn high-level features automatically of data by first recognising pixels, lines, edges and then parts of the object, and finally, the whole object (Mahapatra, 2018). Thousands of masked, unmasked, improper masked wearing, double mask images data shall be used to train deep learning models. The models shall assist the enforcement agencies in performing real-time detection on passers-by.

1.2 Problem Statement

The process of manually detecting whether or not someone is wearing a mask is tedious and inefficient. It is impossible for a human to detect all the passers-by in a location. Moreover, there is not enough manpower to be placed everywhere.

1.3 Project Objectives

a. To detect mask-wearing automatically using deep learning.
b. To evaluate pre-trained deep learning models and select the best in detecting mask-wearing.

1.4 Scope of the Project

The problems can be solved by developing a deep learning model for face mask detection. Jupyter Notebook shall be used for coding purposes.

User Scope Coverage

The target users of this face mask detection using deep learning are those who pass by the sensor of detection to ensure the presence of a face mask.

Research Scope Coverage

The research used deep learning libraries like Keras, OpenCV, TensorFlow, and cv2 to preprocess the data and develop the deep learning model for detecting the mask-wearing. Besides, matplotlib is used for visualising the statistical results.

Scope Not Coverage

The deep learning model cannot detect surgical masks, such as N95.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The practice of wearing face masks in public has become essential during the pandemic to protect us from infection. However, some self-centered individuals do not obey the regulation. Therefore, robust face mask detection must be developed to solve this situation. Global scientific collaboration has risen to unprecedented levels as a result of the coronavirus outbreak. Machine learning and deep learning-based artificial intelligence (AI) have the potential to aid in the battle against Covid-19 in a variety of capacities (Loey et al., 2021). Deep learning enabled researchers and doctors to anticipate COVID-19 distribution by evaluating huge amounts of data. It also can help to ensure the social distancing and detection of the facemask.

Several kinds of the literature review are done to understand about:

1. Traditional Machine Learning
2. Deep Learning Overview
3. Transfer Learning
4. Techniques in coding
5. The existing method in face mask detection

2.2 Traditional Machine Learning

The traditional machine learning process requires feature extraction, which implies that the feature engineer must specify which features the machine learning should look for to distinguish the picture. The methods like Harris Corner Detection, Oriented FAST and Rotated BRIEF (ORB), etc., will be used to descript the feature in object detection. Feature extraction may also require computer vision algorithms like edge and corner detection (O'Mahony et al., 2019). When unstructured data is used, pre-processing must be done to arrange it into a structured format. The challenge with traditional machine learning is determining which features are significant in each image. Feature extraction gets increasingly difficult as the number of classes grows. It became a lengthy trial and error process to determine which

features best describe various object types (O'Mahony et al., 2019). Deep learning innovation has overcome the difficulties of traditional machine learning.

2.3 Deep Learning Overview

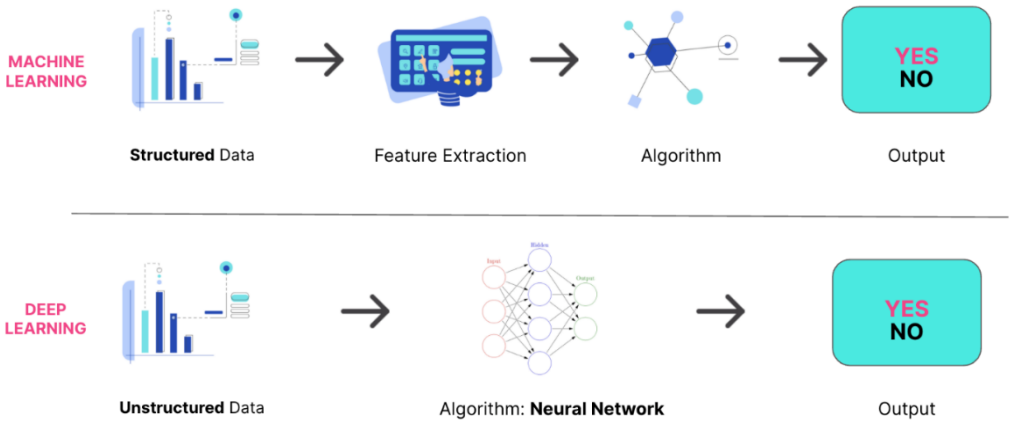


Figure 2.1: Comparison of Machine Learning and Deep Learning

Deep learning is getting more popular due to the big data era since it has a higher accuracy when trained with a massive amount of data (Sharma, Sharma and Jindal, 2021). Big data may be derived from social media, the internet, search engines, e-commerce platforms, and online movies. Often, this data is unstructured and so large that it takes humans decades to extract the features (Sivarajah et al., 2017). Figure 2.1 shows the algorithm in deep learning has eliminated some of the data pre-processing or feature extraction needed in traditional machine learning. Deep learning is capable of ingesting and processing unstructured data such as text and images. Additionally, it can automate feature extraction because it uses hierarchical neural networks like convolutional neural networks (CNNs) or recurrent neural networks (RNNs). CNNs are used to classify images, while RNNs are used to recognise natural language and speech recognition (Sharma, Sharma and Jindal, 2021). The hierarchical structure enables it to adopt a non-linear approach, processing data across a series layer and gradually extracting more complex data features (Shrestha and Mahmood, 2019).

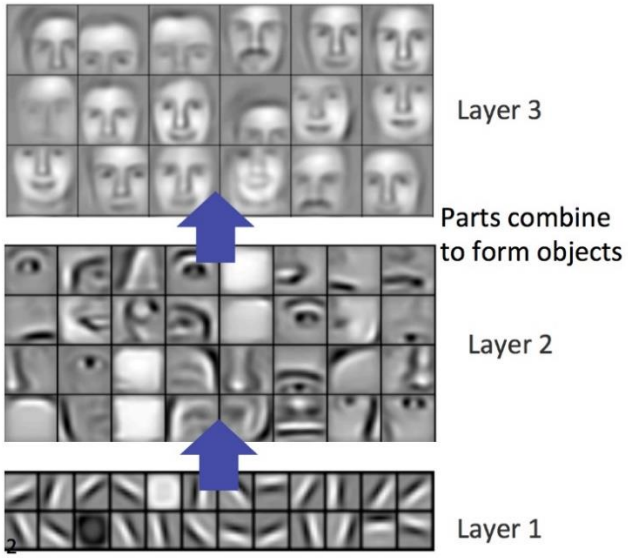


Figure 2.2: Lower-level features progressively combine to form higher-level features (Asghar et al., 2019)

In image recognition, the deep learning system recognises the pixel first, followed by the line and edge. In the next layer, it will recognise more complex shapes, such as eyes and noses, and in the deeper layer, it will learn which shapes and objects may be used to identify a human face as shown in Figure 2.2. Thus, the prediction model gets more complex and accurate with each iteration.

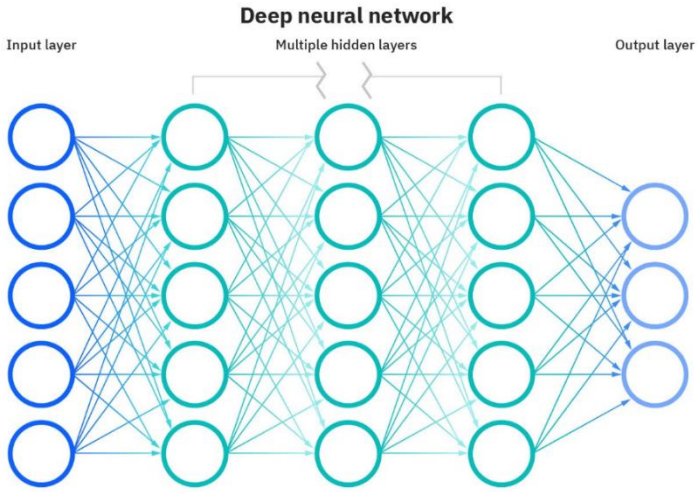


Figure 2.3: Neural Network of Deep Learning (IBM Cloud Education, 2020)

Deep learning simulates the human brain by combining data input, weights, bias or threshold, and an output; the fundamental formula is $Z = \sum_{i=1}^{m} W_i X_i + b$. X represents the non-linear activation function at each layer, W represents the weight between two adjoining unit layers, and b represents the bias, the minimal number

required to pass the threshold. Z will be the output; the output will then be predicted by summing these three inputs together. If the output is greater than the specified bias, the node will activate and transmit data to the network's next layer; otherwise, no data will be transmitted. Since neural networks contain many hidden layers, the process will be performed several times for a single decision. Every hidden layer contains its own activation function, which may be used to transfer information across layers. Once all outputs are produced from the hidden layers, the outputs are used as inputs to compute the neural network's final output (Shrestha and Mahmood, 2019; IBM Cloud Education, 2020). This computation across the network is known as forward propagation. Deep learning also includes backpropagation, which uses an algorithm like gradient descent to allow the function to move backwards through the layers to enhance the model by adjusting the weights and biases when the prediction is incorrectly computed (Shrestha and Mahmood, 2019). By forward and backpropagation, the neural network can generate predictions and fix the error, making the algorithm more accurate.

In short, deep learning trains a computer to learn like a human brain. It can learn high-level features incrementally, which has eliminated the need for feature engineers and can easily achieve incredible accuracy. However, deep learning requires a massive quantity of data and processing power to reach an acceptable degree of accuracy, but this is easily achievable in today's big data and cloud computing era. Hence, it has become the primary option among others.

2.4 Transfer Learning

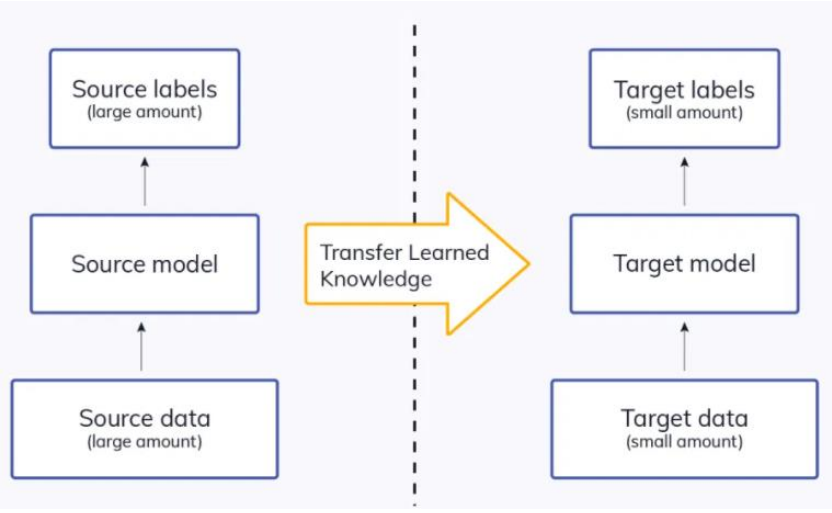


Figure 2.4: Transfer Learning idea

Deep learning is highly dependent on huge data sets compared to traditional learning methods since it needs a large amount of data to discover hidden patterns in data because the size of the model and the quantity of data have a roughly linear correlation (Tan et al., 2018). If a small data set is used to train from scratch, overfitting will occur. Therefore, transfer learning is essential to deep learning due to the scarcity of data and a large amount of time needed, as training may take days or weeks from scratch. Transfer learning is the application of a previously trained model to a new issue. In transfer learning, the knowledge learned in the previous model will be passed to the new training model, so that the new network model can start with pre-trained weights (Krishna and Kalluri, 2019). Figure 2.4 shows a set of small data sets with previously unknown classifications that can be trained via transfer learning by adjusting the internal network, and the resulting new model can recognise the new class. The pre-trained models studied in this literature review are VGG, MobileNet, and YOLO.

2.4.1 Visual Geometry Group (VGG)

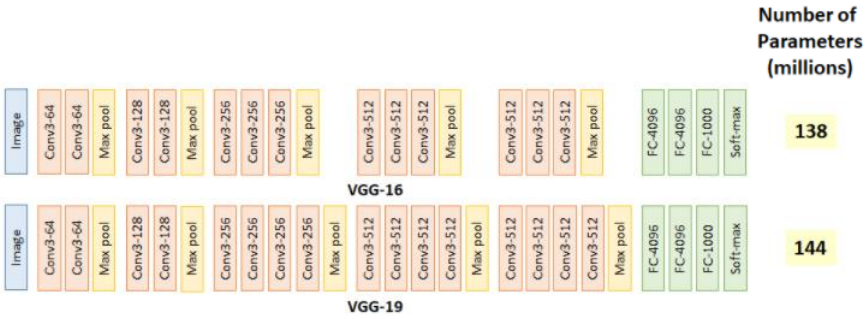


Figure 2.5: VGG-16 and VGG-19 architecture

The most common VGG networks are VGG-16 and VGG-19. The numbers 16 and 19 represent the weight layers in VGG. This network is unique in its simplicity, as it employs only 3x3 convolutional layers stacking on each other to increase the depth. Max pooling will handle the volume reduction. Then end with two 4096 nodes fully connected layers and a softmax classifier. VGG-19 has three additional convolutional layers compared with VGG-16. By using multiple 3x3 filters, it has eliminated the need for large size kernels, allowing us to extract complex features at a low cost (Krishna and Kalluri, 2019).

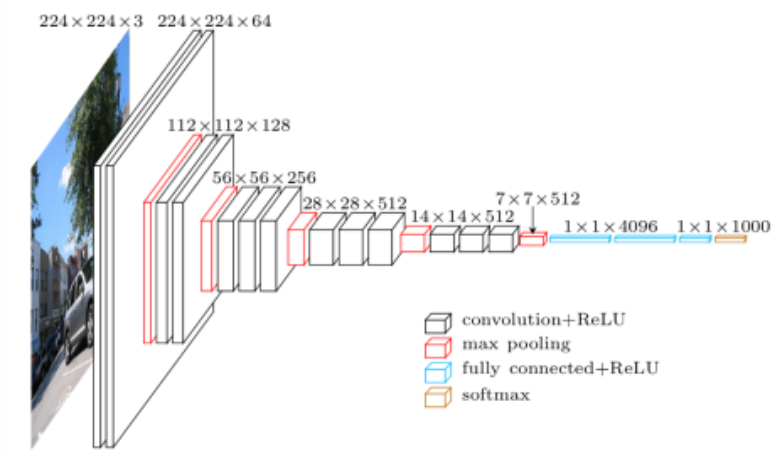


Figure 2.6: VGG-16 architecture's visualisation (Frossard, 2016)

VGG networks use convolution layers with a 3x3 filter, a stride of 1, and equal padding. The max pooling layers were 2x2 filters with a stride of 2. Figure 2.6 illustrates a 224x224 size of the image as the input with 3 filters representing RGB. The first two convolutional layers contain 64 filters, so it ends up with 224x224x64 volume. Then the max pooling layer reduces the height and width from 224 to 112, and the following convolutional layers contain 128 filters, resulting in 112x112x128 dimensions. The process will continue till we get a final 7x7x512 into fully connected layers with a softmax output of 1000 classes.

Table 2.1: Comparison of VGG-16 and VGG-19 (Tsai, 2019)

Model	Memory Size	Accuracy	
		Top-1	Top-5
VGG-16	528MB	0.72	0.910
VGG-19	549MB	0.759	0.929

VGG-19 performs slightly better than VGG-16, but it requires more memory size (Shu, 2019). So, many users prefer VGG-16 as it performs almost as well as VGG-19. However, the VGG network has the disadvantage of being very slow to train and having high architectural weights in terms of disc or bandwidth. VGG requires more than 500MB of memory size due to its depth and amount of fully connected nodes. Therefore, deploying VGG takes a long time, which is why smaller network architectures are typically chosen (Rosebrock, 2017).

2.4.2 MobileNet

A MobileNet model is intended for usage in mobile applications and devices with limited computing capacity. The ability to operate deep networks on mobile devices enhances user experience by providing anytime, everywhere access, as well as extra security, privacy, and energy conservation. So, this neural network developed is entirely lite, as mobile devices cannot afford a large GPU to operate in the background due to space and restriction. Furthermore, MobileNet employs depthwise separable convolutions, including depthwise and pointwise convolutions. Compared to alternative architectures with the same depth in the network, this has substantially reduced the number of parameters. So, MobileNet is a lightweight network (Sandler et al., 2019). This literature study will go over MoblieNetV2 in more detail.

