



ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT100-3-M

DEEP LEARNING

HAND OUT DATE: 14th January 2024

HAND IN DATE : 15th march 2024

STUDENT NAME : Muhammad Arif Bin Jamaluddin

TP NO : TP067696

WEIGHTAGE: 100%

INSTRUCTIONS TO CANDIDATES:

- 1 Submit your assignment Online via Moodle
- 2 Students are advised to underpin their answers with the use of references (cited using the APA System of Referencing)
- 3 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld
- 4 Cases of plagiarism will be penalized

DEEP LEARNING METHOD USING FACE MASK
IMAGE DETECTION

A report submitted in fulfilment of the
requirements for the module
DEEP LEARNING

ASIA PACIFIC UNIVERSITY OF TECHNOLOGY & INNOVATION (APU)
SCHOOL OF COMPUTING AND TECHNOLOGY

.....2024

Abstract

In 2020, the world faced a crisis that changed the landscape of the world forever as covid 19 disrupt many industries and 4 years later in 2024, many industries adapted to these changes and the effect still can be seen today. Covid 19 exposed the weakness of various industries across the globe especially in healthcare industry and thus deep learning can be utilized to mitigate the risk of covid 19 by using neural network algorithm to identify face of people wearing a mask or without a mask. Five neural network models are proposed which are Transfer Learning ResNet50 with the lowest accuracy at 0.72 and Transfer Learning DenseNet121 with 0.99, MobileNetV2 with 0.99, Xception with 0.90 and Inception with 0.90. The models are trained with a 7553 RGB images datasets which are split into images with face mask at 3725 and images without face mask at 3828. The best neural network models are shared with Transfer Learning ResNet50, MobileNetV2 and Inception.

Index terms: Deep Learning, Face mask, Covid 19, Transfer Learning

Table of Contents

Abstract	3
1.0 INTRODUCTION	6
1.1 Face Mask Image Detection.....	10
1.2 Project Requirements, Assumptions and Justifications.....	14
1.3 Problem Statement	17
1.4 Aim.....	19
1.5 Objectives.....	19
1.6 Significance of Study	20
1.7 Research Questions	21
2.0 LITERATURE REVIEW	22
2.1 Introduction	22
2.2 Main Section	22
.....	25
2.3 Problem constraint of the model proposed.	29
2.4 Scope of problem domain	30
2.5 RELATED WORK	31
2.6 Comparison of Related Works	31
2.7 Analysis on Related Works.....	35
3.0 DEEP LEARNING ARCHITECTURES	37
3.1 CNN	37
3.2 Pre-Trained Models and Transfer Learning	43
4.0 METHODOLOGY	45
4.1 Experimental design	45
.....	45
4.2 Business understanding	46
4.3 Data understanding.....	46
4.4 Dataset collection/ description	46
4.4.1 Validation	48
4.4.2 Training and testing datasets.	49
4.4.3 Exploratory data analysis	49
4.4.4 Modelling	51
4.4.5 Evaluation.....	51
4.5 Deployment	52
5.0 MODEL IMPLEMENTATION	53

5.1 Introduction	53
5.2 The Importance of Good Model Implementation to Ensure Accurate and Reliable Results	53
5.3 Python in Google Colab Setup	54
5.4 File Location	54
5.5 The Neural Network architecture & typical workflow of the coding step	55
5.6 CNN	56
5.6.1 Using Pre-trained Neural Network (Transfer Learning ResNet50)	62
5.6.2 DenseNet	67
5.6.3 MobileNetV2.....	70
5.6.4 Xception	77
5.6.5 Inception.....	83
5.6.6 Evaluation.....	90
6.0 TUNING & VALIDATION	91
7.0 VISUALIZATION & CRITICAL ANALYSIS.....	97
8.0 DISCUSSIONS.....	100
9.0 CONCLUSION	104
10.0 REFERENCES.....	105
List of figures.....	108
List of tables.....	110

Assignment part 1

Component 1

1.0 INTRODUCTION

Deep learning, Artificial Intelligence, Machine Learning recently has been widely used and utilized by the industry recently although this field has been around since the 1940s. In recent years many industries or sectors try to make use of the data they gathered and use deep learning techniques to make predictions for future outcomes. It can be deduced that deep learning is a subset of machine learning, and machine learning is a subset of artificial intelligence.

Deep learning can be referred as the application of machine learning that applied complex algorithms and deep neural networks to train a model, and on the other hand, machine learning is the application of artificial intelligence that allow the system implemented to undoubtedly learn from the task repeated and improve from the experience. For example, in video games the computer opponent learns from experience playing against the human opponent and over time will improve to counter the movement of the human opponent. Lastly, artificial intelligence denotes the ability of machines to imitate or act like humans where it can learn actions like normal humans do which is creativity, perception, and logic.

There are various applications of deep learning such as virtual assistant, chatbots, self-driving car, and more. In the entertainment industry, many online streaming services such as Netflix, Disney plus, used classifications to recommended new shows based on user's past search and watched shows. Deep learning is being used in many areas across various industries such as engineering, healthcare, software, finance-commerce and many more. Industry that deals with or generated huge amounts of data stored in excels or any database format will take advantage of deep learning to improve decision making in their business.

Table 1

Deep Learning	Machine Learning
Large size of dataset	Small amount of dataset
Depends on high end machine (Very GPU intensive)	Works well with low end machine (Less GPU intensive)
Time consuming to train data	Less time compute to train the data
Less time needed to test the data	Need more time to test the data
End to end problem can be solved	Usually divided the task into smaller parts and solves it individually and later combined the results

In this assignment, there are various topics that can be chosen from which are computer vision that typically the project suitable for it is object recognition or classification. Computer vision typically handles image datasets, videos, and other visual inputs where it allows computers to see the input of raw images and data and train or test the image datasets. Computer vision is widely used in various industries such as smart manufacturing, energy, and especially automotive industries with self-driving cars.

Another topic that can be explored is Natural Language processing where deep learning is used extensively in sentiment analysis. NLP refers to a technique in machine learning or deep learning that uses computer algorithms and statistical models to process and understand human language, especially text data. Typical example or application of sentiment analysis are fake news detection, sentiment analysis of customer reviews online and more.

Another field that can be explored is generative model where it is known as a statistical model that is able to generate new data instances. Example of generative models are image generation and fake image detection where both computer vision and generative models are connected with one another. Types of generative models are (GANs) Generative Adversarial Network where main applications of generative model generally used in text to image generation and data augmentation. On the other hand, computer vision mainly used image classification, facial recognition and more.

Time series is another topic of deep learning where typically the applications are mostly in the space of stock market predictions and weather forecasts. Usually, deep learning uses time series to make predictions from time sequences data. Some of deep learning methods that are used in

time series are Recurrent Neural Networks (RNNs) where it uses sequential modelling vanishing gradient problems. These are some of time series functions or applications such as financial forecasting where financial institutions can be used to predict stock prices, exchange rates and where it uses historical data. This method is useful for investors, traders, and financial institutions analysts. Before that let's dive deep into the advantages and disadvantages of deep learning as shown in table below.

Table 2

Advantages	Disadvantages
Much more efficient algorithms than other algorithms	Required huge and large amount of data
Used open-source tools such as pytorch, keras.	Does not have strong theoretical foundation
Reduces requirement for feature engineering	Computationally expensive to train

It can be noted that deep learning is a type of supervised machine learning where it has neural networks where this network has several layers and modules. Deep learning is usually non-linear, hierarchical with abstract representation of data and it has flexible models with input and output where it is mostly based on mathematical equations and used in differential functional programming.

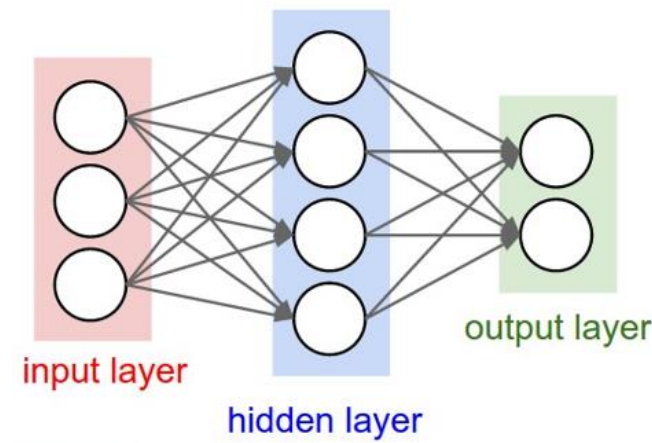


Figure 1

Figure above shows the basic architecture of deep learning system where artificial neural networks are represented by input, hidden and output layer. Figure below shows the comparison between typical machine learning with deep learning method where for machine learning it took information of a car where the information will be transformed into the model and fed

into the model. Next it will analyze the features variable, classification and give predictions it is a car. Hence it can be categorized into supervised learning because label is given to the data.

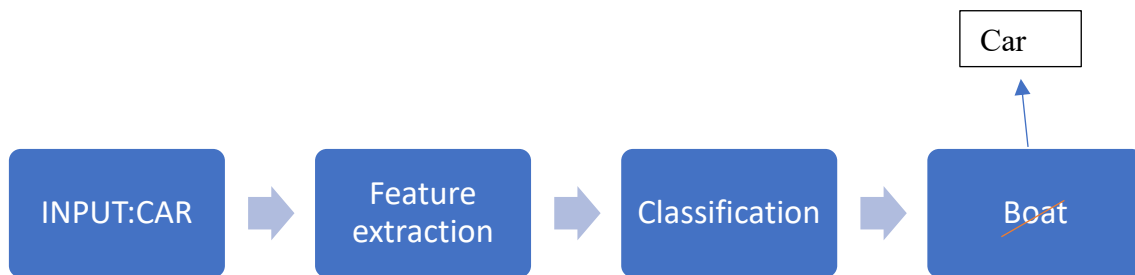


Figure 2

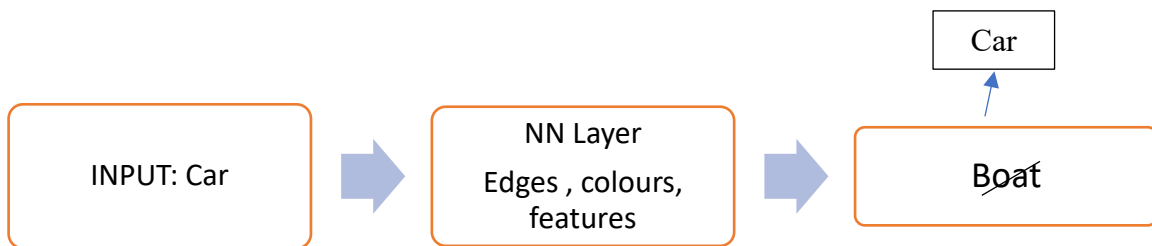


Figure 3

But on the other hand, deep learning used slightly different approaches where instead of having one feature extraction it will have multiple neural networks layers such as edged, contour and features of the input images which will give prediction it is a car. Again, both are still unsupervised machine learning as it used labeled data.

1.1 Face Mask Image Detection

In this era of rapid digitalization millions and billions of data are being generated everyday whether it is in text format, images, audio and many more. Particularly, with the rise of social media platforms and over the years computing and processing powers of chips that run on smartphones, tables and laptops become more powerful and advanced, petabytes and exabytes of data are being generated every day.

For example, platforms like Tik Tok, a social media platform where people constantly consuming videos and hence more people or users upload content and images to the platform. Hence, deep learning comes handy as it can help humans to recognize patterns or features on an image that can't be detected by normal human eyes. Image recognition and object detection works by train computers to understand and interpret the image or visual spatial where usually the method involved in converting the image into numeric or vector form.

There are quite a few differences between image recognition and object detection even though both of them involve interpreting images. Image recognition involves what identifies an entire image represents, like for example identifying a photo whether it is in landscape or portrait mode. For object detection, it is more advanced by identifying and locates certain features of multiple objects in an image.

The mechanism behind image recognition is that it starts with image datasets as input and then pre-processing is executed to remove any noise in the image dataset. After that, the images that had been processed were then fed into machine learning algorithms where some of the features extracted colors, contours, patterns, and shapes. Then, some of the features extracted are put into classifiers which will then interpret the image where the output of classifiers is the prediction. Predictions will tell what the image is about based on the information learned. After that post processing is done to refined the output of the image by filtering it using transfer learning and data augmentation methods.

Table 3

Techniques	Explanation
CNN	It's a type of deep learning method usually used in image recognition where
Deep Learning	Used ANN where it has hidden layers which useful to process large size of unstructured data like images.
Feature extraction	It used feature extraction method such SIFT, SURF and HOG where it can detect unique features in an image such as corner and edges

An example application in real life is an application called Google Lens where it uses image recognition technique where it can identify real life objects with camera. It is a computer vision based that allows computers to capture or record images and identify whether the objects are plants or animals. Other than that, object detection is a type of computer vision application that has become very common in people every day lies from camera surveillance, security system, MRI/CT scans and more.

There is reason object detection is used mainly used in autonomous vehicle where deep learning technology able to reduce human error where distracted driving is responsible 94% from accidents happening (V. Kukkala, et al., 2018). There is evolution in the industry of object detection where 20 years ago, human face detection in real time used Viola Jones detector (L. Jiao, et al, 2019) and several years later a new object detection method called HOG are used in pedestrian street detection (N. Dalal, et al, 2005).

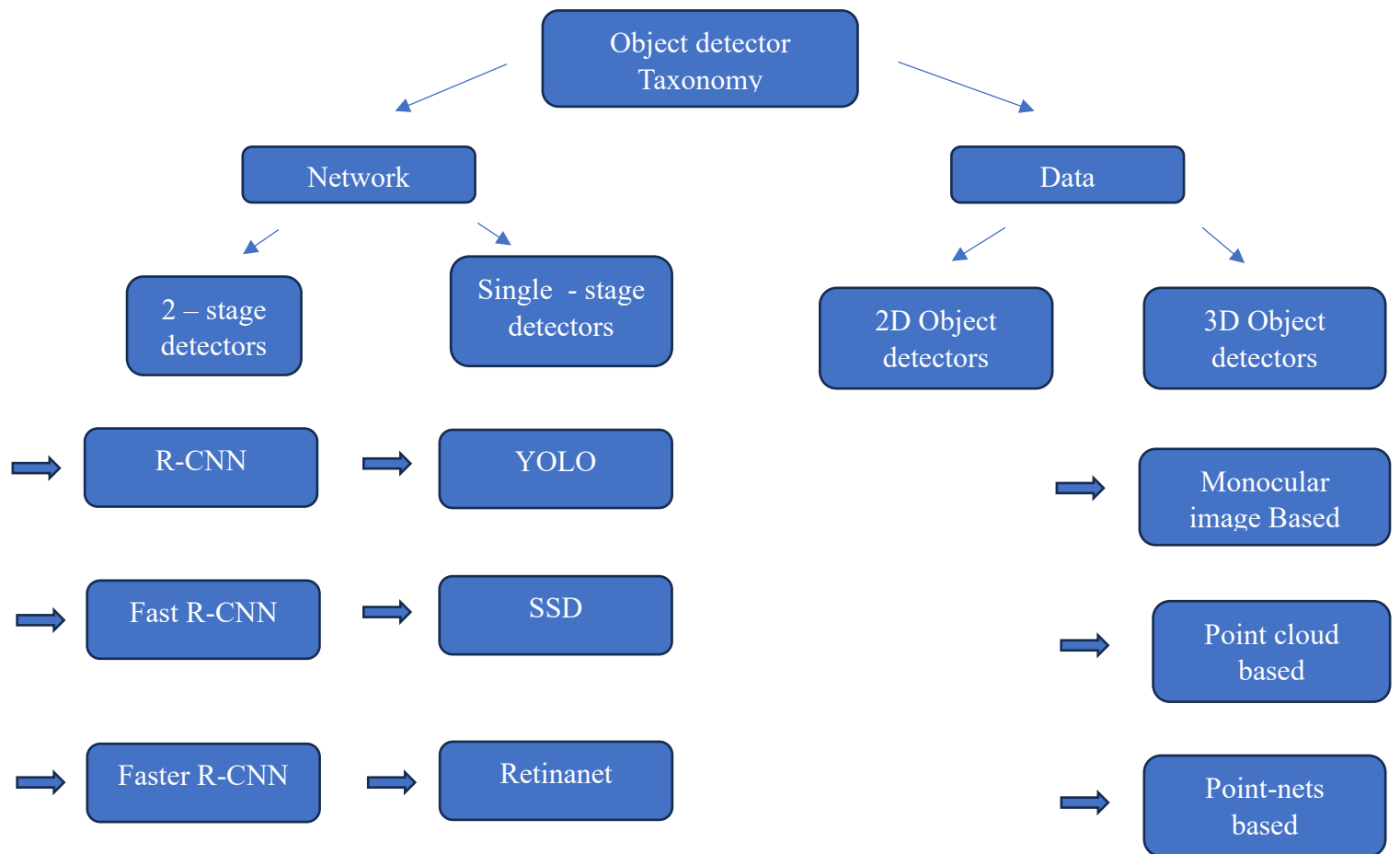


Figure 4

Figure above shows the taxonomy of object detectors. It is known that HOG model detectors are then developed into DPM, which were the pioneer models that focus on multiples images or object detection (P. F. Felzenszwalb, et al, 2010).

Due to event of covid-19 that occurs on the world almost 4 years ago in 2020, this causes most people to wear mask until now even in 2024 although the pandemic already over and the world enter endemic phase in 2022, wearing mask is still recommend if someone are sick going outside in public places. For the past 4 years lot of research had done in the academia about covid 19 where lot of data and information gathered and mined from these diseases, where the project is about covid 19 predictions or predicting the vaccination progress using machine learning.

Hence, this assignment using the domain of healthcare which is regards to covid 19 where it studies about how the algorithm able to detect if someone wearing a mask or no. Moreover, a much more complex project that involves face mask detection is deployed until the application

stage but for this assignment the deployment is only executed on Google Colab. Example a deep learning application is used in many faces' detection application for example FaceID on smartphone like Iphone where images can be analyzed using neural network. This feature allows the computer to recognize the face where the fundamentals of this technology are used in the proposed topic face mask image detection, and it can be seen in most image identification services or applications.

Hence, it is important to identify whether a person wears a face mask or no using deep learning method as covid 19 are easily transmissible in airborne condition especially in crowded area where there is huge crowds or gatherings. Even before covid 19, a novel pandemic happened face mask has been essential part of daily human life where it is the only protection used interacting or communicating with people in public. The event of covid 19 disrupts every industry in the whole world including healthcare, education, tech industry, agriculture and more. Due to covid 19, many industries accelerated adopting more and more digitalization by using machine learning, artificial intelligence and being more tech savvy in their daily day operation. Covid 19 changed people life where back then when the coronavirus happened death are increasing rapidly where it exposed a lot of country healthcare systems (T. Greenhalgh, M. B. Schmid, T. Czypionka, D. Bassler, and L. Gruer, 2020)

Face masks are useful to mitigate against respiratory diseases where WHO already create a guideline for common people and healthcare workers on using face mask and hence the purpose of this assignment is to create efficient face mask detection using deep learning method. The main objective of this assignment is to identify whether someone wear mask or no where the system later can be deployed in real world from input of test and train image datasets from Kaggle.

Face mask detection is very crucial to enhance security where identification procedure can be later implemented based on this face mask detection project that was based upon object detection and image recognition technology. The ability to recognize people using facial recognition technology will be useful in surveillance camera system to catch dangerous criminal and over the years government are adopting technology in recording people data and stored in database. Hence, using deep learning technology to capture people face will be useful for government applications such as ID digital nasional or MySejahtera where it only needs camera for quick verification to collect information easily.

1.2 Project Requirements, Assumptions and Justifications

Project requirements

There are some requirements to adhere to for this assignment where the topic selected must align with the student's interest. For example, for this topic domain interest is generative models about face mask detection where the best deep learning model or technique should be applied to this specific problem. There are various deep learning methods learned in this course such as DMLP, CNN, RNN, RL and many more.

Table 4

Deep Learning	Type	Applications
Deep Multi-Layer Perceptron (DMLP)	Regression or classifications	Used in areas such as financial, engineering, medical and social media analytics
Convolutional Neural Network (CNN)	Imaged recognitions	Used in autonomous and unmanned vehicle or self-driving car (auto pilot)
Recurrent Neural Network (RNN)	Time series problems	Used in Large Language Modelling (LLM), text language translation
Reinforcement Learning (RL)	Adaptive learning	Used in games, robotics application

The table above shows the major categories of deep learning where each of these algorithms has its unique features to solve real life problems where it is revolutionized various fields of machine intelligence where it is used in speech recognition, Natural Language Processing, image recognition, handwriting recognition, and computer vision.

One of the requirements of this project is to choose or select research papers related to the topic chosen, which in this case is face mask image detection. The code which this topic is based on is taken from Kaggle, a popular open source online for data scientists, statisticians, and researchers to study and learn about data science. There are some changes and tweaking that need to be done on the machine learning architecture by tuning the algorithm architecture or do hyperparameter tuning. This will be further explained later on, and this assignment allowed students to continue with capstone project which is about fake news detection using machine

learning algorithm and try with different deep learning but for this assignment new topic proposed which is face mask image detection. This is due to a new opportunity to learn using images datasets instead of textual datasets for face mask detection.

Regarding the literature review, there should be around 13 articles referenced related to face mask image detection as one of the requirements and this will be further explained in component 2, literature review. The main point of literature review is to conduct research related to domain of study, face mask image detection where the main focus would be on results and methodology. Hence, justifications are needed based on literature review and why the proposed model of the deep learning method is used. The key components of methodology should include deep learning architecture, training approaches, fine tuning methods, and evaluation metrics where it will be further explained in component 4.

The next part is selecting the best network architecture where for face mask image detection there is already existing pre trained model available on Kaggle where the architecture and the codes have already been explored. Hence, for this assignment the code is allowed to be modeled upon to solve the face mask image detection problem by ensured all the sources are cited properly. Another requirement is that complex architectures are not advisable to be implemented due to it will take much longer time and requiring high ram to crunch the datasets. There is another approach to study the network architecture by reducing the number of layers, tuning the batch and epochs size and more.

For this assignment face mask image detection datasets are chosen from Kaggle where the description of the datasets will be explained in component 4. On the side note, one of the requirements is to choose datasets with reasonable complexity as datasets with huge size will require more processing time to be executed. The last requirement is project tuning, tuning and analysis where after selecting the best training algorithm, evaluation metrics and model building need to be done where hyperparameter tuning is needed to find better accuracy of the proposed model. Hence, model performance needs to be critically analyzed and comparison is needed with the findings from literature review.

Assumptions & justifications

The reason why deep learning is widely used in face mask detection is that neural networks have better performance than other classification algorithms but there is some drawback using this where it is time consuming and needs high computational power. Deep learning has been proven to be useful in face recognition applications where it can detect object or certain features in a camera or pictures. As the field of artificial intelligence becomes more advanced, the area of object detection goes from heavily relied on engineered features where it was crafted by algorithms towards much more complex architectures using neural networks especially (CNN). CNN used different approaches where it is able to learn and extract features from raw data instantly and automatically.

Many industries in their own respective fields such as retail, healthcare, anti-money laundering bank start to developed for commercial and business purposes using object detection and image recognition to improve their business. It can be seen that some industries used object detection for smart manufacturing to improve efficiency in their production. This can be seen with the terms thrown upon lately in engineering sector such as artificial intelligence, machine learning, internet of things, digital twins, and robotics in manufacturing where smart manufacturing used deep learning technology by integrating computer vision and machine vision to do visual inspection and defect detection.

Hence, object detection is able to solve issues addressed by these industries where deep learning algorithms are able to identify and categorize various types and sizes of data whether it is a cat, an object, house, bicycle and more. Deep learning is quite a complicated architecture where for example in object detection, the image is represented in pixels known as the smallest unit of a digital image or display which are used in convolutional neural network. These neural networks contained fillers where it goes through process where filters are applied where usually it applied visual images or pixels by processing data in grid like topology. Most pixels are in the format of 3-dimensional by 3-dimensional arrays where the images width and height depend on the image resolution and in CNN, every image is represented in the form of arrays of pixel values.

1.3 Problem Statement

A lot of researchers faced some challenges when tried to conduct a project with face mask detection where dealing with image datasets requires huge amounts of time to train and test the model with deep learning algorithms. Moreover, there are issues regarding finding the information about the data where retrieving, storing, and processing image datasets and pre-processing and cleaning the data is one of the challenges. For example, image datasets required more time to upload into the google colab compare to textual datasets and usually to have good analysis on deep learning machine learning algorithms it required quality datasets for this assignment. This is to ensure high accuracy and better classification when identifying people with face or without face mask.

Therefore, face mask detection is used by using machine learning techniques such as OpenCV, Keras and Tensorflow in python using Google Colab where Convolutional Neural Network model is used. There are issues with the existing method where it using manual monitoring system by checking people manually where it is very labor intensive and required officers to check it every time. Hence, by proposing the build model using object detection this algorithm can detect if the person in the image or live stream video is wearing face mask or no and hence able to reduce the risk of covid 19.

There is significant research had been done on face mask detection in the past but there are limitations where in 2001 extensive research had been done on this field where it uses design of handcraft feature and using traditional machine learning algorithm. (L. Nanni, S. Ghidoni, S. Brahnam, 2017)(Y. Jia et al.,2014). But there is drawback on the approach proposed where the feature design has high complexity, and it has weak detection accuracy. There are still some issues need to be addressed in face mask detection problem where there are problems with differencing between detection mask over the face and detection of face under the mask. There is still less literature review done to accurately detect mask over the face and transfer learning method will be used to transfer learned kernels from the trained neural network layers to the face mask image dataset. This will be further explained in component 4 and component 5 section.

It is known that human eyes are limited when it comes detecting anomalies in any image datasets or pixels as machine able to do it more efficiently and many applications can be seen using object detection and facial recognition such as in most smartphone applications such as

Tik Tok able to use algorithms where if the users upload images on the platforms it will learn and understand the behaviours of the users and recommend ‘fyp’ or ‘for you page’ algorithm regards to the image uploads.

The mechanism behind face mask detection is that face identification without face mask is much easier to be implemented with face mask due it is quite difficult to extracted information from a face mask image. There are several features on the face need to be extracted for information such as chin, lips, nose, cheeks and usually face mask are identified by two periods which are the features extraction and face recognitions. Face recognition can be defined as a technology that able to identify individual faces and comparing pattern of the faces with database with known faces. It is some type of biometric identification where the computer able to distinguish people face. Below shows the

Table 5

Steps of face recognition	Explanation
1. Detection	The computer or algorithm able to locates faces for example for face mask it able to detect face with mask or no
2. Alignment	The algorithm able to recognize face to a default position and also able to detect any changes in pose, facial expression.
3. Feature extraction	The algorithm able to retrieve information from the image dataset.
4. Feature matching	The stored image in database is compared with extracted features for validation
5. Verification	It will give end results usually a complete stage until deployment will link through an application software telling if the person verified or not

For this assignment CNN are used for the built model and it will be assessed using the training, testing and validation phase. It can be denoted that researchers had proposed model framework using deep learning techniques for object detection where some of the classifiers are called MobilenetV5 where it is mainly used in object detection, image classification and semantic segmentation (S. Saponara, A. Elhanashi, and A. Gagliardi, 2021). This method able to

drastically reduce the complexity of the architecture, the model size, and the cost because this technique used Depth wise Separable Convolution which is used in devices with low computational power. Other than that, there is another deep learning technique used to solve the issues of vanishing gradient where it is a common problem in training deep neural networks by using ResNet150 where it allows to train more than 150 layers of neural networks. Another CNN architecture called VGG16 where this method used 3X3 fillers with 1 1 stride instead of using hyper-parameters and this method the max pool layers and convolution network are aligned in the same way on the architecture.

1.4 Aim

The aim of this assignment is to proposed a face mask detection using deep learning technique where this computer vision technique is able to accurately identify whether the face or people in the mask wearing a face mask or not.

1.5 Objectives

The objective of this deep learning assignment can be shown below.

- To create an algorithm that can directly detect if a person is wearing a face mask or not.
- To create a robust code scripting with python on Google Colab consisted of necessary steps such as importing the libraries, data preparation, model tuning, validation and more.
- To used deep learning model for the face mask image detection such as Transfer Learning ResNet50, DenseNet121, MobileNetv2, Xception and Inception.

1.6 Significance of Study

Image detection is very important in today's society where for example face mask detection is very important for these obvious reasons as shown below.

Table 6

Importance of face mask detection	Explanation
Disease prevention	The usage of face mask is important to prevent the spread of respiratory disease and nasal infections
Early mitigation or prevention	By implementing face mask detection, it can identify people in real time without mask or no
Health procedure guidelines	In public spaces or event when the world going through endemic phase mask are encouraged to use and hence face mask detection ensure the safety of the public
Safety in workplace	This are useful for the safety of people who are working in an enclosed environment
Monitoring in public spaces	Monitoring system can be implemented for face mask detection in area such as shopping malls and airports to mitigate the risk
Automated surveillance	By using object detection technology, enhance camera security and can reduce human burden in camera surveillance.
Decision making	Data can be derived from this face mask detection where it may contain information about how many people compile on wearing mask, the high-risk area and the trends over the time. This is helpful for business and authority figures to make informed decision based on derived data.

