ANALYSIS OF PERSONAL PROTECTIVE EQUIPMENT CLASSIFICATION METHOD USING DEEP LEARNING

SITI ZAHRAH NUR AIN BINTI SILOPUNG

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

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: SITI ZAHRAH NUR AIN BINTI SILOPUNG
Date of Birth	: 26 JUNE 1999
Title	: ANALYSIS OF PERSONAL PROTECTIVE EQUIPMENT CLASSIFICATION METHOD USING DEEP LEARNING.
Academic Session	: SEMESTER 2 2021/2022

I declare that this thesis is classified as:

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

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

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

Certified by:

ZAHRAH	[illegible]
(Student's Signature)	(Supervisor's Signature)
990626-12-5952	
New IC/Passport Number Date: 17 June 2022	Dr. Rosdiyana Binti Samad Date: 27 June 2022

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

THESIS DECLARATION LETTER (OPTIONAL)

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

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

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

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

[illegible]
(Supervisor's Signature)

Date: 27 June 2022

Stamp: DR. ROSDIYANA SAMAD
PENSYARAH KANAN
FAKULTI TEKNOLOGI KEJURUTERAAN ELEKTRIK DAN ELELTRONIK
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN
PAHANG DARUL MAKMUR
TEL: 09-4246099 FAX: 09-4246111

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

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

SUPERVISOR'S DECLARATION

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

[illegible]

(Supervisor's Signature)

Full Name	: Dr. Rosdiyana Binti Samad
Position	: Senior Lecturer
Date	: 27 June 2022

(Co-supervisor's Signature)

Full Name	:
Position	:
Date	:

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

STUDENT'S DECLARATION

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

ZAHRAH

____

(Student's Signature)

Full Name	: SITI ZAHRAH NUR AIN BINTI SILOPUNG
ID Number	: EA18087
Date	: 17 JUNE 2022

ANALYSIS OF PERSONAL PROTECTIVE EQUIPMENT CLASSIFICATION METHOD USING DEEP LEARNING

SITI ZAHRAH NUR AIN BINTI SILOPUNG

Thesis submitted in fulfillment of the requirements
for the award of the
B.Eng (Hons.) Electrical Engineering (Electronics)

College of Engineering
UNIVERSITI MALAYSIA PAHANG

JUNE 2022

ACKNOWLEDGEMENTS

First and foremost, I would like to dedicate my special thanks of gratitude to Allah S.W.T for His great wisdom and giving me the strength in completing my Final Year Project.

I would like to express my sincere acknowledgement of gratitude to my supervisor, Dr. Rosdiyana Binti Samad whose has given valuable guidance and kind supervision throughout the project especially in providing me platform where I can get to increase my knowledge in machine and deep learning from nothing into successfully completing my Final Year Project.

Next, I would love to express my genuinely thanks to the Final Year Project Presentation's Panels, Professor Madya Dr. Zuwairie Bin Ibrahim, Dr. Mohd Syakirin Bin Ramli, Dr. Mahfuzah Binti Mustafa and Madam Marlina Binti Yakno for your guidance and advise to further improving my Final Year Project.

Special appreciation also to all members of Faculty of Electrical and Electronics, College of Engineering UMP for giving me a chance to do the project and research.

I also would like to express my genuinely thanks to my fellow friends, who always answering to my questions throughout the learning process of the whole journey of Final Year Project learning.

Last but not least, I would like to express my appreciation to my family for their prayers and moral support for me to complete this Final Year Project.