Depthwise separable Convolution:

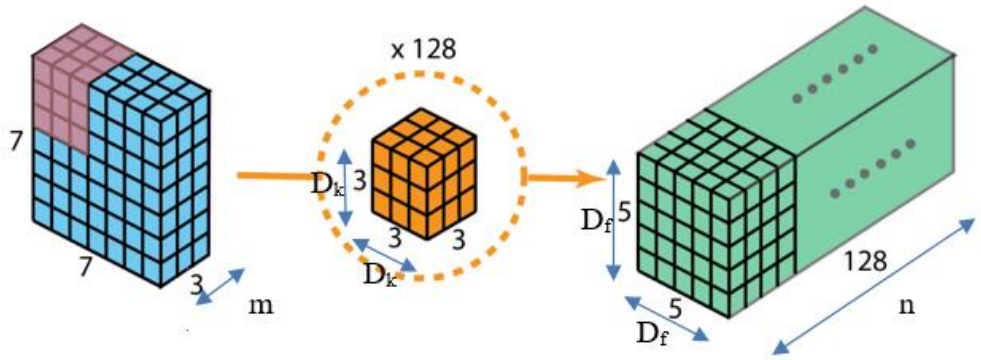


Figure 2.7: Standard convolution

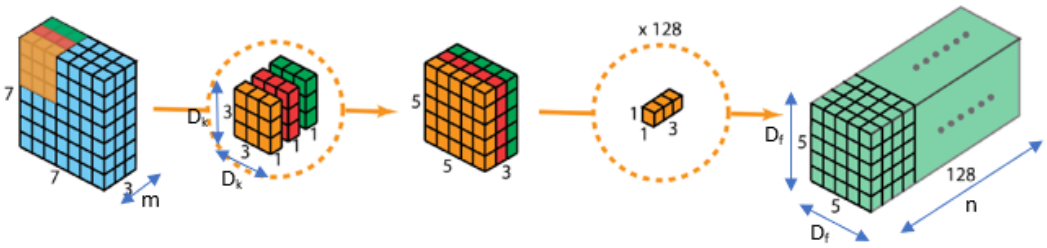


Figure 2.8: Depthwise separable convolution

Depthwise separable convolution is more efficient than standard convolution as it divides the kernels into smaller parts. From Figure 2.7, 128 of 3x3x3 kernels are used, and each move 5x5 times. So, standard convolution has a total of

$128x3x3x3x5x5 = 86{,}400\ multiplications$. In Figure 2.8, the first kernel has 3 of 3x3x1 size that moves 5x5 times. And the second kernel has 128 of 1x1x3 size that moves 5x5 times. The total is $3x3x3x1x5x5 + 128x1x1x3x5x5 = 10275\ multiplication$. So, it only has 10275 multiplication which only costs 11.89% of the standard convolution. The simplified formula for standard convolution is ${D_k}^2 mn{D_f}^2$. In contrast, the formula for Depthwise separable convolution is ${D_k}^2 m{D_f}^2 + mn{D_f}^2$ where m is the input channels' number, n is the output channels' number, D_k is the size of the kernel, and D_f is the size of the feature map. Therefore, with Depthwise separable convolution, it can deduct standard convolution costs by $\frac{1}{n} + \frac{1}{{D_k}^2}$ (Howard et al., 2017).

MoblieNetV2:

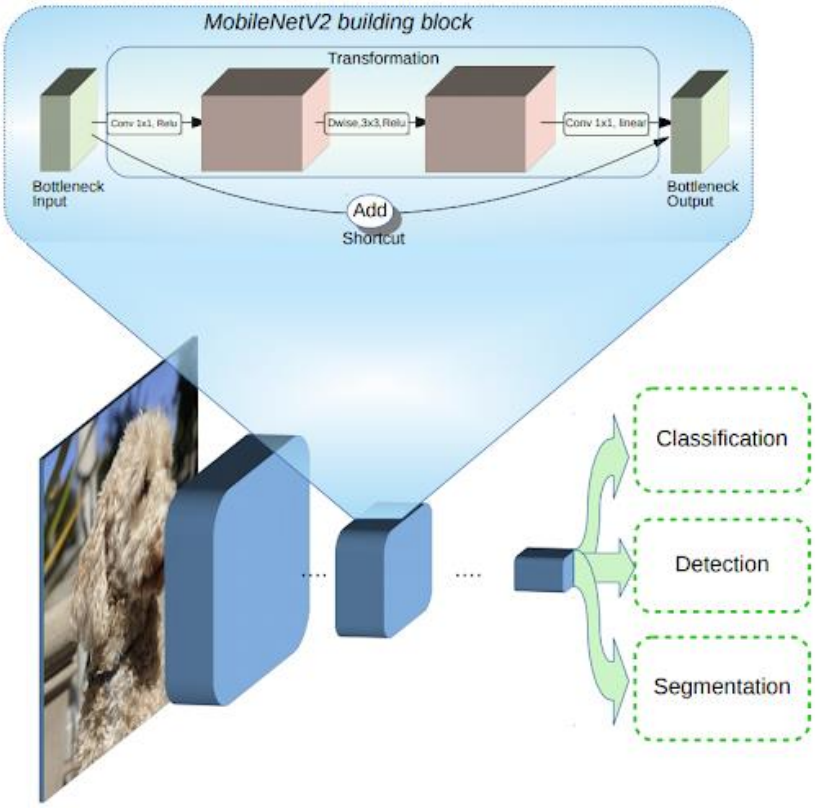
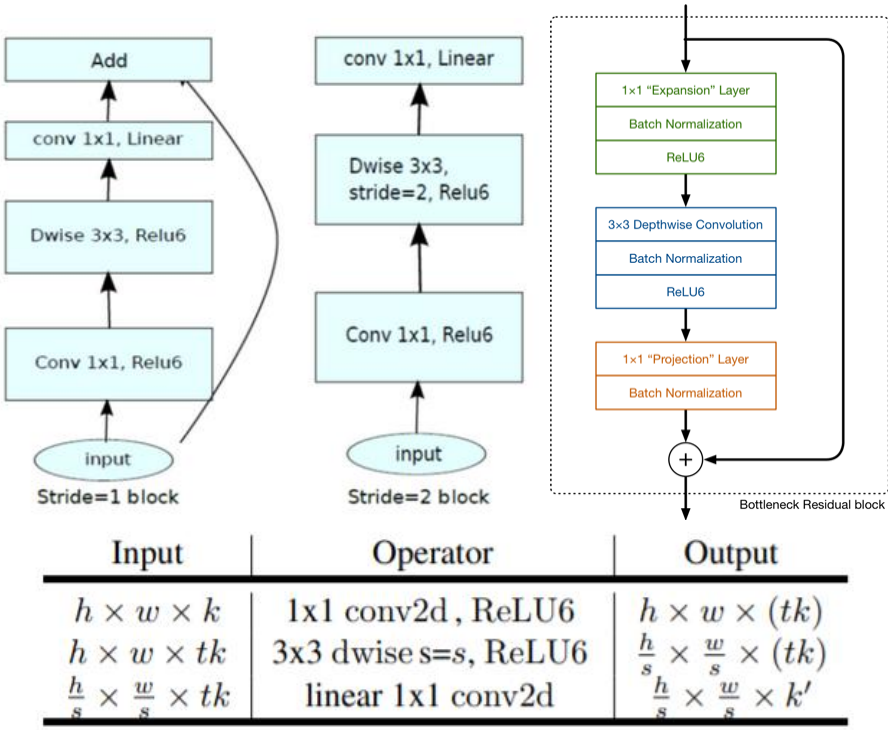


Figure 2.9: Overview of MobileNetV2 architecture (Sinha, 2018)



Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Figure 2.10: MobileNetV2 Bottleneck residual block (Sandler et al., 2019)

MobileNetV2 uses an inverted residual structure with depth separable convolution. It has an architecture that starts with a full convolution layer of 32 filters and 19 residual bottleneck layers. Figure 2.10 shows there are two types of bottleneck residual block, which are stride 1 and stride 2. Stride 2 is used for downsizing the input size, and it does not have a residual connection. There are 3 layers in these blocks. The first layer has 1x1 convolution with ReLU6, the second layer contains depthwise convolution, and the last layer is linear 1x1 convolution. ReLU6 was chosen as a non-linearity because of its robustness in low-precision computation. Every layer contains a batch normalisation layer and an activation function, except the final layer, which does not include ReLU6 since the output form in this layer has low dimension and non-linearity would reduce its performance (Sandler et al., 2019).

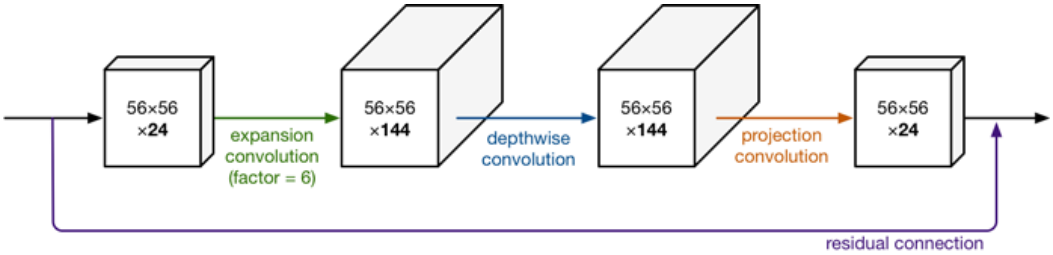


Figure 2.11: Bottleneck residual block visualisation

In MobileNetV2, the expansion factor(t) of 6 is used. If the input tensor is 24 channels, the expansion layer, the first layer in the residual block, will convert it into a new tensor, 144 channels. Then, depthwise convolution will apply its filter to that 144-channel tensor, and finally, the projection layer will project the filter back to 24 channels. These layers are known as bottleneck layers because it lowers the amount of data flowing across the network. The residual connection in MobileNetV2 is only used when the number of channels entering the block equals the number of channels coming out of it, which is the stride of 1. The stride of 2 will not pass through the block residual connection.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Figure 2.12: MobileNetV2 architecture (Sandler et al., 2019)

Figure 2.12 shows the complete architecture of MobileNetV2, where t is the channel expansion rate, c is the number of output channels, n is block repetition time, and s is the stride. MobileNetV2 begins with a standard 3x3 convolution with 32 channels, followed by 17 bottleneck blocks. It ends with a regular 1x1 convolution. Before classification, a global average pooling layer is used, then only followed by the classification layer. The MobileNetV2 achieved a top-5 accuracy of 0.901 when training on 2.5 million parameters. Due to Depthwise separable Convolution, MobileNet requires less computation and parameters, allowing it to perform better in size, latency, and accuracy (Sandler et al., 2019).

2.4.3 ResNet

ResNet is a robust deep neural network that achieved 1st price in the ILSVRC 2015 classification competition with 3.57%. of top-5 error rate. ResNet architectures exist many variants, each with the same basic idea but a different number of layers. The

most popular architectures are ResNet-34, ResNet-50, ResNet-101 and ResNet-152. The digit indicates the number of neural network layers of the Resnet.

In a neural network, several additional layers are often placed in Deep Neural Networks when dealing with complicated problems to increase accuracy and performance. This is because adding additional layers allow that layer gradually learn more complex features. However, He et al. (2016) found that the conventional CNN model has a depth threshold limit.

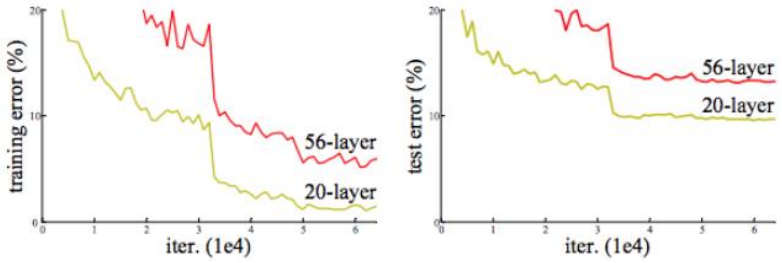


Figure 2.13: Training (left) and test (right) error on CIFAR-10 (He et al., 2016)

From Figure 2.13, He et al. (2016) show that the deeper network will generate greater training and test error. As a result, putting extra layers on top of a network reduces its performance. Because the massive layer in networks is prone to data overfitting. Deep networks are extremely hard to train because of the problem of vanishing gradient, which claims that when a gradient is backpropagated to previous layers, repeated multiplication may cause the gradient to become endlessly tiny. Therefore, its performance degrades as the network becomes deeper. This difficulty has been solved by the invention of ResNet, or residual networks, which are made up of Residual Blocks.

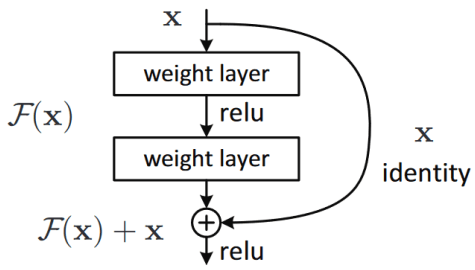


Figure 2.14: Residual building block

In the residual block, there is a direct connection that bypasses some layers in between. This is known as a ‘skip connection’, or identity mapping. There are no

parameters in the skip connection, so the output from the previous layer is just added to the next layer. The layer's output is no longer identical as a result of this skip connection. In the absence of a skip connection, the input 'x' is multiplied by the layer weights and added with a bias $y = f(x, \{w_i\} + b)$ *or* $y = f(x, \{w_i\})$. When skip connection is used, there are two equations:

1st Equation:

$$y = f(x, \{w_i\}) + x \tag{2.1}$$

2nd Equation:

$$y = f(x, \{w_i\}) + w_s x \tag{2.2}$$

where

y = output of data

x = input of data

f (x, {w_i}) = residual mapping to learn

W_s = linear projection

Normally, when a skip connection is employed, the 1st Equation is used. However, x and F(x) do not always have the same dimension. When the dimensions of the input vary from those of the output, this approach will have some issues. Therefore, there are two approaches to address the issue:

1. To enhance the dimensions, the skip connection is padded with additional zero padding. (1st Equation is used)

2. To match the dimension, the projection approach is used, which is achieved by adding 1x1 convolutional layers to the input (2nd Equation is used)

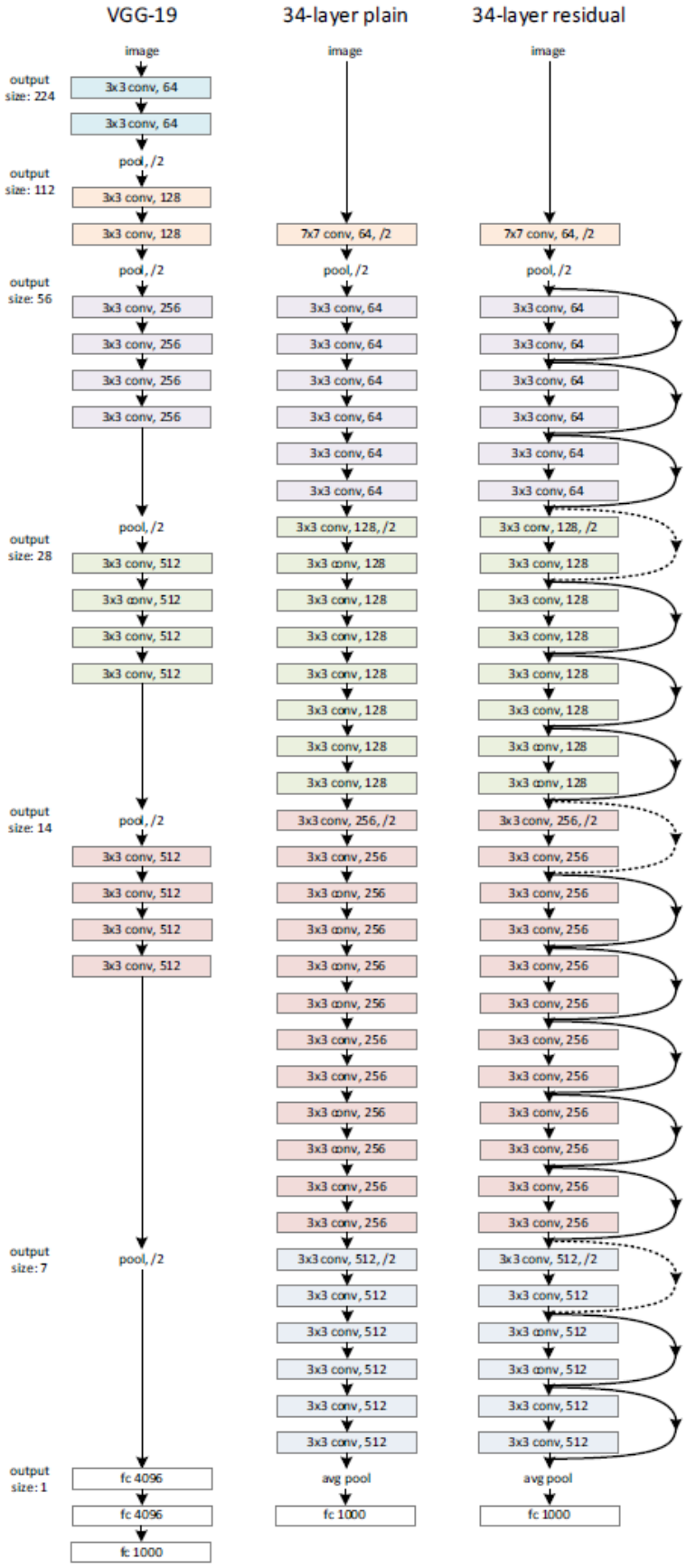


Figure 2.15: ResNet-34 architecture (Right)

Figure 2.15 shows the ResNet with 34 layers, ResNet baselines influenced mostly by VGG networks. To match the dimensions of the inputs, the dotted skip connections in the picture above reflect multiplying the identity mapping by 2nd Equation (He et al.,2016).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.16: ResNet architectures

Figure 2.16 shows the architecture 50-layer above is modified due to concerns about training time, and the building block is redesigned as a bottleneck design for the 50 layers above. Instead of using two levels for each residual function, a three-layer stack is implemented. The three layers are 1 x 1 convolutions, 3 x 3 convolutions, and 1 x 1 convolutions. The first and last 1 x1 layers have the responsibility for lowering and subsequently raising (restoring) dimensions, leaving the 3 x 3 layer as a bottleneck with reduced input or output dimensions. The 34-layer network is modified by switching from a two-layer bottleneck block to a three-layer bottleneck block, resulting in a ResNet of 50 layers. This approach results in more efficient models.

method	top-1 err.	top-5 err.
VGG [41] (ILSVRC'14)	-	8.43†
GoogLeNet [44] (ILSVRC'14)	-	7.89
VGG [41] (v5)	24.4	7.1
PReLU-net [13]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Figure 2.17: Top-5 err (%) on ImageNet validation set

Although the FLOPs and depth rise in all 50/101/152-layer ResNets, they are more accurate than the 34 layers and do not suffer from deterioration as the error rates decrease when going deeper (He et al., 2016).