There are several research on why wearing mask able to mitigate the risk of covid 19 where there is a study conducted by researcher at University of Edinburgh whereby wearing a face mask it can reduced the spread of covid 19 (L.R. Garcia Godoy, et al, 2020). Hence due to this evidence backed by researched and statistics, this results strongly recommended the general

public to wear face mask in the public. There is another research done where Steffen et al. conduct a study to study the difference on the risk of getting covid 19 exposure for people lives in Washington and New York where almost 80% of population wearing weak mask that is only 20% effective, but it could minimize and prevented 17-45% expected death due to covid 19 in New York. (S.E. Eikenberry, et al, 2020). It is found out that wearing face mask able to reduce the daily death rate by 34-58%.

1.7 Research Questions

Several research questions done for this assignment such as;

- What are the best classifiers or algorithms needed for a good accuracy in detecting or identifying whether a person wearing a face mask or not?
- What are the drawbacks of using different hyperparameter tuning techniques and many other optimizations on the accuracy of the models?
- What is the importance of face mask detection in reducing the spread of covid 19?
- What is the best approach to proposed the best deep learning method for this assignment?

Component 2

2.0 LITERATURE REVIEW

2.1 Introduction

It is noted for literature review it is required to do deep analysis and justification on why the selected deep learning algorithm is selected compared to other algorithm for face mask detection. It is required to do detailed survey on some of the relevant deep learning architecture which will be further explored in related work where for this part it also contained table of previous work done related to the proposed topic for this assignment which is face mask image detection.

2.2 Main Section

It is important to know that for this assignment the existing and pre-trained model is used based on the established code online or from Kaggle whereby it is used as reference and some tuning is done needed to improve the accuracy. Hence benchmarking can be done on the performance using the datasets proposed for different algorithms by looking at the evaluation matrices and the accuracy obtained after hyper parameter tuning. But for most data science projects there will be drawbacks, limitations and biases when proposing any algorithm and therefore this assignment requires us to look into improving the bias occurs in the current datasets and for future improvements.

This part here is to discuss and review existing research done regarding face mask image detection where the face mask detection falls under the classes object detection (Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, 2019) where detection technology or detectors usually rely on deep learning than a handcrafted features. This is due to detection technology are very good and provide awesome performance. Another technique is applied where view based approach is used for face scanners with attribute formed by set of feature vectors where the feature or the structure has been revealed to have better accuracy for face or profile detection. (H. Schneiderman and T. Kanade, 2002).

Another approach proposed to use support vector machine SVM, a supervised machine learning model that solve complex classification and regression problem using where it is found that this method that uses local-kernel based is way better than global-kernel based to detect interrupted frontal faces (K. Hotta, 2007).

Another approach used MBH Boost method where it is multiclass boost up algorithm where a Multiview face studied as individual tumble classifier whereby it allocated features into various class (Y.-Y. Lin and T.-L. Liu, 2005). For expedited Multiview face recognition, they utilized the initial cascade levels, incorporating all facial markers, to estimate pose. This approach allows for the efficient measurement of variously modified facial identifiers across each scan window, employing both the initial cascade levels and all facial markers to approximate pose. (J. Yang, J. Wright, T. S. Huang, and Y. Ma, 2010). It can be said that there is a lot of research done in the past regarding to facial recognition technology.

A researcher proposed a method where they suggested convolutional networks, which are neural networks featuring at least one layer utilizing convolution rather than general matrix multiplication. (G. Mesnil, Y. Dauphin, G. Xavier et al, 2012). Moreover, there is many machine classifiers for deep learning method where one of it is ADAM optimizer. Adam is a stochastic optimization algorithm that optimizes objective functions based on adaptive estimations of lower-order moments, employing first-order gradients where this method only need less memory to run it and very efficient computation wise.

ADAM optimizers are mostly used in problems that has huge datasets and complex hyperparameter due to its unique which remains unaffected by diagonal rescaling of the gradients. Another optimizer can be used for face mask detection is ADAGRAD where the optimizer adjusts learning rates for individual parameters based on their update frequency during training, resulting in smaller updates for parameters that are updated more frequently.

PCA also known as principal component analysis can be used in face recognition or face mask detection where it is one of statistical techniques where it is used in pattern recognition and in data analysis it is used for dimensionality reduction. Hence, self-space projection is used in principal component analysis where it is a broad 1-D, one dimensional pixel vector space that comprised of 2-D, two-dimensional face images with elements of spatial function. (Khan, M., Chakraborty, S., Astya, R. and Khepra, S., 2019). It is also found that by using PCA the accuracy of the algorithm decreases when a person wear mask instead of without a mask.

There is another method that used YOLOv3 where it stands as "You Only Look Once version 3," and this method primarily a very robust object detection algorithm and mainly used in computer vision problem. One of the modifications of YOLOv3 is that it applied the CNN architecture where it allows the machine or the algorithm to do real life object detection on images, video, and pictures. The mechanism behind YOLOv3 is that the input image, pictures, or pixels are being split by the network into grid of cells and it allows the machine to predict bounding boxes and the class probability of the object within each of the cell.

Transfer Learning can also be used by adopting the pre-trained model Mobile Net by using the current solutions to solve or new problems that will occurs and a global pooling block method able to convert multi-dimensional map into 1-Dimensional vector with many characteristics (Venkateswarlu, I.B., Kakarla, J. and Prakash, S., 2020). Before that lets dive into what is the explanation behind the terms "Global Pooling Block" and "Transfer Learning".

Transfer learning can be defined as machine learning method that where a model that had been trained for a particular task will be reused again for a model for a new second related task, and instead starting again training new model it allowed to gain knowledge and applied those learned knowledge to solve new problems. Other than "Global Pooling Block" is one of the elements that most people used CNN that is effectively used for image classification problems.

There are two type of Global Pooling Block which are Global Max Pooling and Global Average Pooling which are:

1. Global Max Pooling measure the maximum value of individual feature map across most of the spatial locations.
2. Global Average Pooling on the other hand, calculate the average value of individual feature map for all of the spatial locations that make it into a singular value per feature map.

It is known that deep neural network takes input image data and captures all of the information of that said data and transfer it the information to neurons in which at the end it will give prediction about the image. Another method of CNN used is called MobileNetV2 where for this architecture it is a network model where for its basic unit it uses depth wise separable convolution. (A. G. Howard, M. Zhu, B. Chen et al, 2017)

Figure below shows the proposed framework of MobileNetV2 where MobileNetv2 utilizes an inverted residual structure, featuring residual connections between the bottleneck layers. The intermediate expansion layer employs lightweight depth-wise convolutions to filter features,

providing a source of non-linearity. In its entirety, the MobileNetV2 architecture comprises an initial fully convolutional layer with 32 filters, succeeded by 19 residual bottleneck layers.

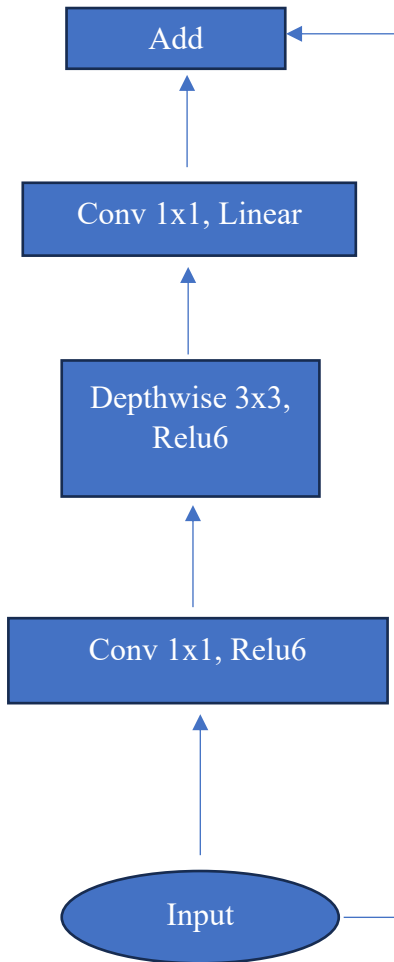


Figure 5

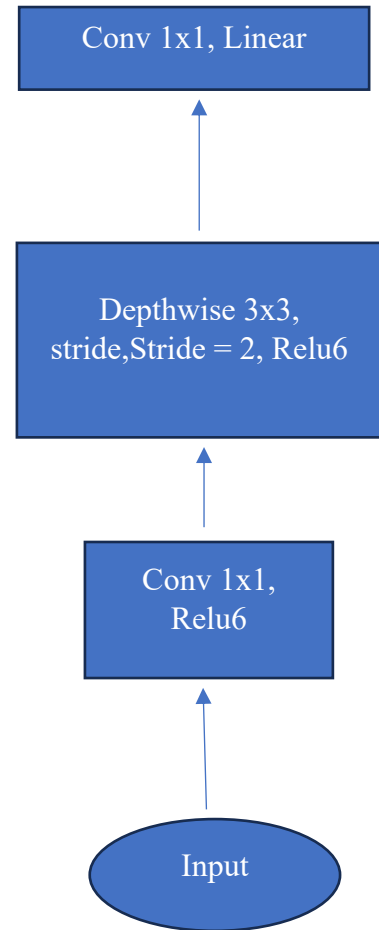


Figure 6

Another approach is to use inception model whereby using this method the inception network provide better accuracy for image classification task and segmentation problems and inception method is useful to reduce computational cost due high computational cost happened it has larger spatial fillers for its convolution neural network (C. Szegedy, V.V and sergeyloffe, j. Shlens and Z. Wohna, 2015). RetinaFaceMask method can also be used for face mask detection where the architecture behind this method it is using the MobileNet and Resnet where it is found that the accuracy is increase and it able to detect a person that wearing small mask (G. Akhila Goud, B. Hemanth, 2015).

It can be said that by using deep learning method, it is found that it has better accuracy and deep learning technique often the time suitable for clustering and classification problem as it has various layers or structures in the neurons which allow it to process multiple tasks at different layers. It can be concluded that these are some of the common deep learning approaches in image detection such as shown below.

1. MobileNetV2
2. VGG16
3. InceptionV3
4. ResNet50
5. YOLOv3

There are several researchers that applied multiple techniques of deep learning with different size of datasets where each of them obtained different results of accuracy for face mask detection.

1. A dataset called Simulated Masked Face with 1099 images are used using InceptionV3 method where image augmentation is used to enhance the train and test model for image processing and image detection. (Chowdary, G. J., Pun, N. S., Sonbhadra, S. K., & Agarwal, S, 2020). It is found out that the training obtain 100% accuracy and testing is 99.92% accuracy.
2. A dataset obtained from Kaggle and PyImageSearch which comprised of mixture of various datasets with 5521 images used SSDMMNV2 deep learning method where it make use of MobilenetV2 classifier and for the face detector it used Single Shot Multibox Detector (Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J, 2021). It is found out that the accuracy obtained is around 92.64%
3. A dataset that contained 5000 image datasets used YOLOV3 to detect people with or without face mask and to detect people face it used Haar Cascade Classifier. The accuracy obtained is 90.1% (Vinh, T. Q.,&Anh, N. T. N, 2020).
4. There is another approach where it used hybrid approach where for this one it used the combination a hybrid deep transfer learning with machine learning model. It used three datasets which are Simulated Masked Face Dataset (SMFD), Real World Masked Face Dataset (RWMFD), and Labeled Faces in the Wild Dataset (LFW) where it adopt ResNet50 deep learning method to obtain information from images dataset before going into three proposed machine classifiers which are SVM, Decision tree and logistic/linear

regression (Loey,M., Manogaran, G., Taha,M. H. N., & Khalifa, N. E. M, 2021). From this analysis it is found that SVM or Support Vector Machine has the highest accuracy compared to other machine learning classifiers where for (LFWD) dataset it has accuracy of 100%, 99.49% for (SMFD) and 99.64% for (RWMFD).

5. Another approached used MobileMetV2 but using OpenCV method for the mask detection where it used 11800 images datasets whereby the validation accuracy results is 99.8% for 800 images trained (Asif, S., Wenhui, Y., Tao, Y., Jinhai, S., & Amjad, K, 2021)
6. Another approached use the same dataset, but it used pre-trained CNN where for the face mask detection, 300 continuous layers of network linked with ResNet-50 model where the validation accuracy obtained is 99% (Sadeddin, S., 2020)

Moreover, in image detection it divided into 2 types which are the single stage detectors and two-stage or multistage detectors and typically in object detection the image in the object are usually categorized and the localization are defined by boundary called as bounding box.

1. Single stage detector

Usually, single stage detector often the time used the method of simple and straightforward regression where it used input image and later studied the class probability and the box coordinates that are bounded where some of the examples are DeepMultiBox and OverFeat method (D. Erhan, C. Szegedy, A. Toshev, D. Anguelov, 2014) (P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, 2014). YOLO deep learning used in single stage detector where although it is used in face detection it has drawbacks such as it small and low localization accuracy although it has high speed, specifically when the image dataset is quite small. (J. Redmon, S. Divvala, R. Girshick, A. Farhadi, 2016)

The YOLO technique works by network partitions an image into a grid of size $G \times G$, generating N predictions for bounding boxes within each grid. Each bounding box prediction is constrained to contain only one class, which limits the network's ability to detect smaller objects. YOLO was then enhanced with YOLOv2, introducing features such as batch normalization, a high-resolution classifier, and anchor boxes. YOLOv3 is then further builds upon YOLOv2 by making improvements in the backbone classifier, multi-scale prediction capability, and a new feature extraction network.

Other than that, there is SSD deep learning method which has much better performance than YOLO method because it has various features maps, able to do predictions in various scales and has small convolutional layers.

2. Two-stage or Multi-stage detector

On the other hand, two-stage detector for face detection used CNN as it has unique features to obtained spatial or image recognition datasets. CNN is one of the unique deep learning methods where used in processing pixel data and there is multiple type of multistage detection such as

1. CNN
2. R-CNN
3. Fast R-CNN
4. Faster R-CNN
5. Mask R-CNN

R-CNN also known as regional convolutional neural network where it is a large-scale adoption for CNN to solve localization and recognition problems (R. Girshick, J. Donahue, T. Darrell, J. Malik, 2015) where a researcher obtained great results by using benchmark datasets of VOC-2012 and ILSVRC- 2013. The mechanism behind R-CNN is that at the early stage it used selective search algorithms to extract set of object proposal and later used SVM to predict related class of object later on. Another method is Fast R-CNN which is built upon R-CNN and SPPNet, where SPPNet stands for Spatial pyramid pooling which obtained features from multiple region proposals and then it is inserted into connected layer for classification (K. He, X. Zhang, S. Ren, J. Sun, 2015).

Fast R-CNN used new layer called Region of Interest (ROI) pooling layer between shared convolutional layers which is used to fine tune the model and this method does not disturb the network configurations by concurrently train detector and the regression model. But there is drawback where the speed for detection is slow compared to single stage detector although it took the best from R-CNN and SPPNet (N.D. Nguyen, T. Do, T.D. Ngo, D.D. Le, 2020)

2.3 Problem constraint of the model proposed.

There are also limitations and biases when conducting data science projects involving deep learning especially for face mask detection where there will be issues arise such as lack of data and it's hard to obtain good accuracy. Another issue is maybe due to collecting the data especially the images data where the inability to collect reliable data is one of the limitations of the assignment.

Hence, libraries like Keras where it is human centric API, it allowed to supply with a very straightforward and API as it is not a machine or computer centric one. Hence it with this API, it can reduce the number of users activities necessary with common use case and giving much clearer and actionable error signals. Keras is able to lower down the cognitive load and comprise implementations of neural networks, activation functions, layers objectives and optimizers.

However, Keras does not entirely able to solve distorted noises in the model as it lacks flexibility in certain complex customization problems, and it is not efficient enough to deploy model in production stage and sometimes there is issues with scalability. There are also issues with large-scale deployments or specialized hardware configurations where it can affect the performance of the model.

Hence Tensorflow can be used to mitigate this problem where it is an open-source platform or dataflow which can be used solved various range of problem as it is a differentiable programming software framework and tool used by researchers, data scientists and statisticians. It allows users to import mathematical library that applied machine learning applications such as neural networks and it is mostly used for research and production purposes.

2.4 Scope of problem domain

The domain problem of this assignment is healthcare because the topic selected is about face mask detection where the main goal is to prevent or curb the spread of covid 19 infection in general. The mechanism behind face mask detection using deep learning approach particularly CNN used object detection or facial recognition technology, and this shows that how significance is the machine learning technology and its application as it can be used in many different area and field.

This shows that computer technology is able to significantly improve human life and make people's life easier for daily life routine. Another similar application to the healthcare domain can be seen on image scanning especially for medical diagnostics such as medical imaging where object detection is used to detect any brain tumor from the scanned images of the patient brain images. Deep learning is utilized in a lot of MRI scans to diagnose any possible tumor, and this also can be applied in other part as well especially to detect diabetic retinopathy, ultrasound detection of breast nodules and identifying early case of Alzheimer's in old people.

This shows that the field of mathematics and computer science can help medical professionals, especially doctors and healthcare workers to prevent the spread of fatal diseases especially in this context covid 19. Image recognition allow to revolutionize the healthcare industry for example at the entry checking point in a hospital, camera system allow to identify people face with and without face mask and hence the information gathered can be used by the hospital to validate the importance of face mask as people without face mask are prone to covid 19 disease.

2.5 RELATED WORK

2.6 Comparison of Related Works

This part here is to summarize the research paper about face mask image detection where it shows the various article and research paper published where the table below shows how different authors proposed different method and machine learning method with performance and accuracy. Below shows the details of the reviewed research papers, journal, and articles.

Table 7

No	Authors	Datasets	Techniques	Accuracy/Performance	Problem solved	Hyperparameter	Strength/limitations	Future work
1	Chowdary, et al., 2020	-	Data augmentation InceptionV3	Training (99.9%) Testing(100%)	With mask Without face mask	80 epochs with 42 steps each	Applied in surveillance camera. Improper face mask is harder to detect	Implement facial recognition
2	Ejaz et al. [6]	ORL face dataset & additional images	PCA (Principal Component Analysis)	With mask (68.75%) Without mask(96.25%)	-	-	This method works for people with normal face and hence people with face mask does not work	-
3	Loey, et al., 2021a	-	Hybrid machine learning Classification problem Resnet-50	The accuracy achieved for SVM is 99.64% for testing	With mask Without face mask	-	People with improper face masks can't be detected. Very computationally expensive	-

			ML algorithm – SVM highest accuracy, ensemble, and decision tree					
4	Li et al. [7]	Wider face, CelebA & FDDB	YOLOv3	93.9%	-	-	This approach only used synthetic dataset not real time dataset	-
5	Nagrath, et al., 2021	-	SSDMNV2 SSD (Single Shot Detector) ResNet10 MobileNetV2 Data augmentation	Accuracy achieved 92.64% F1 score achieved 93% If no data augmentation the accuracy achieved is 87.51%	With mask Without face mask	Optimizer used is ADAM Epochs :100 The weight used is Imagenet	During real time detection it need less resources It's quite complex	
6	Das et al. [8]	FMD dataset (2 datasets)	Keras Tensorflow OpenCV	95.77% 94.58%	-	-	The datasets are too small. When people face in motion it fail to detect the movement of the face	-
7	Jiang, et al., 2021	-	Used Squeeze and Excitation (SE) YOLOv3 Backbone: YOLOv3 + SE block	The image size is 416x416 $AP_{50} = 98.6\%$ $AP_{75} = 86.3\%$	With mask Without face mask Improper face mask	The initial learning rate (ILR) obtained is 0.0005 for 100 epochs The optimizer used is ADAM	Can be implemented in Raspberry Pi	Use SE-YOLOv3 on lightweight device

			Using data augmentation where it is mixed up augmentation and image.	$mAP = 71.9\%$ The detection time obtained is 43.2ms	Blockade on the face image dataset can be detected	Depletion of 0.9 for ILR/100 epochs Ther lose function used is categorical entropy. Batch size is 1		
8	Loey et al. [9]	RMFRD, SMFD and LFW	ResBNet5o with the combination of machine learning algorithm	Testing accuracy for RMFD dataset is 99.64%, for SMFD is 99.49% and LFW is 100%	-	-	There is issues with the RMFD dataset where where in the image datasets there is some mistakes in labelling the people who wear mask or no	-
9	Wang, Zhao and Chen, 2021	-	It used 2-stages method such as Faster R-CNN with InceptionV2 For verification it used BLS	F1 score is 94.19%	With mask Without mask Mask image with only impending chin Mask image that cover chin and mouth part	-	Cant detect small object The classifier cant detect image of face that has blockade or face mask image with goggles	The lack of features problem can be solved with image super resolution method

10	Khandelwal et al. [13]	The datasets used is primary datasets where it contained the 4225 images comprised of 1900 for face mask and 2300 for without face mask. The rest of 25 images	MobileNetV2	Accuracy is 97.6%	-	-	The machine algorithm could not detect if the height of the camera is 10 feet and people with slightly hidden faces can't be detected	-
----	------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------	-------------------	---	---	---------------------------------------------------------------------------------------------------------------------------------------	---

2.7 Analysis on Related Works

This part here is to analyze the related works where evaluation needs to be done on the where the experiments and results of this related works will be compared to the results and the model or deep learning techniques proposed for this assignment later on. The summary part here is to justified the model proposal in relation to what was done by others.

Table 8

No	Author	Reference list
1	Chowdary, et al., 2020	Chowdary, G.J., Punna, 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.
2	Ejaz et al. [6]	M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker, "Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition," May 2019, doi: 10.1109/ICASERT.2019.8934543
3	Loey, et al., 2021a	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.10828
4	Li et al. [7]	C. Li, R. Wang, J. Li, and L. Fei, "Face detection based on YOLOv3," in Advances in Intelligent Systems and Computing, 2020, vol. 1031 AISC, pp. 277–284, doi: 10.1007/978-981-13-9406-5_34.
5	Nagrath, et al., 2021	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.
6	Das et al. [8]	A. Das, M. Wasif Ansari, and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," Dec. 2020, doi: 10.1109/INDICON49873.2020.9342585
7	Jiang, et al., 2021	Jiang, X., Gao, T., Zhu, Z. and Zhao, Y., 2021. Real-Time Face Mask Detection Method Based on YOLOv3. Electronics, 10(7), p.837.
8	Loey et al. [9]	M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection

		in the era of the COVID-19 pandemic,” Meas. J. Int. Meas. Confed., vol. 167, p. 108288, Jan. 2021, doi: 10.1016/j.measurement.2020.108288.
9	Wang, Zhao, and Chen, 2021	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.
10	Khandelwal et al. [13]	P. Khandelwal, A. Khandelwal, S. Agarwal, D. Thomas, N. Xavier, and A. Raghuraman, “Using Computer Vision to enhance Safety of Workforce in Manufacturing in a Post COVID World,” May 2020, Accessed: Dec. 20, 2021. [Online]. Available: https://arxiv.org/abs/2005.05287v2 .

From table of summary of related work above shows the list of authors with its reference list where it is found out that some of the common machine learning techniques used are InceptionV3, YOLOv3, YOLOv2, ResNet50, and MobiNetV2. It is known also that some of the limitations on the research work done about the face mask image detection is that some of the machine learning classifiers not able to detect accurately and hence some of the results of the accuracy, precision and f1-score is relatively low. Most of the common issues with this proposed topic assignment is that most of the datasets extracted and used can't be detected properly by the machine learning classifiers where the problems exist such as;

1. Cannot detect people with improper face mask.
2. High computational cost.
3. Need more training time.
4. The image dataset that has occlusion can't be properly detected.

Hence for this assignment some of these issues might be addressed especially on the hyperparameter part where batch size, activation function, and different optimization can be altered to give better results and the accuracy and also by tuning the epoch. This part will be further explained on assignment part 2 under component 4 where much more detailed explanation from importing the datasets until to the discussion part.

Component 3

3.0 DEEP LEARNING ARCHITECTURES

3.1 CNN

This part here is just to have brief understanding of CNN where it stands for Convolutional Neural Network, and it is mainly designed to identify visual or optical patterns from any extracted pixels. Figure below shows the example of how CNN works to identify whether the machine algorithm can identify the input picture as rose or orchid.

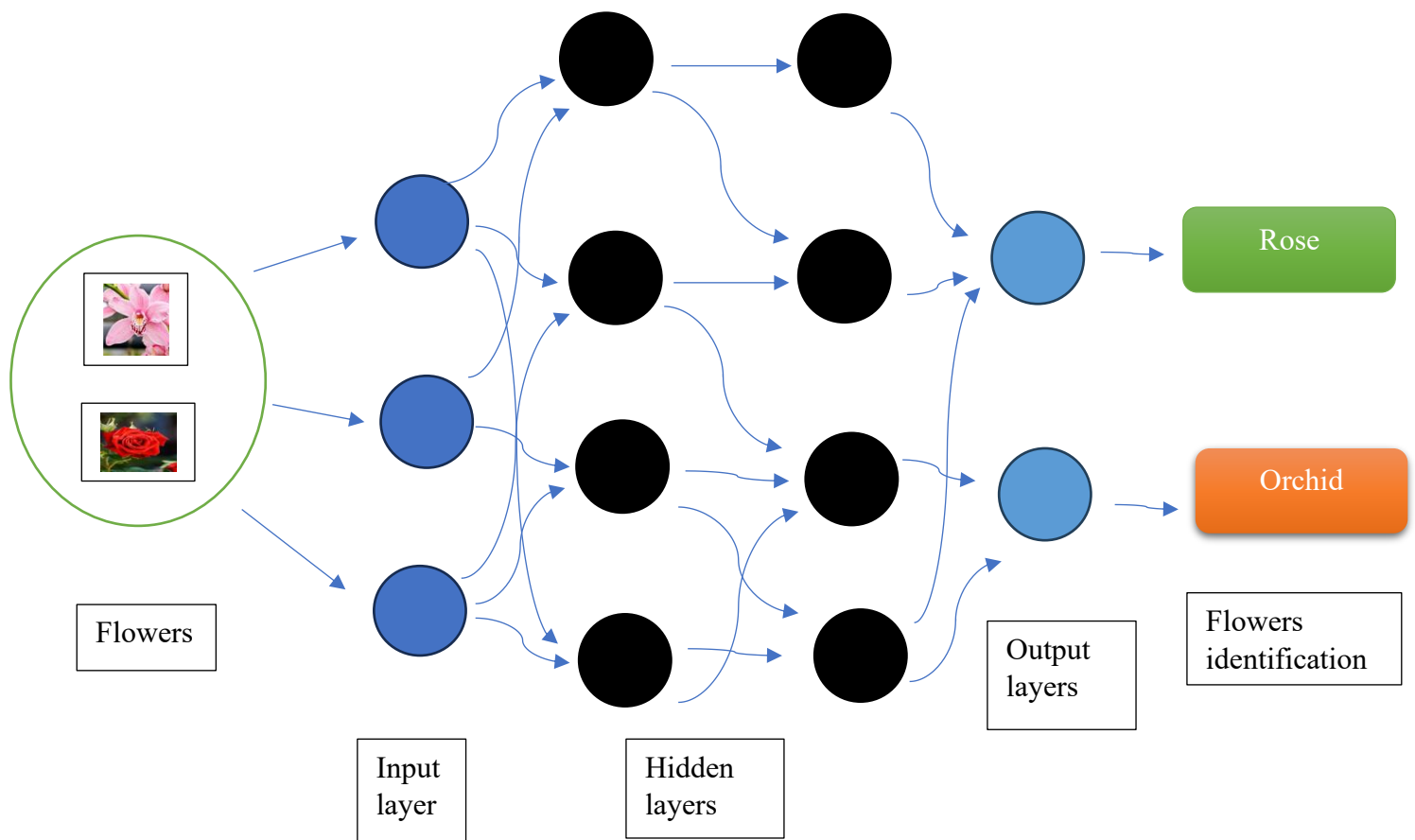


Figure 5

CNN usually visualize image data in grid topology as shown below where figure below shows that for example in CNN, every input image is in the form arrays of pixel values where if example real image of digit 8 is an input, it will transform into array form and eventually as matrix of numbers.