ABSTRAK

“Personal Protective Equipment” (PPE) dalam bahasa inggerisnya adalah peralatan yang dipakai di tempat kerja untuk mengurangkan pendedahan kepada bahaya yang menyebabkan kecederaan serius dan penyakit terhadap pekerja. Dalam projek ini, terdapat lima jenis PPE yang diambil kira, yang mana juga turut diiktiraf oleh “Occupational Safety and Health Administration“ (OSHA) yang mana merupakan satu badan persatuan keselamatan diiktiraf untuk memastikan PPE seperti topeng muka, pelindung muka, goggle keselamatan, topi keledar keselamatan dan jaket keselamatan. Untuk mengelakkan kerja yang memenatkan dalam menyemak secara manual sama ada pekerja memakai PPE atau tidak, pengelas PPE automatik dibina dengan menggunakan algoritma pembelajaran mendalam yang dipanggil “Convolutional Neural Network”, (CNN). Pengelasan ini dilakukan menggunakan Perisian Anaconda dan Jupyter Notebok yang menggunakan Python sebagai bahasa pengaturcaraan. Terdapat sejumlah 7500 imej dalam set data PPE, 6000 imej untuk latihan dengan dan 1500 imej lagi untuk ujian. Pengelasan dilakukan dalam dua cara, satu adalah dengan mengklasifikasikan PPE kepada 5 kelas dan satu lagi adalah dengan mengklasifikasikan ke dalam 2 kelas selepas beberapa parameter terbaik digabungkan dengan melalui pelbagai latihan dan ujian dengan menukar setiap parameter seperti nilai iterasi, fungsi pengaktifan, pengoptimum dan bilangan lapisan pada algoritma CNN. Dengan menggunakan pengelasan kepada 5 kelas, ketepatan latihan akhir ialah 89.39% dan ketepatan ujian 62.23%. Sebaliknya, dengan mengklasifikasikan PPE ke dalam kelas binari, PPE mempunyai ketepatan ujian akhir sehingga 88% untuk semua jenis PPE. Pelitup separuh muka mempunyai ketepatan akhir 95.60%, pelindung muka 94.32%, pelindung mata keselamatan 89.79%, topi keledar keselamatan 98.90% dan yang terakhir jaket keselamatan mempunyai ketepatan ujian 88.45%. Berdasarkan keputusan tersebut, algoritma CNN adalah algoritma yang baik kerana klasifikasi binari PPE mencapai hasil ketepatan yang tinggi.

ABSTRACT

Personal Protective Equipment (PPE) is equipment worn in the workplace in order to reduce the exposure to hazards that causing serious injuries and illness to people. In this project, there are five types of PPE to be considered which is also recognized by Occupational Safety and Health Administration (OSHA) such as face mask, face shield, safety goggle, safety helmet and safety jacket. To avoid a tedious work in manually checking whether workers wear PPE or not, an automatic PPE classifier is constructed by utilizing a deep learning algorithm called Convolutional Neural Network (CNN). This classification is performed using Anaconda and Jupyter Notebok Software that use Python as the programming language. There are total of 7500 images in the PPE dataset, 6000 images for training with and another 1500 images for testing. The classification is done in two ways, one is by classifying the PPE into 5 classes and another one is by classifying into binary class after the best combined parameters are obtained using multiple training and testing by changing the parameters such as epoch, activation function, optimizer and filter layer. By using classifying into 5 classes, the final training accuracy is 89.39% and testing accuracy of 62.23%. On the other hand, by classifying the PPE into binary class, the PPE has final testing accuracy up 88% for all PPE. Face mask has final accuracy of 95.60%, face shield 94.32%, safety goggle 89.79%, safety helmet 98.90% and lastly safety jacket has 88.45% testing accuracy. Based on the result, CNN algorithm is a good algorithm as the binary classification of PPE achieved high accuracy result.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

%	Percentage

LIST OF ABBREVIATIONS

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
ANSI	American National Standards Institute
BLS	Bureau of Labor Statistics
CDC	Centres for Disease Control and Prevention
CIFAR	Canadian Institute For Advance Research
CNN	Convolutional Neural Network
COCO	Common Objects In Context
Conv2D	2D Convolution Layer
COVID-19	Coronavirus Disease 2019
FC	Few-Shot Classification
GMM	Gaussian Mixture Models
HDF	Hierarchical Data Format
HM-SVM	Hybrid Multiclass Support Vector Machine
KNN	K-Nearest Neighbours
NADAM	Nesterov-accelerated Adaptive Moment Estimation
NumPy	Numerical Phyton
OpenCV	Open-Source Computer Vision Library
OSH	Occupational Safety and Health
OSHA	Occupational Safety and Health Administration
PPE	Personal Protective Equipment
R-CNN	Region-Based Convolutional Neural Network
ReLu	Rectified Linear Unit
RMSprop	Root Mean Square Propagation
SGD	Stochastic Gradient Descnt
SSD	Single-Shot Detector
SVM	Support Vector Machine
UMP	Universiti Malaysia Pahang
YOLO	You Only Live Once