2.4.4 Inception

The Inception network was complex because it used several methods to enhance speed and accuracy. Its ongoing development resulted in the formation of many network versions such as v1, v2, v3, and so on. The Inception network has implemented a method to employ varied filter sizes within its convolutional layers.

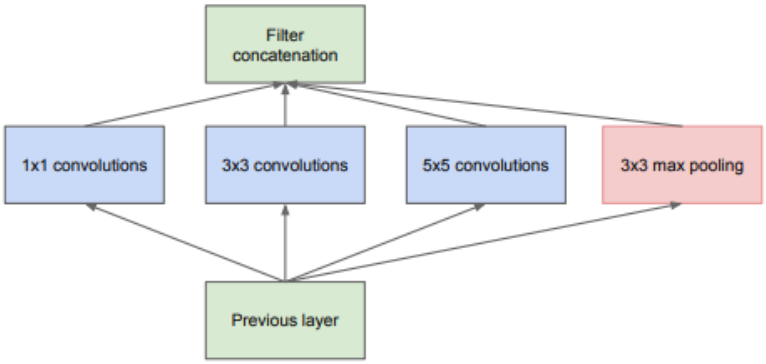


Figure 2.18: Naive version of inception module (Szegedy et al., 2015)

Before the emergence of Inception, researchers had to figure out which filter sizes to use in deep convolutional neural networks to get the best results. Inception eliminates the necessity for such selections by using several filter sizes 1x1, 3x3, and 5x5 together. 1x1 convolutions will minimise the dimensions of data travelling through the network. It is also able to increase the network's breadth and depth and learn patterns throughout the depth of the input. Furthermore, the use of 3x3 and 5x5 convolutions allows the network to learn a variety of spatial patterns at different scales. In the naive inception model, it performs convolution on an input, using these 3 kinds of filters and a max-pooling layer. Before the costly filter sizes of convolutions, the 1x1 was employed to compute reduction (Szegedy et al., 2015). The outputs are then concatenated and passed to the next inception module. The pooling layer downsamples the input data by producing a smaller output with a lower height and width. Padding will be added to the pooling layer to ensure that the pooling layer's output can be concatenated with the output of the convolution layers.

Inception-v3 is reviewed in this literature study. It has a 42-layer deep learning network and a low error rate, making it the first runner-up in ILSVRC 2015 (Szegedy et al., 2016).

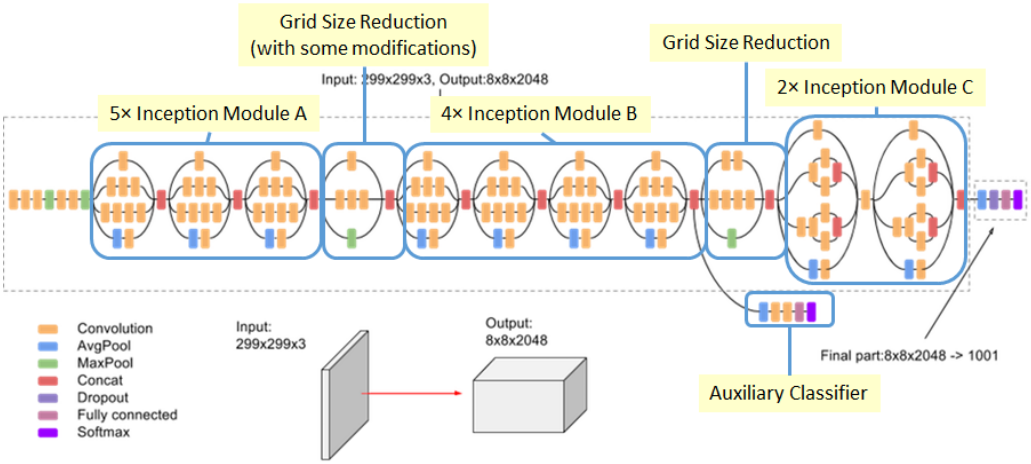


Figure 2.19: Inception-v3 architecture (Tsang, 2018)

Few improvements have been done in Inception-v3:

1. Convolution factorisation

   The goal of convolution factorisation is to decrease the number of connections and parameters while maintaining network efficiency. There are 3 types of factorisation techniques (Szegedy et al., 2016):

   a. Factorisation into smaller convolutions

      This method is accomplished by substituting two 3x3 convolutions for the 5x5 convolution. This strategy is able to reduce the number of parameters by 28%. Because when using the 5x5 convolution, the number of parameters is $5\ x\ 5 = 25\ parameters$. When using two 3x3 convolutions, the parameters will be $(3\ x\ 3)\ x\ 2 = 18\ parameters$.

      So, the modules will be updated to the following:

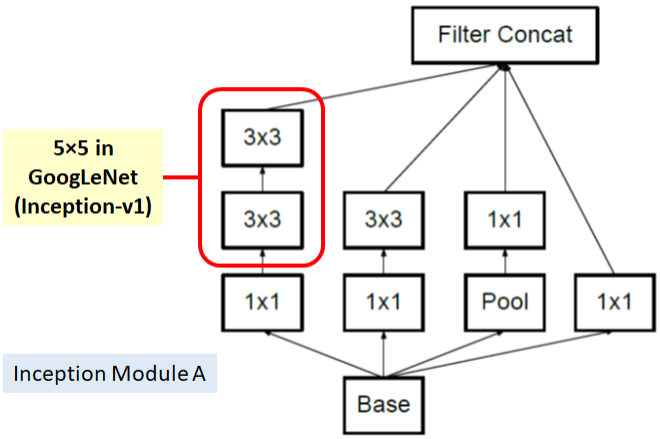


Figure 2.20: Updated Module by replacing 5x5 convolution (Tsang, 2018)

b. Factorisation into asymmetric convolutions

This method is done by replacing n x n convolution with n x 1 convolution and 1 x n convolution. With this method, the number of parameters is reduced by 33%. Assume using a 3x3 convolution, the number of parameters is $3\ x\ 3 = 9\ parameters$. When using the 1 x 3 and 3 x 1 convolution, the number of parameters will be $(1 \times 3) + (3 \times 1) = 6\ parameters$.

So, the modules will be updated to the following:

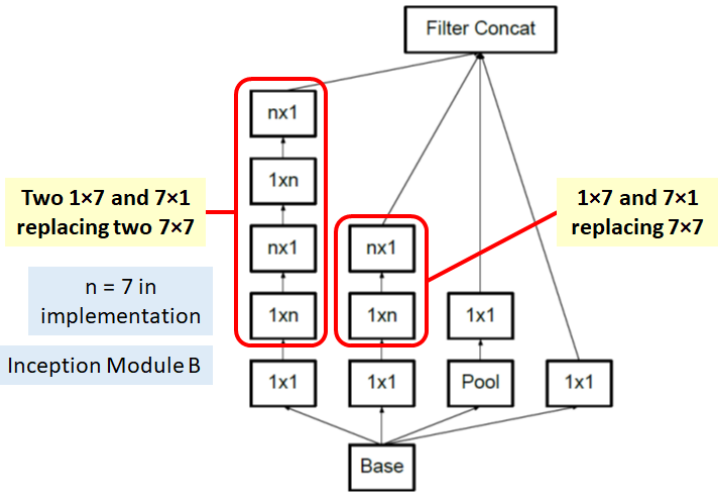


Figure 2.21: Updated Module by replacing 7x7 convolution (Tsang, 2018)

c. Expend filter bank

To eliminate the representational bottleneck, the filter banks in the module were expanded. 1x1 convolution is used to make the module wider rather than deeper to encourage the high-dimensional

representations. As the dimensions of the module would be dramatically decreased if it were made deeper, resulting in information loss.

So, the modules will be updated to the following:

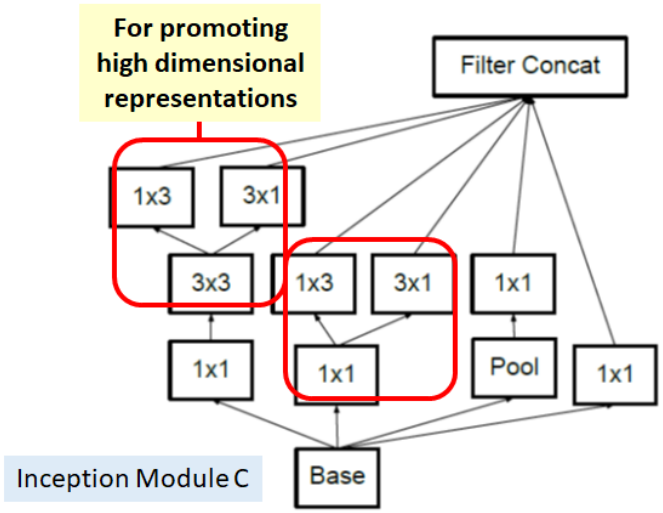


Figure 2.22: Updated Module by expending filter bank (Tsang, 2018)

2. Auxiliary Classifier

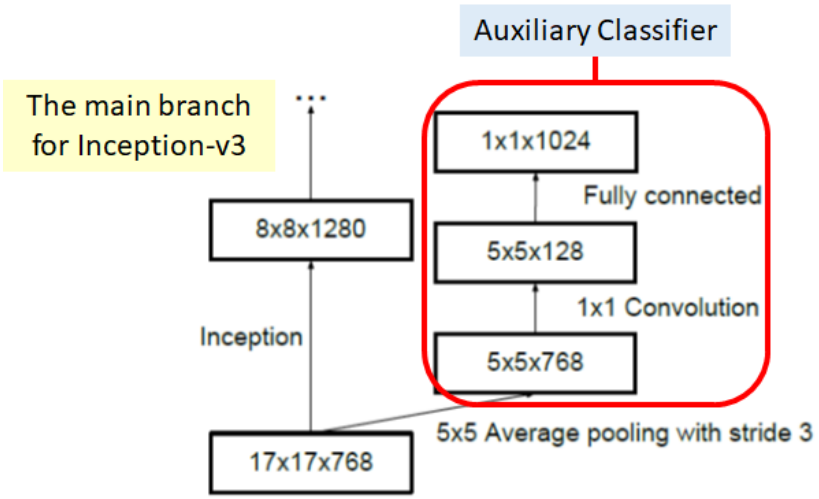


Figure 2.23: Auxiliary Classifier (Tsang, 2018)

Auxiliary classifiers are used in Inception-v1 to have a deeper network. However, in Inception-v3, the authors observed that auxiliary classifiers did not make a significant contribution until the completion of the training. Therefore, the Auxiliary Classifier serves as the regularisers in Inception-v3 and there is only one auxiliary classifier used in the model, which is placed on top of the final 17x17 layer (Szegedy et al., 2016).

3. Efficient Grid Size Reduction

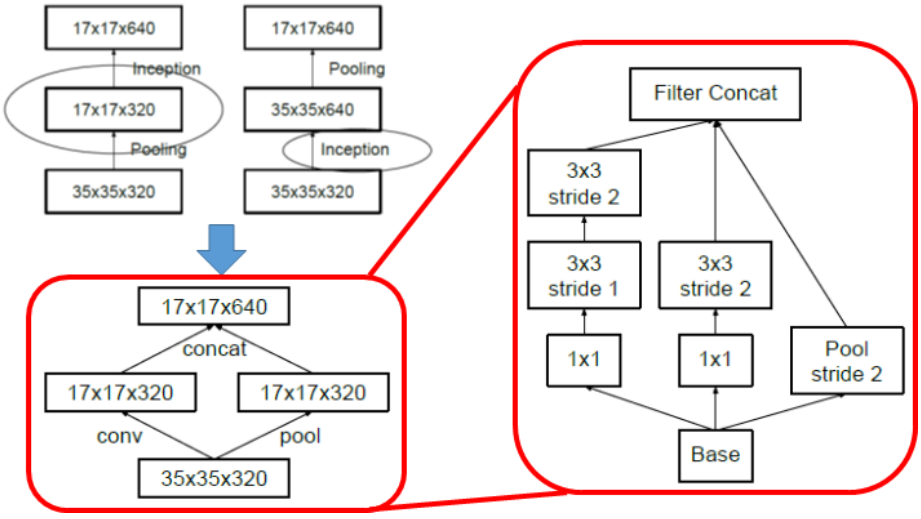


Figure 2.24: Efficient Grid Size Reduction (Tsang, 2018)

Traditionally, feature map reduction in networks has been performed through the use of max pooling. However, it is extremely greedy if max pooling is followed by convolution, and excessively expensive if convolution is followed by max pooling. As a result, a method for efficiently reducing the grid size is proposed. With this approach, convolution with stride 2 generates 320 feature maps, while max pooling produces another 320. Next, both feature maps are concatenated to create 640 feature maps. All generated feature maps will then forward to the following module level (Szegedy et al., 2016).

4. Label smoothing

   A regularising component is introduced to the loss formula to prevent the network from getting overconfident toward a class, which prevents overfitting (Szegedy et al., 2016).

Due to the design of the Inception network, which enables the use of variable convolutional filter sizes, the Inception network is capable of extracting features from input data at varied scales. This enables the inception network to attain excellent performance while maintaining a low computation cost.

2.4.5 Xception

Xception stands for extreme inception, which is a more advanced version of Inception. The Xception architecture uses the same amount of parameters as Inception-v3 but performs better due to more efficient usage of model parameters (Chollet, 2017).

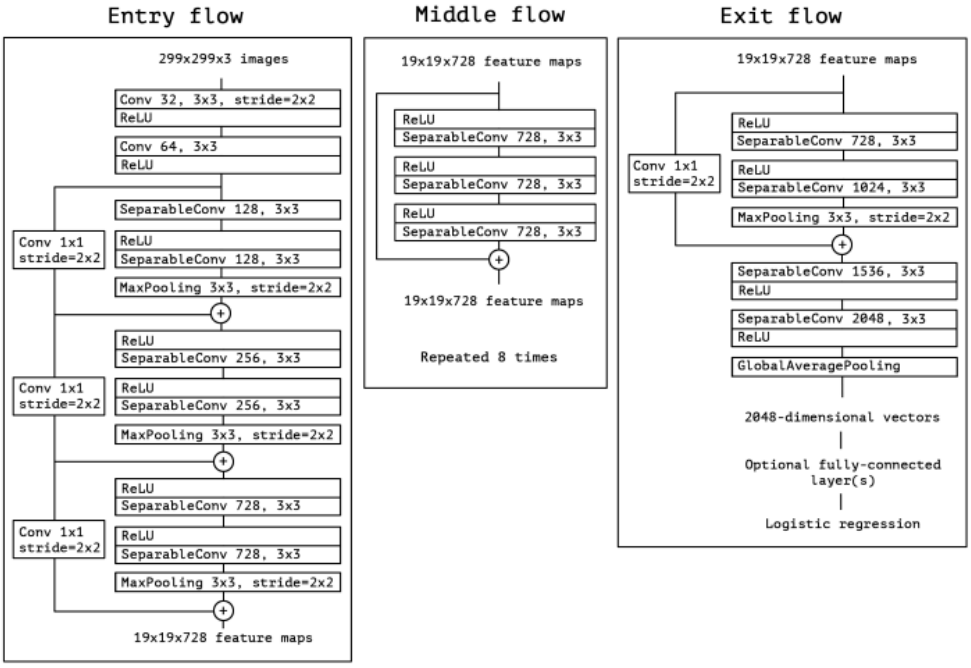


Figure 2.25: Xception architecture (Chollet, 2017)

In Xception, the data will move through three flows: first, the 'input flow,' second, the 'middle flow,' and it will be repeated eight times, and at the end, the 'exit flow'. In Xception, batch normalisation is performed in all Convolution and Separable Convolution layers. Then, a depth multiplier of one is used by all Separable Convolution layers. Xception outperformed Inception-v3 due to two significant changes: updated depthwise separable convolutions and a modification of the non-linearity (Chollet, 2017).