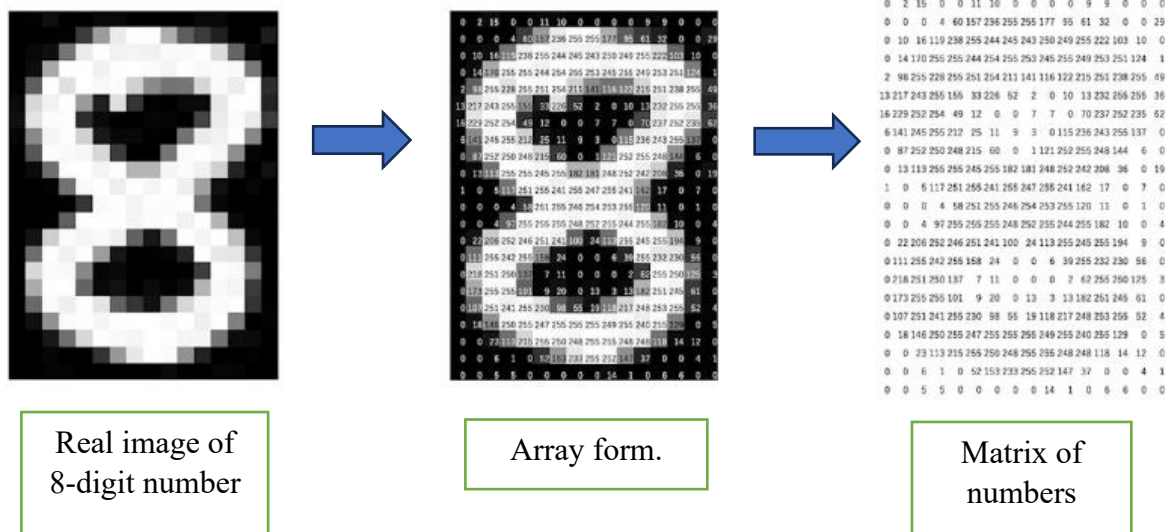


Figure 6

CNN can also be called ConvNet where convolutional layer is one of its building blocks and it usually contained set of filters called kernels. The size of the filters tends to be smaller than the actual image and every individual filter convolves with the image, and it creates an activation map. Complex models can be created by stacking the convolutional layers where over the time it can learn more features from the input images. On the other hand, kernels, and filters function as highlighters of a certain feature in the image where it creates activation function map of the feature in an image. Below shows the feature extraction method where it is executed by convolving the image with filter where the filter can be detected vertically or horizontally.

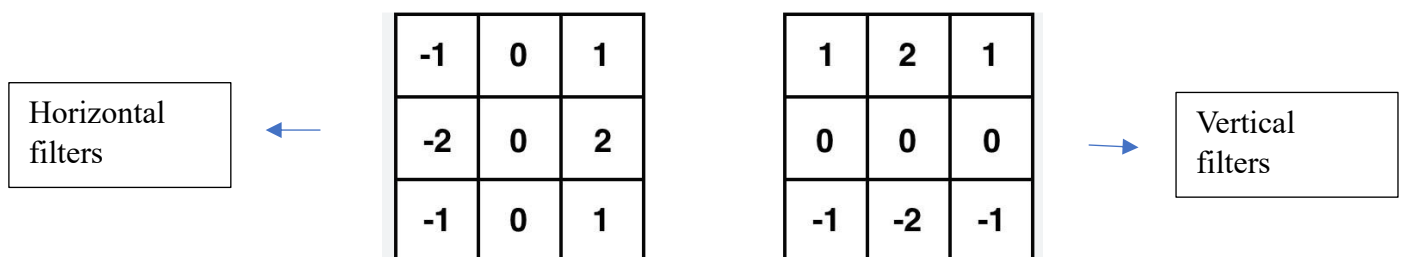


Figure 7

The horizontal filter works by identifying horizontal patterns in an input data where for example horizontal filter able to detect horizontal patterns or edges in an image that lay flat

after the filter convoluted with input image. For vertical filter it is for object or pattern that stand upright.

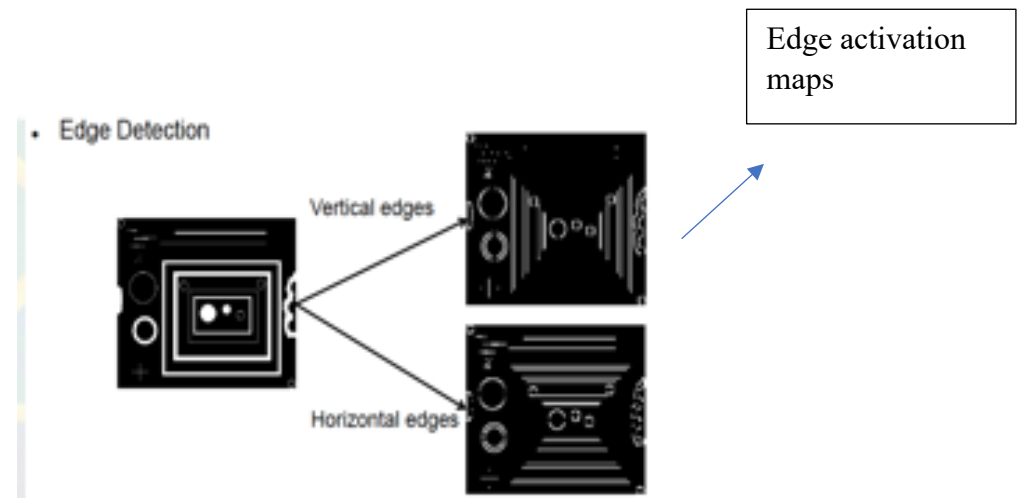


Figure 8

Another concept of CNN is padding where in CNN, convolutional neural networks, it is to add extra or additional pixels at the input image before convolution where the main goal is to control the spatial dimensions of the output volume. Padding helps to prevent the reduction of spatial dimensions after each convolutional layer as it can prevent shrinking effect as network advance through layers.

There are 2 type of padding such as

1. Valid padding

It is also known as no padding where the filter only applied to valid pixels of the input and thus input feature is bigger than the feature map due to not enough pixels for the filter to cover.

2. Same padding

This happened when the output dimension is the same as input and this can be obtained by adding zeros around the input image symmetrically and hence it preserved the spatial dimensions of the input.

Below shows the same padding method where for example if the image has size of 28x28, and thus by adding padding of 1 on each side the input size will be 30x30. Hence, by adding padding the output dimension is the same as input.

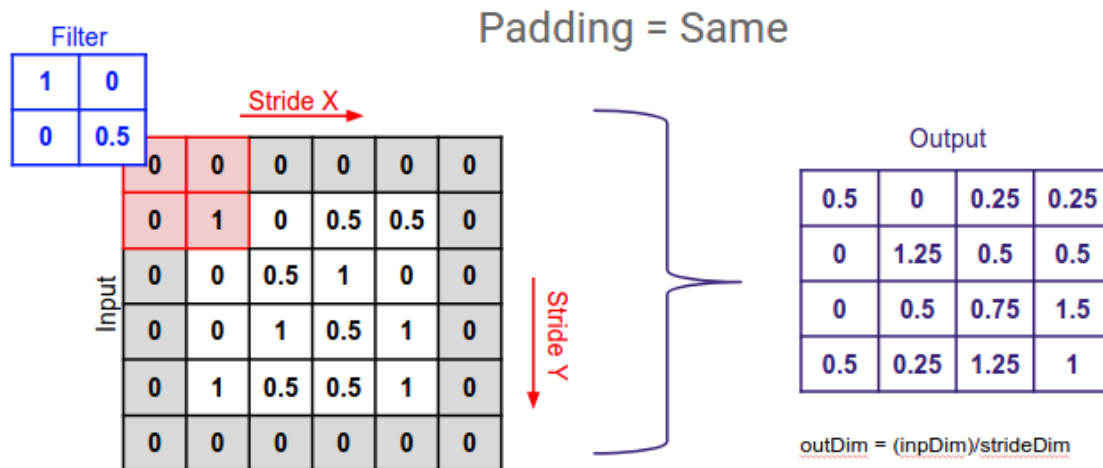


Figure 9

Kernels on the other hand are one of the fundamental components in CNN where it is a small sized matrix that purposely used for feature extraction and typically the more the number of kernels is it can detect more features. In the early layers it used less and fewer kernels as it detects simple features whereby more kernels are used for later layers as it detects much more complex features. There are variations in kernel size where some of the common sizes are 5x5 or 3x3 and larger kernel indicates more processing time.

Stride in CNN indicates the step size where the kernel moves across the input or feature map and stride are used because sometime all the information or data is not required to be captured hence , some of the neighboring cells can be skipped. There are 2 types of strides which are;

1. Stride of 1

For stride 1, dense convolution happened because the kernel moves one pixel at a time and the kernel moves along the vertical direction and horizontal of the input volume . Below shows the figure of stride 1 and this is one of the common methods in CNN.

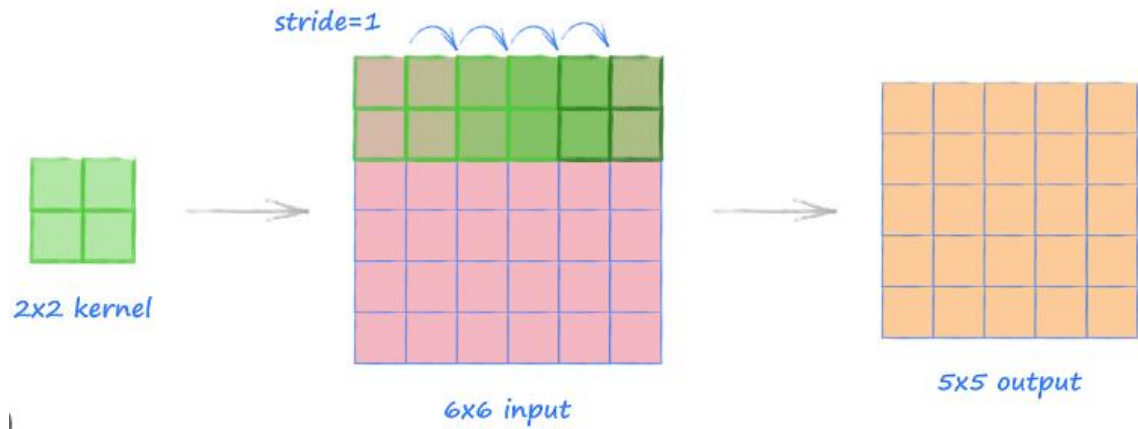


Figure 10

2. Stride of 2

On the other hand, for stride 2 at each dimension the kernel would move 2 pixels at a time and by having larger size of stride, it able to cut down the spatial dimensions of the output feature map compared to input volume. Figure below shows stride =2

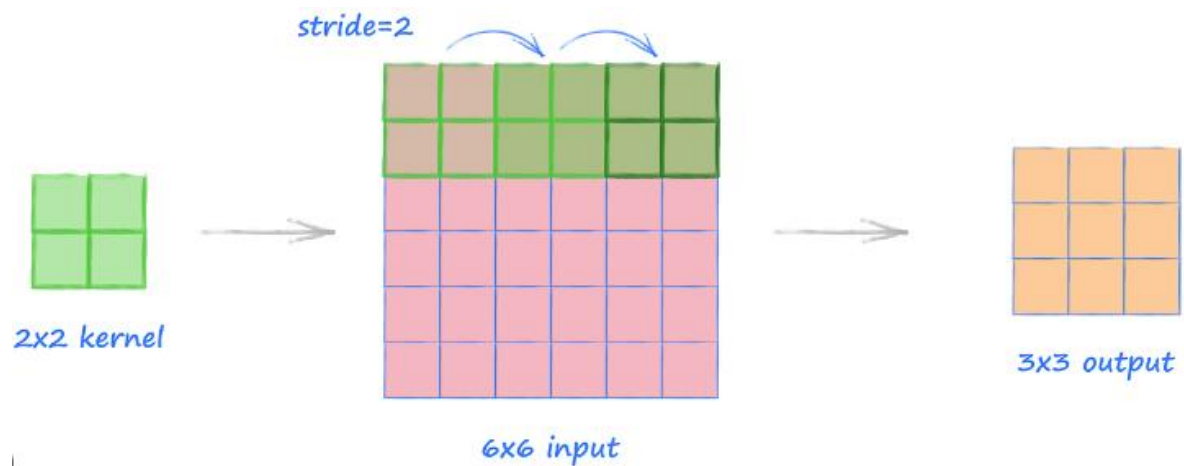


Figure 11

Pooling layer is commonly used in CNN where it is a type of layer that is implemented to cut down the spatial dimensions of feature map or input volume meanwhile maintaining the essential information. The characteristic of pooling layer is that it allows to reduce the spatial size of the input, it requires less memory, training time become faster, able to cut down number of parameters and much easier to process.

There are 2 types of pooling layer which are;

1. Max pooling

It takes the maximum amount from each feature map, and it is very effective to maintain the important features of individual region of input feature map.

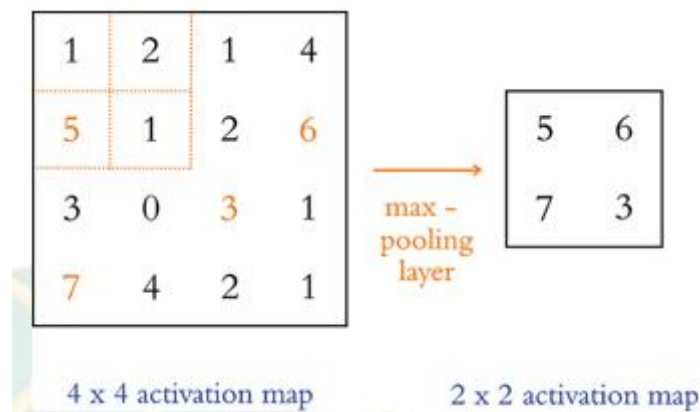


Figure 12

2. Average pooling

It takes the average value for individual region of the input feature map and average value of all pixels in that region.

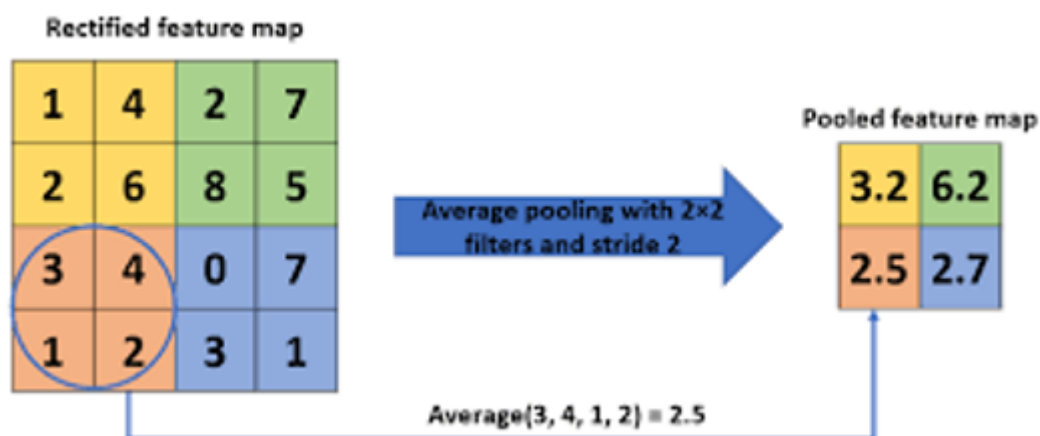


Figure 13

Flatten layer allows to collapse various dimensional array down to one dimension where for example it can convert 3x3 image into 9x1 as shown in figure below. In CNN, flatten is considered as measure of converting multi-dimensional output of the previous pooling and convolutional layers into a one-dimensional vector space.

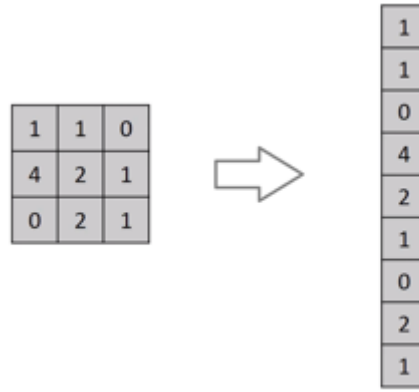


Figure 14

Hence it can be deduced that CNN architecture consisted of convolutional, pooling, activation, dropout, and fully connected layer.

3.2 Pre-Trained Models and Transfer Learning

For face mask image detection, deep learning-based transfer learning will be used where it is known that it involved in taking pre-trained neural networks where large datasets sometimes are trained using ImageNet, Resnet, MobileV2, VGG, BERT, and other deep learning technique and fine-tuning is done on it on a new task or dataset. There are three steps of transfer learning which are pre- training where CNN are trained on a selected dataset in this case face mask image dataset and then transfer where model or information from pre-training transferred to a new task and last one is fine-tuning where the weight or parameters can be adjusted to a new dataset.

The benefits of doing transfer learning are that the training time can be reduced where with pre-trained model, transfer learning allowed reduces the amount of labeled data and hence the computational cost run new trained model would be lower. Domain specific knowledge can be transferred easily whereby it helps to improve performance on new tasks.

There are many pre-trained models such as MobileNet, ResNet50 that trained 14 million images from ImageNet dataset (G.J. Chowdary, N.S. Pun, S.K. Sonbhadra, S. Agarwal, 2020) for this one research paper where the model proposed used ReNet50 for the pre-trained model. For this assignment the pre-trained model used will be further explained in component 4 and component 5. This approach adds five new layers for the chosen ResNet50 where the new layer included average pool layer with size of 5x5, ReLU layer of 128, flattening layer, dropout out of

0.5 and softmax activation function using binary classification. Figure below shows the fine tuning of the author pre-trained model using ResNet50.

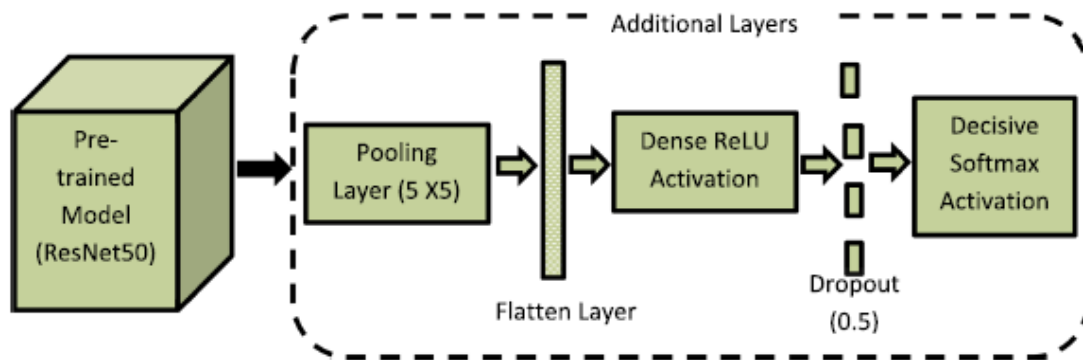


Figure 15

Component 4

4.0 METHODOLOGY

4.1 Experimental design

Experimental design is one of the key components where usually it consists of workflow of data science project where consist of few components such as business understanding, data understanding, data pre- processing, modelling, evaluation and at the end is the outcome will tell if the proposed model used reliable or no.

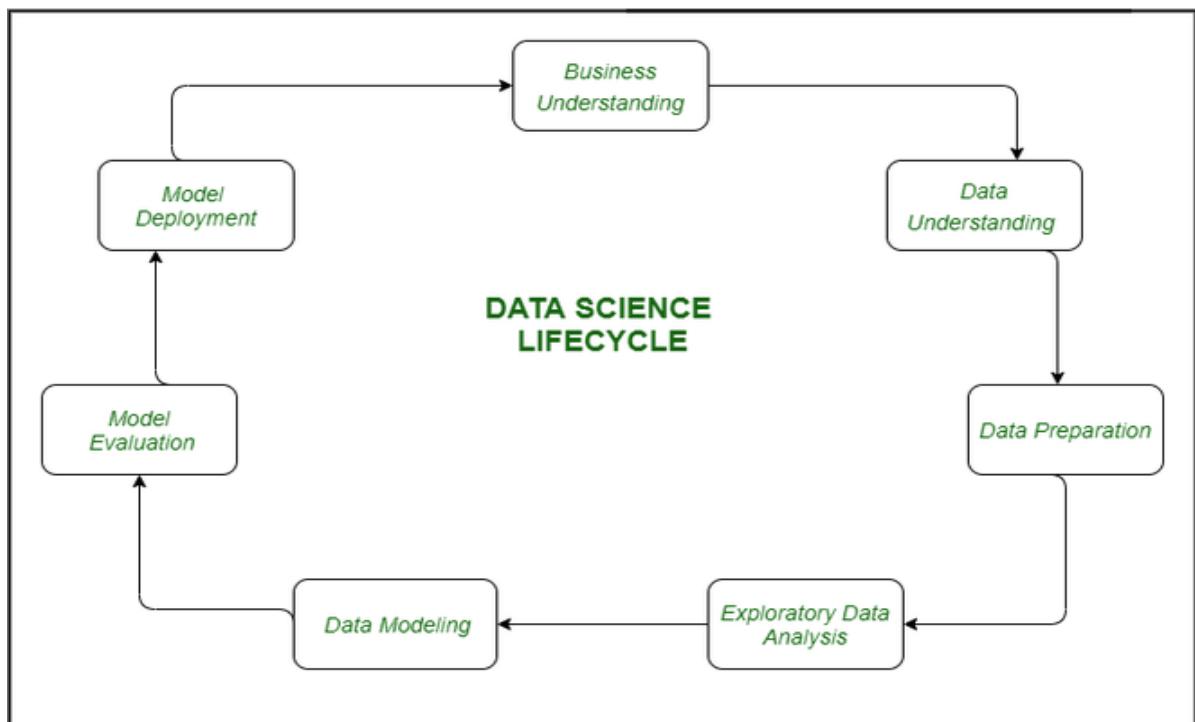


Figure 16

Figure above shows the data science life cycle or also known as CRISP-DM where it is one of crucial workflow in any data science project. On the business perspective, gather information and insights is very crucial to make informed decisions and prediction based on the past data collected. The rise of big data and the improvement been made recently with artificial intelligence more and more data need to be gathered and mined to be analyse using statistical software and to draw conclusion from the data gathered.

4.2 Business understanding

It is known that on the business perspective, object detection is very crucial for people where it is a type of computer vision task where it will be trained to identify if the image of the person wearing a mask or no. To understand the methodology of face mask image detection the problem understanding of face mask image detection start with datasets collection where in this case it used secondary dataset. A secondary dataset is a type of dataset where it's already collected by someone else other than a primary user. Next is training and testing of the dataset and eventually deployment phase.

4.3 Data understanding

It is known that the dataset used for this assignment is an image dataset which falls under the category of computer vision that is mostly used in object or image classification. Data understanding is important as for example textual datasets about fake news detection in the format of csv or excel file are suitable for sentiment analysis which fall under natural language processing. Hence, knowing the type of dataset used is important before proceeding into later stage from data understanding until deployment.

4.4 Dataset collection/ description

Kaggle is one of the main sources of the dataset where the dataset taken from Kaggle is called “Face Mask Detection Dataset” where the dataset is a secondary dataset type. The overview of the dataset is that it is known that recently 3 years ago from 2020 until 2021, the world undergoes worldwide lockdown due covid 19 outbreak that causes people to stay home to curb the spread of covid 19 diseases. Hence, wearing a face mask is mandatory to go outside and thus deep learning approach become more popular to detect people wearing mask or no. The model proposed by the secondary dataset author is that a model had been trained on 7553 images with 3 colour channels and their model used custom CNN architecture model with training accuracy of 94% and validation accuracy of 96%.

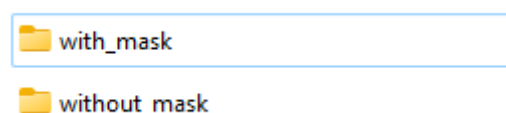


Figure 17

Figure above shows the content of the dataset where the dataset consists of 7553 RGB images in 2 folders which are labeled with_mask and without_mask. The images with faces mask are

3725 and without face mask are 3828 and figure below shows the image of the datasets used for this assignment which will be further explained in component 5.



Figure 18

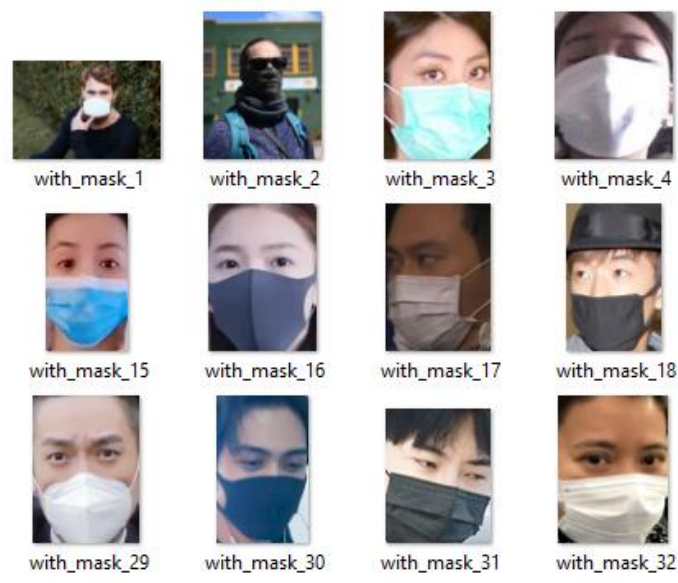


Figure 19

Below shows the methodology of face mask image detection where it shows the path or workflow of image detection model. In the load model there are different deep learning techniques such as Resnet50, MobileNet, Inception and DenseNet but the validation part will be further discussed in component 5.

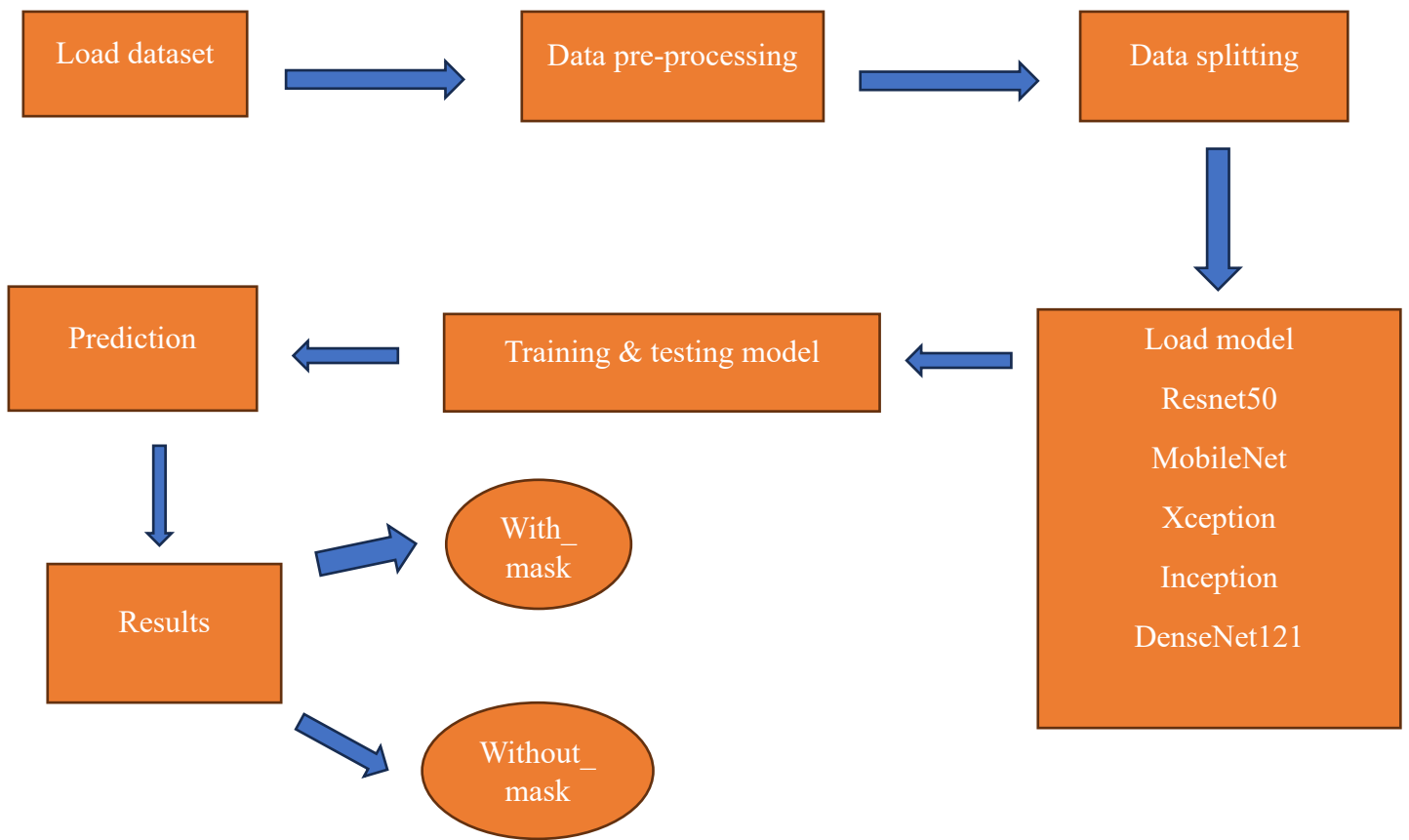


Figure 20

But there are drawbacks and challenges faced using face mask image detection datasets such as;

1. Different size and shape of face mask such as N95 mask, surgical mask need to be trained with different range of machine learning model.
2. Lighting conditions able to affect the display of the face mask.
3. There is occlusion and disruptions on the face mask image will cause the machine learning algorithm to have poor performance.
4. Face orientation and expression of the people cause it not to be able detect whether a person is wearing a face mask or no.