CHAPTER 1

INTRODUCTION

1.1 Introduction

Since the last December 2019, a novel coronavirus forced global citizens to a new norm where various regulations have been mandating a national lockdown, closure of many sectors, social distancing and including wearing face mask for air filtration (World Health Organization, 2020). In Malaysia, the most significant impact of COVID-19 outbreak that can be observed is the usage of face mask for every individual has been set to be a mandatory action when going to the public and open places. This is one of the prevention steps to avoid being infected by the coronavirus. The face mask has been indirectly can be categorized as a personal protective equipment for every person nowadays.

According to the Occupational Safety and Health, (OSH, 2022), personal protective equipment, generally referred as PPE, is equipment worn to minimize the exposure to hazards that affect in a serious injuries and illness in workplace. Simply saying, PPE is item that has the ability to shield and protect the user against safety and health risks at workplace. There are many types of PPE recognized by OSH which are hard hat or safety helmet for head protection, safety goggles for eye protection, face mask for breathing protection, face shield for face protection, safety jackets for body protection and many more (Occupational Safety and Health Administration, 2004).

The utilization of PPE for this study can be considered in two situations which are firstly in the industrial workplace such as construction site where PPE such as safety helmet, safety goggles, safety jackets as well as face mask due to the recent pandemic. Secondly, in a workplace which involves the health frontliners such as hospital where face mask and face shield PPE are being used. Hence, this application is capable of detecting PPE used by person in these two working spaces or conditions.

In this study, an application to classify the presence and absence PPE using deep learning method is developed. It is done by setting dataset that consists of a set of PPE images in suitable directory and implementing significant algorithm. The deep learning algorithm that has the ability to extract features from the images then further classify the images according to the similar feature characteristics it learns from the feature extraction process. There are few training parameters that plays important role in extracting features of the PPE images. Generally, these parameters are utilized almost in every deep learning algorithm especially neural networks such as dropout, batch normalization, optimizer and activation function including learning rate and momentum, (Varma & Das, 2018). The parameters contribute the most in helping the classification algorithm to understand about the PPE characteristics and information to produce desired output of the classification.

The method of deriving useful information from images can be considered as a part of Artificial Intelligence (AI). AI is a term used to describe a software or machines that capable to mimic human's functions such as performing task, learning and problem solving. Deep learning algorithm is type of machine learning and a part of AI. Computations will be performed on large amount of data using artificial neural networks and training the software or machine to learn from examples. There will be sets of data and deep learning algorithm will train and test the data such as images or videos to derive the desired output images. In this application, the output should be the classification images of the PPE.

1.2 Problem Statement

The usage of PPE has been long implemented in the workspace and currently the usage of face mask and face shield are very significant due to the coronavirus outbreak. The absence of PPE in the workplace can cause a serious injury to the employee for example not wearing helmet will cause a serious head injury. Besides, starting 1 April 2022, Malaysia has begun the "Transition to Endemic" phase and even so the new Standard Operating Procedure (SOP) is still implemented mandatory to wear face mask in public places and encourage workers to wear face mask in workplace, including wearing appropriate PPE, (Geyzel, 2022). That is because person without face mask would lead to infection and quick spread of the virus among the workers. This shows how important the presence of PPE to be utilized accordingly weather in industrial site as well as in the clinical workspace. Unfortunately, not all personnel will be abiding the rule to use PPE in which there will always few persons who will not cope. These may be because the worker sometimes forgets to bring along their PPE when coming to work especially when they are in rush. The problem in this scenario is faced by the person in charge to check whether workers are wearing PPE or not as the manually checking process is tedious. Secondly, the person in charge also will take more time since the manually checking process is time consuming and inconvenience especially in the morning time where a lot of workers will come at workplace almost at the same time.

1.3 Objectives

i. To utilize deep learning method which is Convolutional Neural Network (CNN) in classification of Personal Protective Equipment (PPE).

ii. To analyse the performance of Convolutional Neural Network (CNN) algorithm in term of classification accuracy.