1. Modified depthwise separable convolutions

   Normally, depthwise separable convolution is performed first, followed by pointwise convolution. The phrase depthwise convolution refers to the n x n spatial convolution channel-wise. If there are three channels, we will have three spatial convolutions of size n x n x 1. Then, the 1 x 1 convolution that is used to modify the dimension is called pointwise convolution. The Xception

model has updated the depthwise separable convolutions by reordering the depthwise and pointwise convolutions, which implies that pointwise convolutions will be performed first, followed by depthwise convolutions. This modification is motivated by the fact that Inception-v3's module conducts 1 x 1 convolution before doing any n x n spatial convolutions. However, Chollet mentioned that this is irrelevant for performance improvement, since when it is employed in a stacked configuration, only minor differences occur at the beginning and end of Inception modules (Chollet, 2017).

2. Non- linearity

   In Inception, ReLU non-linearity is used in the operations. However, depthwise separable convolutions are often done without non-linearity.

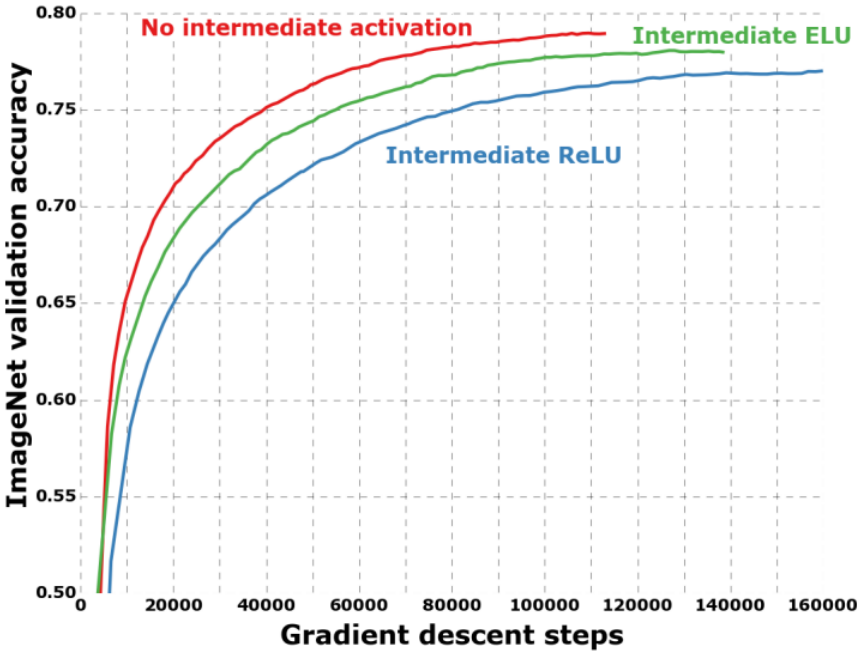


Figure 2.26: Performance of Xception with non-linearity

As seen in the above figure, the Xception with no intermediate activation has the greatest accuracy when compared to those that use ELU or ReLU (Chollet, 2017).

Table 2.2: Comparison summary for the 5 deep learning model

	Version Used	Advantages	Disadvantages
VGG (Simonyan and Zisserman, 2014)	VGG-16	- Network simplicity - Eliminated the need for large size kernel	- Slow to train - High architectural weights (549MB for VGG-16)
MobileNet (Sandler et al., 2019)	MobileNetV2	- Lightweight (Depthwise separable Convolution) - Small storage required - High speed - Suitable in mobile	- Depends on optimization strategy - Tuning problem
ResNet (He et al., 2016)	ResNet-50, ResNet-152	- Allows you to build a deeper network (skip connection) - Tackling the vanishing gradient problem (skip connection)	- deeper network usually requires weeks for training - making it practically infeasible in real-world applications.
Inception (Szegedy et al., 2016)	Inception-v3	- Eliminated the need of selecting filter size (employ various convolutional filter parallelly) - Improve performance while maintaining low computation cost - Convolution factorisation	- require a lot of memory when performing computation due to the width of the convolution layer
Xception (Chollet, 2017)	Xception	- More efficient use of model parameters (modified depthwise separable convolution and non-linearity)	- require a lot of memory when performing computation due to the width of the convolution layer

The model chosen in this research is shown in the table above. The model's unique characteristics have resulted in the model's advantages, which is why the model was chosen in this research.

2.5 Techniques in coding

2.5.1 Data augmentation

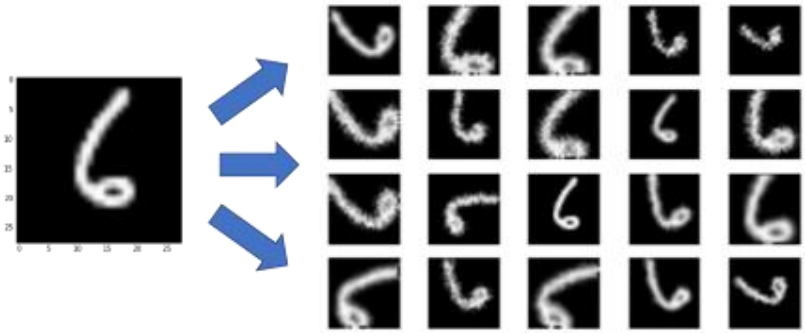


Figure 2.27: Before and after data augmentation

A large quantity of data sets is required during training; to address the insufficient amount of data, the data augmentation method can be employed. It may create several variations of the same picture by rotating, flipping, zooming, shearing, and shifting (Nagrath et al., 2021). This exposes the model to various aspects of the training data, reducing overfitting (Jiang et al., 2021).

2.5.2 Transfer learning

1. **Create a base model**

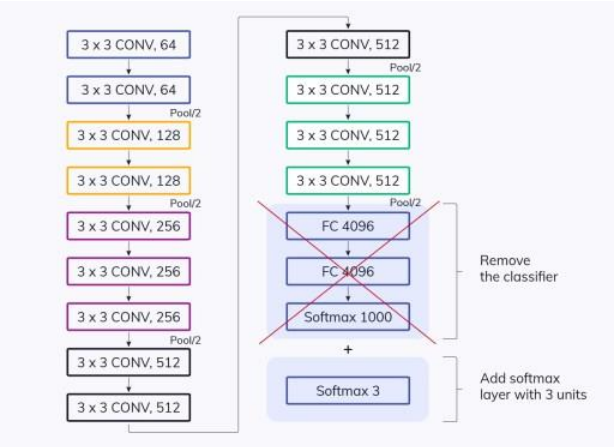


Figure 2.28: Base model

The initial stage in transfer learning is to build a base model using CNN architectures, and weight must be assigned in the model (e.g. weights= "imagenet"). If no weight is given, the model will be trained from scratch using that architecture. Figure 2.28 shows that when creating a base model, the final output layer must be removed by "include_top=False",

and a new output layer suitable for the new problems must be created and attached to the base model.

2. Feature extraction (freeze layers) and add new trainable layers

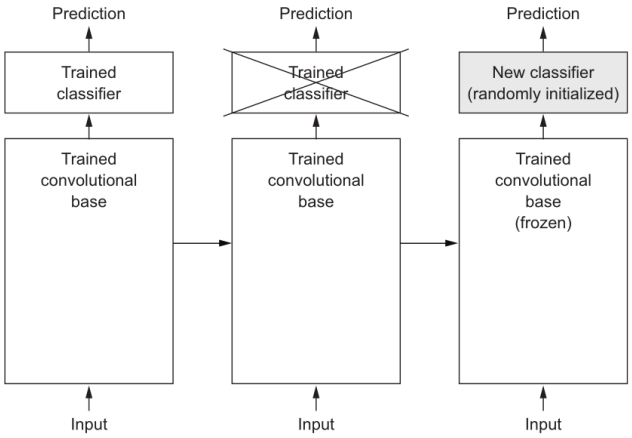


Figure 2.29: Feature extraction

The layers from the pre-trained model must be frozen by setting base_model.trainable = False. This is done to prevent reinitialising the weights in those layers. If the weight is lost, the model will be trained from scratch. So, freezing the layers enables the new data set to pass through the previously trained convolutional base and the new classifier needs to be added on top for training, so the new prediction can be made. The stacked classifier may be a stack of fully connected layers or a single global pooling layer. Both classifiers are followed by a dense layer with softmax. Using a global pooling layer is recommended to reduce overfitting as there are no parameters to tune in this layer (Lin, Chen and Yan, 2013).

2.5.3 Optimisation method to improve deep learning performance

1. Fine-tuning the learning rate

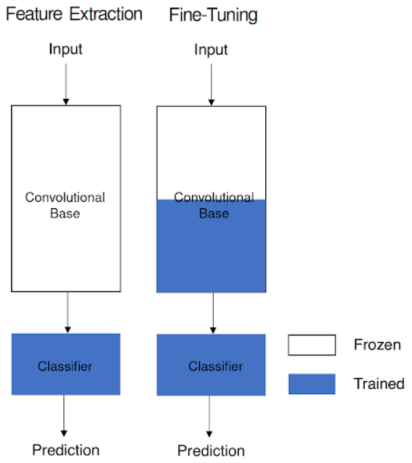


Figure 2.30: Frozen type in the base model

To improve the model performance, we can fine-tune the model. It is done by unfreezing the whole base model or portion of it, and it will be training the whole model again on the entire data set. The code is base_model.trainable = True or by specific layers which is layer.name.startswith('name').trainable = True. The learning rate assigned must be low, as it will enhance the model's performance and prevent overfitting as a low learning rate ensures the weight of the pre-trained model will not distort. A high learning rate will cause the risk of losing prior knowledge to increase. To prevent the consumption of time, a callback can be used to monitor the training loss. If 5 consecutive epochs do not improve the model, Keras will terminate the training.

2. Batches

The batch size determines the number of samples that will be sent to the network at once. If a total of 100 images are used with a batch size of 10, a complete epoch needs to iterate 10 times. The equation is $batches\ in\ epoch = \frac{size\ of\ training\ image}{batch\ size}$. The bigger the batch size, the faster the model completes each epoch. The model's quality may degrade as a larger batch size is used, and larger batch sizes require massive computing resources to process all images in parallel. So, it is necessary to fine-tune the batch size during training. Gradient descent is an optimisation method used to determine the weights of deep learning. It enables the model to make

predictions on training data and utilise the prediction error to update the model to enhance performance. There are three learning algorithms in batches:

i. Batch Gradient Descent (BGD)

batch size = size of image in training set OR batch size = per epoch

This algorithm computes the error for each training data set or per epoch. It only updates the model after evaluating all training data. It is more computationally efficient since it has fewer updates. The reduced update frequency produces a more stable error gradient and convergence. On the other hand, the stable error gradient may lead the model to prematurely cover a set of less ideal parameters. It also needs the whole training data set to be kept in memory so that it will become slow with a large data set.

ii. Stochastic Gradient Descent (SGD)

batch size =1

This algorithm provides immediate feedback on model performance and improvement rate. It is also simple to apply and comprehend. A higher update frequency may result in faster learning on certain problems and prevent premature convergence. However, frequent updates require high computation capacity and a longer time to train the model. It may also cause higher variance over training epochs.

iii. Mini-Batch Gradient Descent

1 < batch size < size of image in training set

Mini-Batch Gradient descent is the most used algorithm in deep learning. The most popular batch sizes are 32, 64, and 128. This algorithm allows the model to update more often, resulting in more robust convergence and avoiding premature convergence. It is more computationally efficient than SGD and does not need all training data stored in memory like BGD.

3. **Epochs**

Epoch is a hyperparameter that has a connection to batches size. It determines the number of times the training algorithm iterates through the whole provided training data set. When a single epoch is used, every data in the data set will have a chance to update the parameters of the internal model. As mentioned in "**b. Batches**", an epoch may contain one or more batches. Often, the epochs used are large, like 100 or 1000 times, allowing the training algorithm to run until the model's error is sufficiently minimised. There are no standard epochs. With every iteration, the loss will continue to decrease. The validation loss will first reduce, but it will increase when the model starts overfitting. Therefore, the lowest validation loss is an ideal endpoint.

2.6 The existing method in face mask detection

This study investigates a total of 20 works of literature. There are several methods for identifying the presence or absence of a face mask in deep learning in these existing models. For example, training a CNN, using transfer learning of the CNN model like VGG, Inception, ResNet, MobileNet or using an object detection model which has a CNN backbone. In an object detection model, there are two types of frameworks: region proposal based, which is a two-stage object detector like RCNN, Faster RCNN, and so on, and another framework is regression or classification based, which is a one-stage detector like YOLO, SSD, and so on. One-stage detector regresses the bounding boxes in a single step. However, the two-stage detector will produce region proposals first. Then the proposals will be subsequently fine-tuned in the second stage. Therefore, a two-stage detector has a better performance, but a slower speed than a one-stage detector. The most common model employed in these 20 papers is a one-stage detector, followed by the CNN model, a two-stage detector

model, and training a new CNN model. Several problems have been resolved, including face with a mask, face without a mask, improper mask wear, detect occlusion on the face, type of mask, and social distancing. Mostly only solved faces with a mask and without a mask. Different evaluation metrics are used to evaluate performance like testing or training accuracy, F1 score, average precision, and frame per second (FPS). This review will cover techniques used, problem solved, best tuned hyperparameter, strength and limitation, performance, and future work. Table 2.2 will show the overview of these 20 models.

The majority of these 20 existing methods are mostly focused on the classifications of no mask wearing, single mask wearing, and inappropriate mask wearing. In May 2021, Tan Sri Dr Noor Hisham Abdullah, the Director General of Health Malaysia, encouraged Malaysians to use double face masks because they may limit COVID-19 transmissions by up to 96.4 percent (Rashvinjeet and Jo, 2021). Since wearing a double masks is important in Malaysia, I decided to try a new approach in this research which is detecting double masks. As a result, this will be a new challenge for this research.

Table 2.3: The overview of Reviewed Paper

No	Authors	Techniques used	Problem solved	Hyperparameter	Strength/ Limitation – [author no. mention] xxx	Performance	Future Work
1	Chowdary, et al., 2020	- InceptionV3 - Data augmentation	- With mask - without mask	- Epochs: 80, each 42 steps	- Can be used in surveillance cameras - Cannot detect improper mask wear	- Training accuracy: 99.9% - Testing accuracy: 100%	- Identify mask type - Facial recognition - Identify person with mask on
2	Loey, et al., 2021a	- Hybrid method - Feature extraction: ResNet-50 - Detection/ Classification: SVM, decision tree and ensemble (SVM highest accuracy)	- With mask - without mask	-	- [7] Cannot detect improper mask wear - [12] easy to deploy - [12] Limit the scenario to one person in each picture, with the face covering most of the image. - [12, 18] Complex training process - [16] High computational costs - [18] High computation, unsuitable for real-time processing	- Testing accuracy (SVM): 99.64%	- Use deep learning in neutrosophic categorization domain feature extraction
3	Loey, et al., 2021b	- Feature Extraction: ResNet-50 - Detection: YOLOv2 - Data augmentation	- With mask - without mask	- Epochs: 60 - Batch size: 64 - Learning rate (LR): 0.001 - Optimizer: SGDM - Epochs: 60 - Batch size: 64 - LR: 0.001 - Optimizer: Adam	- The detection will specifically detect for medical mask - Cannot detect improper mask wear - [12] Reasonable number of anchor box - [12] High speed - [12] Medium performance - [12] Low confidence for predicted results	Average Precision (AP): 61% AP: 81%	- Detect masked face in image and video-based
4	Nagrath, et al., 2021	SSDMNV2 - Backbone: Single Shot Multibox Detector (SSD) Object Detection Model and ResNet10	- With mask - without mask	- Weight: Imagenet - Epochs: 100 - Optimizer: Adam	- Lightweight - Allow less resources used in real-time detection - Can integrate in surveillance cameras and devices like Raspberry Pi - Cannot detect improper mask wear	Accuracy score: 92.64% F1 score: 93% Without data	- Face recognition, facial landmark, and facial part detection