4.4.1 Validation

Overall, in face mask image detection, validation is very important to evaluate and assess the accuracy and capability of the trained models. One of significance of validation is that it allows to assess the model performance when a model undergoes training, as it tries to fit the training data and see and test how well it performs on a new dataset. Other than that, validation is

necessary to prevent overfitting where overfitting happened when the model is able to capture the noise and underlying pattern of the data and usually the model has low bias and high variance. Bias is defined as difference between the average prediction of the model and the correct or true value. Variance on the other hand, is defined as the variability of the model prediction for a given data point or a value which enables to tell the spread of the data.

One of the other aspects of validation is hyperparameter tuning where some of the methods used are batch size, epochs, learning rate where during training of the dataset, some of these hyperparameter are fine tuned to optimize the performance. In face mask image detection, model selection is crucial as experimentation can be done with another type of algorithm to eventually seek the best architecture for the task.

4.4.2 Training and testing datasets.

Train and test dataset is one of the essential roles in face mask image detection where it is known that the proposed deep learning or machine learning model is trained from the training dataset. For this assignment, the training dataset consisted of an image with mask and image without mask in which during model training, the model learns to distinguish between a masked face and unmasked face. Typically, much a quality training data, a sensible size of dataset and data that had been pre-processed and cleaned will result in optimal model performance and hence choosing good quality dataset is very crucial for training and testing. Testing dataset on the other hand, is to use to assess the final performance of the trained model and used to equip unbiased evaluation on the final performance of the proposed model. Typically, data splitting is done based on the percentage chosen, for example 80/20, and 70/30 where 80 is the training set and 20 is the testing set. To assess the model performance the performance metrics, for example precision, accuracy, f1-score, and recall are used.

4.4.3 Exploratory data analysis

For the exploratory data analysis or EDA, it basically involved the patterns and characteristics of the datasets used in this assignment. So, for this assignment, the folder with mask contains 3725 images datasets while for folder without mask contain 3828 images datasets. For EDA, data visualization can be done on the datasets to study the class distributions, the image characteristics, and the correlation analysis.



with_mask_3725

Figure 21



without_mask_38
28

Figure 22

Below shows the class distribution of the training dataset where it is in the format of pie chart.

```
# Directory containing your dataset
dataset_dir = "/drive/My Drive/COLABNOTEBOOK/ODL/Data/data"

# Count the number of images in each category
categories = os.listdir(dataset_dir)
image_counts = []

for category in categories:
    category_dir = os.path.join(dataset_dir, category)
    if os.path.isdir(category_dir):
        num_images = len(os.listdir(category_dir))
        image_counts.append(num_images)

# Plot the pie chart
plt.figure(figsize=(4, 4))
plt.pie(image_counts, labels=categories, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Images with and without Mask')
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Distribution of Images with and without Mask

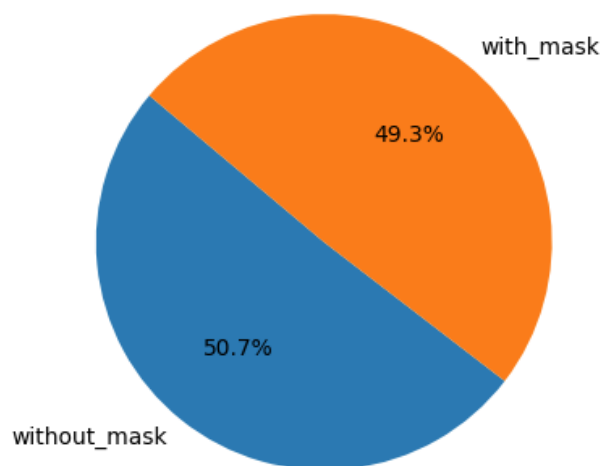


Figure 23

It shows that at 50.7% of the image datasets are without face mask and 49.3% of the image datasets are with face mask. This suggests that population without face mask are higher than with face mask.

4.4.4 Modelling

For the modelling part there are quite a few selected neural networks method such as Mobilenetv2, VGG19, , Inception, Xception and Pre-trained Resnet50V2 with transfer learning model where each of these neural networks has its own unique capabilities and features. For example, Mobilenetv2 exclusively implemented depthwise separable convolutions where it split into depthwise and pointwise convolution. Moreover, it also has inverted residuals with linear bottlenecks where it helps efficiently acquired complex features.

Other than that, VGG19 is one of them is a deep neural network that in the family of Visual Geometry Group and it comprised of 16 convolutional layers that grouped in block and each individual block has convolutional layers and also max pooling layer. The uniform architecture of VGG19 is that it adopts 3x3 filters with stride of 1 and same padding and also with ReLU activation functions. Later, in assignment part 2, most of neural network architecture can be improved using dropout, regularizers, early stopping and hyperparameter tuning.

Another pre-trained ResNet50v2 model transfer learning also can be implemented in the model implementation where it has been significantly pre-trained on large datasets such as ImageNet where it has millions of labeled image dataset. By fine-tuning the pre-trained model on small dataset, it can be used later on huge dataset. Other neural networks technique is Inception created by Google where the unique features of it is it has multi-branch architecture, and another one is Xception where it took inspiration from Inception but replaces the standard convolutional layers with depth wise separable convolutions.

4.4.5 Evaluation

Evaluation is done on the test dataset where the metrics performance as shown below where it shows the formula f1-score, precision, recall and accuracy.

$$F1\ Score = \frac{2 * Precision * recall}{Precision + recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy also can be known as the value sum of correct predictions over the total value of all predictions.

		Real Label	
		Positive	Negative
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 24

4.5 Deployment

Deployment in face mask image detection is usually done by taking the trained model and insert or integrate it into web applications or mobile applications where it can identify the image dataset with face mask or no. But in this assignment deployment will not be done as it's only covered until model implementation where hyperparameter tuning is done on google colab which will be further discuss in model implementation in component 5 (assignment part 2).

Assignment part 2

Component 5

5.0 MODEL IMPLEMENTATION

5.1 Introduction

This part is for assignment part 2 where in model implementation the main goal is to document all of the code snippets done for this assignment. The code snippets are done using Google Colab where Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. Based on the instructions given. Model implementation starts with data pre-processing, model initialization, training, and model evaluation.

5.2 The Importance of Good Model Implementation to Ensure Accurate and Reliable Results

It is known that the significant of proper and model implementation for face mask image detection can be shown as below.

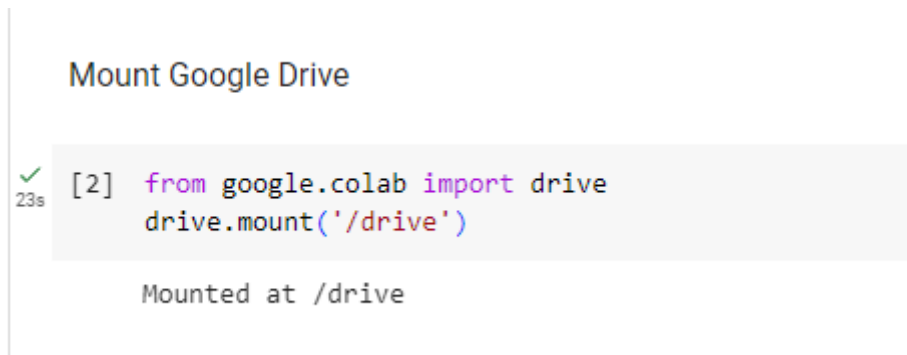
Table 9

The importance	Explanation
Model performance	A good model performance will result in better accuracy for face mask image detection
Training quality	The model capability of seeing unseen or unpredicted data is being largely influenced by data pre-processing, hyperparameter and data augmentation.
Robustness	Variations in the input images for example face expressions, lighting and angles where be able to be handled by good model implementation
Efficiency	For example, face mask image detection usually deployed to make sure crowds wearing mask for example in an entry point of a premises or building wearing mask, thus good implementation model ensures fast processing in real time

Interpretability	A well robust and good implementation model minimize errors and bias as it allowed easier debugging of the codes
Scalability	A good model implementation is very adaptable and scalable as one model deploy can be used in other environment and also it can be adapted with different datasets.
Maintainability	A good model implementation also ensures a good code maintenance and collaboration with developers as having latest model implementation is essential for better integration system.

5.3 Python in Google Colab Setup

This part here is to discuss the setting of Google Colab where Colab is a free Jupyter Notebook environment that operate in the cloud. Google colab can be run in web browsers and can be run with the access of internet with the main purpose of it is to study coding about machine learning and artificial intelligence. It allows easy collaboration with other users as well because python code can be run with easy configurations and beginner friendly for entry level coders. It is also known that the python library is already predefined and updated automatically and below shows the start of the code implementation part of this assignment part 2 where google drive is mounted.



```

Mount Google Drive

✓ 23s [2] from google.colab import drive
      drive.mount('/drive')

Mounted at /drive

```

Figure 25

5.4 File Location

Below shows the file location of the datasets of the with_mask and without_mask where it is stored in drive>Mydrive>COLABNOTEBOOK>ODL>Data>data>with_mask, without_mask, Face_mask_1.zip. Face mask 1 is the colab file for one of the network architectures of the neural network proposed for this assignment and there are other 4 neural networks proposed but will be further discussed and comparison will be made with each one of the architecture proposed. The library in colab acts as a shared information and resources

where it helps to collaborate work in much safer and secured environments as it is cloud shared networks where users can edit online on website with internet access.

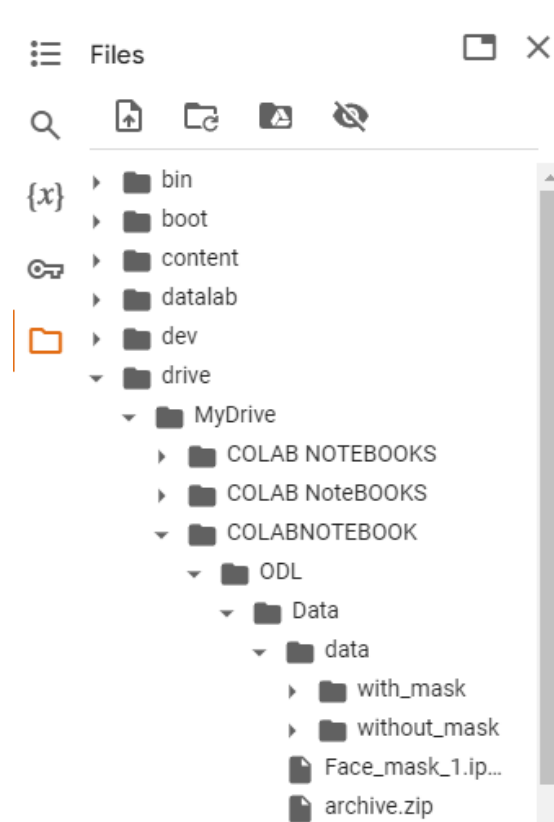


Figure 26

5.5 The Neural Network architecture & typical workflow of the coding step

It is known that five neural networks will be used and the usual step of coding in Google Colab starts with importing the libraries or dependencies, importing the datasets, data exploration, model building, hyperparameter tuning where it includes the following method to improve the accuracy of the model building such as

1. Dropout
2. Regularizers
3. Early stopping
4. Tune the batch and epoch size

The last one is plotting the graph where it shows the train and test accuracy and also train and test loss.

5.6 CNN

This part here is to showcase the coding step and explanation of the first CNN architecture proposed.

Import Dependencies

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
```

Import the datasets

```
[ ] with_mask_files = os.listdir('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])

['with_mask_3457.jpg', 'with_mask_346.jpg', 'with_mask_3452.jpg', 'with_mask_3450.jpg', 'with_mask_3472.jpg']
['with_mask_1646.jpg', 'with_mask_1656.jpg', 'with_mask_1636.jpg', 'with_mask_1643.jpg', 'with_mask_1635.jpg']
```

```
without_mask_files = os.listdir('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])

['without_mask_3535.jpg', 'without_mask_3558.jpg', 'without_mask_3540.jpg', 'without_mask_3561.jpg', 'without_mask_3557.jpg']
['without_mask_1703.jpg', 'without_mask_1734.jpg', 'without_mask_1709.jpg', 'without_mask_1730.jpg', 'without_mask_1723.jpg']
```

Exploring the datasets

```
[ ] print('Number of with mask images:', len(with_mask_files))
print('Number of without mask images:', len(without_mask_files))
```

```
Number of with mask images: 3725
Number of without mask images: 3828
```

Figure 27

Creating label for the two class of images

with mask = 1 without mask = 0

```
[ ] #create the labels

with_mask_labels = [1]*3725

without_mask_labels = [0]*3828

print(with_mask_labels[0:5])
print(without_mask_labels[0:5])

[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
[ ] print(len(with_mask_labels))
print(len(without_mask_labels))

3725
3828
```

```
[ ] labels = with_mask_labels + without_mask_labels
print(len(labels))
print(labels[0:5])
print(labels[-5:])

7553
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

Figure 28

Figure above shows that for this architecture it starts with importing the dependencies and later import the datasets where with mask files the path and without mask files is.

```
'/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/with_mask'
'/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/without_mask'
```

Data exploration is done where there is 3725 number of photos with face mask and 3828 number of photos without mask.

Figure on the left shows the step to create label where with mask it labelled as 1 and without face mask labelled as 0

Displaying images

```
# displaying with mask image
img = mpimg.imread('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/with_mask/with_mask_2197.jpg')
imgplot = plt.imshow(img)
plt.show()
```



Figure 29

```
img = mpimg.imread('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/without_mask/without_mask_20.jpg')
imgplot = plt.imshow(img)
plt.show()
```

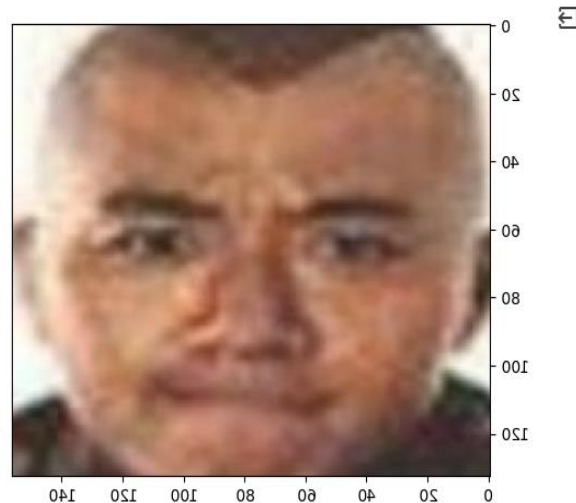


Figure 30

Both of the figures above show the step to display images of the photo dataset where it can display the images of people wearing a face mask or no. Any random photo number can be chosen based on the figure above photo number 2197 is selected for with mask and for without face mask it is number 20.

Image processing

Resize the image

convert images to numpy arrays

```
[ ] #convert images to numpy arrays

with_mask_path = '/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/with_mask/'
data = []
for img_file in with_mask_files:
    image = Image.open(with_mask_path + img_file)
    image = image.resize((128,128))
    image = image.convert('RGB')
    image = np.array(image)
    data.append(image)

without_mask_path = '/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data/without_mask/'

for img_file in without_mask_files:
    image = Image.open(without_mask_path + img_file)
    image = image.resize((128,128))
    image = image.convert('RGB')
    image = np.array(image)
    data.append(image)

/usr/local/lib/python3.10/dist-packages/PIL/Image.py:996: UserWarning: Palette images with Transparency expressed in bytes should be converted to RGBA images
  warnings.warn(
```

Figure 31

```
[ ] type(data)

list

[ ] len(data)

7553

▶ data[0]

ndarray (128, 128, 3) hide data
array([[ 123, 125, 112],
       [ 134, 134, 122],
       [ 134, 134, 122],
       ...,
       [  78,  71,  63],
       [  59,  58,  46],
       [  40,  41,  34]],

      [[ 129, 131, 118],
       [ 140, 141, 128],
       [ 137, 137, 124],
       ...,
       [  71,  64,  54],
       [  50,  48,  36],
       [  37,  39,  27]],

      [[ 140, 142, 127],
       [ 147, 149, 134],
       [ 139, 140, 126],
       ...,
       [  70,  63,  52],
       [  43,  42,  29],
       [  32,  34,  21]],

      ....
```

Figure 32

Figure above shows the step for image processing where it resizing the image by converting the image into numpy arrays. It is very common for computer vision task where numpy is used due to it can provide effective method to represent image in multi-dimensional array. The goal of converting the image into numpy array is standardization where standardize data is important for computer vision problem.

The figure on the left shows the pixel values where the images are being converted into numpy array and also shows the type of data as list and the total length of the data is 7553.

```
[ ] type(data[0])
numpy.ndarray

[ ] data[0].shape
(128, 128, 3)

# converting image list and label list to numpy arrays
X = np.array(data)
Y = np.array(labels)

[ ] type(X)
numpy.ndarray

[ ] type(Y)
numpy.ndarray

[ ] print(X.shape)
print(Y.shape)
(7553, 128, 128, 3)
(7553,)
```

Figure 33

Train and Test Split

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)

[ ] print(X.shape, X_train.shape, X_test.shape)
(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)

# scaling the data
X_train_scaled = X_train/255
X_test_scaled = X_test/255

X_train[0]
ndarray (128, 128, 3)
array([[105, 115, 124],
       [106, 116, 125],
       [108, 118, 127],
       ...,
       [102, 112, 124],
       [102, 112, 124],
       [101, 111, 123]],
      dtype=uint8)

[[110, 120, 129],
 [111, 121, 130],
 [113, 123, 132],
 ...,
 [ 99, 109, 121],
 [ 99, 109, 121],
 [ 98, 108, 120]],
 dtype=uint8)

[[106, 116, 125],
 [107, 117, 126],
 [109, 119, 128],
 ...,
 [105, 115, 124],
 [106, 116, 125],
 [108, 118, 127]],
 dtype=uint8)
```

Figure 34

Step above shows the method for train and test split where `x_train` represents training images, `Y_train` represents training labels, `X_test` represents testing images and `Y_test` represents testing label. It is known that the dataset is split into training and testing where for this 80/20 is used for training and testing dataset. Next is scaling the data where scaled is applied upon the `x_train` and `x_test` where previously the colour range from 0 to 255 and now after scaled it become 0 to 1.

Figure on the left shows the shape of the image where 128 is the width and height and 3 is the colour channel.

Next step is to convert the image and label list into numpy arrays where

(7553, 128, 128, 3) => 7553 represent number of images

(7553,) => this one represent the label

```

X_train_scaled[0]
array([[0.41176471, 0.45098039, 0.48627451],
       [0.41568627, 0.45490196, 0.49019608],
       [0.42352941, 0.4627451 , 0.49803922],
       ...,
       [0.4       , 0.43921569, 0.48627451],
       [0.4       , 0.43921569, 0.48627451],
       [0.39607843, 0.43529412, 0.48235294]],

       [[0.43137255, 0.47058824, 0.50588235],
        [0.43529412, 0.4745098 , 0.50980392],
        [0.44313725, 0.48235294, 0.51764706],
        ...,
        [0.38823529, 0.42745098, 0.4745098 ],
        [0.38823529, 0.42745098, 0.4745098 ],
        [0.38431373, 0.42352941, 0.47058824]],

       [[0.41568627, 0.45490196, 0.49019608],
        [0.41960784, 0.45882353, 0.49411765],
        [0.42745098, 0.46666667, 0.50196078],
        ...,
        [0.38823529, 0.42745098, 0.4745098 ],
        [0.38823529, 0.42745098, 0.4745098 ],
        [0.38823529, 0.42745098, 0.4745098 ]],

       ...,

       ...,

```

Figure on the left shows the result of the x_{train} after being scaled where the numpy array lie between 0 to 1.

Figure 35

Building a convolutional neural network (CNN)

```

[ ] import tensorflow as tf
    from tensorflow import keras

num_of_classes = 2

model = keras.Sequential()
model.add(keras.layers.Conv2D(32, kernel_size=(3,3), activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation = 'relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation = 'relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(num_of_classes, activation = 'sigmoid'))

```

Figure 36

Figure above shows the most important part of the CNN architecture where this part here shows the model building for the convolutional neural network where the input shape indicates number of 3 which shows the image datasets is coloured and not grey scale. Conv2D is convolutional layer and MaxPool is maxpooling layer is used where it is one of the most important components in CNN and activation function of ReLU and Sigmoid are used. Dropout is also applied to this where it is a method used to minimize overfitting.

```
[ ] # compile the neural network

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])

#training the neural network
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=10)

Epoch 1/10
170/170 [=====] - 148s 860ms/step - loss: 0.4342 - acc: 0.8133 - val_loss: 0.2969 - val_acc: 0.8777
Epoch 2/10
170/170 [=====] - 140s 823ms/step - loss: 0.2874 - acc: 0.8823 - val_loss: 0.2647 - val_acc: 0.8661
Epoch 3/10
170/170 [=====] - 140s 821ms/step - loss: 0.2490 - acc: 0.8963 - val_loss: 0.2527 - val_acc: 0.8959
Epoch 4/10
170/170 [=====] - 140s 825ms/step - loss: 0.1977 - acc: 0.9215 - val_loss: 0.2298 - val_acc: 0.9074
Epoch 5/10
170/170 [=====] - 149s 878ms/step - loss: 0.1767 - acc: 0.9288 - val_loss: 0.2549 - val_acc: 0.8959
Epoch 6/10
170/170 [=====] - 143s 842ms/step - loss: 0.1441 - acc: 0.9448 - val_loss: 0.2543 - val_acc: 0.9107
Epoch 7/10
170/170 [=====] - 144s 845ms/step - loss: 0.1193 - acc: 0.9538 - val_loss: 0.2563 - val_acc: 0.9256
Epoch 8/10
170/170 [=====] - 143s 843ms/step - loss: 0.0986 - acc: 0.9619 - val_loss: 0.2996 - val_acc: 0.8992
Epoch 9/10
170/170 [=====] - 141s 829ms/step - loss: 0.1016 - acc: 0.9601 - val_loss: 0.2556 - val_acc: 0.9107
Epoch 10/10
170/170 [=====] - 141s 827ms/step - loss: 0.0854 - acc: 0.9674 - val_loss: 0.2528 - val_acc: 0.9207
```

Figure 37

Figure above shows the neural network compile and the training code of the neural network where the optimizer used is “adam” and for the epochs it uses 10 epochs number. Based on the training of the model fit, the loss represents the difference between true value and predicted value and accuracy represent how many correct predictions the neural network makes. Hence if loss value decreases, accuracy increases. The train accuracy achieved is 0.9674 and the validation accuracy is 0.9207 which is quite good.

Model evaluation

```
[ ] loss, accuracy = model.evaluate(X_test_scaled, Y_test)
    print('Test Accuracy =', accuracy)

48/48 [=====] - 19s 393ms/step - loss: 0.2036 - acc: 0.9365
Test Accuracy = 0.9364659190177917

h = history
# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```

Figure 38

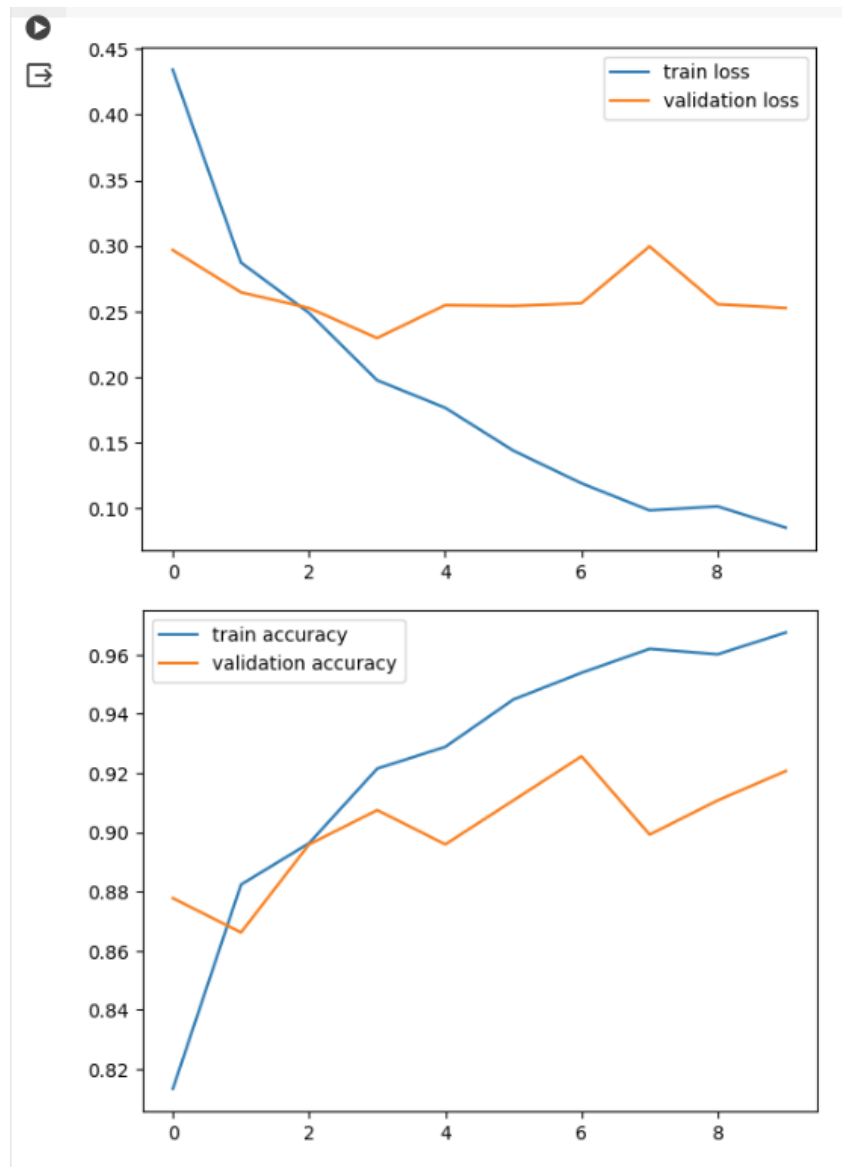


Figure 39

Both figures above show plot for the (train loss, validation loss) and (train accuracy, validation accuracy).

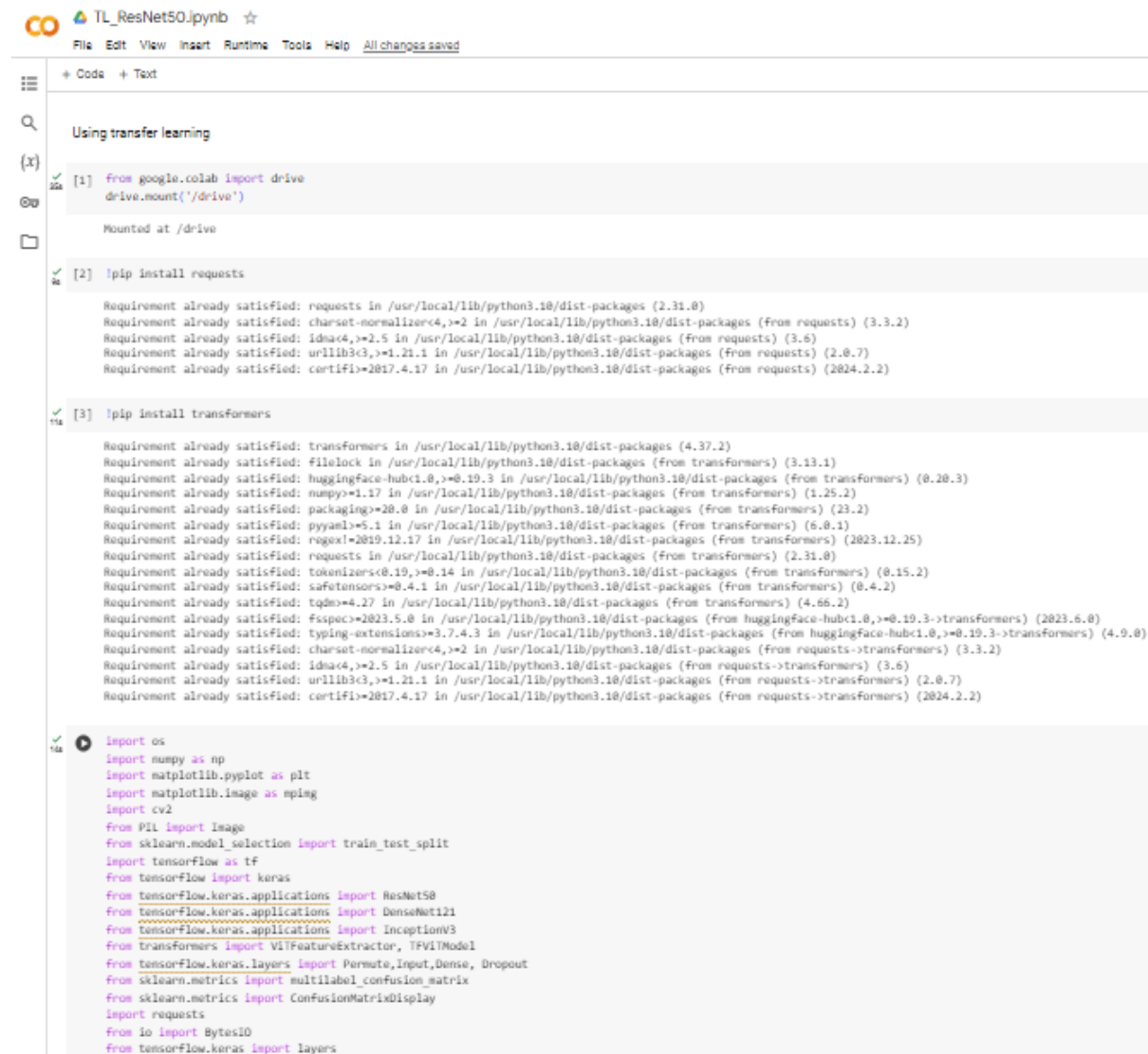
5.6.1 Using Pre-trained Neural Network (Transfer Learning ResNet50)

Another neural network approach is to use ResNet50 where its pre-trained large dataset where using ImageNet and ResNet50 consisted of 50 layers. This architecture used residual blocks that allow it to study and effectively learn data representations and it can reduce the problems with vanishing gradient problem. Below shows the step to use ResNet50 with transfer learning.

- 1) Load the pre-trained ReNet50 model.
- 2) Pre-processing step where images are resized, and the pixels are normalize into (0,1) range.

- 3) Customization where new layers can be added on top of the Resnet50 to adapt to the face mask detection model. Other than that, dense layers can be added with softmax layers with 2-unit output.

There are several advantages of using transfer learning such as feature reuse, faster convergence and it can significantly improve performance. Figure below shows the code in Google colab for Transfer learning ResNet50.