1.4 Project Scope

The PPE classifier application has few scopes and limitations such as only five types of PPE are being considered in this project. The PPE are yellow safety helmet, safety goggle, face mask, face shield with blue band and yellow safety jacket. This project is a fully simulation project in which no hardware arrangement is needed. Next, the dataset used is a combination between online images from sources like Kaggle, 123RF and iStock and own images that have been captured using smartphone's camera of 64 megapixels. The example of dataset is shown in Table 1.1.

Table 1.1 : Example of Dataset Images

Type of PPE	Example images in PPE dataset
Safety Helmet	
Safety Goggle	
Face Mask	
Face Shield	Face Shield Protective Isolation Mask Face Shield Direct Splash Protection
Safety Jacket	

The own images are captured at Universiti Malaysia Pahang (UMP) dormitory and the datasets are all in .jpg format. The size of images are between 100x100 pixels to 480x480 pixels. Other than that, there are in total of 7500 images which divided into 6000 images for training while another 1500 images for testing. The ratio of training to testing is 80% to 20% respectively. This ratio is chosen because the 80% training set of the total images will provide sufficient and more data to be learn by the classifier, (Gholamy, Vladik Kreinovich, & Olga Kosheleva, 2018). Next, phyton is used as the programming language and software like Anaconda and JupyterLab are used as platform to execute the coding in this project. There are also few libraries that have been utilized such as NumPy, Keras, TensorFlow and OpenCV. Last but not least, the laptop specifications include Intel Core i5 processor, 8GB RAM and 475GB SSD storage.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Personal Protective Equipment (PPE) that protects workers from workplace injuries and other hazards is addressed in specific standards by OSHA. It generally divided according to the expertise such as general industry, maritime and construction. The standards have been originated by the American National Standards Institute (ANSI) for the equivalent categories of the PPE (United States Department of Labor, 2022). According to United States Department of Labor, eye and face protection lies under 1910 Subpart I, standard number 1910.133 and section (a)(1) stated that the employer should make sure that employee utilize appropriate eye or face protection when being exposed to eye and face hazards such as flying particles, liquid, gases or vapours chemical, molten metal or potentially light radiation (United States Department of Labor, n.d.).

Other than that, head protection lies under standard number 1910.135 section (a)(1) which requires employer to endure that employee uses a protective helmet when working in workplace that can cause injury to the head due to falling objects (United States Department of Labor, n.d.). Besides, as for respiratory protection at standard number 1910.134 section (a)(1) states that to prevent of those occupational diseases due to the respiring of the polluted or contaminated air with harmful dusts, fumes, fogs, gases, mists, smokes, or vapours (United States Department of Labor, n.d.). The main objective is to halt atmospheric contamination and the same concept applied for nowadays pandemic as the contaminated air also means the existence of Covid-19 viruses in the air.

In conjunction to this matter, OSHA has worked together with the Centres for Disease Control and Prevention (CDC) to propose a guidance preventing the spread of

Covid-19 in the workplace on January 29,2021 which includes the obligation to wear face mask in the workplace for both vaccinated and unvaccinated people (United States Department of Labor, 2021). Furthermore, the utilization of face mask has long being implemented by the clinical practitioners in the clinical workplace such as hospital and clinics. Similarly with the face shield which often use by the frontliners as one of the protective equipment for Covid-19 primarily. Due to the strict regulation to wear PPE in the workplace, it would be very convenience to have an automatic PPE classification system instead of just manually checking the PPE compliance by the person in charge. Therefore, reviews from articles and journals are important in order to get the main idea of how to conduct the classification or figuring out the best method to do the PPE classifier.

2.2 Object Detection Using Machine Learning

Generally, machine learning is a term to refer when computers learn from data. According to (Wolfewicz, 2022), machine learning describes the intersect of computer science and statistics where algorithms are used to perform specific task without being explicitly programmed. Machine learning recognizes patterns in the data and able to make predictions when a new data comes. The examples of commonly used machine learning algorithms are linear regression, Support Vector Machine (SVM) and K-Nearest Neighbours. Machine learning is also used to perform operation like object classification, recognition and detection.