		- Classifier: MobilenetV2 - Data augmentation			- [16] Too complex	aug: accuracy score is 87.51%	
5	Jiang, Fan and Yan, 2020	RetinaFaceMask (Single stage object detector): - Backbone: MobileNet - Neck: feature pyramid network FPN - Head: novel context attention module	- With mask - without mask - Able to detect the occlusion on face	- Input image: 840x 840 - Epochs: 250 - Batch size: 2 - Weight: Imagenet - SGD with LR: 0.001 and momentum: 0.9	- Comparable in speed and accuracy as two-stage detector - MobileNet backbone may be used in both high and low computation with accuracy trade off. - [7] Cannot detect improper mask wear - [12] Strong ability in extracting robust features - [12] Efficient - [12] More training times - [12] Post-processing is difficult and sensitive to related hyper-parameters. - [18] No suitable in low power devices due to compute and memory requirements	Face precision: 83% Mask precision: 82.3%	-
		Single stage object detector (RetinaFaceMask): - Backbone: ResNet - Neck: feature pyramid network FPN - Head: context attention module		- Input image: 640x 640 - Epochs: 250 - Batch size: 32 - Weight: Imagenet - SGD with LR: 0.001 and momentum: 0.9		Face precision: 91.9% Mask precision: 93.4%	
6	Sethi, Kathuria and Kaushik, 2021	- Two stages (TS) and single stages (SS) detector - Backbone: ResNet50 - Neck: Image complexity predictor for face detection 1. Soft images (TS) - MobileNer-SSD 2. Hard image (SS) - ResNet50 - Head: Identity predictor - Data augmentation	- Detect mask over face - Able to detect the occlusion on face - Identify person violating mask norm - Solve high computation time in two-stage detector	- SGD with LR: 0.03 - Lose function: cross-entropy loss	- Fast inference - Require less memory - Easy to deploy - Cannot detect improper mask wear - 11.07% and 6.44% higher precision and recall compared to RetinaFaceMask [5] due to optimized face detector used	Accuracy: 98.2%	- Integrate into high resolution video surveillance device and not restricted to mask detection only - Detect facial landmarks with a mask on for biometric purposes
7	Jiang, et al., 2021	Squeeze and Excitation YOLOv3 (SE-YOLOv3) - Backbone: YOLOv3 + SE	- With mask - Without mask - Incorrect mask	- Initial learning rate (ILR): 0.0005 (100 epochs) - Optimizer: Adam	- Increased accuracy with negligible additional computational cost - Able to embed in Raspberry Pi, etc.	Image size 416x416: - AP_{50}: 98.6%	- Collect more data to balance the data - Deploy SE-YOLOv3 on lightweight device

		block - Data augmentation (image and mixup augmentation)	- Able to detect the occlusion on face	- Exponential decay (attenuation of 0.9 for ILR/100 epochs) - Batch size:1 - Lose function: categorical cross-entropy		- AP_{75}: 86.3% - mAP: 71.9% - Detection time: 43.2ms	
8	Wang, Zhao and Chen, 2021	Two stages method: - Prediction: Faster RCNN with InceptionV2 - Verification: BLS	- Correct mask - Incorrect mask - Mask only converging chin - Mask cover mouth and chin	-	- Fail to detect small objects - Fail to detect face occlusion by protective clothing or mask with goggles	- F1: 94.19 %	- Apply image super-resolution to solve the lack of features
9	Ge, et al., 2017	LLE-CNNs	- Detect simple mask - Complex mask - Human body cover the mask position - Mask with eyes occluded glasses	-	- Poor performance in heavy occlusion face (4 regions are occluded) - Any type of mask will detect as masked face - [3] Detect any face mask as medical mask	- AP: 76.1%	- Predict mask type and occlusion degree
10	Rahman, et al., 2020	Deep learning architecture with 17 layers	- With mask - Without mask - Notify the appropriate authorities of a non-masked person's location	- Epochs: 100	- Able embedded in CCTV and alert authority when detecting a non-masked face. - Difficult to classify occluded face by hands - Unable to track a person in transportation - Need convert image into grayscale - Cannot detect improper mask wear - [8] only able to process 64x64 size image	- Accuracy: 98.7%	- Social distancing
11	Militante and Dionisio, 2020	RTFR - VGG-16 - Data augmentation	- With mask - Without mask - Generate alarm	- Input image: 224x 224 - Epochs: 100 - Batch size: 64 - Optimizer: Adam - LR: 0.0001	- Can send alarm and voice notice if notice a non-masked face - Able to be embedded in Raspberry Pi, etc. - [12] Easy to deploy - [12] Highly depend on preprocessing	- Accuracy: 97%	- Social distancing

					- [12] Will result in mistakes in some complicated situations as only local feature were used		
12	Zhang, et al., 2021	Context-Attention R-CNN - Backbone: VGG-16	- With mask - Without mask - Incorrect mask - Able to detect the occlusion on face	- Input size: 600x1000 - Batch size: 1 for 11 epochs - Optimizer: SGD 0.9 momentum - ILR: 0.001 (decrease 10 factors after 10 epochs)	- Strong capability in extracting the distinguish feature map - Simplicity in implementation (including training and testing) and deployment	- mAP: 84.1%	- Investigate the imbalance issue and improve attention architecture. - Investigate hyperparameter optimization as CNN-based detector sensitive to hyperparameter
13	Qin.and Li, 2020	SCRNet	- With mask - Without mask - Incorrect mask	-	- Only detects 10 images per second, doesn't meet basic video frame of 24fps - [12] good performance on low quality image - [12] complicated training process	- Accuracy: 98.7% - 0.03s	- Collect more image and video type data set - Identify wearing condition of face mask
14	Mercald and Santone, 2021	- MobileNetV2	- With mask - Without mask	- Epochs: 20	- Can operate on limited-resource devices	- Accuracy: 98%	- Improve performance through exploiting series of transfer learning
15	Hussain, et al., 2021	SSDWG (2 model – mask detection and type of mask) - 1. VGG-16 - 2. MobileNetV2	- With mask - Without mask - Incorrect mask - Type of mask (surgical and N-95)	- ILR: 0.001 (decrease 0.1 factor every 7 epochs) - Lose function: cross-entropy	- May not be effective in real time detection due to noisy data used in training	- Accuracy: 99.81% (mask detection) - 98.17% (type) - Accuracy:99.6 % (mask detection) - 97.37% (type)	- Social distancing - Detect transparent face mask
16	Yu and Zhang, 2021	- Backbone: YOLO-v4	- With mask - Without mask - Incorrect mask	- Activation function: H-swish	- Lowe training cost - Low model complexity - No consider in insufficient of light	- AP_{50}: 98.3% - AP_{75}: 98.5% - mAP: 84.7% - FPS: 54.57 - Training time: 2.834h	- Data set expansion based on standard mask wearing condition

17	Cao, et al.., 2020	MaskHunter - Backbone: YOLOv4 with backbone of CSPDarknet19 - Neck: SSPP, FPN, PAN	- With mask - Without mask	- Neck activation function: Mish	- Fast operating speed - Good performance - Able to perform in a low-light situation	- AP: 94.0% - FPS: 74	-
18	Said, Y., 2020.	Pynq-YOLO-net - MobileNet V2 + YOLO	- With mask - Without mask		- Able to embed in low power devices - Small model size and rapid processing - Low computational complexity	- Accuracy: 97%	- To implement on video surveillance system to be tested on real conditions
19	Fan, X., Jiang, M. and Yan, H., 2021.	Single shot lightweight face mask detector (SL-FMDet) - Backbone: MobileNet - Learn more discrimination features: synthesized Gaussian heatmap regression (SGHR)	- With mask - Without mask - Non-mask occlusion		- Effective on small or blur faces - Heatmap generation requires more computation.	- mAP: 93.8%	- Improper mask wearing detection - Use zero-shot learning to train the model detect improper mask wearing
20	Yadav, S., 2020.	Single shot Detector Multibox (SSD – it used VGG-16) - Backbone: MobileNet V2 - Data augmentation	- With mask - Without mask - Social distancing	- ILR: 0.0001 - Epochs: 20 - BS: 32 - Binary cross-entropy - Adam optimizer with 100 steps	- Able to embed in Raspberry PI 4 but its power consumption is too high	- mAP: 91.2% - FPS: 28.07	- Coughing and sneezing detection - Temperature screening

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 The Proposed Model Workflow

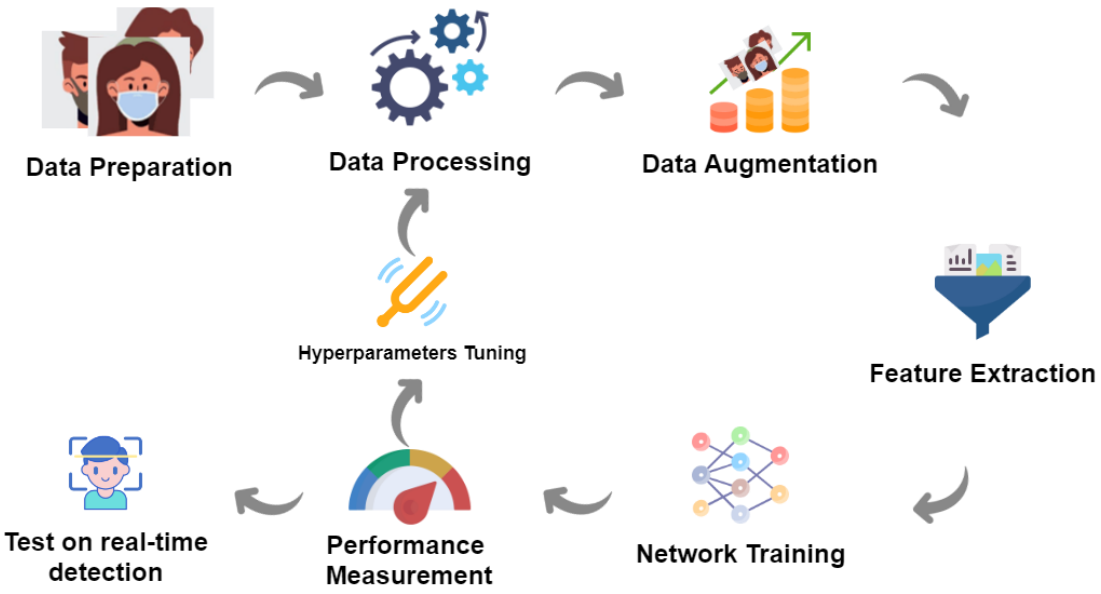


Figure 3.1: Face mask Recognition Model Workflow Summary

The architectural design of the proposed detector model is shown in Figure 3.1. The data set is labelled with three types of labels: (i) faces without a mask, (ii) faces with a single mask, (iii) incorrect mask-wearing and (iv) faces with double masks. Before feeding the data set to the neural network, some data pre-processing and data augmentation will be performed to ensure that all the data sets are labelled accurately and prevent overfitting. The model is typically developed with a pre-trained model based on convolutional neural networks (CNN) - MobileNetV2, VGG-16, ResNet-50, ResNet-152, Inception-v3 and Xception. Transfer learning is used to preserve the weight of earlier training; this technique is used due to a lack of data and has the ability to save time and computational cost. Transfer learning behaves as a feature extractor, extracting image features for use in network training. The convolutional base will be frozen for the feature extraction, and a classifier will append at the top of the pre-trained model to build a new model. The new model will then be used to fit the training data set for training. After finishing training, the model will be used to

predict the test data set. The performance of the model will be evaluated and printed out. A grid search approach is used to fine-tune the six models to identify the optimum combination of hyperparameters to tune this model. After founding the best model, the models will be evaluated using images and deployed on a webcam to test real-time detection.

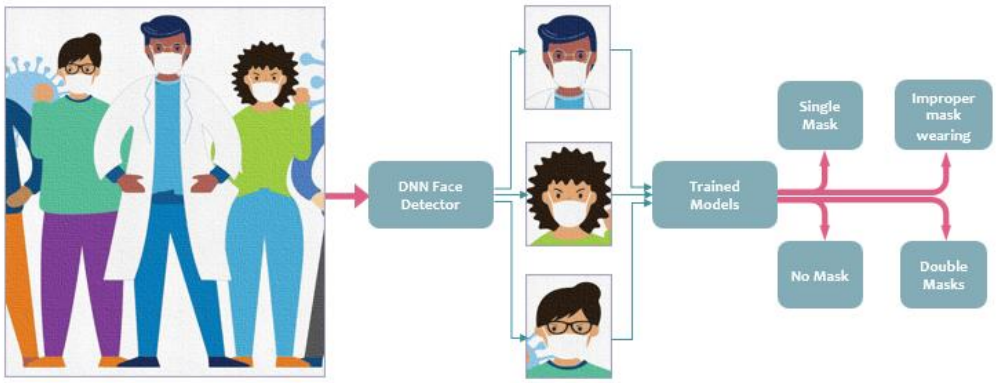


Figure 3.2: Face mask detection technique

During real-time detection, the face mask detection approach is proposed. Video frames are captured and sent to a DNN face detector to extract the faces; the extracted faces are then scaled and pre-processed according to the pre-trained model. The processed image of the faces is then sent to the trained model for prediction.

3.2 Data Preparation

3.2.1 Data Collection

The data set for training, validation, and testing was prepared by collecting four different mask-wearing images from various online resources because only one resource is not able to collect a balanced data set. 4 main resources were used to collect data sets. In addition, some data were collected from search engines, social media and artificially created. Each class contains 500 images, and 4 classes have 2,000 images. This was done to produce a balanced class across the data set, as unbalanced data will lead to bias toward the class labelled with the majority of the data.

1. Medical mask (MM) data set (https://humansintheloop.org/mask-dataset-download/?submissionGuid=add801a1-b11b-4e08-8825-3f3a1d2cce2c)

Figure 3.3: samples of data from MM

This data set contains 6024 images, with single and multiple people per image. This data set includes annotation files. Following the extraction of the faces from the photos, there are a total of 2,068 faces with no mask data, 6,715 faces with single mask data and 235 incorrect face mask wear data.

2. MAFA data set (https://www.kaggle.com/datasets/rahulmangalampalli/mafa-data)

Figure 3.4: Samples of data from MAFA

This data set contains 30,811 images. These data sets are divided into three categories: faces without masks, faces with single masks, and incorrect face mask wear. To balance the data set, some incorrect mask data is extracted here.

3. MaskedFace-Net (https://github.com/cabani/MaskedFace-Net)

Figure 3.5: Incorrectly masked face

This data set is generated based on the FFHQ (https://github.com/NVlabs/ffhq-dataset). The data set is artificially created as there is currently no publicly accessible large data set of masked face images that can be used to determine whether a person has been properly wearing a mask (Cabani et al., 2021).

4. FMD data set (https://www.kaggle.com/andrewmvd/face-mask-detection)

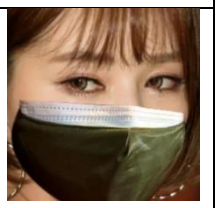
Figure 3.6: FMD sample1

Figure 3.7: FMD sample2

This data set contains 853 images with 3 labels. It comes with an annotations file that draws a coordinate box to determine the person's face and with a label. To use this data set, the image needs to be cropped and extract the label from the annotations file. It has a total of 4072 data in those 853 images, which are 717 for no mask-wearing, 3232 for single mask, and 123 for incorrect mask-wearing.

Due to a scarcity of online double mask images, different search engines were applied to obtain them, and artificially made double mask images were created. There were a total of 471 doubled masks collected, with the remaining 29 produced artificially as training data.

Table 3.1: Artificially created doubled mask data

Input				
Mask				
Output				

3.2.2 Data Finalisation

FMD data set was used as testing data because the data in FMD is largely from candid photos and seems more real-life. Each category extracted 30 photographs from the FMD for different distribution data testing. However, the 30 double mask testing images were still collected from the search engine. For the training data set, 450 faces with mask and single mask data were taken from the MM data set, and all incorrect mask wear was used. Some of the incorrect mask wear data come from MAFA and MaskedFace-Net. According to Assawiel (2019), it is crucial for the training data set to be similar to where you want the model to predict. So, the FMD data set, which looks like a real-life data set, will extract 50 cropped images for each class to shuffle in the training data set to make the model more robust in prediction. Before data pre-processing, all data must be cropped 1 person per picture. As a result, the final data for each class look like this:

Figure 3.8: Sample training data set

3.3 Data Pre-processing

In data pre-processing, all images will be cropped into one person per image, and the data set will then convert from RGB to BGR as OpenCV read image in BGR format. Then all the images will be resized to according to the trained size of pretrained model, which are 224x224 for MobileNetV2, ResNet50, ResNet152 and VGG, and 229 for Inception-v3 and Xception. Firstly, the without mask will be labelled with 0, the single mask face will be labelled with 1, the incorrect mask wear will be labelled with 2 and the doubled mask wear will be labelled with 3. Then the label will be converted to categorical. Next, the image will be pre-processed by importing the model-specific preprocess_input method. Finally, the data augmentation method will be applied to the data set to increase the images number in the training stage.

3.4 Data Augmentation

Data augmentation was carried out with 0.1 zoom range, 25 rotation range, 0.1 width shift range, 0.1 height shift range, 0.15 shear range, flip horizontally and with nearest fill mode.

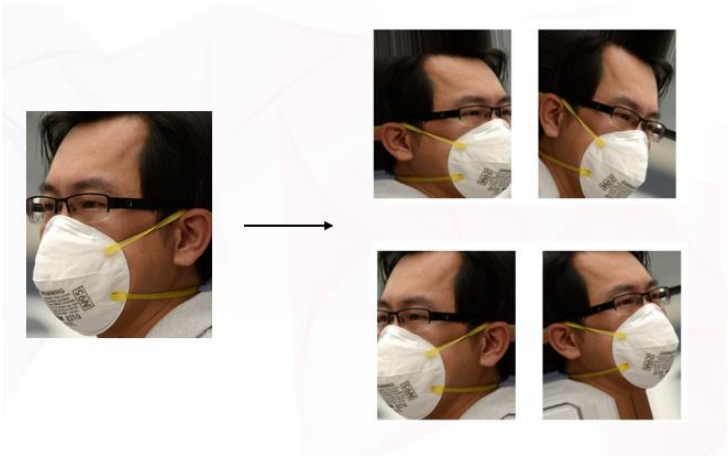


Figure 3.9: Augmented data

3.5 Training, Validation and Testing

To train a deep learning model, the data set with 2000 images will be divided into three categories, which is training, validation, and testing. Due to the small data set size used in this project, the ratio chosen to divide the data set is 80:10:10. The model learns and fits the parameters using the training data set. Then, the validation data sets were used to tune the model's hyperparameters as it was able to provide an unbiased evaluation of the model. Furthermore, the testing data set which comes from that 10% will be used to evaluate the final model as the same distribution testing.