```

from google.colab import drive
drive.mount('/drive')

Mounted at /drive

!pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2024.2.2)

!pip install transformers

Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.37.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.8.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.15.2)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.2)
Requirement already satisfied: fspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3->transformers) (2023.5.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3->transformers) (4.9.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)

import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from PIL import Image
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.applications import InceptionV3
from transformers import ViTFeatureExtractor, TFViTModel
from tensorflow.keras.layers import Permute, Input, Dense, Dropout
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import requests
from io import BytesIO
from tensorflow.keras import layers

```

Figure 40

```

num_of_classes = 2

resnet = ResNet50(include_top=False, weights='imagenet', input_shape=(128, 128, 3))


for layer in resnet.layers:
    layer.trainable = False

model_ResNet50 = keras.Sequential()
model_ResNet50.add(resnet)
model_ResNet50.add(keras.layers.GlobalAveragePooling2D())
model_ResNet50.add(keras.layers.Dense(128, activation='relu'))
model_ResNet50.add(keras.layers.Dense(64, activation='relu'))
model_ResNet50.add(keras.layers.Dropout(0.3))
model_ResNet50.add(keras.layers.Dense(32, activation='relu'))

model_ResNet50.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))

model_ResNet50.summary()

```

 Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [=====] - 1s 8us/step
Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 4, 4, 2048)	23587712
global average pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense (Dense)	(None, 128)	262272
dense_1 (Dense)	(None, 64)	8256
dropout (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2080
dense_3 (Dense)	(None, 2)	66

Total params: 23860386 (91.02 MB)
Trainable params: 272674 (1.04 MB)
Non-trainable params: 23587712 (89.98 MB)

```

[22] model_ResNet50.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

[23] history_ResNet50 = model_ResNet50.fit(X_train_scaled, Y_train, validation_data=(X_test_scaled, Y_test), epochs=10, batch_size=30)

```

```

Epoch 1/10
282/282 [=====] - 521s 3s/step - loss: 0.6835 - acc: 0.5498 - val_loss: 0.6518 - val_acc: 0.6248
Epoch 2/10
282/282 [=====] - 515s 3s/step - loss: 0.6391 - acc: 0.6397 - val_loss: 0.6755 - val_acc: 0.5976
Epoch 3/10
282/282 [=====] - 461s 2s/step - loss: 0.6197 - acc: 0.6687 - val_loss: 0.5851 - val_acc: 0.7114
Epoch 4/10
282/282 [=====] - 507s 3s/step - loss: 0.5764 - acc: 0.7087 - val_loss: 0.5563 - val_acc: 0.7287
Epoch 5/10
282/282 [=====] - 508s 3s/step - loss: 0.5618 - acc: 0.7178 - val_loss: 0.5876 - val_acc: 0.6757
Epoch 6/10
282/282 [=====] - 458s 2s/step - loss: 0.5379 - acc: 0.7362 - val_loss: 0.5873 - val_acc: 0.7690
Epoch 7/10
282/282 [=====] - 458s 2s/step - loss: 0.5193 - acc: 0.7502 - val_loss: 0.4946 - val_acc: 0.7631
Epoch 8/10
282/282 [=====] - 506s 3s/step - loss: 0.5018 - acc: 0.7590 - val_loss: 0.4895 - val_acc: 0.7670
Epoch 9/10
282/282 [=====] - 506s 3s/step - loss: 0.5095 - acc: 0.7531 - val_loss: 0.5061 - val_acc: 0.7525
Epoch 10/10
282/282 [=====] - 507s 3s/step - loss: 0.5003 - acc: 0.7569 - val_loss: 0.4721 - val_acc: 0.7842

```

Figure 41

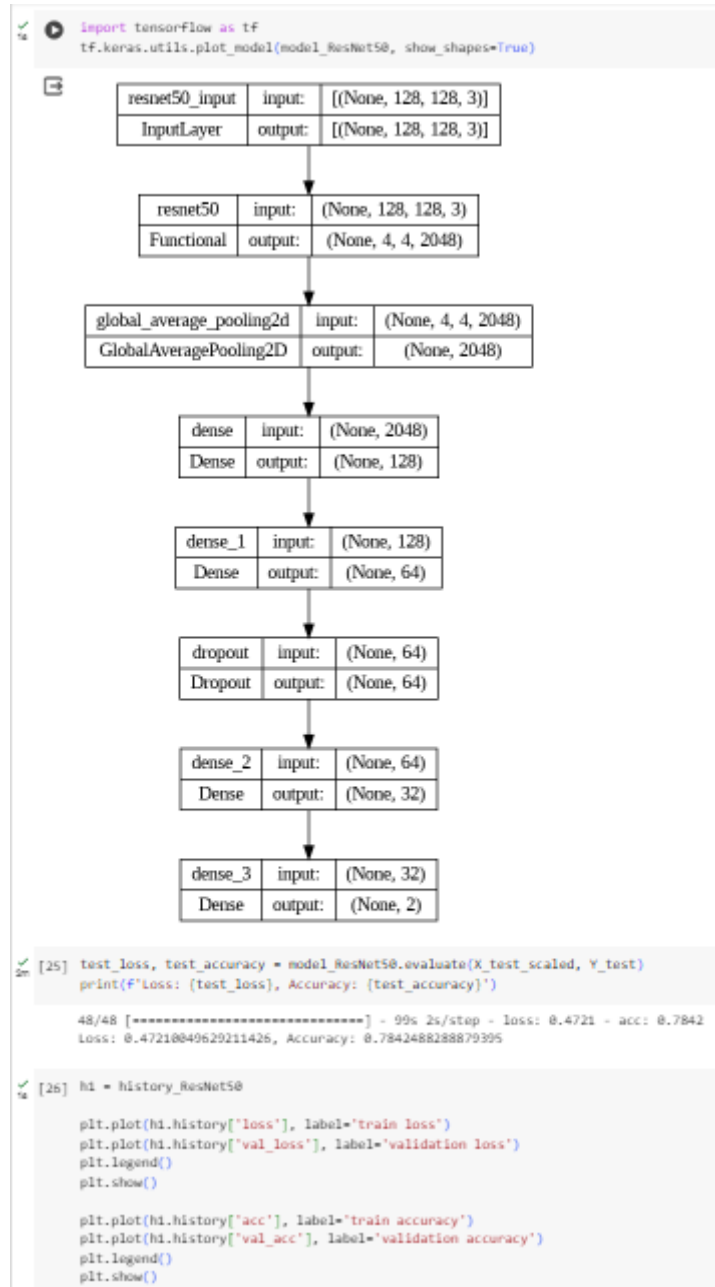


Figure 42

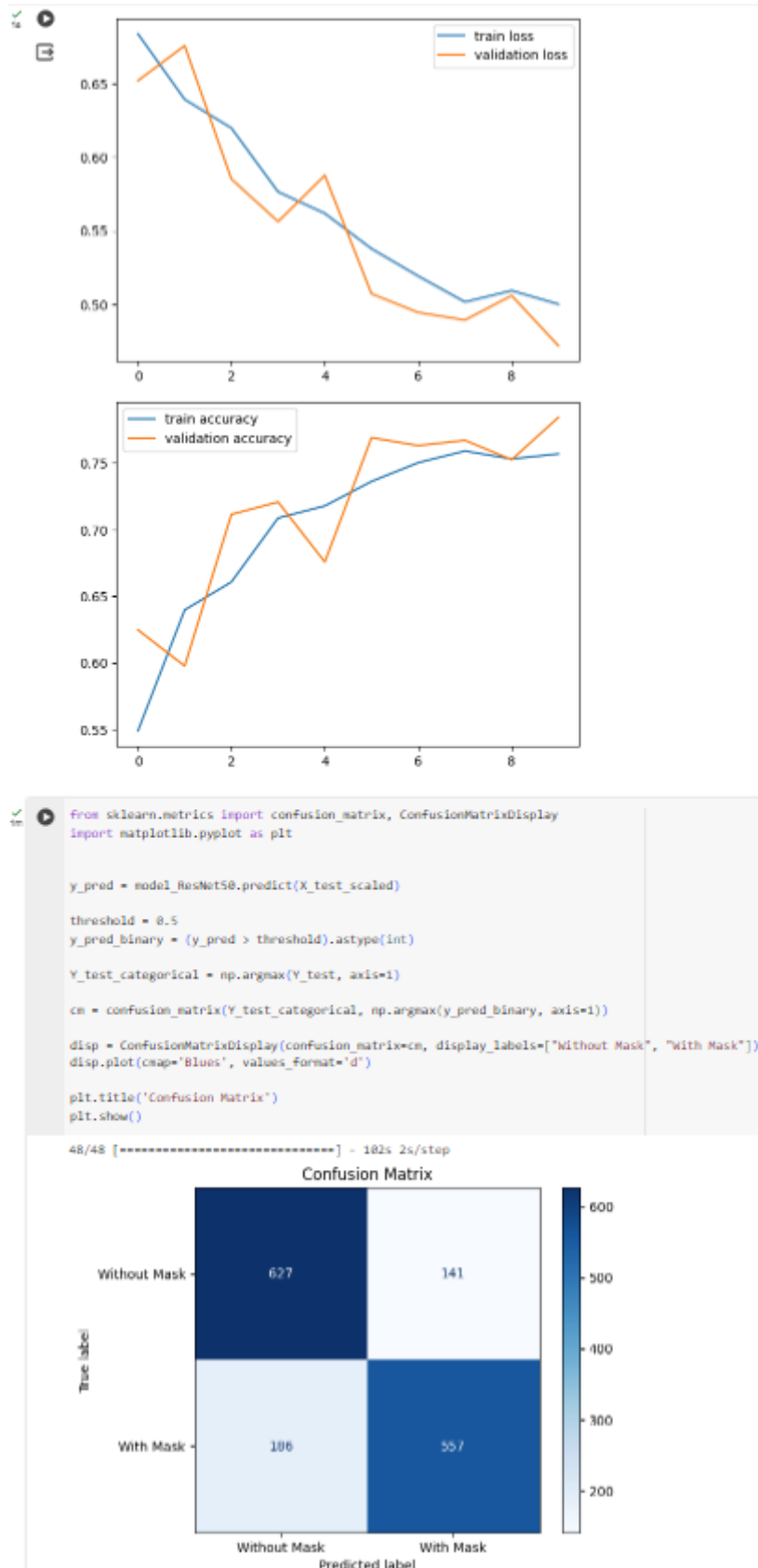


Figure 43

5.6.2 DenseNet

DenseNet is one of the architectures of CNN where it helps with vanishing gradient problems. Each layer of the networks is linked with a layer of feed forward fashion. This is quite distinct compared to traditional CNN whereby it linked each layer to the following layer. Dense block comprised of various layers where individual layers received input from previous layers and this dense connectivity applied feature reuse and eventually reduce vanishing gradient problem. DenseNet is used due to its good performance, efficiency and suitable for computer vision tasks. Figure below shows the python code using DenseNet121 architecture.

```
Using DenseNet121

num_of_classes=2
densenet = DenseNet121(include_top=False, weights='imagenet', input_shape=(128, 128, 3))

for layer in densenet.layers:
    layer.trainable = False

model_DenseNet = keras.Sequential()
model_DenseNet.add(densenet)
model_DenseNet.add(keras.layers.Flatten())
model_DenseNet.add(keras.layers.Dense(128, activation='relu'))
model_DenseNet.add(keras.layers.Dropout(0.5))
model_DenseNet.add(keras.layers.Dense(64, activation='relu'))
model_DenseNet.add(keras.layers.Dropout(0.5))
model_DenseNet.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))

model_DenseNet.summary()
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5
29084464/29084464 [=====] - 2s 0us/step
Model: "sequential_1"

Layer (type)	Output Shape	Param #
=====		
densenet121 (Functional)	(None, 4, 4, 1024)	7037504
flatten (Flatten)	(None, 16384)	0
dense_4 (Dense)	(None, 128)	2097280
dropout_1 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 64)	8256
dropout_2 (Dropout)	(None, 64)	0
dense_6 (Dense)	(None, 2)	130
=====		
Total params: 9143170 (34.88 MB)		
Trainable params: 2105666 (8.03 MB)		
Non-trainable params: 7037504 (26.85 MB)		

Figure 44

```

✓ [29] model_DenseNet.compile(optimizer='adam',loss='binary_crossentropy',metrics=['acc'])
0s

✓ 1h ▶ history_DenseNet = model_DenseNet.fit(X_train_scaled, Y_train, validation_data=(X_test_scaled, Y_test), epochs=10, batch_size=30)

Epoch 1/10
202/202 [=====] - 429s 2s/step - loss: 0.2531 - acc: 0.9330 - val_loss: 0.0594 - val_acc: 0.9881
Epoch 2/10
202/202 [=====] - 410s 2s/step - loss: 0.0840 - acc: 0.9676 - val_loss: 0.0450 - val_acc: 0.9868
Epoch 3/10
202/202 [=====] - 395s 2s/step - loss: 0.0871 - acc: 0.9695 - val_loss: 0.0564 - val_acc: 0.9821
Epoch 4/10
202/202 [=====] - 387s 2s/step - loss: 0.0645 - acc: 0.9828 - val_loss: 0.0348 - val_acc: 0.9894
Epoch 5/10
202/202 [=====] - 393s 2s/step - loss: 0.0731 - acc: 0.9806 - val_loss: 0.0409 - val_acc: 0.9887
Epoch 6/10
202/202 [=====] - 394s 2s/step - loss: 0.0430 - acc: 0.9879 - val_loss: 0.0385 - val_acc: 0.9907
Epoch 7/10
202/202 [=====] - 394s 2s/step - loss: 0.0518 - acc: 0.9848 - val_loss: 0.0500 - val_acc: 0.9901
Epoch 8/10
202/202 [=====] - 394s 2s/step - loss: 0.0404 - acc: 0.9882 - val_loss: 0.0577 - val_acc: 0.9894
Epoch 9/10
202/202 [=====] - 394s 2s/step - loss: 0.0440 - acc: 0.9844 - val_loss: 0.0389 - val_acc: 0.9927
Epoch 10/10
202/202 [=====] - 396s 2s/step - loss: 0.0382 - acc: 0.9868 - val_loss: 0.0417 - val_acc: 0.9901

✓ [31] test_loss, test_accuracy = model_DenseNet.evaluate(X_test_scaled, Y_test)
1m
print(f'Loss: {test_loss}, Accuracy: {test_accuracy}')

48/48 [=====] - 78s 2s/step - loss: 0.0417 - acc: 0.9901
Loss: 0.041694801300764084, Accuracy: 0.9900727868080139

✓ [32] h2 = history_DenseNet
2s

plt.plot(h2.history['loss'], label='train loss')
plt.plot(h2.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

plt.plot(h2.history['acc'], label='train accuracy')
plt.plot(h2.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()

```

Figure 45

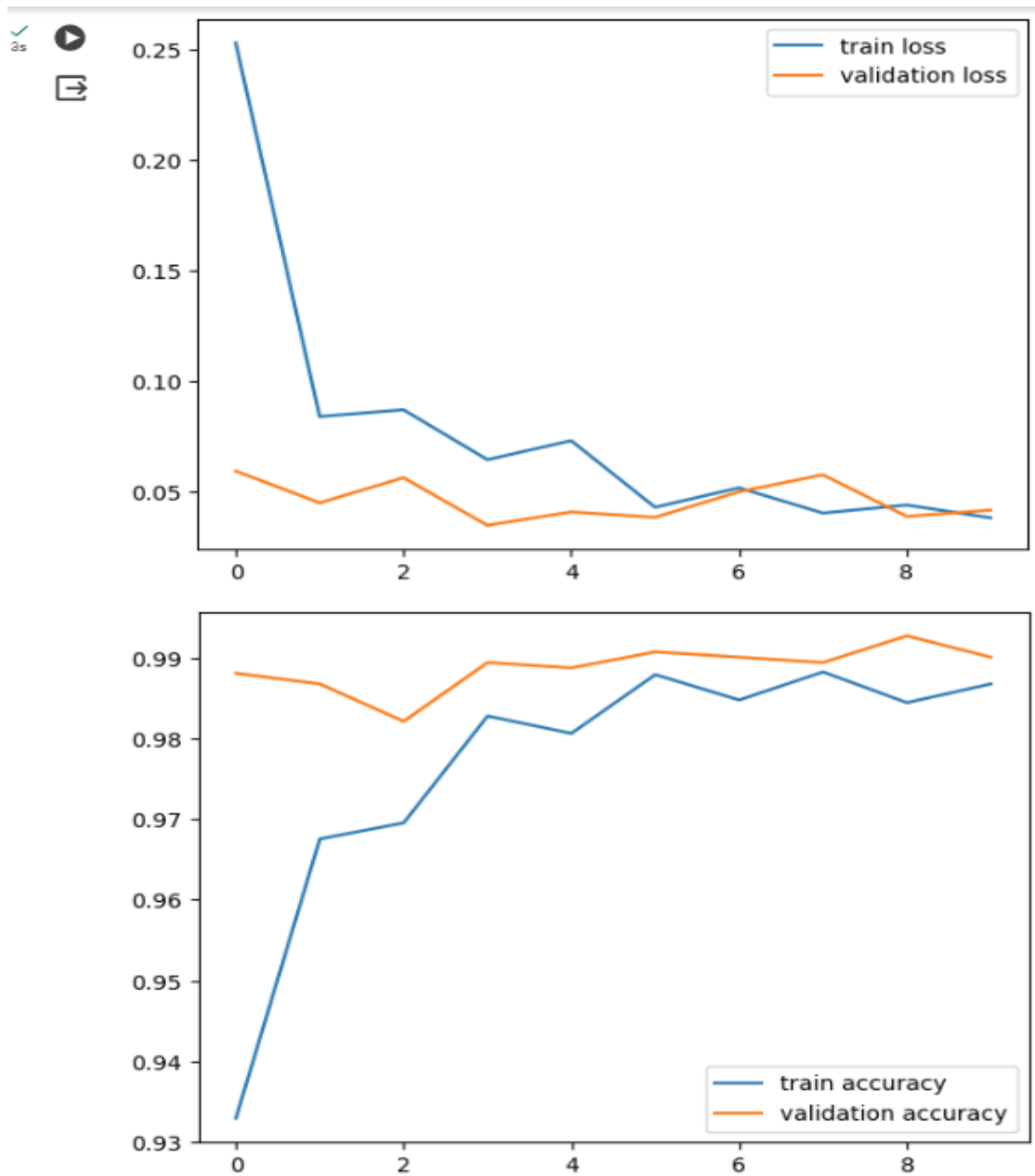


Figure 46

```

✓ 1m ▶ from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

y_pred = model_DenseNet.predict(X_test_scaled)

threshold = 0.5
y_pred_binary = (y_pred > threshold).astype(int)

Y_test_categorical = np.argmax(Y_test, axis=1)

cm = confusion_matrix(Y_test_categorical, np.argmax(y_pred_binary, axis=1))

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Without Mask", "With Mask"])
disp.plot(cmap='Blues', values_format='d')

plt.title('Confusion Matrix')
plt.show()

```

48/48 [=====] - 85s 2s/step

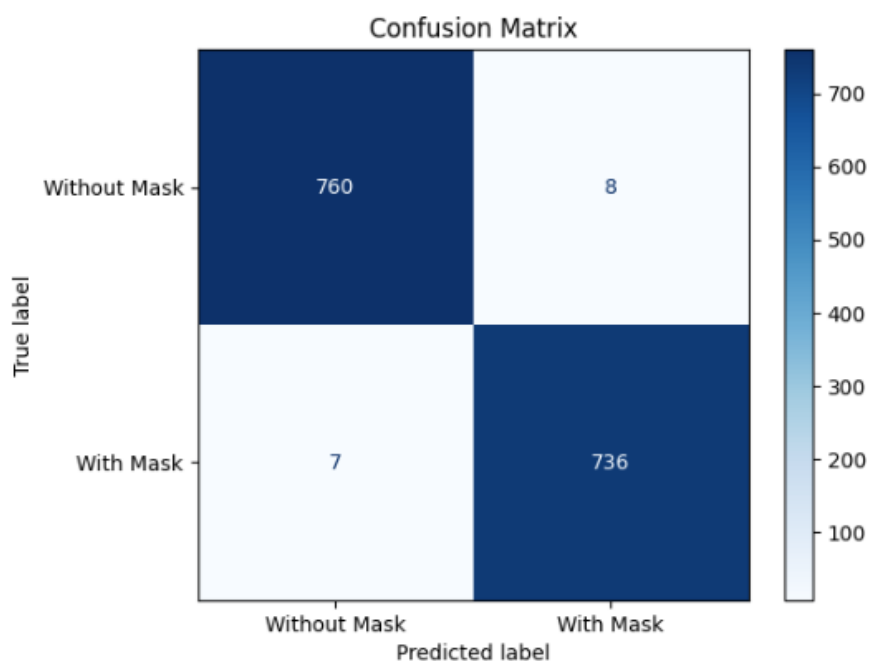


Figure 47

5.6.3 MobileNetV2

MobileNetV2 is one of the common methods in face mask image detection where many researchers used this method for computer vision and image classification. It is design to be used in mobile devices where it is much more efficient than traditional convolutional neural networks because it uses depthwise convolutional layers. The advantages of using MobileNetV2 are that it is very efficient, provides speed and very versatile and figure below shows the python code for MobileNetV2.



```
plt.figure(figsize=(10,8))
ax = sns.countplot(x=df.label)
ax.set_xlabel("Class")
ax.set_ylabel("Count")
plt.title('The Number Of Samples For Each Class')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```

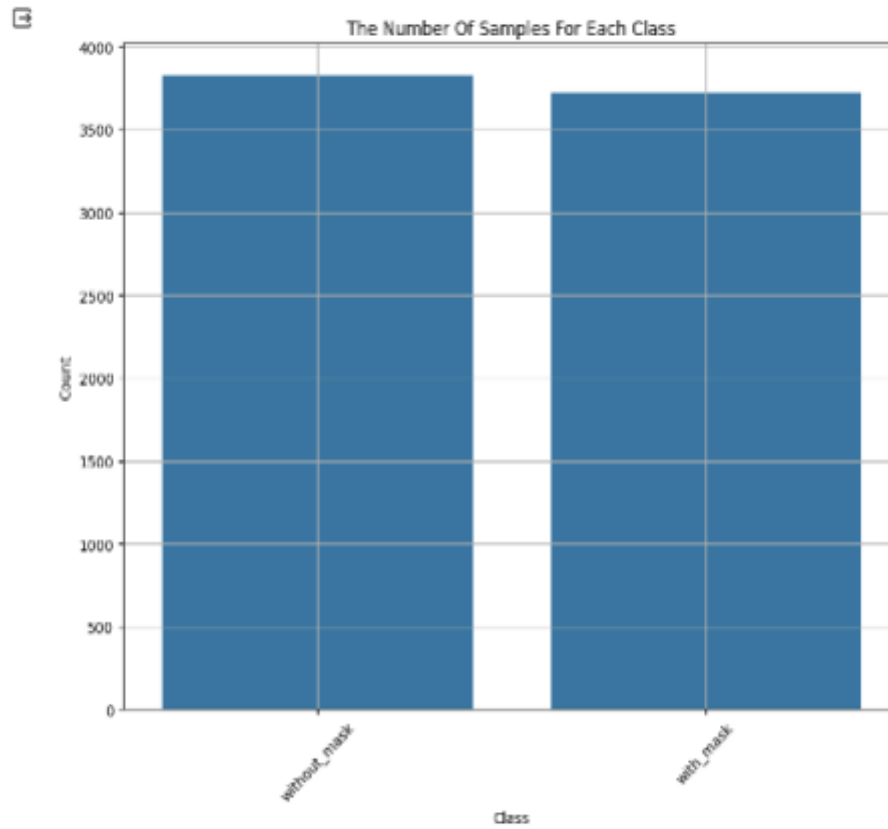


Figure 49


```
[6] X_train, X_test1, y_train, y_test1 = train_test_split(df['image'], df['label'], test_size=0.3, random_state=42, shuffle=True, stratify=df['label'])
X_val, X_test, y_val, y_test = train_test_split(X_test1, y_test1, test_size=0.5, random_state=42, shuffle=True, stratify=y_test1)
df_train = pd.DataFrame({'image': X_train, 'label': y_train})
df_test = pd.DataFrame({'image': X_test, 'label': y_test})
df_val = pd.DataFrame({'image': X_val, 'label': y_val})
```

```
image_size = (128,128)
batch_size = 20
datagen = ImageDataGenerator(
    rescale=1./255
)
train_generator = datagen.flow_from_dataframe(
    df_train,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)
test_generator = datagen.flow_from_dataframe(
    df_test,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
val_generator = datagen.flow_from_dataframe(
    df_val,
    x_col='image',
    y_col='label',
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)
```

Found 5287 validated image filenames belonging to 2 classes.
Found 1133 validated image filenames belonging to 2 classes.
Found 1133 validated image filenames belonging to 2 classes.