2.2.1 Object Detection Using Linear Regression Algorithm

Linear regression algorithm is a supervised machine learning model in which the model discovers the best fit linear between the independent and dependent variable. There are two types of linear regression which are simple linear regression where only a single independent variable is present and multiple linear regression that has more than one independent variables for the model to find the relationship (Deepanshi, 2021). In a paper by (Zhang, Wang, Davoine, & Pan, 2012), the linear regression algorithm was implemented to detect the skin colour in a multi-classification approach and the result had showed a good generalization ability compared to other two approach called Gaussian Mixture Model (GMM) and Bayesian classifier. Linear regression had better performance in discriminating skin colours even from the similar skin tone.

Other than that, the linear regression algorithm also had been implemented in analysis of the head pose without the need for facial feature detection in a research by (Yu & Liu, 2014). The experiment was conducted on multiple databases consists of 337 subjects. In each subject, there were 13 different views where the face's images were taken and also various number of images for each pose. At the end of the research, result showed that linear regression was a good method to estimate the head pose and it had the ability to avoid exactly extracting the face features in which useful for estimation under uncontrolled conditions. Thus, linear regression is a good machine learning algorithm to perform object detection.

2.2.2 Object Detection Using Support Vector Machine (SVM).

Similar like linear regression, SVM is also categorized under supervised machine learning algorithm that mostly can be used for classification method. In SVM, the classification is performed by finding the hyper-plane that differentiates between two different classes where initially the value of each feature being the value of a particular coordinate in the x-y plane of the SVM algorithm (Ray, 2017). SVM works by selecting the hyper-plane which classifies the classes correctly according to maximizing margin and has a feature to ignore outliers. In a paper wrote by (Yang, Wang, & Yang, 2012), the SVM classifier for microcalcification had achieved quite a high sensitivity which was up to 94% while a small false positive rate at 0.5%. The overall classification performance by SVM was a notably higher in true detection rate and SVM was effective to achieved a high detection accuracy.

Another research conducted by (Selamat & Rais, 2015) showed that SVM had achieved a consistent and high accuracy rather than classic strategies approximately ranging from 4.5% to 18.1%. The SVM was implemented to recognize face and also using polynomial kernel to assists with the Hybrid Multiclass SVM (HM-SVM). Other than that, (Liu, Dai, & He, 2011) had achieved an accurate and efficient performance of up to 91% hit rate accuracy in real-time on-road vehicle detection by applying SVM approach with a less computation cost. The test datasets consist of five scenes with more than 13000 monochromatic images at 1024 x 384 pixels resolution. This had showed that SVM classifier is a good algorithm to perform the detection as well as had a low cost in computation.

2.2.3 Object Detection Using K-Nearest Neighbour (KNN).

KNN is a machine learning classification technique but it can be used in both classification and regression application. According to (Srivastava, 2018), KNN is widely used because it is easy to interpret output, low calculation time and good in prediction. Besides, the "K" in KNN refers to the nearest neighbour and "K" is determined by considering the training error rate and validation error. As for instance, the value of "K" is always accurate when it is the closest to any point itself. A research conducted by (George, Potty, & Jose, 2014) showed that KNN achieved accuracy of 66.6% when it was implemented in the smile detection application. The classification was done by extracting mouth and eye pair from the images and setting the value of "K" to 1.

Besides, a study by (Bouaich & Tairi, 2019) that implement KNN in vehicle detection gave a satisfactory result and able to identify region of interest when tested on three videos of different weather condition which were windy, raining and sunny. In the vehicle detection application, KNN was used to subtract the background based on adaptive and non-adaptive probability density as well as threshold to identify shadow. So, the application able to detect pixels that belong to the foreground which was the vehicle as well as differentiate between vehicle and shadow.

2.3 PPE Detection Using Deep Learning Algorithm

The object detection using deep learning has been long implemented by the previous researchers. Unlike machine learning, in this section, the deep learning algorithm will directly further discuss on PPE as the object for classification and detection. Specifically, the detection of the PPE was mostly on detecting the face mask by using various methods under deep learning. The methods are Convolutional Neural Networks (CNN), Single Shot Detector (SSD) and You Only Look Once (YOLO).