In this research, two other testing data sets which come from other distributions and video captured from real life will be used to evaluate the model's performance further.

3.6 Feature Extraction

In feature extraction, a pre-trained model was used to train a new model in the project. The pre-trained model used is MobileNetV2, VGG-16, ResNet-50, ResNet-152, Inception-v3 and Xception. Since all models are available in Keras, they can be imported directly from the Keras framework. These pre-trained models will be used by freezing the entire convolutional base as only a small data set is used to train the model. Weight must be assigned in the model to prevent it from training from scratch. In these two pre-trained models, ImageNet was selected as the weight to train the model.

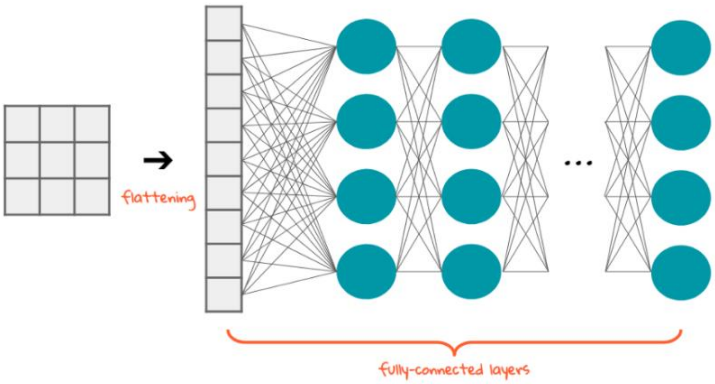


Figure 3.10: Flattening of the multidimensional array

The pre-trained classifier will then be removed and replaced with a new classifier. A flatten layer is added to make the image flat, which is into a 1-dimensional array as it is a multidimensional array, as shown in Figure 3.9. Then, a dense layer with Relu activation is added since relu deals better with images. To avoid overfitting the model, a dropout layer will be added. Lastly, a dense layer with softmax activation will be added to the final layer. Softmax is used as the data set is a multi-classification as softmax needs to be applied when there are more than two classes. The neuron for the last layer will be set to 4 as only 4 classes will be predicted.

3.7 Hyperparameter Tuning

To train a deep learning model more efficiently, hyperparameter tuning is essential and should not be ignored. Therefore, a Grid search CV will be used to find the appropriate hyperparameter for MobileNetV2, VGG-16, ResNet-50, ResNet-152, Inception-v3 and Xception. The learning rate and dropout rate are the hyperparameters that will be tuned in this project. A ParameterGrid will be used to assign the desired hyperparameter, and a loop will be created to loop the hyperparameter and generate the accuracy of the model. Using the desired learning rate will prevent the pre-trained model from distorting, so an appropriate learning rate is important; the learning rate used to tune will be 0.1, 0.01, 0.001, 0.0001 and 0.00001. Dropout is a technique in which randomly selected neurons are ignored during training, with 1 indicating no dropout and 0 indicating no output will transmit to the subsequent layer; the dropout values used in this model will be 0.3 and 0.5.

The Grid Search will be used to determine the best combination of learning rate and dropout rate.

3.8 Model Evaluation

To evaluate the model, a training and validation loss graph will be plotted to see the model's behaviour with different hyperparameters and pre-trained models. And, to evaluate the model's performance, the classification report class will be used to evaluate each model's performance by examining the accuracy, precision, recall, and the f1-score. In this project, the outcome is 4 classes. Therefore, a confusion matrix will be 4x4, which is shown in Table 3.2 below.

Table 3.2: Confusion Matrix for 4 classes

		Predicted Class				
		No mask	Single mask	Incorrect wear	Double mask	FN
Actual	No mask	TP_1	a	b	c	a + b + c
	Single mask	d	TP_2	e	f	d + e + f
	Incorrect wear	g	h	TP_3	i	g + h + i
	Double mask	j	k	l	TP_4	j + k + l
	FP	d + g + j	a + h + k	b + e + l	c + f + i	

The precision, recall and F1-score for each class can be calculated by (where n = class type):

Precision:

$$Precision_n = \frac{TP_n}{TP_n + FP} \tag{3.1}$$

where

n = 1,2,3

Recall:

$$Recall_n = \frac{TP_n}{TP_n + FN} \tag{3.2}$$

where

n = 1,2,3

F1-score:

$$F1_n = 2\frac{Precision_n * Recall_n}{Precision_n + Recall_n} \quad (3.3)$$

where

n = 1,2,3

The accuracy can be calculated by:

$$Accuracy = \frac{TP_1 + TP_2 + TP_3}{Total} \quad (3.4)$$

3.9 Pseudocode for this project

The algorithm below shows the pseudocode to train and test the model:

This pseudocode has covered all the methods explained in the previous section like reading data sets, data processing, data augmentation, feature extraction, network training, hyperparameter tuning and performance measurement for each model.

1. **def** read_dataset(img_size, preprocess_input)
    a. read image from directory
    b. process the images using preprocess_input
    c. return **images, labels**
2. **def** dataset_train_test_split(images,labels)
    a. split data set to 8:1:1
    b. return **x_train,y_train,x_test,y_test,x_val,y_val**
3. Define an ImageDataGenerator for data augmentation
4. **def** return_create_model(pretrained_model)
    a. **def** create_model (learning_rate, dropout_rate,optimizer)
        i. Define a base model from Keras library using pretrained_model
        ii. For layer in the base model, assign trainable to false
        iii. Define a head model by flatten, dense layer, dropout_rate and dense layer with class number and softmax activation function
        iv. Define an optimizer with learning_rate
        v. Compile the appended model with the loss of categorical_crossentropy, optimizer and metrics of accuracy

vi. return **model**

b. return **create_model**

5. **def** grid_search (create_mode)
    a. define KerasClassifier
    b. define param_grid dict with learning_rate and dropout_rate
    c. define GridSearchCV with cv =3
    d. Fit the grid with x train and y train
    e. Store all grid search result in excel
    f. return **best_param**
6. **def** train_model_with_best_param(best_param, create_mode)
    a. create model with best param
    b. define callback with checkpoint and early stopping
    c. Fit the model with the train and validation set, batch_size=32, epoch=20, and ImageDataGenerator
    d. Plot the loss and accuracy
    e. Save the model as .h5
    f. return **trained_model**
7. Define models (MobileNetV2, ResNet50, ResNet152, InceptionV3, Xception, VGG16)
8. For each model:
    a. read_dataset
    b. dataset_train_test_split
    c. define create_model function
    d. grid_search(create_model)
    e. train_model_with_best_param(best_param, create_model)
    f. load the best trained model
    g. Predict the model with the test set (same distribution)
    h. Print the classification_report
    i. Predict the model with the test set (different distribution)
    j. Print the classification_report
    k. Predict the model with framed based testing data set
    l. Print the classification_report
9. End

3.10 Work Breakdown Structure of the Project

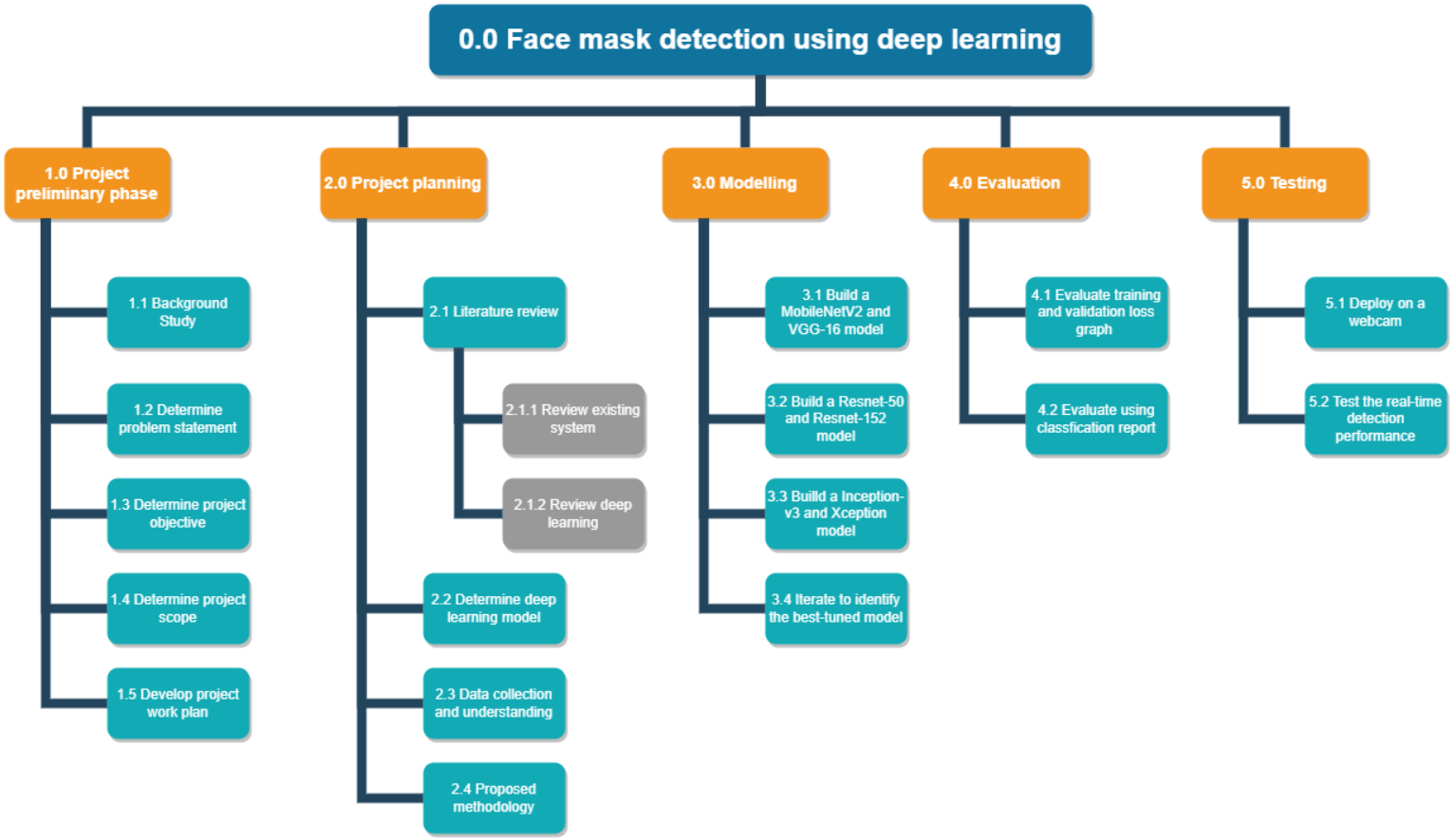


Figure 3.11: Work Breakdown Structure of the Project

3.11 Gantt Chart of Project

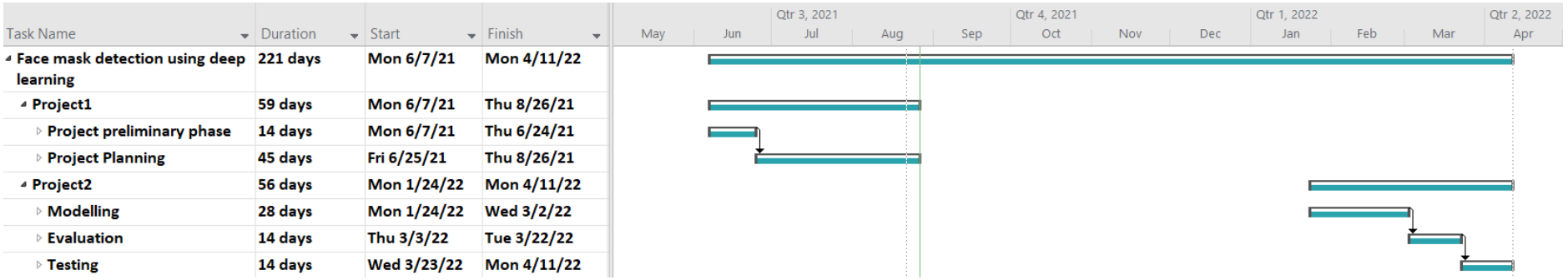


Figure 3.12: Gantt Chart of the Overall Project

The WBS and Gantt Chart for this study are shown in the figure above. Project 1 is more concerned with the preparatory phase, planning, analysis, and data collection, while Project 2 is more concerned with modelling, evaluation, and testing. During the preliminary phase, some background research will be conducted to get an idea of the research issue and to complete the project overview. Then, during the design stage, literature review is conducted to better understand deep learning, transfer learning models, coding techniques, and current face mask detection models. This is to get a concept of how to build the model for the research. Following that, data collection will be carried out in order to gather data for the training model in Project 2. To discover the optimum model, Project 2 will repeat the modelling, evaluation, and testing.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

The goal of this research is to predict face mask detection using the transfer learning approach that was selected. MobileNetV2, VGG-16, ResNet-50, ResNet-152, Inception-v3, and Xception were selected as transfer learning methods. The model will initially use grid search to tune the hyperparameter. Then, the optimal parameter will use to further training the model, and the best-trained model will be used to predict the testing data set. These models will be evaluated based on their accuracy and recall. The results that will be discussed in this chapter are:

1. Grid Search Result
2. Model Evaluation
3. Result Summary
4. Deploy the best model on a webcam

4.2 Grid Search Result

To discover the optimal combination of hyperparameters, a grid search with 3 cv approach is used. Due to computing resource constraints, the batch size 32 and epochs 5 are fixed during the gird search. In grid search, the Adam optimizer, learning rate (0.1, 0.01, 0.001, 0.0001, and 0.00001), and dropout rate (0.3 and 0.5) are all examined. The optimal parameters will then be chosen to retrain for 20 epochs with early stopping and checkpoints, allowing the best-tuned model to load for evaluation.

1. MobileNetV2

Table 4.1: Grid search result for MobileNetV2

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.384995	10
		0.01	0.660586	7
		0.001	0.784368	3

		0.0001	0.802497	1
		0.00001	0.756243	5
	0.5	0.1	0.436202	9
		0.01	0.658127	8
		0.001	0.768129	4
		0.0001	0.794369	2
		0.00001	0.750005	6

The optimal combination for MobileNetV2 is dropout rate 0.3, learning rate 0.0001, Adam optimiser.

2. VGG-16

Table 4.2: Grid search result for VGG-16

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.450674	10
		0.01	0.728709	5
		0.001	0.767509	4
		0.0001	0.768115	3
		0.00001	0.655638	8
	0.5	0.1	0.468105	9
		0.01	0.710597	6
		0.001	0.773747	1
		0.0001	0.77314	2
		0.00001	0.664377	7

The optimal combination for VGG-16 is dropout rate 0.5, learning rate 0.001, Adam optimiser.

3. ResNet-50

Table 4.3: Grid search result for ResNet-50

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.332394	10

		0.01	0.766859	7
		0.001	0.828746	4
		0.0001	0.851248	1
		0.00001	0.81499	5
	0.5	0.1	0.424952	9
		0.01	0.67939	8
		0.001	0.833741	3
		0.0001	0.850622	2
		0.00001	0.809365	6

The optimal combination for ResNet-50 is dropout rate 0.3, learning rate 0.0001, Adam optimiser.

4. ResNet-152

Table 4.4: Grid search result for ResNet-152

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.471253	9
		0.01	0.803747	7
		0.001	0.846255	2
		0.0001	0.851248	1
		0.00001	0.82312	4
	0.5	0.1	0.418745	10
		0.01	0.769994	8
		0.001	0.822498	5
		0.0001	0.838128	3
		0.00001	0.817501	6

The optimal combination for ResNet-152 is dropout rate 0.3, learning rate 0.0001, Adam optimiser.

5. Inception-v3

Table 4.5: Grid search result for Inception-v3

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.323773285	10
		0.01	0.686288019	7
		0.001	0.784381847	5
		0.0001	0.811258435	1
		0.00001	0.799388905	3
	0.5	0.1	0.346856534	9
		0.01	0.685632189	8
		0.001	0.781872094	6
		0.0001	0.805006862	2
		0.00001	0.796892007	4

The optimal combination for Inception-v3 is dropout rate 0.3, learning rate 0.0001, Adam optimiser.

6. Xception

Table 4.6: Grid search result for Xception

Optimiser	Dropout rate	Learning rate	Mean test score	Ranking
Adam	0.3	0.1	0.446293	10
		0.01	0.753734	8
		0.001	0.81562	5
		0.0001	0.819369	4
		0.00001	0.835003	1
	0.5	0.1	0.496907	9
		0.01	0.763772	7
		0.001	0.80063	6
		0.0001	0.826262	2
		0.00001	0.823751	3

The optimal combination for Inception-v3 is dropout rate 0.3, learning rate 0.00001, Adam optimiser.