```
[8] base_model = tf.keras.applications.MobileNetV2(input_shape=(128,128,3),include_top=False,weights='imagenet')
base_model.trainable = False
model=keras.models.Sequential()
model.add(base_model)
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(256,activation=tf.nn.relu))
model.add(keras.layers.Dropout(.3))
model.add(keras.layers.Dense(2,activation=tf.nn.softmax))
model.summary()
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_128_no_top.h5
9486464/9486464 [=====] - 0s 0us/step
Model: "sequential"

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_128 (Functional)	(None, 4, 4, 1280)	2257984
flatten (Flatten)	(None, 20480)	0
dense (Dense)	(None, 256)	5243136
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

Total params: 7501634 (28.62 MB)
Trainable params: 5243650 (20.00 MB)
Non-trainable params: 2257984 (8.61 MB)

Figure 50



Figure 51

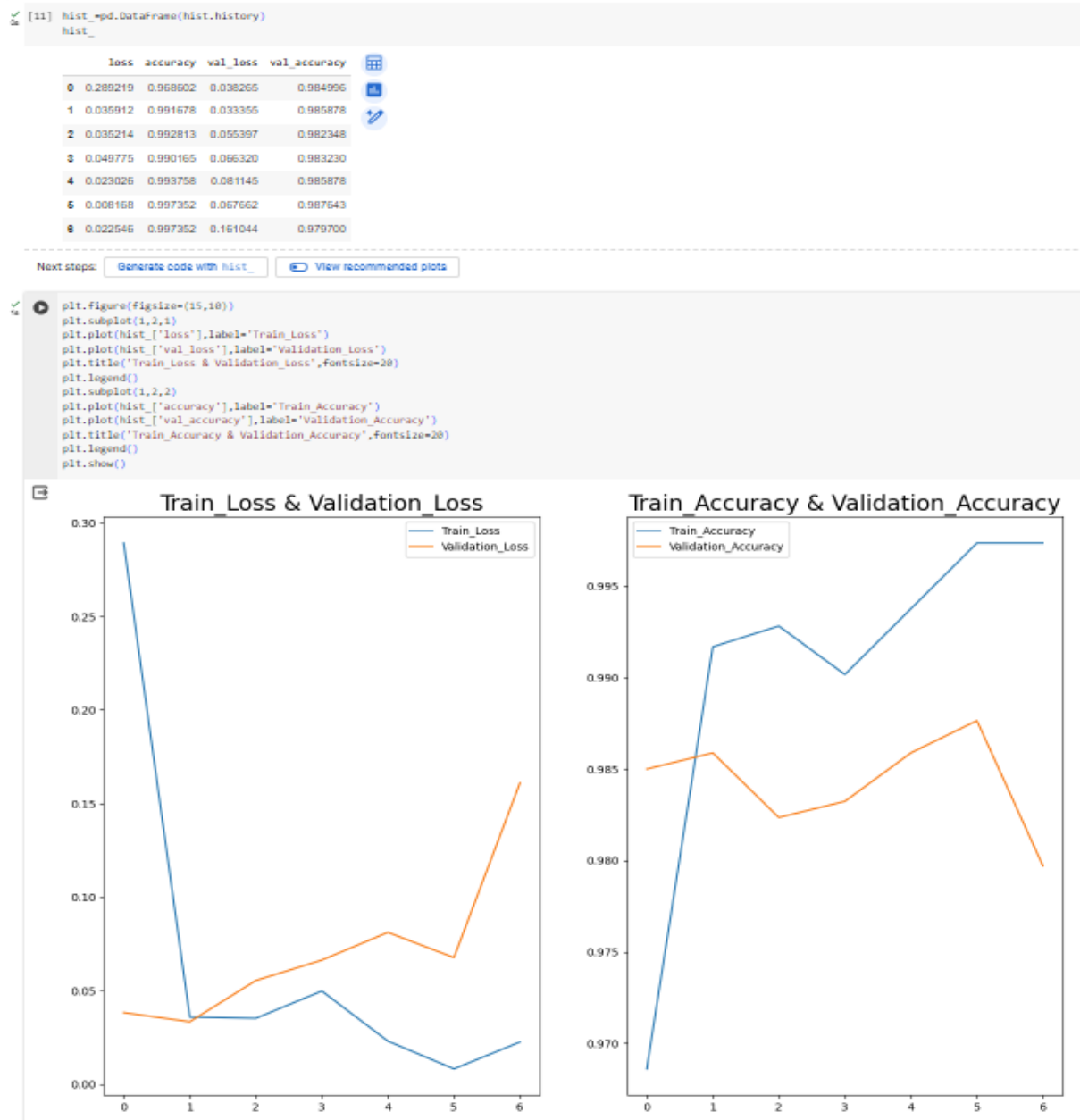


Figure 52

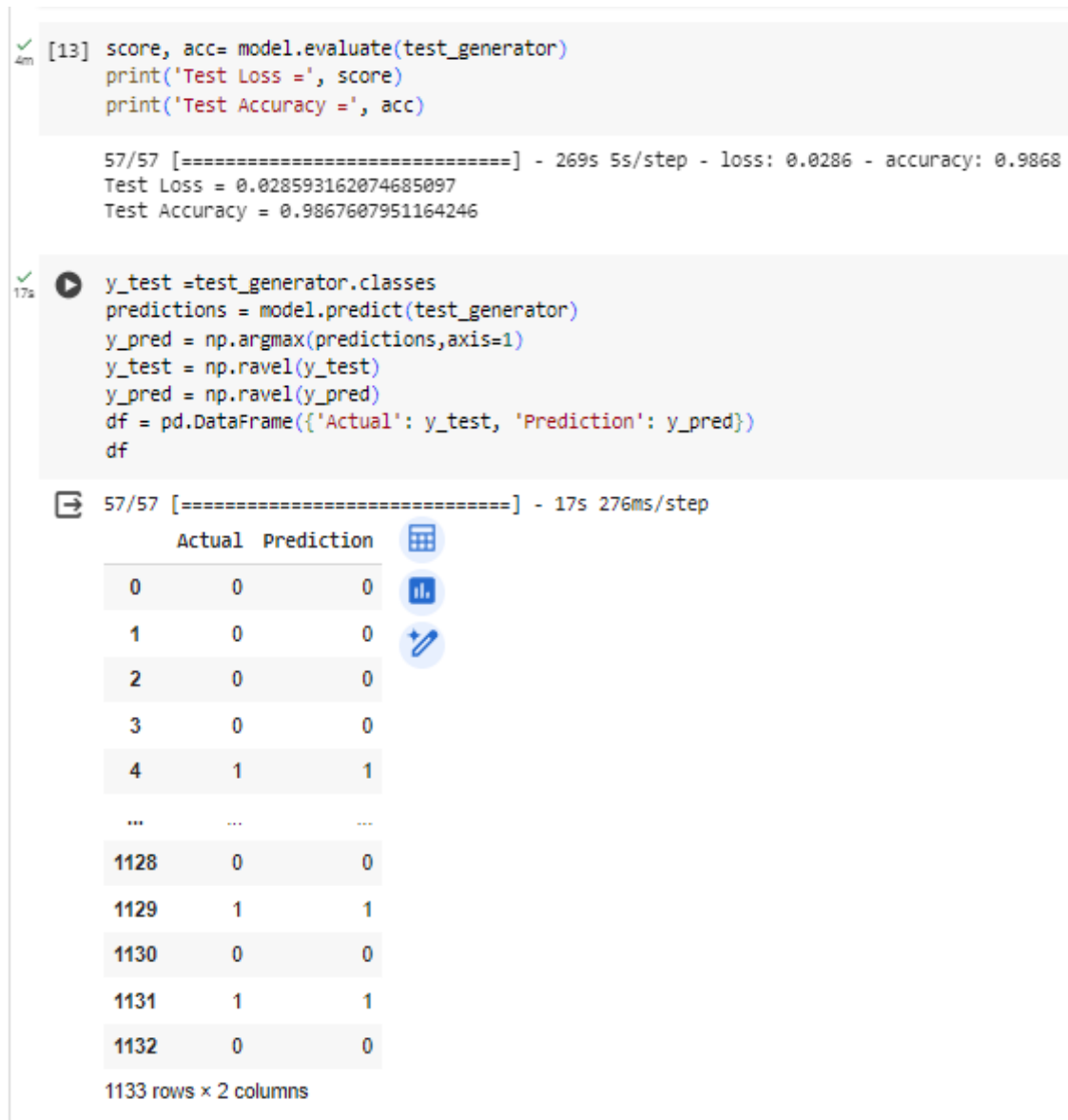


Figure 53



Figure 54

5.6.4 Xception

Xception also known as “Extreme Inception” where it is one of the CNN architectures and it has five major components which are depthwise separable convolutions, extreme depthwise separable convolutions, bottleneck design, linear architecture and global depthwise convolutions. Some of the advantages of using Xception is that it is very efficient for image classification and object detection problems. Other than that, it provides parameter efficiency where it can reduce the number of parameters. Also, it has expressive power where it is able to maintain very much high expressive power and understand complex input data. Below shows

the figure of the coding for Xception. Below shows the python code for the Xception in google colab.

```
[ ] import libarchive

import datetime as dt
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style('darkgrid')

import os
from keras.applications import xception
from keras.preprocessing import image
from mpl_toolkits.axes_grid1 import ImageGrid
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import cv2
from scipy.stats import uniform

from tqdm import tqdm
from glob import glob

from keras.models import Model, Sequential
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding, Masking
from keras.utils import to_categorical

import keras
from keras.callbacks import EarlyStopping, ModelCheckpoint

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Figure 55

```
[ ] keras.applications.Xception(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception\_weights\_tf\_dim\_ordering\_tf\_kernels.h5
91884032/91884032 [*****] - 1s 0us/step
ckeras.src.engine.functional.Functional at 0x7ebf24d9e320>
```

```
[ ] # Copying the pretrained models to the cache directory
cache_dir = os.path.expanduser(os.path.join('~', '.keras'))
if not os.path.exists(cache_dir):
    os.makedirs(cache_dir)
models_dir = os.path.join(cache_dir, 'models')
if not os.path.exists(models_dir):
    os.makedirs(models_dir)

# Specify the correct file path for Xception models
xception_model_path = '../Input/keras-pretrained-models/xception*'

# Copy the Xception models to the cache directory
!cp {xception_model_path} ~/.keras/models/

# Show the copied models
!ls ~/.keras/models

cp: cannot stat '../Input/keras-pretrained-models/xception*': No such file or directory
xception_weights_tf_dim_ordering_tf_kernels.h5
```

```
[ ] data_folder = '/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data'
categories = ['with_mask', 'without_mask']
len_categories = len(categories)
```

```
[ ] #show number of images per category
for category in categories:
    print('{} {} images'.format(category, len(os.listdir(os.path.join(data_folder, category)))))

with_mask 3725 images
without_mask 3828 images
```

```
train_data = []
for i, category in tqdm(enumerate(categories)):
    class_folder = os.path.join(data_folder, category)
    for path in os.listdir(os.path.join(class_folder)):
        train_data.append(['{}/{}'.format(category, path), category, i])
df = pd.DataFrame(train_data, columns=['filepath', 'class', 'label'])

#reduce the data
SAMPLE_PER_CATEGORY = 500
df = pd.concat([df[df['class'] == i][:SAMPLE_PER_CATEGORY] for i in categories])

print('DATAFRAME SHAPE: ', df.shape)
df.head()
```

2it [00:00, 13.41it/s]
DATAFRAME SHAPE: (1000, 3)

	filepath	class	label
0	with_mask/with_mask_3457.jpg	with_mask	0
1	with_mask/with_mask_346.jpg	with_mask	0
2	with_mask/with_mask_3452.jpg	with_mask	0
3	with_mask/with_mask_3450.jpg	with_mask	0
4	with_mask/with_mask_3472.jpg	with_mask	0

Figure 56

```
[ ] # function to get an image
def read_img(filepath, size):
    img = image.load_img(os.path.join(data_folder, filepath), target_size=size)
    #convert image to array
    img = image.img_to_array(img)
    return img

[ ] nb_rows = 3
    nb_cols = 5
    fig, axs = plt.subplots(nb_rows, nb_cols, figsize=(10, 5));
    plt.suptitle('SAMPLE IMAGES');
    for i in range(0, nb_rows):
        for j in range(0, nb_cols):
            axs[i, j].axis.set_ticklabels([]);
            axs[i, j].yaxis.set_ticklabels([]);
            axs[i, j].imshow((read_img(df['filepath'].iloc[np.random.randint(998)], (255,255)))/255);
    plt.show();
```

SAMPLE IMAGES



```
[ ] # function to sharpen the images
def sharpen_image(image):
    image_blurred = cv2.GaussianBlur(image, (0, 0), 3)
    image_sharp = cv1.addWeighted(image, 1.5, image_blurred, -0.5, 0)
    return image_sharp / 255

INPUT_SIZE = 255
X_train = np.zeros((len(df), INPUT_SIZE, INPUT_SIZE, 3), dtype='Float')

for i, file in tqdm(enumerate(df['filepath'])):
    img_sharpen = sharpen_image(read_img(file, (255,255)))
    X_train[i] = xception.preprocess_input(np.expand_dims(img_sharpen.copy(), axis=0))

1000it [03:44, 4.45it/s]

[ ] print('Train Image Shape: ', X_train.shape)
    print('Train Image Size: ', X_train.size)

Train Image Shape: (1000, 255, 255, 3)
Train Image Size: 195075000

[ ] #split the data
y = df['label']
train_x, train_val, y_train, y_val = train_test_split(X_train, y, test_size=0.2, random_state=101)

[ ] xception_bf = xception.Xception(weights='imagenet', include_top=False, pooling='avg')
bf_train_x = xception_bf.predict(train_x, batch_size=32, verbose=1)
bf_train_val = xception_bf.predict(train_val, batch_size=32, verbose=1)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 [=====] - 0% 0us/stop
25/25 [=====] - 244s 10%/stop
7/7 [=====] - 60s 8%/stop
```

Figure 57


```
[ ] #print shape of feature and size
print('Train Shape: ', bf_train_x.shape)
print('Train Size: ', bf_train_x.size)

print('Validation Shape: ', bf_train_val.shape)
print('Validation Size: ', bf_train_val.size)

Train Shape: (800, 2048)
Train Size: 1638400
Validation Shape: (200, 2048)
Validation Size: 409600

[ ] #optimizer
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=1e-3,
    decay_steps=10000,
    decay_rate=0.9)
opt = keras.optimizers.Adam(learning_rate=lr_schedule)

#keras model
model = Sequential()
model.add(Dense(units = 256, activation = 'relu', input_dim=bf_train_x.shape[1]))
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dense(units = 1, activation = 'sigmoid'))
model.compile(optimizer = opt, loss = 'binary_crossentropy', metrics = ['accuracy'])
model.summary()

Model: "sequential"
Layer (type) Output Shape Param #
-----
dense (Dense) (None, 256) 524544
dense_1 (Dense) (None, 64) 16448
dense_2 (Dense) (None, 1) 65
-----
Total params: 541057 (2.06 MB)
Trainable params: 541057 (2.06 MB)
Non-trainable params: 0 (0.00 Byte)

#set callbacks
callbacks = [EarlyStopping(monitor='val_loss', patience=2),
             ModelCheckpoint(filepath='best_model.h5', monitor='val_loss', save_best_only=True)]

#fit the data
history = model.fit(bf_train_x, y_train, batch_size=32, epochs=500, callbacks=callbacks)

WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 11ms/step - loss: 0.0076 - accuracy: 0.9987
Epoch 487/500
23/25 [=====] - ETA: 0s - loss: 0.0080 - accuracy: 0.9986WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 10ms/step - loss: 0.0085 - accuracy: 0.9987
Epoch 488/500
24/25 [=====] - ETA: 0s - loss: 0.0086 - accuracy: 0.9987WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 10ms/step - loss: 0.0085 - accuracy: 0.9987
Epoch 489/500
24/25 [=====] - ETA: 0s - loss: 0.0077 - accuracy: 0.9987WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 10ms/step - loss: 0.0076 - accuracy: 0.9987
Epoch 490/500
25/25 [=====] - ETA: 0s - loss: 0.0082 - accuracy: 0.9987WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 14ms/step - loss: 0.0082 - accuracy: 0.9987
Epoch 491/500
24/25 [=====] - ETA: 0s - loss: 0.0104 - accuracy: 0.9986WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 18ms/step - loss: 0.0187 - accuracy: 0.9912
Epoch 492/500
24/25 [=====] - ETA: 0s - loss: 0.0145 - accuracy: 0.9987WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 1s 21ms/step - loss: 0.0141 - accuracy: 0.9987
Epoch 493/500
24/25 [=====] - ETA: 0s - loss: 0.0472 - accuracy: 0.9779WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 1s 21ms/step - loss: 0.0454 - accuracy: 0.9787
Epoch 494/500
24/25 [=====] - ETA: 0s - loss: 0.0221 - accuracy: 0.9948WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 0s 10ms/step - loss: 0.0226 - accuracy: 0.9950
Epoch 495/500
24/25 [=====] - ETA: 0s - loss: 0.0096 - accuracy: 0.9987WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
25/25 [=====] - 1s 28ms/step - loss: 0.0094 - accuracy: 0.9987
Epoch 496/500
25/25 [=====] - ETA: 0s - loss: 0.0101 - accuracy: 0.9979WARNING:tensorflow:Early stopping conditioned on metric 'val_loss' which is not available. Available metrics are: loss,accuracy
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
```

Figure 58



Figure 59

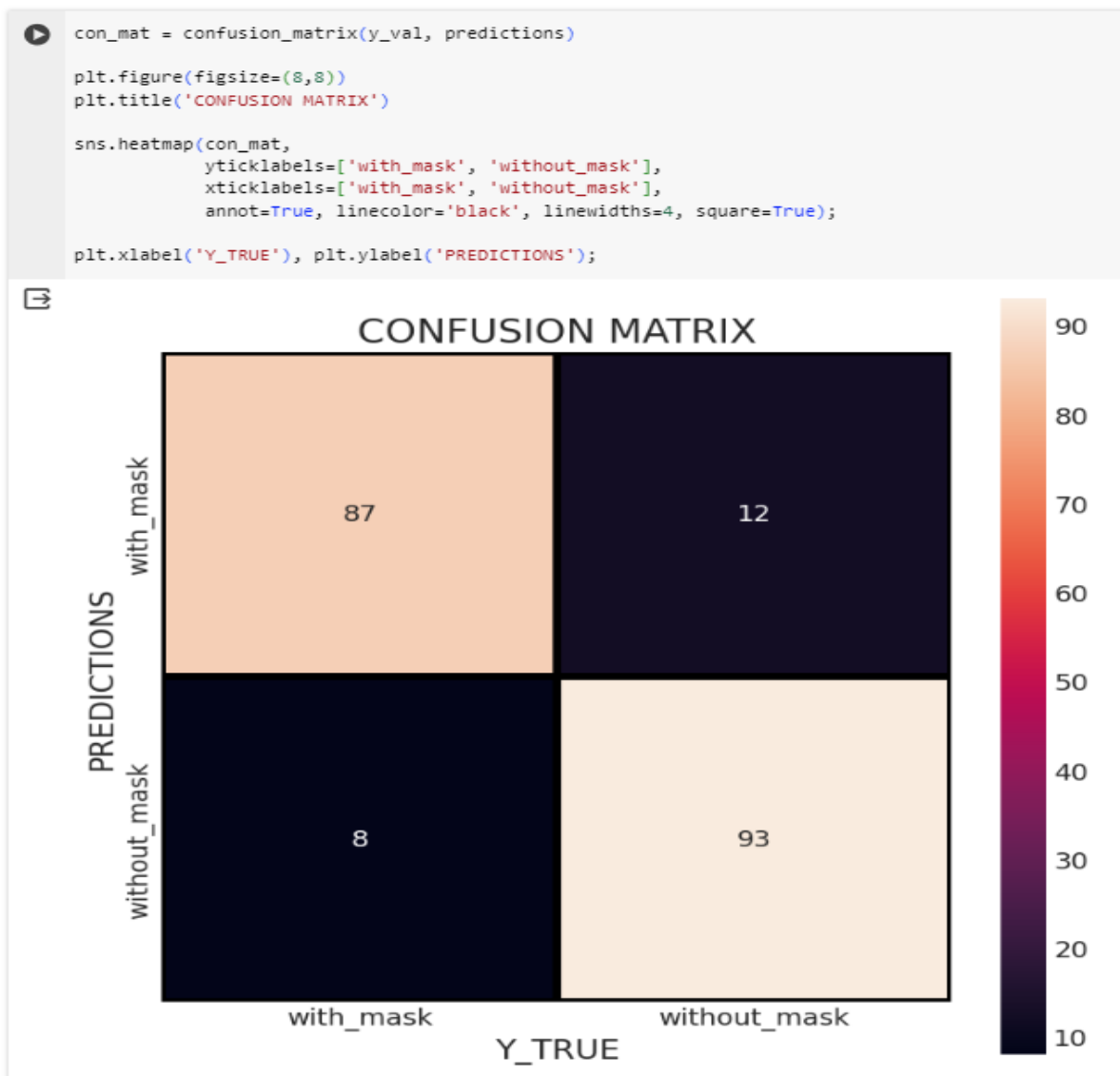


Figure 60

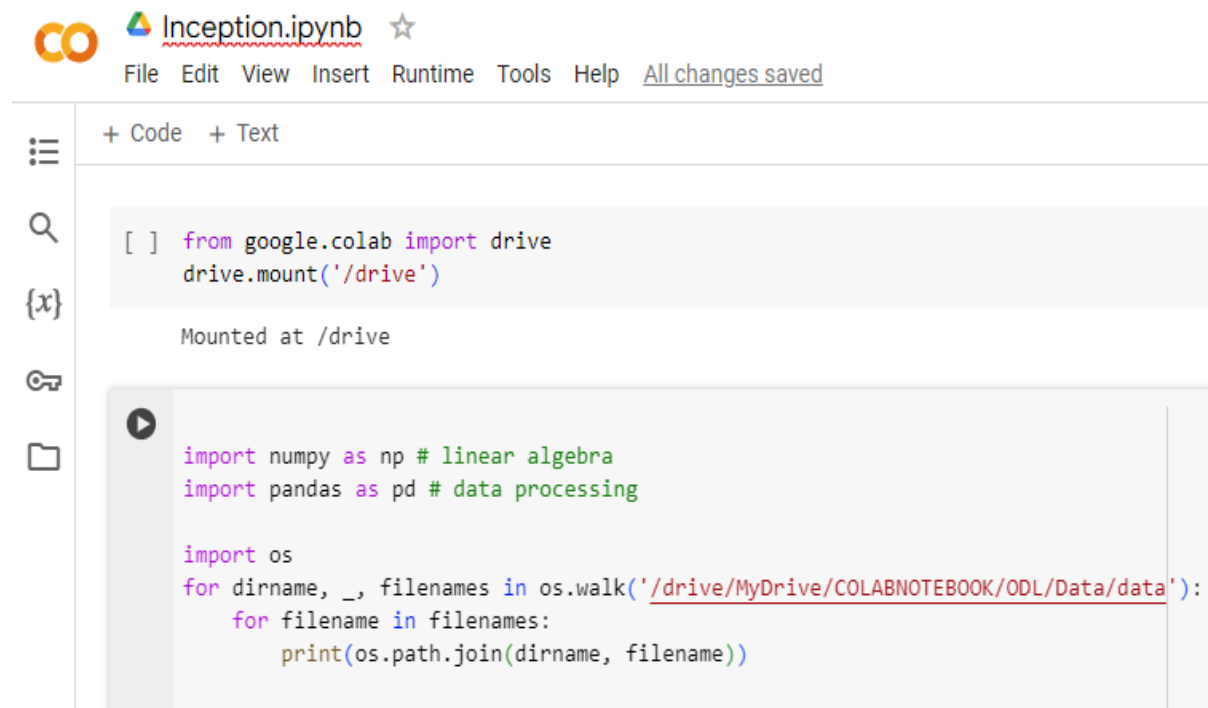
5.6.5 Inception

Inception is one of the CNN architectures where it uses inception module which allows it to obtain features at various scales and resolutions in parallel manners. There are four main components of Inception which are.

- 1) Inception modules : the main building blocks of inception architecture where it performs convolution operation with filter of various size.
- 2) Parallel convolutions : The input data are being applied parallel with convolutional layers with distinct filter size for example 1 by 1, 5 by 5, etc.
- 3) Dimensionality reductions : 1 by 1 convolutions are applied in dimensionality reductions prior applying filter size.

- 4) Auxiliary classifiers : This architecture also used auxiliary classifiers at intermediate layers besides the main task of classification output.

The figure below shows the code for InceptionV3 and InceptionV1 method.



```
[ ] from google.colab import drive
drive.mount('/drive')

Mounted at /drive

import numpy as np # linear algebra
import pandas as pd # data processing

import os
for dirname, _, filenames in os.walk('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Figure 61

```
[ ] import os
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Model
import tensorflow as tf
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, GlobalAveragePooling2D, Input, BatchNormalization, Activation, Add, concatenate
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.mobilenet_v2 import preprocess_input
from keras.applications import InceptionV3
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
▶ import os
import cv2
from keras.preprocessing import image
categories = ['with_mask', 'without_mask']
data = []
for category in categories:
    path = os.path.join('/drive/MyDrive/COLABNOTEBOOK/ODL/Data/data', category)
    label = categories.index(category)
    for file in os.listdir(path):
        img_path = os.path.join(path, file)
        img = cv2.imread(img_path)
        img = cv2.resize(img, (128, 128))
        data.append([img, label])
```

```
[ ] import random
random.shuffle(data)
image_data = []
y = []
for features, label in data:
    image_data.append(features)
    y.append(label)

image_data = np.array(image_data)
y = np.array(y)
image_data = preprocess_input(image_data)
print(image_data.shape)
print(y.shape)
```

```
(7553, 128, 128, 3)
(7553,)
```

```
▶ def inception_module(x, filters):
    conv1x1_1 = Conv2D(filters[0], (1, 1), padding='same', activation='relu')(x)
    conv1x1_3 = Conv2D(filters[1], (1, 1), padding='same', activation='relu')(x)
    conv3x3 = Conv2D(filters[2], (3, 3), padding='same', activation='relu')(conv1x1_3)
    conv1x1_5 = Conv2D(filters[3], (1, 1), padding='same', activation='relu')(x)
    conv5x5 = Conv2D(filters[4], (5, 5), padding='same', activation='relu')(conv1x1_5)
    maxpool = MaxPooling2D((3, 3), strides=(1, 1), padding='same')(x)
    conv1x1_pool = Conv2D(filters[5], (1, 1), padding='same', activation='relu')(maxpool)
    inception_block = concatenate([conv1x1_1, conv3x3, conv5x5, conv1x1_pool], axis=-1)
    return inception_block
```

Figure 62

```

def create_inception_model(input_shape, num_classes):
    input_layer = Input(shape=input_shape)

    # Stem
    stem = Conv2D(64, (7, 7), padding='same', strides=2, activation='relu')(input_layer)
    stem = MaxPooling2D((3, 3), padding='same', strides=2)(stem)
    stem = BatchNormalization()(stem)
    stem = Conv2D(64, (1, 1), activation='relu')(stem)
    stem = Conv2D(192, (3, 3), activation='relu')(stem)
    stem = BatchNormalization()(stem)
    stem = MaxPooling2D((3, 3), padding='same', strides=2)(stem)

    # Inception blocks
    inception1 = inception_module(stem, [64, 128, 32, 32, 96, 16])
    inception2 = inception_module(inception1, [128, 192, 96, 64, 128, 32])
    inception2 = MaxPooling2D((3, 3), padding='same', strides=2)(inception2)

    inception3 = inception_module(inception2, [192, 288, 48, 64, 96, 16])
    inception4 = inception_module(inception3, [160, 224, 64, 64, 112, 24])
    inception5 = inception_module(inception4, [128, 256, 64, 64, 128, 24])
    inception6 = inception_module(inception5, [112, 288, 64, 64, 144, 32])
    inception7 = inception_module(inception6, [256, 320, 128, 128, 160, 32])
    inception7 = MaxPooling2D((3, 3), padding='same', strides=2)(inception7)

    inception8 = inception_module(inception7, [256, 320, 128, 128, 160, 32])
    inception9 = inception_module(inception8, [384, 384, 128, 128, 192, 48])

    global_avg_pooling = GlobalAveragePooling2D()(inception9)
    dropout = Dropout(0.4)(global_avg_pooling)

    # Fully connected layers
    dense = Dense(1000, activation='relu')(dropout)
    output_layer = Dense(num_classes, activation='softmax')(dense)

    # Model
    model = Model(inputs=input_layer, outputs=output_layer)
    return model

```

```

[ ] # Example usage:
input_shape = (128, 128, 3)
num_classes = 3 # Number of output classes
inception_model = create_inception_model(input_shape, num_classes)

# Display the model summary
inception_model.summary()

```

max_pooling2d_10 (MaxPooling2D)	(None, 4, 4, 576)	0	['concatenate_6[0][0]']
conv2d_46 (Conv2D)	(None, 4, 4, 320)	184640	['max_pooling2d_10[0][0]']
conv2d_48 (Conv2D)	(None, 4, 4, 128)	73856	['max_pooling2d_10[0][0]']
max_pooling2d_11 (MaxPooling2D)	(None, 4, 4, 576)	0	['max_pooling2d_10[0][0]']
conv2d_45 (Conv2D)	(None, 4, 4, 256)	147712	['max_pooling2d_10[0][0]']
conv2d_47 (Conv2D)	(None, 4, 4, 128)	368768	['conv2d_46[0][0]']
conv2d_49 (Conv2D)	(None, 4, 4, 160)	512160	['conv2d_48[0][0]']
conv2d_50 (Conv2D)	(None, 4, 4, 32)	18464	['max_pooling2d_11[0][0]']
concatenate_7 (Concatenate)	(None, 4, 4, 576)	0	['conv2d_45[0][0]', 'conv2d_47[0][0]', 'conv2d_49[0][0]', 'conv2d_50[0][0]']
conv2d_52 (Conv2D)	(None, 4, 4, 384)	221568	['concatenate_7[0][0]']
conv2d_54 (Conv2D)	(None, 4, 4, 128)	73856	['concatenate_7[0][0]']
max_pooling2d_12 (MaxPooling2D)	(None, 4, 4, 576)	0	['concatenate_7[0][0]']
conv2d_51 (Conv2D)	(None, 4, 4, 384)	221568	['concatenate_7[0][0]']
conv2d_53 (Conv2D)	(None, 4, 4, 128)	442496	['conv2d_52[0][0]']
conv2d_55 (Conv2D)	(None, 4, 4, 192)	614592	['conv2d_54[0][0]']
conv2d_56 (Conv2D)	(None, 4, 4, 48)	27696	['max_pooling2d_12[0][0]']
concatenate_8 (Concatenate)	(None, 4, 4, 752)	0	['conv2d_51[0][0]', 'conv2d_53[0][0]', 'conv2d_55[0][0]', 'conv2d_56[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 752)	0	['concatenate_8[0][0]']