2.3.1 Detection of PPE Using Convolutional Neural Network (CNN)

CNN is widely known a type of supervised Artificial Neural Network (ANN) that usually used for object classification or recognition. It takes images or videos as its input and has a known target output so that the machine will learn to recognize those classifications through training process that passing through a set of parameters such as layers, filter, activation function, and layers.

According to an article that utilized CNN to detects and tracking moving objects using TensorFlow object detection Application Programming Interface (API), it had achieved specificity of 91.24% and good accuracy of 90.88%, (Mane & Prof.Supriya Mangale, 2018). The detection able to detect the object's location by 150 frames per second throughout obstacles like illumination and occlusion. Other than that, another research that used Region Based Convolutional Neural Network (R-CNN) to detect high resolution masked and non-masked faces, (Gathani & Shah, 2020). R-CNN is an extension of CNN that mainly focuses on extracting region proposals from the input image and will be computed by a convolutional neural network. As a result, the non-masked accuracy was 98.61% which was quite high and a little lower accuracy in detecting masked face, 68.72%. The detection was successful since this method was using R-CNN which exists the localization process where the exact location of the object in that particular image can be extracted so that bounding box can be drawn for detection purpose. In addition, another research was using CNN model and applying to a base version of an algorithm for detection had results in obtaining high performance result accuracy, precision, recall rate and F1-score up to 96%, (Gathani & Shah, 2020). In another paper, an accuracy of 79.14% with precision of 80% were achieved when training 13000 images using dataset of Residual Network (ResNet), (Delhi & Thomas, 2020). It has utilized CNN for classifying the object then had a Region Proposal Network (RPN) that generates region proposal that helps in detecting the particular object.

Processing time of CNN may be slower compared to any other advance deep learning algorithm since CNN is the basic of deep learning algorithm. However, observing from the results of the papers, CNN method is also good in term of accuracy especially when performing the object classification. In June 2017, research shown that CNN was capable of achieving record-breaking results which was to obtain a lesser error rate than the previous research when classifying a large amount of dataset using purely supervised learning CNN, (Krizhevsky, Sutskever, & Hinton, 2017) . This shows that in term of object classification, CNN is definitely one of the effective options to perform any image classification.

2.3.2 Detection of PPE Using Single Shot Detector (SSD)

SSD is one of the well-known algorithms in deep learning where it utilizes the transfer learning to produce a great result. SSD has a high-performance quality and that is why it usually used for object detection in real-time. According to an article that had used SSD for detection and monitoring system, (Mingyuan Xu, Wang, & Li, 2020). SSD had fast detection speed and good performance because it had bigger layer so the larger the target will be. Nevertheless, although the performance was good but SSD also unable to meet requirement for detections when directly applied to certain object due to various kind of features and scene complexity.

In addition, SSD network also able to generate good scores for the detection of presence object because it had combined predictions from a few feature maps with different resolution. This is a good step to handle the various sizes of images in the dataset. SSD is straightforward and easy to train because SSD requires object proposals that crucial in a detection component, (Liu, et al., 2016). Overall, SSD is able to achieve a competitive accuracy method and it is quite unsuitable to be used for detecting because there are still detection requirements that left unsatisfied. Moreover, SSD algorithm also using VGG-16 network, under CNN architecture that crucial in extracting features and improving results.

2.3.3 Detection of PPE Using You Only Look Once (YOLO)

YOLO is one of a famous deep learning algorithm and has well performing architectures especially for real time object detection. YOLO is based on regression that predicts classes and bounds boxes for the entire picture and run only once instead of just selecting a certain part from the image. This benefits YOLO in term of faster detection time.