According to the results, a learning rate 0.1 will always contribute to the last ranking, and most learning rates selected are usually low, indicating that when the learning rate is too high, it is very difficult for the neural network to learn because high learning rate will cause the model to quickly converge on a suboptimal solution, therefore the scores for high learning rate is always unsatisfactory.

4.3 Model Evaluation

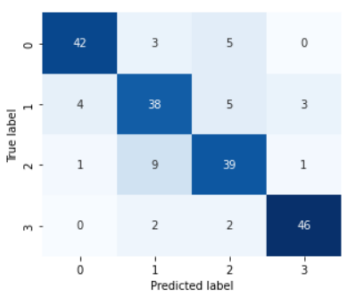
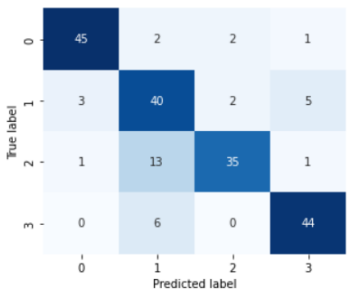
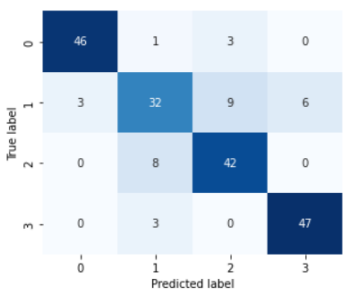
4.3.1 Test set from the same distribution (200 images)

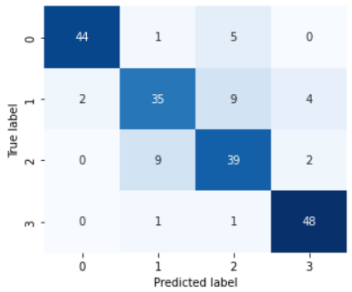
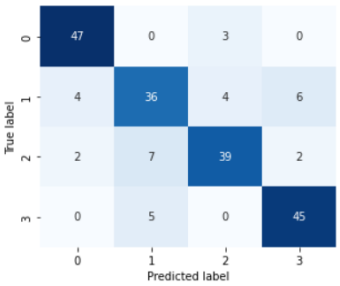
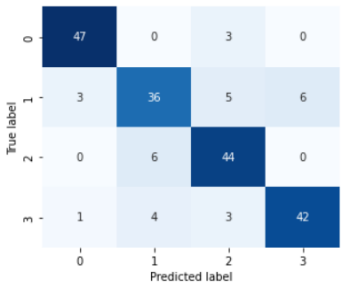
200 images are extracted from the training data set to evaluate the pretrained model in the same distribution

Table 4.7: Results tested on test set (same distribution)

Model		MobileNetV2	VGG-16	ResNet-50	ResNet-152	Inception-v3	Xception
Accuracy		0.8250	0.8200	0.8350	0.8300	0.8350	**0.8450**
Testing time (ms/step)		**20**	53	44	102	52	92
0 (no mask-wearing)	precision	0.89	0.92	0.94	0.96	0.89	0.92
	recall	0.84	0.90	0.92	0.88	0.94	0.94
	F1	0.87	0.91	**0.93**	0.92	0.91	**0.93**
1 (single mask)	precision	0.73	0.66	0.73	0.76	0.75	0.78
	recall	0.76	0.80	0.64	0.70	0.72	0.72
	F1	**0.75**	0.72	0.68	0.73	0.73	0.75
2 (incorrect mask wearing)	precision	0.76	0.9	0.78	0.72	0.85	0.80
	recall	0.78	0.70	0.84	0.78	0.78	0.88
	F1	0.77	0.79	0.81	0.75	0.81	**0.84**
3 (double mask)	precision	0.92	0.86	0.89	0.89	0.85	0.88
	recall	0.92	0.88	0.94	0.96	0.90	0.84
	F1	**0.92**	0.87	0.91	**0.92**	0.87	0.86

Table 4.8: Confusion matrix (same distribution)

		
Confusion matrix (MobileNetV2)	Confusion matrix (VGG-16)	Confusion matrix (ResNet-50)

		
Confusion matrix (ResNet-152)	Confusion matrix (Inception-v3)	Confusion matrix (Xception)

When we evaluate the model using data from the same distribution, we can see that the accuracy of six models is pretty good, with results that are over 80% accurate. Among the other models, the Xception had the best accuracy. We can also see that all models perform well in identifying no mask-wearing class.

4.3.2 Test set from other distribution (120 images)

This data set has 30 images from each class. The FMD data set (https://www.kaggle.com/andrewmvd/face-mask-detection) is used to extract the classes 0 (no mask mask-wearing), 1 (single mask), and 2 (incorrect mask wear). Due to the scarcity of double mask images in the data set, the double mask data obtained from the search engine was used. This testing set will be used to examine the model's performance in evaluating data for other distributions.

Sample Data:

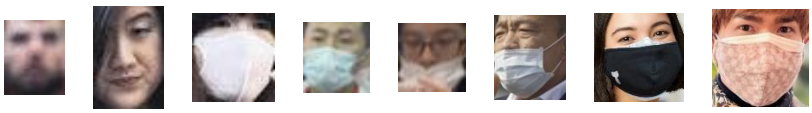


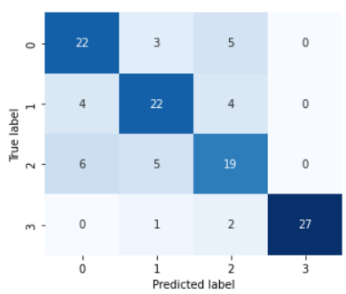
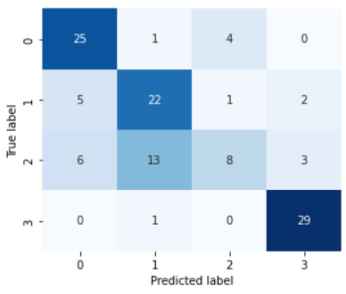
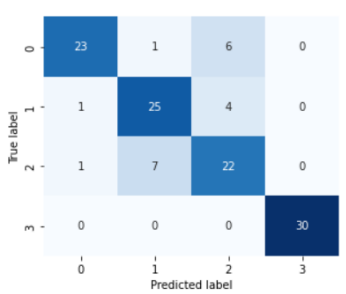
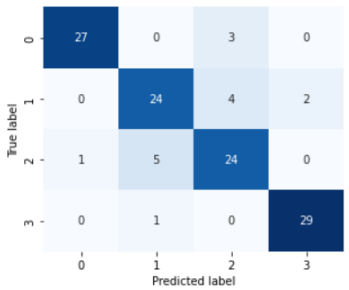
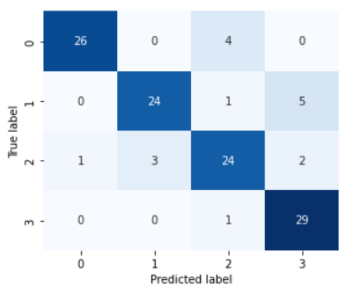
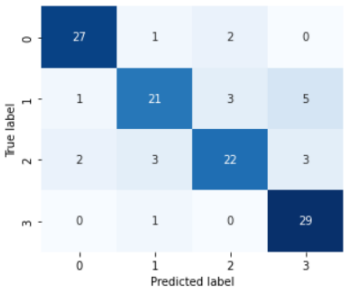
Figure 4.1: Sample data for test set (other distribution)

Table 4.9: Results tested on test set (other distribution)

Model		MobileNetV2	VGG-16	ResNet-50	ResNet-152	Inception-v3	Xception
Accuracy		0.7500	0.7000	0.8333	**0.8667**	**0.8583**	0.8250
Testing time (ms/step)		22	57	46	101	58	96
0 (no mask-wearing)	precision	0.69	0.69	0.92	0.96	0.96	0.90
	recall	0.73	0.83	0.77	0.90	0.87	0.90
	F1	0.71	0.76	0.84	0.93	0.91	0.90
1 (single mask)	precision	0.71	0.59	0.76	0.80	0.89	0.81
	recall	0.73	0.73	0.83	0.80	0.80	0.70
	F1	0.72	0.66	0.79	0.80	0.84	0.75

2 (incorrect mask wearing)	precision	0.63	0.62	0.69	0.77	0.80	0.81
	recall	0.63	0.27	0.73	0.80	0.80	0.73
	F1	0.63	0.37	0.71	0.79	0.80	0.77
3 (double mask)	precision	1.00	0.85	1.00	0.94	0.81	0.78
	recall	0.90	0.97	1.00	0.97	0.97	0.97
	F1	0.95	0.91	1.00	0.95	0.88	0.87

Table 4.10: Confusion matrix (other distribution)

		
Confusion matrix (MobileNetV2)	Confusion matrix (VGG-16)	Confusion matrix (ResNet-50)
		
Confusion matrix (ResNet-152)	Confusion matrix (Inception-v3)	Confusion matrix (Xception)

When the model was evaluated using data from different distributions, ResNet-152 had the greatest accuracy, followed by Inception-v3 and RestNet-50. When MobileNetV2 and VGG-16 predict data from different distributions, their accuracy drops dramatically. We can see that the double mask data work very well in this case since the data used is gathered in the same way as the training data set due to the scarcity of the data.

4.3.3 Test set from video frame (12 videos – 3 videos per category)

Total 12 videos are captured and collected by recording around the university and condominium. Every frame of the video is processed, and the faces are extracted using the DNN face detector model. The incorrect faces discovered by the DNN face detector are manually removed to ensure that no incorrect data is sent into the face mask detection model.

Sample video:

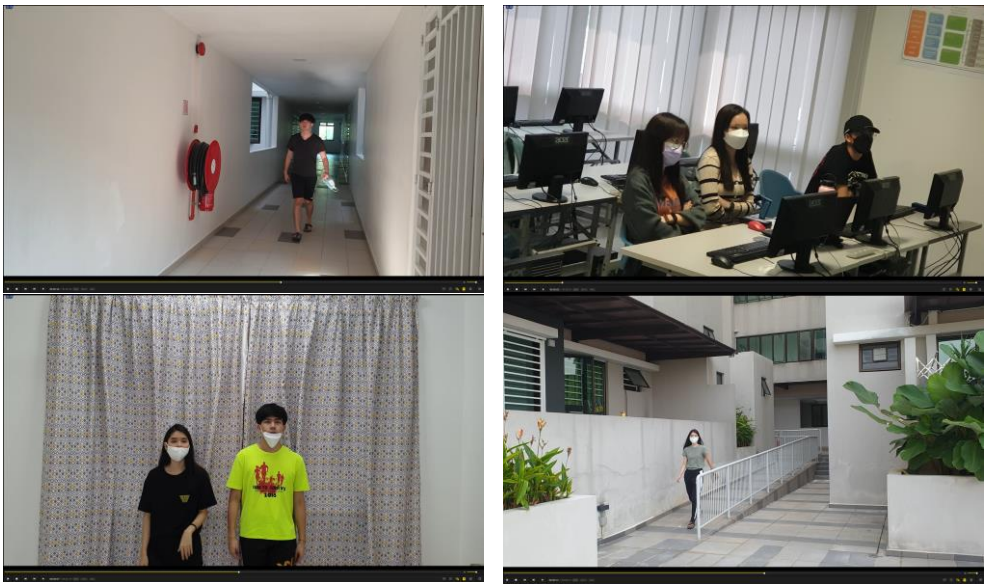


Figure 4.2: Sample videos captured

Sample cropped data from video frame:

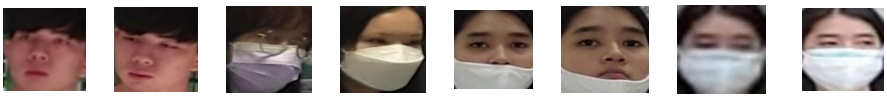


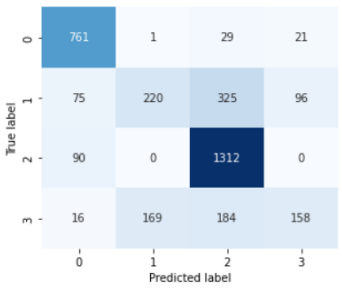
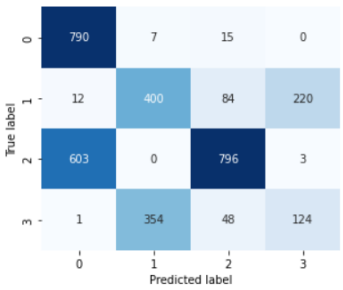
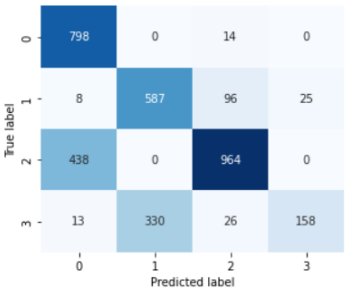
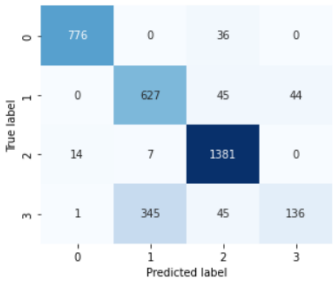
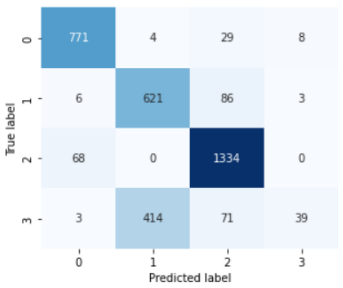
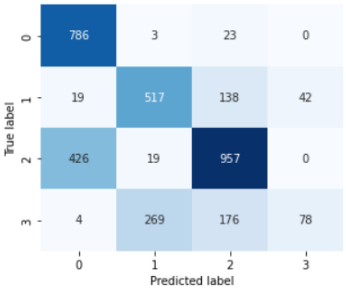
Figure 4.3: Sample data from the video frame

Table 4.11: Results tested on test set (video frame)

Model		MobileNetV2	VGG-16	ResNet-50	ResNet-152	Inception-v3	Xception
Accuracy		0.7090	0.6104	0.7252	**0.8447**	**0.7998**	0.6763
Testing time (ms/step)		22	57	49	110	62	97
0 (no mask-wearing)	precision	0.81	0.56	0.63	0.98	0.91	0.64
	recall	0.94	0.97	0.98	0.96	0.95	0.97
	F1	0.87	0.71	0.77	0.97	0.93	0.77
1 (single mask)	precision	0.56	0.53	0.64	0.64	0.60	0.64
	recall	0.31	0.56	0.82	0.88	0.87	0.72
	F1	0.40	0.54	0.72	0.74	0.71	0.68
2 (incorrect mask wearing)	precision	0.71	0.84	0.88	0.92	0.88	0.74
	recall	0.94	0.57	0.69	0.99	0.95	0.68
	F1	0.81	0.68	0.77	0.95	0.91	0.71
3 (double	precision	0.57	0.36	0.86	0.76	0.78	0.65

mask)	recall	0.30	0.24	0.30	0.26	0.07	0.15
	F1	0.39	0.28	0.45	0.38	0.14	0.24

Table 4.12: Confusion matrix (video frame)

		
Confusion matrix (MobileNetV2)	Confusion matrix (VGG-16)	Confusion matrix (ResNet-50)
		
Confusion matrix (ResNet-152)	Confusion matrix (Inception-v3)	Confusion matrix (Xception)

When using the real-life video frame for model evaluation, all model accuracy is dropped. ResNet-152 outperformed the other 5 models and achieved 0.96 recall for no mask-wearing, which is quite satisfactory as no mask-wearing is the most serious in the pandemic It also achieves 0.88 recall for single mask, 0.99 recall for incorrect mask wear but only 0.26 for double mask.

Figure 4.4: video frame for double mask detection (ResNet-152)

The recall for double masks is quite poor. This is because when the detected face is far away, the ResNet-152 model may not be able to identify the double mask; the model will only work effectively when the face is close enough to the lens. According to the confusion matrix of ResNet-152 and Inception-v3, double mask data is more likely to be predicted as single mask data.

4.4 Result summary

Table 4.13:Grid Search Result Summary

Model	Learning rate	Dropout rate	Mean test score
MobileNetV2	0.0001	0.3	0.8025
VGG-16	0.001	0.5	0.7737
ResNet-50	0.0001	0.5	0.8512
ResNet-152	0.0001	0.3	0.8512
Inception-v3	0.0001	0.3	0.8112
Xception	0.00001	0.3	0.8350

Table 4.14: Model Evaluation Summary

Model	Test set					
	200 images, from same distribution		120 images, from other distribution		Total 12 videos – 3 videos per category	
	Accuracy	Testing time (ms/step)	Accuracy	Testing time (ms/step)	Accuracy (Frame based)	Testing time (ms/step)
MobileNetV2	0.8250	20	0.7500	22	0.7090	22
VGG-16	0.8200	53	0.7000	57	0.6104	57
ResNet-50	0.8350	44	0.8333	46	0.7252	49
ResNet-152	0.8300	102	**0.8667**	101	**0.8447**	110
Inception-v3	0.8350	52	**0.8583**	58	**0.7998**	62
Xception	0.8450	92	0.8250	96	0.6763	97

Table 4.14 shows that ResNet-152 performed well in all three types of test sets, with an accuracy of 0.8447 when dealing with video frame prediction. It is the most stable model across the three test sets, making it the most robust model. ResNet-152, on the other hand, has the longest testing duration due to the depth of the model. In this research, the second-best model is Inception-v3, followed by ResNet-50. The MobileNetV2 has the shortest testing time; however, it only predicts video frames

with 0.7090 accuracy. Since the testing time is still in milliseconds, the ResNet-152 remains the best model as the testing time is so small that it may be negligible.