Figure 63

```
[ ] inception_model.compile(
    optimizer= tf.keras.optimizers.Adam(),
    loss = tf.keras.losses.sparse_categorical_crossentropy,
    metrics=['accuracy']
)

[ ] from sklearn.model_selection import train_test_split

# Assuming X is your feature matrix and y is the target variable
X_train, X_test, y_train, y_test = train_test_split(image_data, y, test_size=0.2, random_state=42)

[ ] history = inception_model.fit(X_train, y_train, epochs =10 ,batch_size= 128, validation_data=(X_test,y_test))

Epoch 1/10
48/48 [=====] - 624s 13s/step - loss: 0.7337 - accuracy: 0.5619 - val_loss: 0.7694 - val_accuracy: 0.5856
Epoch 2/10
48/48 [=====] - 599s 13s/step - loss: 0.2684 - accuracy: 0.9810 - val_loss: 0.5254 - val_accuracy: 0.7856
Epoch 3/10
48/48 [=====] - 598s 12s/step - loss: 0.1682 - accuracy: 0.9346 - val_loss: 0.4491 - val_accuracy: 0.9107
Epoch 4/10
48/48 [=====] - 597s 12s/step - loss: 0.1188 - accuracy: 0.9563 - val_loss: 0.3245 - val_accuracy: 0.9113
Epoch 5/10
48/48 [=====] - 598s 12s/step - loss: 0.1217 - accuracy: 0.9545 - val_loss: 0.2914 - val_accuracy: 0.9014
Epoch 6/10
48/48 [=====] - 598s 12s/step - loss: 0.0856 - accuracy: 0.9695 - val_loss: 0.1856 - val_accuracy: 0.9610
Epoch 7/10
48/48 [=====] - 598s 12s/step - loss: 0.0880 - accuracy: 0.9715 - val_loss: 0.1322 - val_accuracy: 0.9584
Epoch 8/10
48/48 [=====] - 599s 12s/step - loss: 0.0722 - accuracy: 0.9753 - val_loss: 0.0937 - val_accuracy: 0.9662
Epoch 9/10
48/48 [=====] - 594s 12s/step - loss: 0.0483 - accuracy: 0.9834 - val_loss: 0.1117 - val_accuracy: 0.9576
Epoch 10/10
48/48 [=====] - 593s 12s/step - loss: 0.0422 - accuracy: 0.9876 - val_loss: 0.1069 - val_accuracy: 0.9656

plt.figure(figsize=(9, 4))

# Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot Training Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

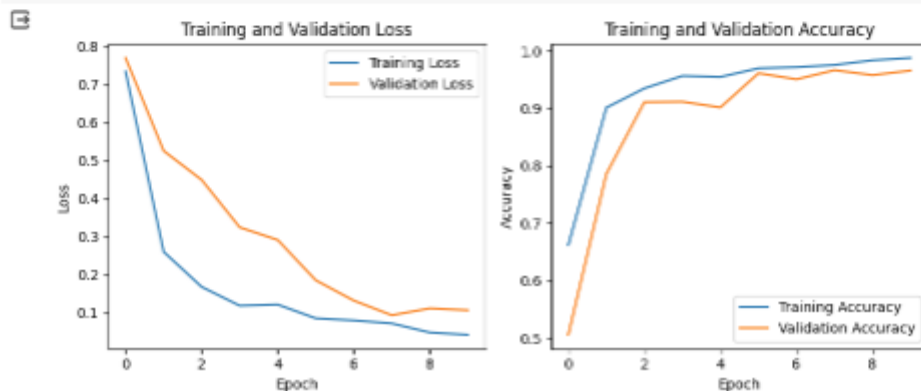


Figure 64

```
[ ] # Create the pretrained model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
x = GlobalAveragePooling2D()(base_model.output)

# Fully connected layers
dense = Dense(1000, activation='relu')(x)
predictions = Dense(3, activation='softmax')(dense)
pre_trained_model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model
pre_trained_model.compile(optimizer='adam',
                          loss='sparse_categorical_crossentropy',
                          metrics=['accuracy'])

# Fit the model
history = pre_trained_model.fit(X_train, y_train, epochs=5, batch_size=128, validation_data=(X_test, y_test))
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [-----] - 3s 0us/step

```
Epoch 1/5
48/48 [=====] - 1048s 21s/step - loss: 0.2963 - accuracy: 0.9341 - val_loss: 8.9313 - val_accuracy: 0.5268
Epoch 2/5
48/48 [=====] - 983s 21s/step - loss: 0.0571 - accuracy: 0.9834 - val_loss: 7.5585 - val_accuracy: 0.8716
Epoch 3/5
48/48 [=====] - 1026s 21s/step - loss: 0.0207 - accuracy: 0.9939 - val_loss: 0.0663 - val_accuracy: 0.9828
Epoch 4/5
48/48 [=====] - 1022s 21s/step - loss: 0.0057 - accuracy: 0.9978 - val_loss: 0.1046 - val_accuracy: 0.9808
Epoch 5/5
48/48 [=====] - 1026s 21s/step - loss: 0.0008 - accuracy: 0.9970 - val_loss: 0.0392 - val_accuracy: 0.9901
```

```
# Display the model summary
pre_trained_model.summary()
```

batch_normalization_82 (Batch Normalization)	(None, 2, 2, 448)	1344	['conv2d_137[0][0]']
activation_80 (Activation)	(None, 2, 2, 448)	0	['batch_normalization_82[0][0]']
conv2d_134 (Conv2D)	(None, 2, 2, 384)	491520	['mixed8[0][0]']
conv2d_138 (Conv2D)	(None, 2, 2, 384)	1548288	['activation_80[0][0]']
batch_normalization_79 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_134[0][0]']
batch_normalization_83 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_138[0][0]']
activation_77 (Activation)	(None, 2, 2, 384)	0	['batch_normalization_79[0][0]']
activation_81 (Activation)	(None, 2, 2, 384)	0	['batch_normalization_83[0][0]']
conv2d_135 (Conv2D)	(None, 2, 2, 384)	442368	['activation_77[0][0]']
conv2d_136 (Conv2D)	(None, 2, 2, 384)	442368	['activation_77[0][0]']
conv2d_139 (Conv2D)	(None, 2, 2, 384)	442368	['activation_81[0][0]']
conv2d_140 (Conv2D)	(None, 2, 2, 384)	442368	['activation_81[0][0]']
average_pooling2d_7 (Average Pooling2D)	(None, 2, 2, 1280)	0	['mixed8[0][0]']
conv2d_133 (Conv2D)	(None, 2, 2, 320)	409600	['mixed8[0][0]']
batch_normalization_80 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_135[0][0]']
batch_normalization_81 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_136[0][0]']
batch_normalization_84 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_139[0][0]']
batch_normalization_85 (Batch Normalization)	(None, 2, 2, 384)	1152	['conv2d_140[0][0]']
conv2d_141 (Conv2D)	(None, 2, 2, 192)	245760	['average_pooling2d_7[0][0]']
batch_normalization_78 (Batch Normalization)	(None, 2, 2, 320)	960	['conv2d_133[0][0]']
activation_78 (Activation)	(None, 2, 2, 384)	0	['batch_normalization_80[0][0]']
activation_79 (Activation)	(None, 2, 2, 384)	0	['batch_normalization_81[0][0]']

Figure 65


```
[ ] # Plot Training Loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# Plot Training Accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

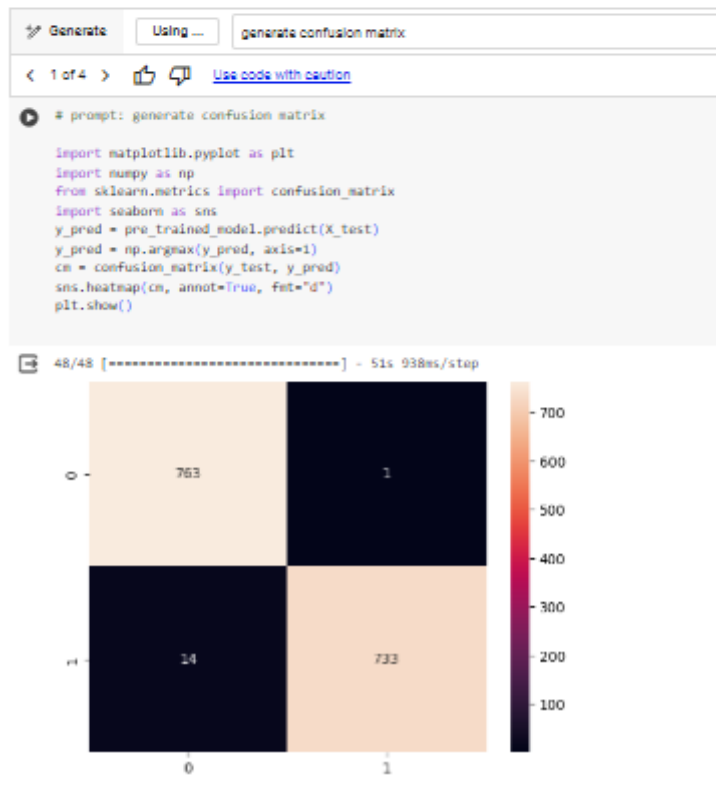
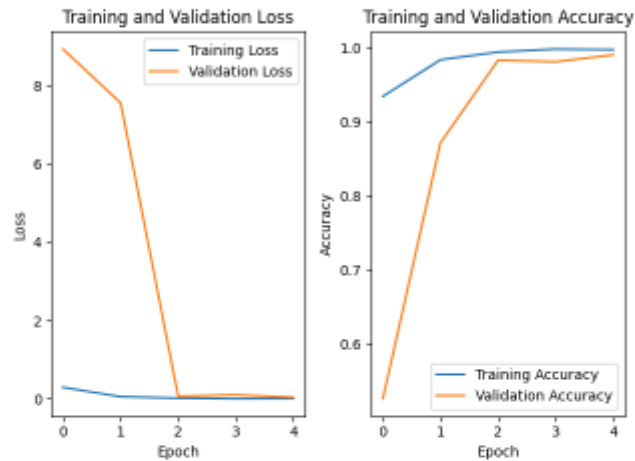


Figure 66

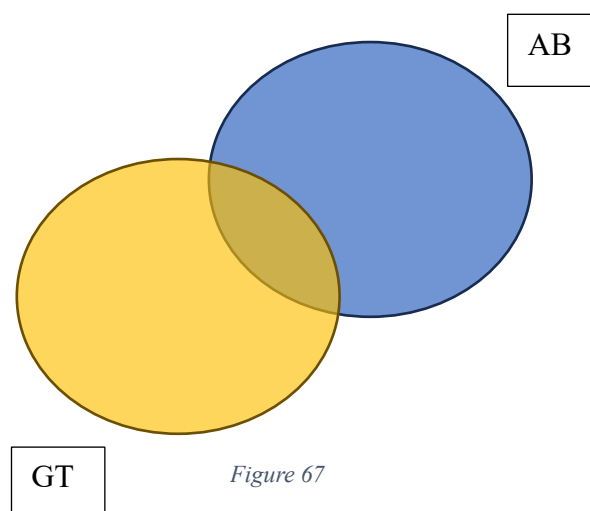
5.6.6 Evaluation

Hence, it can be concluded that there are so many methods that can be used to assess the performance of CNN, convolutional neural networks using confusion matrix. In this assignment there are six neural networks architectures proposed where each of these neural networks has its own unique architecture. The proposed architecture is :

- 1) Resnet50
- 2) DenseNet121
- 3) MobileNetV2
- 4) Xception
- 5) Inception

In computer vision Region Proposals Network operate by passing a sliding window over CNN feature map and at each individual window, the outputting potential bounding boxes and scores where it indicates the performance each of these individual boxes. Object detection and classification for single object works with classification and localization where the input images usually pre-trained on ImageNet (Transfer Learning)

Region proposal network works with fully convolutional layer where it works smoothly with Faster R-CNN where for the feature extraction the input image passed through CNN architecture where it is used to extract feature map. ResNet and Inception method is a common method of pre-trained CNN with transfer learning where features are extracted from the input images which afterwards passed through layers of network of regression and classification. IoU and mAP are used to assess the performance of the networks where IoU stands for Intersection over Union. It is a metric used in object recognition where it is a measure of degree of overlap and below shows the formula of IoU.



$$IoU = \frac{Area(AB) \cap Area(GT)}{Area(AB) \cup Area(GT)}$$

It is used to evaluate the performance of object detection and image classification problems, especially tasks that involve segmentation and box regression. Other than that, mAP where it stands for mean Average precision, it is vastly used in computer vision problems. Depthwise separable convolutions are used MobileNet and Xception where it is mostly designed for mobile devices.

6.0 TUNING & VALIDATION

This part here is to discuss the model hyper parameter tuning for the model implementation done previously for the ResNet50, DenseNet121, MobileNetV2, Xception and Inception where tuning and validation is needed to improve the model accuracy. The importance of tuning and validation is that for example, in hyperparameter tuning it often involves to improve the accuracy of the model, obtained better convergence and improved generalization by tuning the batch size, number of layers, the dropout rate and the learning rate.

For example, in the transfer learning ResNet50, dropout rate of 0.3 and activation function of ReLU and Sigmoid are used where these three methods are widely used in deep learning problems as it can reduced the problem of vanishing gradient. Vanishing gradient problems happened especially in training deep neural networks where particularly with many layers such as RNN, CNN and deep feedforward neural networks. Dropout on the other hand, is one of the common methods in deep learning to minimize or prevent overfitting where dropouts are able to minimize and prevent overfitting. It is able to improve the generalization of the model by drop out neurons from neural networks during training phase.

It is known that in deep learning, especially for computer vision input image size is in the shape of (224,244,3) where 3 is the colour channel of the RGB and it is very common to use batch size of 64. In this assignment the input image size is reduced for the majority of the neural network's architecture as size of input image can affect in many ways such as

1. Computational complexity : It is more computationally expensive as the input image size become larger as for example when setting the input image size at 244 it took around 8 hours for one of the neural networks architectures to compute and thus reducing it to 128 able to reduce the computing time.

2. Number of parameters : It is known that the larger the input size, it requires more learning time for the network parameters as it can lead to overfitting if the size of the dataset is not enough.
3. Spatial dimensions : Spatial dimensions refer to the width and height of the input image where it can influence the number of spatial positions as larger input size increases the number of parameters and computational complexity.
4. Feature resolution : It is known that overfitting will happen as larger input size will affect the feature resolution that the network is able to learn. Vice versa a smaller input size will minimize over fitting.
5. Data augmentation: By having large input size it allows more flexibility in data augmentation as it is a method to artificially increase size of dataset.

Hence there are various factors that can be considered when doing tuning and validation such as the parameter that can be manipulated, input size of the image, dropout rate, activation function, batch size tuning and more. It is known that for spatial images dataset the neural network coding is a little bit different than using a textual dataset where rescaling the dataset is done using one hot encoding for image dataset. One hot encoding is one of the methods used to represent categorical data which include class labels before being fed into machine learning models. One hot encoding is very famous with computer vision task especially for image dataset classification and below shows the mechanism behind one hot encoding for image or spatial datasets.

1. Class labels : Class labels are used in image classification where for example each of the images are assigned with number ranging from 0 to 9. This indicates that the image is associated with class label and hence shows the category where it belongs to.
2. Integer coding: For example, the class label represents number or digits from 0 to 9 for each image dataset (integers)
3. One hot encoding: The function of one hot encoding is to convert the integer label into binary class such 0 and 1 where for 0 is represented image dataset without mask and for 1 it represents image with mask.

The reason why one hot encoding is crucial in training machine learning model where it converts the class labels into a format that can be read by the neural network for example in this assignment the class labels are interpreted as categorical variables and later during the building the

sequential model and compiling the sequential model loss functions are used such as categorical cross-entropy or binary cross entropy.

It is known that in model building, training and test of the dataset is done where training data refers to one of the subset data to train machine learning model. The main function of training data is to use the neural network model so that it is able to learn the underlying patterns and make new predictions or classification for unseen data. It is known that in training data it consisted of input features where for example in this assignment it contained the with and without mask image datasets. The target variable usually is the main objective of the machine learning model for example, the target is to identify image with mask and without mask and hence face mask image classification use binary label as target value. Hence, the binary labels will indicate the and identify the different classes into image with face mask that is label as positive class and on the other hand without face mask label as negative class.

The table below shows the test loss and test accuracy table for the proposed neural network model and from the table below it can be seen that different neural network models will resulted in different (%) result for test loss and test accuracy.

Table 10

Neural network model	Test loss	Test accuracy
ResNet50	0.6730	0.6640
DenseNet121	0.0470	0.9910
MobileNetV2	0.0285	0.9867
Xception	0.0108	0.9987
Inception	0.0098	0.9970

It is found that from the table above that Xception model has the highest accuracy at 99.87% followed by Inception at 99.70%, DenseNet121 at 99.10%, MobileNetV2 at 93.89% and the lowest test accuracy at 66.40% which is the Resnet50.

On the other hand, for the test loss the highest value for test loss is ResNet50 at 0.6730 followed by DenseNet121 at 0.0470, MobileNetV2 at 0.0285, Xception at 0.0108 and lowest value for test loss is Inception at 0.0098. There is a trend where if the test accuracy is high the test value will become lower, and a reminder is that test loss and test accuracy show the performance of a model able to generalize and predict unseen data. Hence, the best model is Xception with the highest test accuracy compared to the rest of the model.

Figure below shows the train loss, validation loss and train accuracy, validation accuracy graph for all of the five neural networks model and training and validation accuracy shows how the model able to fit the training data and generalize the brand-new data during training process.

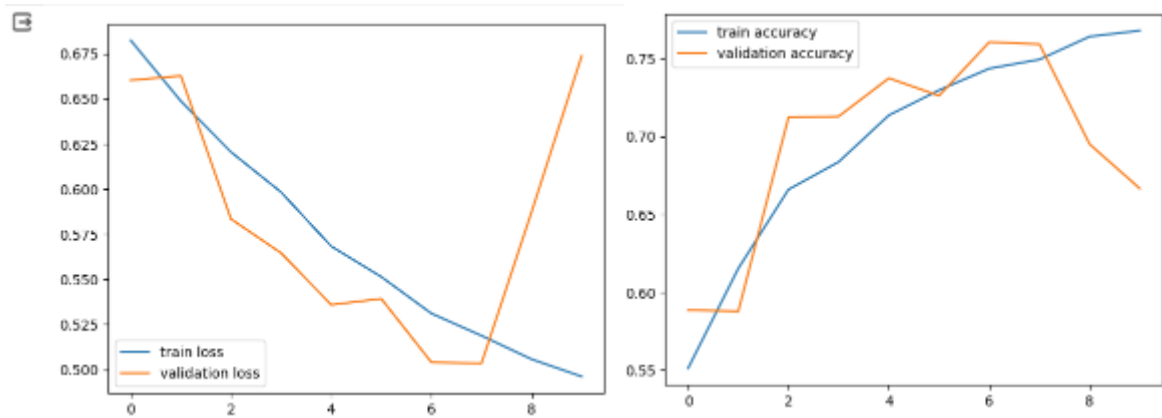


Figure 68 : Transfer Learning ResNet50

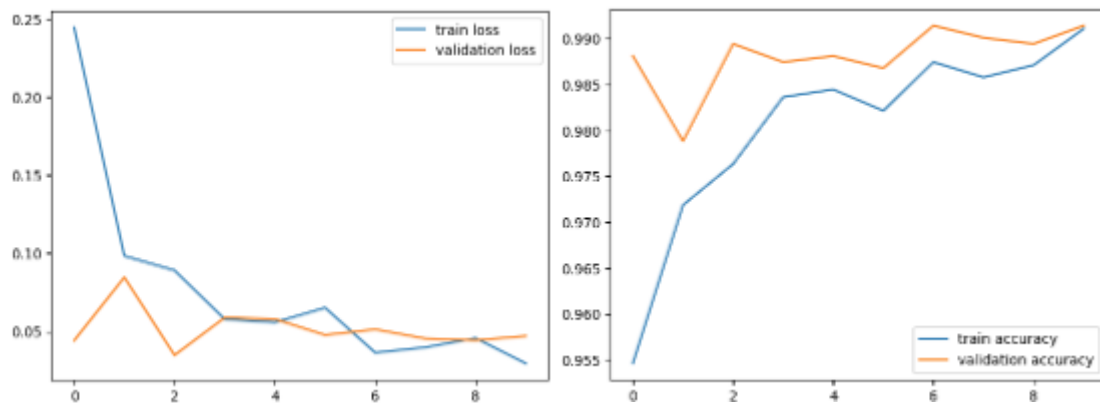


Figure 69 : DenseNet121

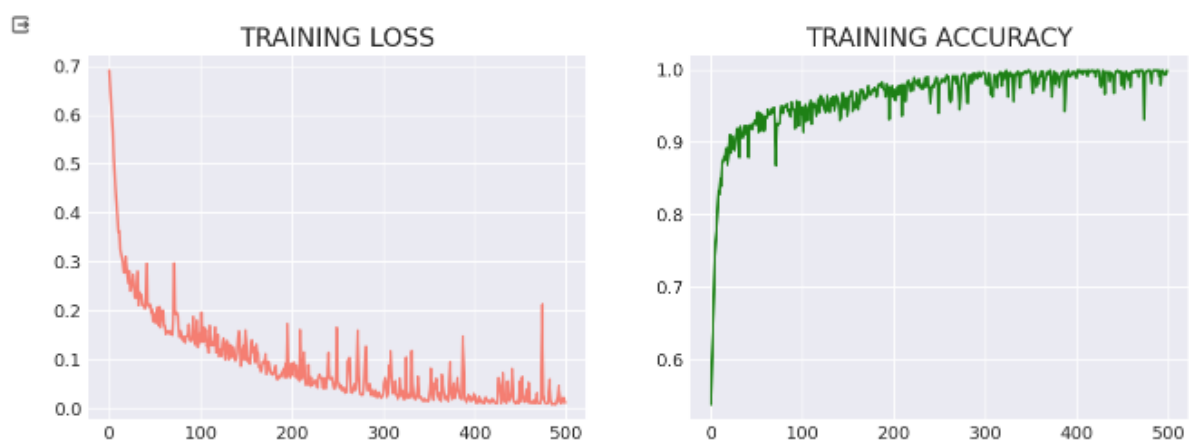


Figure 70 : Xception

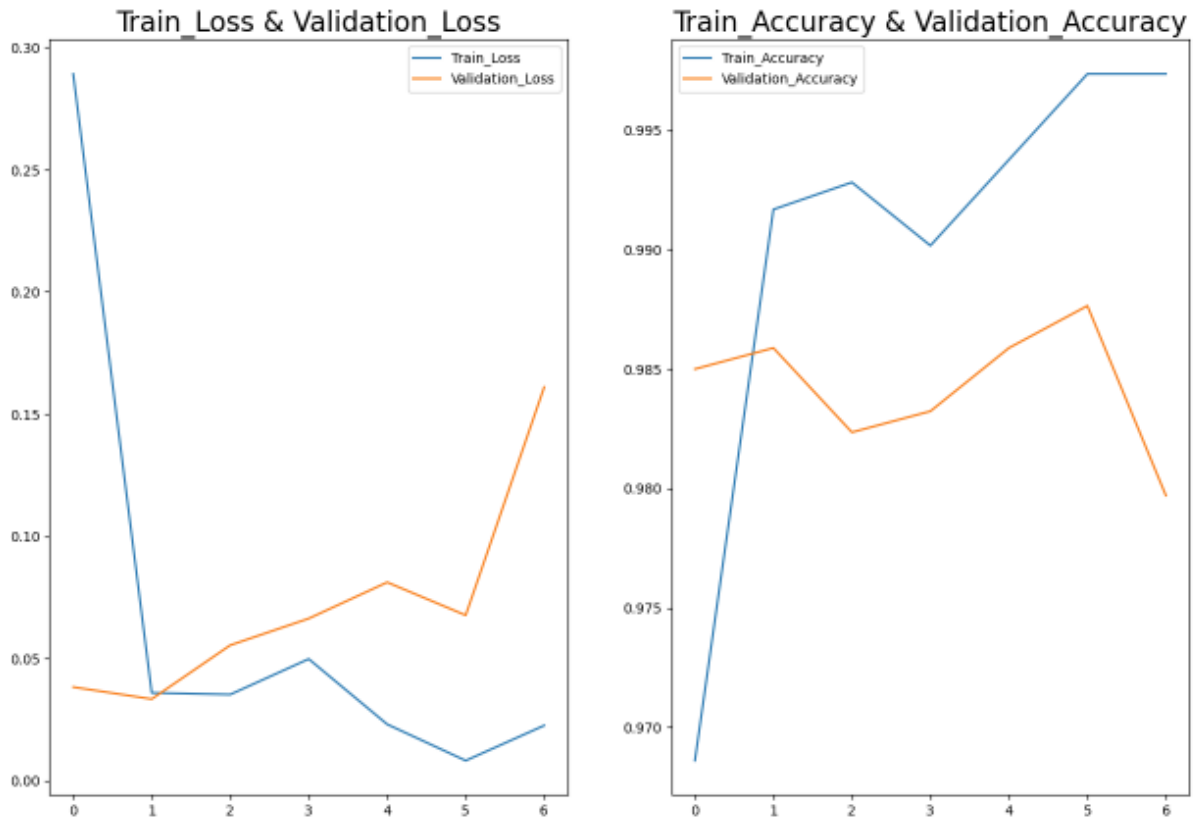


Figure 71 : MobileNetV2

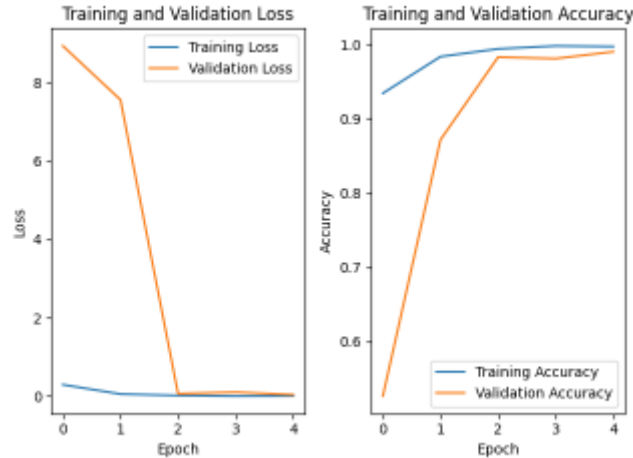


Figure 72 : Inception

It can be seen that there is an underlying pattern can be seen on the graph plotted for the train loss, validation loss and train accuracy, validation accuracy whereby for all of the five model proposed the value of train accuracy and validation accuracy increasing over the time. For example, the value of train accuracy for densenet121 increase from (0, 0.955) \rightarrow (9, 0.990) and this can also be seen with MobileNetV2 where it increases from (0, 0.960) \rightarrow (6, 0.998). But

there is an anomaly for Transfer Learning ResNet50 where although train accuracy and validation accuracy increases, it is the lowest among the others. The train accuracy goes from (0, 0.55) \rightarrow (9, 0.75) and validation accuracy goes from (0, 0.558) \rightarrow (9, 0.66). It can be deduced that the highest training and validation accuracy is Inception and followed by DenseNet121 and MobileNetV2. The lowest validation accuracy and train accuracy is Transfer Learning ResNet50.

There is also train loss and validation loss where train loss is the measure of how the performance of the machine learning model does during the training phase. It computes the difference between the actual target variable and the model prediction. Below shows how the training loss works:

1) Loss Function:

- It is known that loss function usually depends on the type of the dataset and the problem or the nature of the target variable where image classification or classification tasks it uses loss function such as binary cross-entropy and regression usually adopted MSE loss.

2) Training Loss:

- During the sequential model building, each iteration of training of epoch, loss function is applied to the model predictions.
- It also can be defined as the total of individual losses that have been calculated for each training data.
- This also can be defined as the measure of how well the model is fitting the training data.

3) Monitoring Training Progress

- It is required that in order to evaluate model performance, the training loss are monitored and if the training loss keep on decreasing, it shows that that the machine learning model is learning, adapting, and improving its ability to make predictions on training data.
- It is also very crucial to avoid overfitting and hence methods such as regularization and validation data are done to minimize overfitting.

4) Optimization

- Ther model performance can be improved by using optimization algorithms such as “Adam” optimizer, RMSprop and more.

On the other hand, validation loss is a metric used and proposed by statisticians to study and understand the performance of deep learning on validation dataset during model training. Comparison can be made to evaluate model efficacy by comparing it to a subset of data that is

called validation dataset where validation loss is done by summing up the mistakes of each validation set. Overfitting can happen if validation loss is higher than training loss and good fit model only can be achieved if both training and validation loss decrease and stabilize at certain point of iteration after it converged. It can be deduced a good model occurs when the value of train accuracy is almost similar with validation accuracy as can be seen with DenseNet121 where the train accuracy goes from (0, 0.955) \rightarrow (9, 0.990) and validation accuracy goes from (0, 0.987) \rightarrow (9, 0.990).

There are so many methods to prevent overfitting such as regularization by adding L1,L2 regularization, dropout, and early stopping. Early stopping pauses and stop training when the validation set seems to start to deteriorate and hence prevent the model from becoming overfit and dropout selectively and randomly dropout neurons during training of the dataset.

7.0 VISUALIZATION & CRITICAL ANALYSIS

This part here is to do visualization or in other words the model architecture workflow of the proposed neural network model for this assignment and at the same time doing critical analysis on the architecture and the performance of the model. It can be noted that there are 5 proposed deep learning method used for this assignment such as

1. Transfer Learning Resnet50
2. Transfer Learning DenseNet121
3. MobileNetV2
4. Xception
5. Inception

Where transfer learning is quite common to be use as it able to reduce amount of training time and it uses pre-trained model approaches and all of these model architectures are very useful for image classification problems where in this example face mask image detection. Figure below shows the visualization and the model workflow on how the architecture works for face mask image detection.

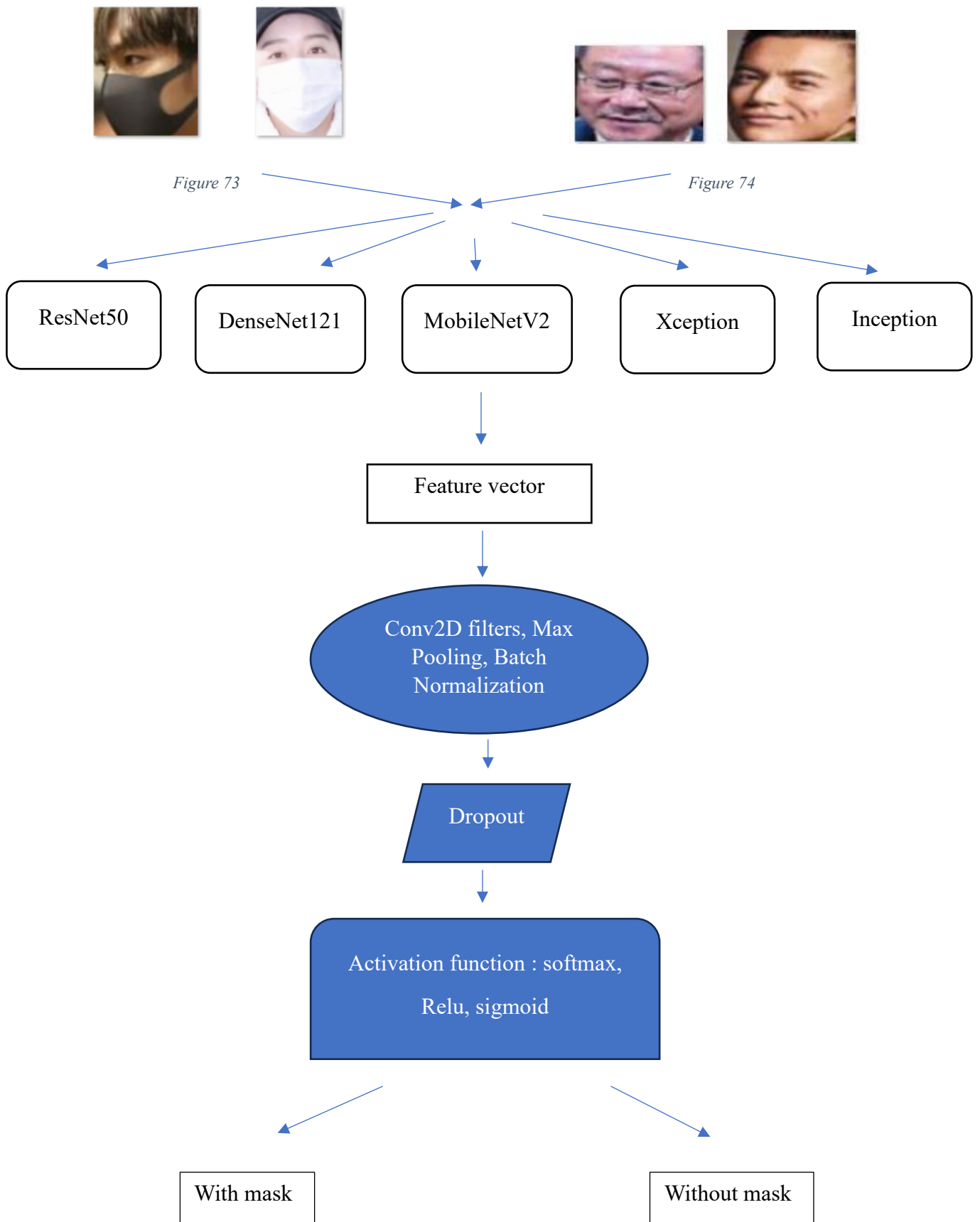


Figure 75

The main goal is eventually to classify image pixels with and without face mask but there are differences between all of the five proposed deep learning model where although most all of the technique has the same objectives but each of them has its own unique architecture and design principles.

For example, ResNet is known as Residual Networks where it uses residual learning in which the residual blocks allow network to learn residual functions, hence it's easier to train the hundreds of layers of deep neural networks without suffering from vanishing gradient problem. ResNet architecture such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152 where the higher the number of the variants it shows it has more layers.

DenseNet also known as Densely Connected Convolutional Networks where it passes information through dense connection of layers where these individual layers obtained features map from previous layers. It is the same as ResNet where DenseNet includes DenseNet-121, DenseNet-169, DenseNet-201, and DenseNet-264 and it is a good deep learning model that is able to reduce parameter count and increase efficiency and performance.

Xception used depthwise separable convolutions where it factorizes the normal and standard convolutions into depthwise convolutions. Inception on the other hand, adopts the concept of various or multi-scale feature extraction where it uses inception module. Inception also has its varying degree such as Inception V1,V2,V3 and V4. MobileNetV2 uses inverted residual blocks with linear bottlenecks, and it usually aims for low latency and good efficiency especially with embedded devices.

It can be seen with the codes in the previous model implementation where these architectures contained various and multiple layers of activation function, pooling, convolutional and normalization where the explanation had been done on the CNN architecture section previously. ResNet50 consists of 50 convolutional layers where and other type of layers such as pooling layers, fully connected layers, and normalization layers.

For example, for ResNet50 it consists of input layer where for the ResNet50 architecture the input image size is 128 by 128 pixels and which layer it is divided into few residual blocks and each block contained convolutional layers, activation layers and batch normalization layers. The residual blocks are also known as identity blocks, and it is a building block of ResNet50. It also contains pooling layers which include average pooling and max pooling and towards the end of the network, it has fully connected layers where it is used for classification and feature aggregation. Batch normalization is added to increase training speed and stabilization where

output layer on the other hand comprised of SoftMax function because it is mainly used for image classification.

Hence it can be concluded that, convolutional layers are very crucial in deep learning as large image dataset is needed to have a good quality of deep learning or neural networks training as during training process, it associates the local features in the input image with reciprocal regions of the mask. It is known that convolutional layer's main task is to extract information from input image where first layer starts with simple edge detectors and later on the layers become more complex as it will learn more complex features.

ResNet50, DenseNet121, MobileNetV2, Xception and Inception, all of them has one main goal where the final classification of the input image where in this case image datasets of face mask image is classified into few predefined classes. Probability distribution over the classes is created by connecting the final output of connected layer to activation function such as softmax, ReLU or sigmoid and the output will produce probability map which label the image or pixels into with mask or without mask class. Hence, the output of face mask image detection has label indicating image with face mask label as 1 and without is 0 and further explanation will be discussed on discussion section.

8.0 DISCUSSIONS

In the discussion part it focusses more on the confusion matrix of all of the five neural networks proposed where there will be confusion matrix of all the five neural networks and each of them has its results of the precision, recall, f1-score, and support. Confusion matrix is often used to assessed and evaluate the performance of classification model by presenting it against the actual labels of the dataset. The structure of confusion matrix can be shown below:

1. True positive : Also known as TP where it's the number of instances that are correctly predicted as belonging to positive class.
2. False positive : Also known as FP where it's a number of instances that are incorrectly predicted as belonging to the positive class.
3. True Negative : Also known as TN where it's a number of instances that are correctly predicted as not belonging to positive class.
4. False Negative : Also known as FN where it's a number of instances that are correctly predicted as not belonging to positive class.

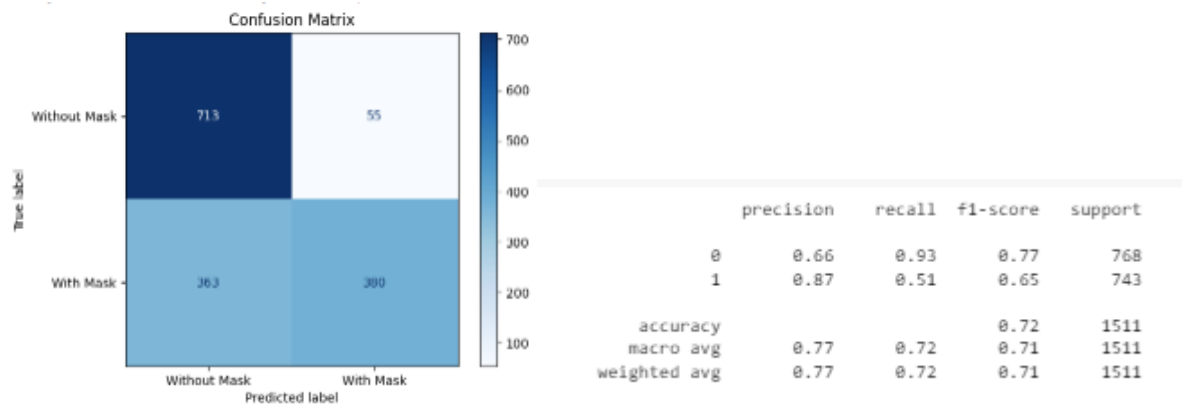


Figure 76 Transfer Learning ResNet150

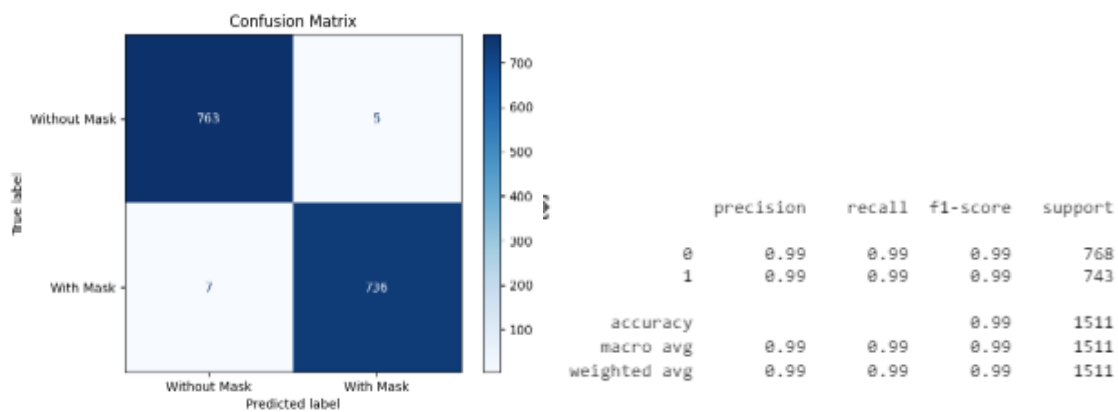


Figure 77 Transfer Learning DenseNet121

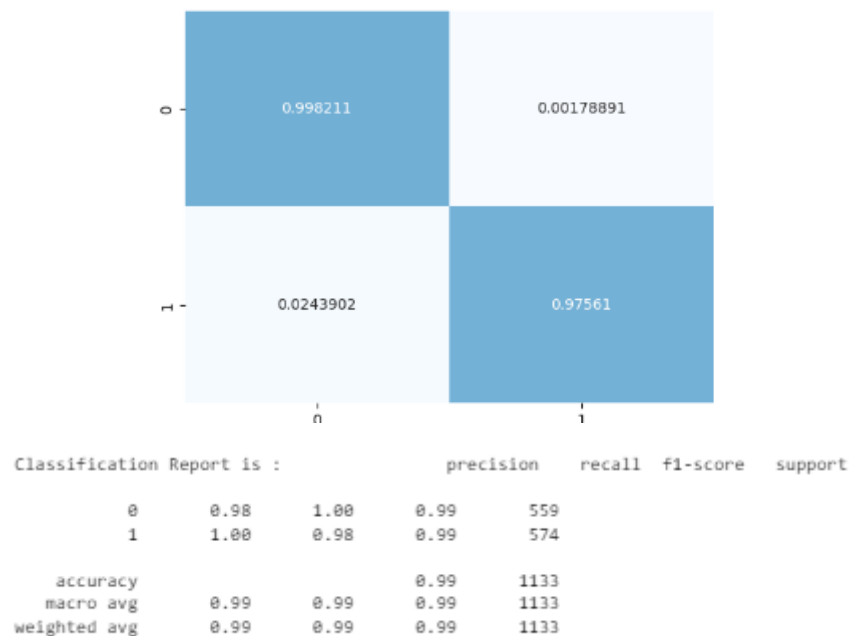


Figure 78 MobileNetV2

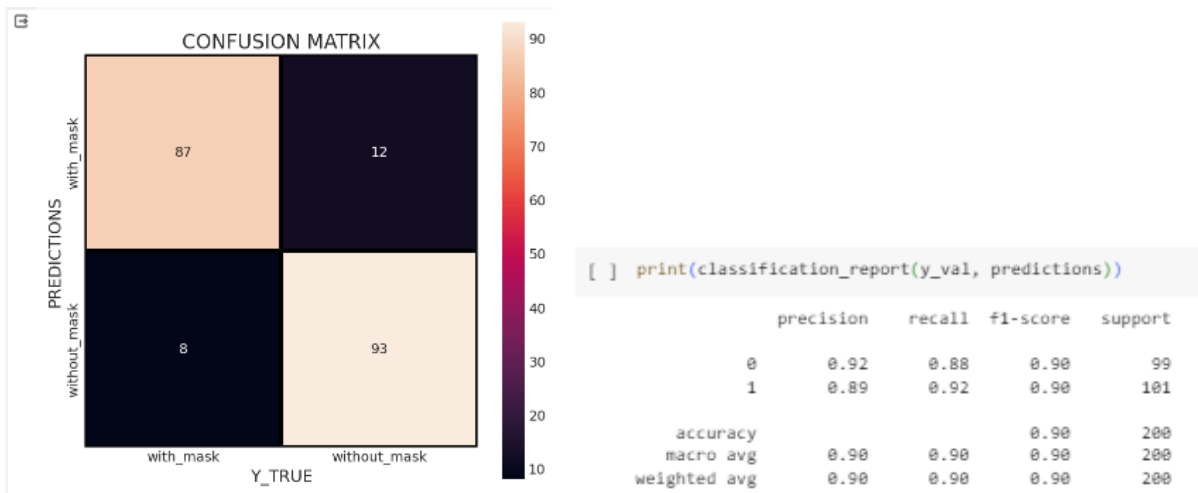


Figure 79 Xception

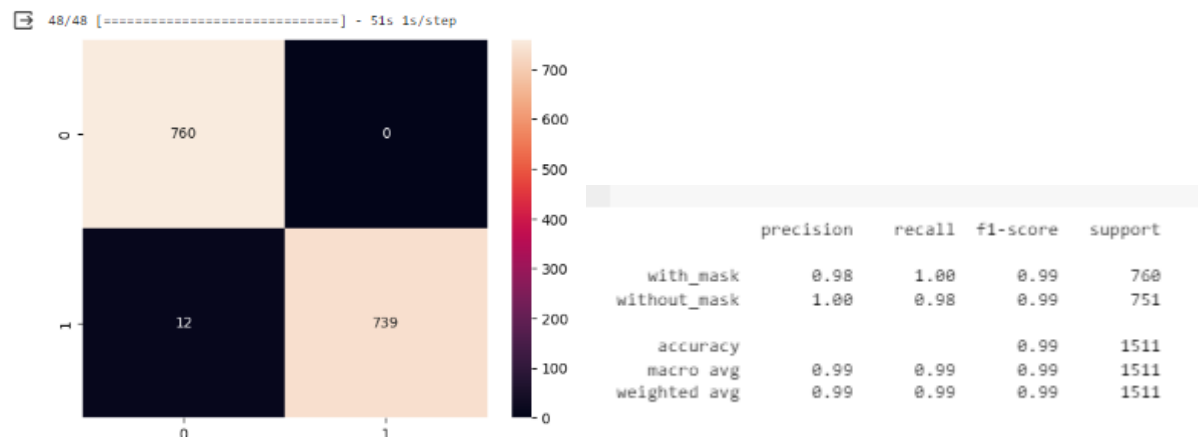


Figure 80 Inception

Figure above shows all of the confusion matrix for the neural network proposed where confusion matrix is used mainly to understand more of the classification models as most of data might have imbalanced class distributions. There are also a few important matrices that evaluates the performance of confusion matrix such as accuracy, precision, recall and f1-score.

Below shows some of the factors that resulted in higher value for precision, recall, f1-score, and accuracy as out of the five proposed neural networks, the one with the highest values of precision, recall, f1-score, and accuracy are Transfer Learning DenseNet121, MobileNetV2, and Inception. The lowest value of out of them is ResNET50 followed by Xception.

1) High accuracy

It is known that high accuracy shows that correct predictions can be made correctly and the neural network model able to classify correctly different class and labels. At the same time, high accuracy shows that the model able to generalize unseen data effectively and identifies any underlying patterns from the input dataset. The lowest value of accuracy is Transfer

Learning ResNet50 which at 0.72 and followed by Xception at 0.90 and the rest has the same accuracy value at 0.99.

2) High precision

Precision can be defined as how the proportion of true positive predictions out of all positive predictions from the model and high precision often the time shows that if the model positive class, the likelihood it to be correct is high. Thus, a high precision shows that it can reduced the likelihood of false positives and neural network model that has lowest precision is Transfer Learning ResNet50. This shows that this model is not suitable for face mask image detection.

3) High Recall

Recall on the other hand, is how the proportion of true positive predictions out of all positive instances in the face mask image datasets. A high recall shows that the machine learning model able to capture a huge part of the positive instances in the face mask image dataset and model with high recall as usual Transfer learning DenseNet121, MobileNetV2, and Inception.

4) High F1-score

F1-score shows the balance between the precision and recall if it has a high f1-score it shows that the neural network model already achieved good balance between the metrics of precision and recall.

There are so many reasons such as Transfer learning DenseNet121, MobileNetV2, and Inception has high values of accuracy, precision, recall and f1-score is due to all of these architectures are being trained on large datasets such as ImageNet and use TensorFlow for its deep learning method. This can be a good starting point as the pre-trained models undergoes fine-tuning on a specific task. Hyperparameter tuning able to influence the model performance where below shows the factors:

1. Tune batch size: Batch size refers to the number of samples processed by the neural network model for every iteration during the training process. Size of batch size plays a role as a large batch size has faster training time, but the drawbacks is that it requires more memory. On the other hand, smaller batch size leads to slower convergence but it excels in understands the data better.

2. Regularization techniques : It is one of the methods in deep learning where it is used to prevent overfitting. Some of the methods used are L1 regularization, L2 regularization, dropout, batch normalization and early stopping.
3. Tune number of epochs : Number of epochs refers to the number of times the whole dataset is being passed forwards and backward through neural networks during training process.
4. Learning rate : Is one of the essential hyperparameters technique where it is used to find out the size of steps taken during optimization stage to update the weights of the network. The model to overshoot the optimal solution can happened if the learning rate is high and slow learning rate will make the neural network model slowly converge.

Hence, the hierarchy the highest accuracy obtained is Transfer Learning DenseNet121 (0.99), MobileNetV2 (0.99), Inception (0.99), Xception (0.90), and Transfer Learning ResNet50 (0.72). It can be deduced there are limitations on using Transfer Learning ResNet50 where ResNet50 are specifically designed not for face mask image detection and hence it does not able to capture details related to the face mask image such as the type of face mask, face position and postures. ResNet50 is a very complex model, and it requires higher training time particularly fine-tuning face mask image datasets that is small and also its prone to overfitting especially with small datasets. Hence, to fix low accuracy for ResNet50 a bigger dataset must be used. Other than that, Xception has low accuracy for face mask image detection task as it is very complex architecture as it contained huge number of parameters and hence make it very computationally expensive to train.

9.0 CONCLUSION

It can be deduced that there is a different approach of neural networks as different deep learning models will results in different values of accuracy, precision, f1-score, and recall. This can be seen with the proposed neural network model with Transfer Learning ResNet50, Transfer Learning DenseNet121, MobileNetV2, Xception and Inception where each of these architectures has its own unique capabilities and features for face mask image detection. It can be concluded that deep learning is very beneficial to mitigate the risk of covid 19 pandemic as it ensures public safety and able to detect a person wearing a face mask or not in real time.

10.0 REFERENCES

- V. Kukkala, et al. “Advanced Driver Assistance Systems: A Path Toward Autonomous Vehicles “, IEEE Consumer Electronics, Sept 2018.
- L. Jiao, et al. "A Survey of Deep Learning-Based Object Detection," in IEEE Access, vol. 7, pp. 128837-128868, 2019
- N. Dalal, et al. "Histograms of oriented gradients for human detection", Proc. IEEE CVPR, pp. 886-893, Jun. 2005.
- P. F. Felzenszwalb, et al. "Object Detection with Discriminatively Trained Part-Based Models," in IEEE TPMAI, vol. 32, no. 9, Sept. 2010.
- T. Greenhalgh, M. B. Schmid, T. Czypionka, D. Bassler, and L. Gruer, “Face masks for the public during the covid-19 crisis,” *BMJ*, vol. 369, p. 1435, 2020.
- L. Nanni, S. Ghidoni, S. Brahmam, Handcrafted vs. non-handcrafted features for computer vision classification, *Pattern Recogn.* 71 (2017) 158–172, <https://doi.org/10.1016/j.patcog.2017.05.025>.
- Y. Jia et al., Caffe: Convolutional architecture for fast feature embedding, in: *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, 2014, doi: 10.1145/2647868.2654889.
- S. Saponara, A. Elhanashi, and A. Gagliardi, “Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19,” *Journal of Real-Time Image Processing*, vol. 11, pp. 1–11, 2021.
- L.R. Garcia Godoy, et al., Facial protection for healthcare workers during pandemics: a scoping review, *BMJ, Glob. Heal.* 5 (5) (2020), e002553, <https://doi.org/10.1136/bmjgh-2020-002553>.
- S.E. Eikenberry, et al., To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic, *Infect. Dis. Model.* 5 (2020) 293–308,.
- Wearing surgical masks in public could help slow COVID-19 pandemic’s advance: Masks may limit the spread diseases including influenza, rhinoviruses and coronaviruses – ScienceDaily. (n.d.). <https://www.sciencedaily.com/releases/2020/04/200403132345.htm>.

Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: a review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

H. Schneiderman and T. Kanade, “A statistical method for 3D object detection applied to faces and cars,” vol. 1, pp. 746–751, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, vol. 1, pp. 746–751, IEEE, Hilton Head, SC, USA, June 2002.

K. Hotta, “Robust face detection under partial occlusion,” *Systems and Computers in Japan*, vol. 38, no. 13, pp. 39–48, 2007.

Y.-Y. Lin and T.-L. Liu, “Robust face detection with multiclass boosting,” vol. 1, pp. 680–687, in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 680–687, IEEE, San Diego, CA, USA, June 2005.

J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image superresolution via sparse representation,” *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

G. Mesnil, Y. Dauphin, G. Xavier et al., “Unsupervised and transfer learning challenge: a deep learning approach,” in *Proceedings of the ICML Workshop on Unsupervised and Transfer Learning*, vol. 27, pp. 97–110, Washington, WA, USA, 2012.

Khan, M., Chakraborty, S., Astya, R. and Khepra, S., 2019, October. Face Detection and Recognition Using OpenCV. In 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS) (pp. 116-119). IEEE.

Venkateswarlu, I.B., Kakarla, J. and Prakash, S., 2020, December. Face mask detection using mobilenet and global pooling block. In 2020 IEEE 4th Conference on Information & Communication Technology (CICT) (pp. 1-5). IEEE.

A. G. Howard, M. Zhu, B. Chen et al., “Mobilenets: efficient convolutional neural networks for mobile vision applications,” 2017, <https://arxiv.org/abs/1704.04861>

C. Szegedy, V.V and sergeyloffe, j. Shlens and Z. Wozhna, “Rethinking the inception architecture for computer vision,” *CoRR*, vol. abs/1512.00567, 2015

G. Akhila Goud, B. Hemanth, “Face Recognition using Edge Detection of Layer” (2015) *IJSRSET* | Volume 1.

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

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & 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, 102692

Vinh, T. Q., & Anh, N. T. N. (2020, November). Real-time facemask detector using YOLOv3 algorithm and Haar cascade classifier. In 2020 International Conference on Advanced Computing and Applications (ACOMP) (pp. 146–149). IEEE. doi:.

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

Asif, S., Wenhui, Y., Tao, Y., Jinhai, S., & Amjad, K. (2021, May). Real Time Face Mask Detection System using Transfer Learning with Machine Learning Method in the Era of Covid-19 Pandemic. In *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)* (pp. 70–75). IEEE.

Sadeghin, S. (2020). *Face mask detection trained on ResNet - 50*. The Notebook Archive | Wolfram Foundation. <https://notebookarchive.org/face-mask-detection-trained-on-resnet-50--2020-12-6z8vw5z/>

D. Erhan, C. Szegedy, A. Toshev, D. Anguelov, Scalable Object Detection using Deep Neural Networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 2147–2154, <https://doi.org/10.1109/CVPR.2014.276>.

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, 2014.

J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, vol. 2016-Decem, pp. 779–788, doi: 10.1109/CVPR.2016.91.

R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-based Convolutional Networks for Accurate Object Detection and Segmentation, *IEEE Trans. Pattern Anal. Mach. Intell.* 38 (1) (2015) 142–158, <https://doi.org/10.1109/TPAMI.2015.2437384>.

K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2015), <https://doi.org/10.1109/TPAMI.2015.2389824>.

N.D. Nguyen, T. Do, T.D. Ngo, D.D. Le, An Evaluation of Deep Learning Methods for Small Object Detection, J. Electr. Comput. Eng. 2020 (2020), <https://doi.org/10.1155/2020/3189691>.

G.J. Chowdary, N.S. Punna, S.K. Sonbhadra, S. Agarwal, Face Mask Detection using Transfer Learning of InceptionV3, in: International Conference on Big Data Analytics, Springer, Cham, 2020, pp. 81-90, pp. 1–11, doi:10.1007/978-3-030-66665-1_6.

List of figures

Figure 1	8
Figure 2	9
Figure 3	9
Figure 4	12
Figure 5	37
Figure 6	38
Figure 7	38
Figure 8	39
Figure 9	40
Figure 10	41
Figure 11	41
Figure 12	42
Figure 13	42
Figure 14	43
Figure 15	44
Figure 16	45
Figure 17	46
Figure 18	47
Figure 19	47
Figure 20	48
Figure 21	50
Figure 22	50
Figure 23	50
Figure 24	52
Figure 25	54
Figure 26	55
Figure 27	56
Figure 28	56
Figure 29	57
Figure 30	57

Figure 31	58
Figure 32	58
Figure 33	59
Figure 34	59
Figure 35	60
Figure 36	60
Figure 37	61
Figure 38	61
Figure 39	62
Figure 40	63
Figure 41	64
Figure 42	65
Figure 43	66
Figure 44	67
Figure 45	68
Figure 46	69
Figure 47	70
Figure 48	71
Figure 49	72
Figure 50	73
Figure 51	74
Figure 52	75
Figure 53	76
Figure 54	77
Figure 55	78
Figure 56	79
Figure 57	80
Figure 58	81
Figure 59	82
Figure 60	83
Figure 61	84
Figure 62	85
Figure 63	86
Figure 64	87
Figure 65	88
Figure 66	89
Figure 67	90
Figure 68 : Transfer Learning ResNet50	94
Figure 69 : DenseNet121	94
Figure 70 : Xception	94
Figure 71 : M,obileNetV2.....	95
Figure 72 : Inception.....	95
Figure 73	
Figure 74	98
Figure 75	98
Figure 76 Transfer Learning ResNet150.....	101

Figure 77 Transfer Learning DenseNet121	101
Figure 78 MobileNetV2	101
Figure 79 Xception	102
Figure 80 Inception	102

List of tables

Table 1	7
Table 2	8
Table 3	11
Table 4	14
Table 5	18
Table 6	20
Table 7	31
Table 8	35
Table 9	53
Table 10	93