4.5 Deploy the best model on a webcam

The best model (ResNet-152) is deployed on a webcam to determine if the model is able to work on a webcam.

Result:

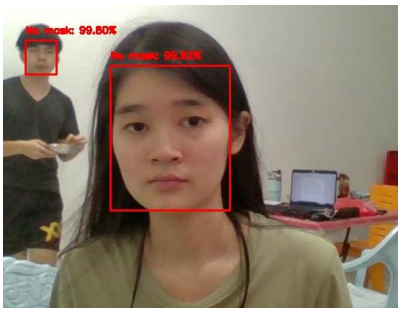


Figure 4.5: No mask-wearing

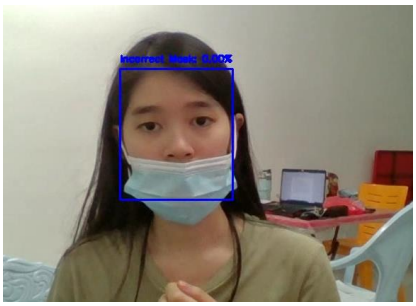


Figure 4.6: Incorrect mask-wearing

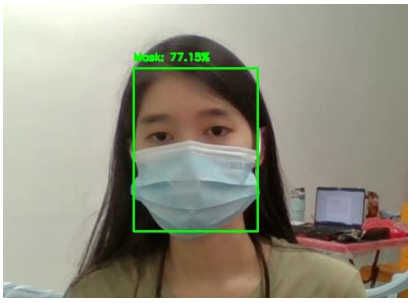


Figure 4.7: Single mask-wearing

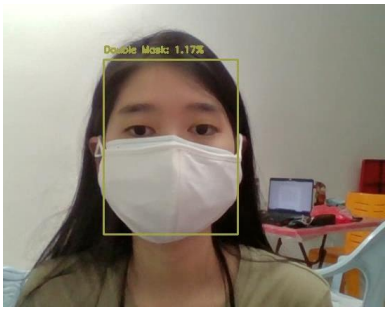


Figure 4.8:Double mask-wearing

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Transfer learning with 6 deep learning models MobileNetV2, VGG-16, ResNet-50, ResNet-152, Inception-v3 and Xception are trained and evaluated on the detection of mask-wearing in 4 classes: no wearing, single mask, improper mask-wearing and double mask. The research has achieved all the objectives as the proposed transfer learning model is able to detect the mask-wearing automatically by predicting the 4 classes. All trained models were evaluated, and the ResNet152 model performed the best in this research. ResNet152 has achieved the highest accuracy scores in both testing images and videos frame, which are 0.8867 and 0.8447, respectively.

5.2 Limitation

5.2.1 Poor performance when detecting individuals from far

When detecting an individual from a far distance, the trained model will fluctuate throughout classification due to data gathering that is mostly close to the lens.

5.2.2 Poor performance in detecting double masks wearing

The double masks wearing in this research have achieved the lowest performance which is only 0.26 recall for the ResNer-152 model, this is because the model always identifies the double masks as a single mask when the individual is far from the lens. This is due to the lack of real-life data sets for double masks wearing are used in training.

5.3 Recommendations for future work

5.3.1 Gather more data set

Recently, the proposed model only works well when the faces are closed to the lens. To enhance the performance, more data sets shall be gathered in the future and used in training models to improve the model’s ability to detect masks from far.

5.3.2 Improve performance of detecting double masks

The double mask detection performance in the real-life data set is quite poor. This is due to the fact that the images of the double mask are mostly obtained from social media (Instagram), and the majority of photos collected are the face close to the lens and come with a full-frontal face. Therefore, real-world shooting should be carried out in order to capture more realistic data for the double mask in order to increase its performance.

5.3.3 Explore more transfer learning model

The transfer learning model adopted for this research is mostly the Keras pre-trained model. Several other models, such as two-stage object detectors (eg. RCNN and Faster RCNN) and one-stage object detectors (eg. YOLO and SSD), may be implemented to achieve the research objectives and improve performance.

REFERENCES

Asghar, M.Z., Abbas, M., Zeeshan, K., Kotilainen, P. and Hämäläinen, T., 2019. Assessment of deep learning methodology for self-organizing 5g networks. Applied Sciences, 9(15), p.2975.

Assawiel, N., 2019. What to do when your training and testing data come from different distributions. [online] Available at: < https://www.kdnuggets.com/2019/01/when-your-training-testing-data-different-distributions.html> [Accessed 21 April 2022].

Cabani, A., Hammoudi, K., Benhabiles, H. and Melkemi, M., 2021. MaskedFace-Net–A dataset of correctly/incorrectly masked face images in the context of COVID-19. Smart Health, 19, p.100144.

Cao, Z., Shao, M., Xu, L., Mu, S. and Qu, H., 2020. MaskHunter: real-time object detection of face masks during the COVID-19 pandemic. IET Image Processing, 14(16), pp.4359-4367.

Centers for Disease Control and Prevention, 2021. How to Protect Yourself & Others. [online] Available at: < https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/prevention.html> [Accessed 19 June 2021].

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251-1258).

Chowdary, G.J., Punn, N.S., Sonbhadra, S.K. and Agarwal, S., 2020, December. Face mask detection using transfer learning of inceptionv3. In International Conference on Big Data Analytics (pp. 81-90). Springer, Cham.

Fan, X., Jiang, M. and Yan, H., 2021. A Deep Learning based Light-weight Face Mask Detector with Residual Context Attention and Gaussian Heatmap to Fight Against COVID-19. IEEE Access.

Frossard, D., 2016. VGG in TensorFlow. [online] Available at: <https://www.cs.toronto.edu/~frossard/post/vgg16/> [Accessed 18 July 2021].

Ge, S., Li, J., Ye, Q. and Luo, Z., 2017. Detecting masked faces in the wild with lle-cnns. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2682-2690).

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. and Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Hussain, S., Yu, Y., Ayoub, M., Khan, A., Rehman, R., Wahid, J.A. and Hou, W., 2021. IoT and Deep Learning Based Approach for Rapid Screening and Face Mask Detection for Infection Spread Control of COVID-19. Applied Sciences, 11(8), p.3495.

IBM Cloud Education. 2020. Neural Networks. [online] Available at: <https://www.ibm.com/cloud/learn/neural-networks> [Accessed 1 July 2021].

Jiang, M., Fan, X. and Yan, H., 2020. Retinamask: A face mask detector. arXiv preprint arXiv:2005.03950.

Jiang, X., Gao, T., Zhu, Z. and Zhao, Y., 2021. Real-Time Face Mask Detection Method Based on YOLOv3. Electronics, 10(7), p.837.

Krishna, S.T. and Kalluri, H.K., 2019. Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE), 7(5S4), pp.427-432.

Lin, M., Chen, Q. and Yan, S., 2013. Network in network. arXiv preprint arXiv:1312.4400.

Loey, M., Manogaran, G., Taha, M.H.N. and Khalifa, N.E.M., 2021a. A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. Measurement, 167, p.108288.

Loey, M., Manogaran, G., Taha, M.H.N. and Khalifa, N.E.M., 2021b. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustainable cities and society, 65, p.102600.

Mahapatra, S., 2018. Why deep learning over traditional machine learning. Towards Data Science.

Mayo Foundation for Medical Education and Research, 2021. COVID-19 (coronavirus): Long-term effects. [online] Available at: < https://www.mayoclinic.org/diseases-conditions/coronavirus/in-depth/coronavirus-long-term-effects/art-20490351> [Accessed 18 June 2021].

Mercaldo, F. and Santone, A., 2021. Transfer learning for mobile real-time face mask detection and localization. Journal of the American Medical Informatics Association.

Militante, S.V. and Dionisio, N.V., 2020, August. Real-time facemask recognition with alarm system using deep learning. In 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC) (pp. 106-110). IEEE.

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P. and Hemanth, J., 2021. SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. Sustainable cities and society, 66, p.102692.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G.V., Krpalkova, L., Riordan, D. and Walsh, J., 2019. Deep learning vs. traditional computer vision. In Science and Information Conference (pp. 128-144). Springer, Cham.

Qin, B. and Li, D., 2020. Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19. Sensors, 20(18), p.5236.

Rahman, M.M., Manik, M.M.H., Islam, M.M., Mahmud, S. and Kim, J.H., 2020, September. An automated system to limit COVID-19 using facial mask detection in smart city network. In 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS) (pp. 1-5). IEEE.

Rashvinjeet, S. and Jo, T.B., 2021. Covid-19: Wearing of double face masks recommended, says Health DG. [online] Available at: <https://www.thestar.com.my/news/nation/2021/05/22/covid-19-wearing-of-double-facemasks-recommended-says-health-dg> [Accessed 11 May 2022].

Rampal, L. and Liew, B.S., 2021. Malaysia's third COVID-19 wave-a paradigm shift required. The Medical Journal of Malaysia, 76(1), pp.1-4.

Rosebrock, A., 2017. ImageNet: VGGNet, ResNet, Inception, and Xception with Keras. [online] Available at: < https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/> [Accessed 19 July 2021].

Said, Y., 2020. Pynq-YOLO-Net: An Embedded Quantized Convolutional Neural Network for Face Mask Detection in COVID-19 Pandemic Era. International Journal of Advanced Computer Science and Applications, 11(9).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. and Chen, L.C., 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

Sharma, N., Sharma, R. and Jindal, N., 2021. Machine Learning and Deep Learning Applications-A Vision. Global Transitions Proceedings, 2(1), pp.24-28.

Sethi, S., Kathuria, M. and Kaushik, T., 2021. Face mask detection using deep learning: An approach to reduce risk of coronavirus spread. Journal of Biomedical Informatics, 120, p.103848.

Shrestha, A. and Mahmood, A., 2019. Review of deep learning algorithms and architectures. IEEE Access, 7, pp.53040-53065.

Shu, M., 2019. Deep learning for image classification on very small datasets using transfer learning.

Sinha, S., 2018. Why Google's MobileNetV2 Is A Revolutionary Next Gen On-Device Computer Vision Network. [online] Available at: <https://analyticsindiamag.com/why-googles-mobilenetv2-is-a-revolutionary-next-gen-on-device-computer-vision-network/> [Accessed 19 July 2021].

Sivarajah, U., Kamal, M.M., Irani, Z. and Weerakkody, V., 2017. Critical analysis of Big Data challenges and analytical methods. Journal of Business Research, 70, pp.263-286.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C. and Liu, C., 2018, October. A survey on deep transfer learning. In International conference on artificial neural networks (pp. 270-279). Springer, Cham.

Tsai, C.Y., Chou, Y.S., Wong, C.C., Lai, Y.C. and Huang, C.C., 2019. Visually Guided Picking Control of an Omnidirectional Mobile Manipulator Based on End-to-End Multi-Task Imitation Learning. IEEE Access, 8, pp.1882-1891.

Tsang, S.H., 2018. Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015/ [online] Available at: < https://sh-tsang.medium.com/review-

inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c> [Accessed 20 April 2022].

Wang, B., Zhao, Y. and Chen, C.P., 2021. Hybrid Transfer Learning and Broad Learning System for Wearing Mask Detection in the COVID-19 Era. IEEE Transactions on Instrumentation and Measurement, 70, pp.1-12.

World Health Organization, 2021. WHO Coronavirus (COVID-19) Dashboard. [online] Available at: <https://covid19.who.int/> [Accessed 18 June 2021].

Yadav, S., 2020. Deep learning based safe social distancing and face mask detection in public areas for covid-19 safety guidelines adherence. International Journal for Research in Applied Science and Engineering Technology, 8(7), pp.1368-1375.

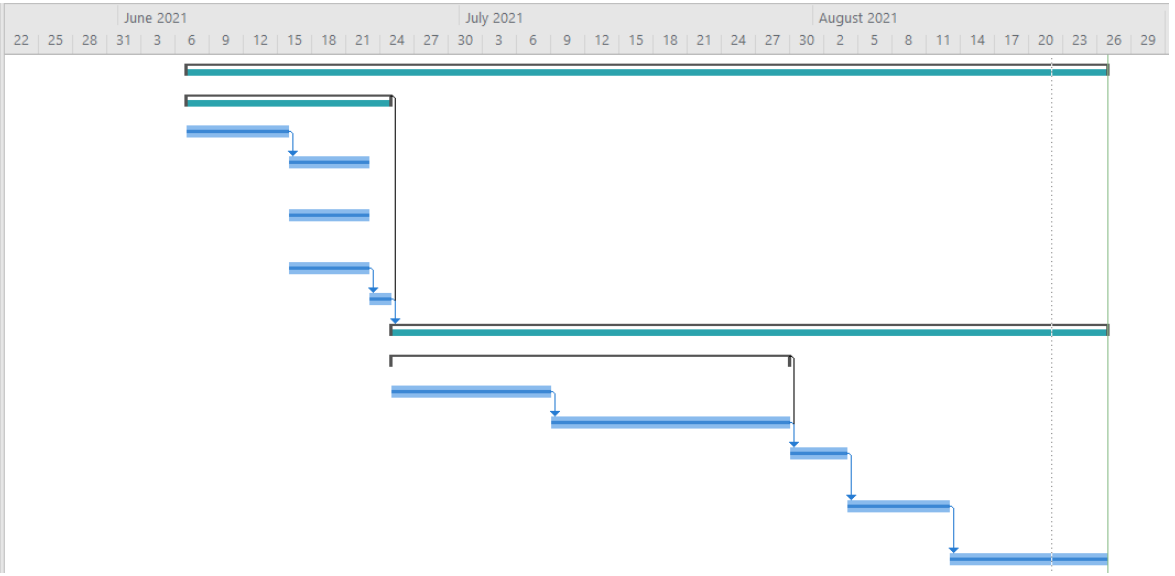
Yu, J. and Zhang, W., 2021. Face mask wearing detection algorithm based on improved YOLO-v4. Sensors, 21(9), p.3263.

Zhang, J., Han, F., Chun, Y. and Chen, W., 2021. A Novel Detection Framework About Conditions of Wearing Face Mask for Helping Control the Spread of COVID-19. IEEE Access, 9, pp.42975-42984.

APPENDICES

APPENDIX A: Detailed of the Gantt Chart in Project 1

Task Name	Duration	Start	Finish
Project1	**59 days**	**Mon 6/7/21**	**Thu 8/26/21**
Project preliminary phase	**14 days**	**Mon 6/7/21**	**Thu 6/24/21**
Background Study	7 days	Mon 6/7/21	Tue 6/15/21
Determine problem statement	5 days	Wed 6/16/21	Tue 6/22/21
Determine project obejectives	5 days	Wed 6/16/21	Tue 6/22/21
Determine project scope	5 days	Wed 6/16/21	Tue 6/22/21
Develop project workplan	2 days	Wed 6/23/21	Thu 6/24/21
Project Planning	**45 days**	**Fri 6/25/21**	**Thu 8/26/21**
Literature review	**25 days**	**Fri 6/25/21**	**Thu 7/29/21**
Review existing system	10 days	Fri 6/25/21	Thu 7/8/21
Review deep learning	15 days	Fri 7/9/21	Thu 7/29/21
Determine deep learning model	3 days	Fri 7/30/21	Tue 8/3/21
Data collection and understanding	7 days	Wed 8/4/21	Thu 8/12/21
Proposed methodology	10 days	Fri 8/13/21	Thu 8/26/21



APPENDIX B: Detailed of the Gantt Chart in Project 2

Task Name	Duration	Start	Finish
Project2	**56 days**	**Mon 1/24/22**	**Mon 4/11/22**
Modelling	**28 days**	**Mon 1/24/22**	**Wed 3/2/22**
Build a MobileNetV2 and VGG-16 model	3 days	Mon 1/24/22	Wed 1/26/22
Build a Resnet-50 and Resnet-152 model	3 days	Thu 1/27/22	Mon 1/31/22
Builld a Inception-v3 and Xception model	4 days	Tue 2/1/22	Fri 2/4/22
Iterate to identify the best-tuned model	18 days	Mon 2/7/22	Wed 3/2/22
Evaluation	**14 days**	**Thu 3/3/22**	**Tue 3/22/22**
Evaluate training and validation loss graph	7 days	Thu 3/3/22	Fri 3/11/22
Evaluate using classfication report	7 days	Mon 3/14/22	Tue 3/22/22
Testing	**14 days**	**Wed 3/23/22**	**Mon 4/11/22**
Deploy on a webcam	7 days	Wed 3/23/22	Thu 3/31/22
Test the real-time detection performance	7 days	Fri 4/1/22	Mon 4/11/22