According to (Jian & Lang, 2021) that compares the accuracy and detection speed between YOLO and Faster R-CNN, result had shown that YOLO was an efficient and faster detection speed than the CNN model and had up to 86% mean average precision (mAP) for mask detection in real time. In other articles that evaluate performance of YOLO versus other method, it used Haar Cascade method as classifier before processed by YOLO to do object detection and resulting in achieving the highest accuracy compared to the other method for the detection, (Vinh & Anh, 2020). Next, YOLO had a higher accuracy because it only needs a single forward propagation pass that able to detect object on one input image. Besides, in detecting face with mask, YOLO had results in up to 90% precision, 70% F1-score, and a faster execution time, (Liu & Ren, 2021). It had used quite a large size of datasets which have 3145 images for training and 853 images for testing that being classified into three different classes, “With mask”, “Without mask” and “Mask worn incorrectly”. In term of its overall validation set performance, YOLO also had shown a higher percentage performance in extracting feature when being trained using the traditional classification network Few-Shot Classification (FC) layer predictor. However, when comparing the performances of YOLO to a different model, Faster R-CNN, this extension of CNN tends to perform up to 3% better than YOLO in term of their precision and F1-Score.

Other than that, (Protik, Rafi, & Siddique, 2021) had developed an automated system for PPE detection such as face mask, face shield and hand gloves using YOLO with Roboflow computer vision developer framework for image augmentation in training set with 8000 iterations and Tensorflow weight format to check performance of real time detection. The result shows that the average precision of individual classes was between 70% to 87% which indicate that the detector had a good performance. Besides, the precision, recall rate, and F1-Score metrics scores 78%, 80% and 79% respectively.

Based on the articles and research, YOLO definitely had a promising good performance for PPE detection especially for the real time application. YOLO gave a faster execution time and quicker processing time for object detection but in term of its overall performance, CNN algorithm had higher precision and F1-Score than YOLO does. It can be concluded that YOLO sacrifices accuracy in order to achieve a higher speed detection time.

Furthermore, there are also situation where YOLO only perform detection process after being passed from the classifier which was done by the other algorithm such as Haar Cascade (Vinh & Anh, 2020) . Moreover, in most of the references that implement YOLO algorithm to detect object has utilized a pre-trained model from various open-sources platform such as Common Objects In Context (COCO) datasets (COCO, n.d.), Canadian Institute for Advance Research (CIFAR) and many more (Papers With Code, n.d.). Thus, the pre-trained datasets are not really practical this project since it provides less opportunity to learn how to train the datasets manually and improves knowledge in utilizing deep learning for PPE detection.

2.4 Summary

In summary, both machine learning and deep learning approach have the ability to perform object classification or detection. However, by comparing both approach, deep learning provides a good result especially in term of accuracy. In a paper by (George, Mary, & K S , 2013) that comparing the accuracy between ANN and KNN in vehicle detection and classification from acoustic signal application using 160 vehicles in different categories. Result had showed that KNN achieved average accuracy which was 50.62% while the ANN achieved higher accuracy which was 73.42%. As had been discussed in previous explanation, KNN which is one of machine learning algorithm recorded a low performance compared to ANN, which is categorized under deep learning. Therefore, it is proved that deep learning algorithm is a good choice to be utilized in the PPE classification application in this project.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter explains in detail about the chosen method that being used to perform the classification of the five types of PPE which are face mask, face shield, safety goggle, safety helmet and safety jacket. The method majorly is inspired by the philosophical research conducted in the literature reviews mainly to obtain the fundamental understanding about various methods and choosing the appropriate one to be implemented in this project. Besides reviewing articles and journals, researching from online platforms such as websites related to object detection or classification also has influenced in the methodological choice. Those websites basically utilize python as it is one of a commonly used programming language worldwide. In this chapter also includes specific explanation of the step-by-step project flow from start to finish.

3.2 Overview of Overall Project Flow

Figure 3.1 illustrates the overall flowchart of this project. It starts with visualising the dataset that contains pictures of the PPE which are face mask, face shield, safety goggle, safety helmet and safety jacket. Then follows by the image augmentation and pre-processing where the images are being rotated, shifted, normalized, sheared, zoomed, and flipped. Next, the model is built in which appropriate layers, activation function, and optimizer are applied. After that, model training process is then completed. The model is trained using two ways which are firstly training for multiple classes and secondly binary class. In multiple classes, there are 5 classes that classify each type of PPE into different class for example class 0 belong to face mask, class 1 belong to face shield, class 2 is for safety goggle, class 3 is for safety helmet and lastly class 4 belong to safety jacket. Moreover, in binary class, each of the PPE is trained separately and only classify into two classes at the end which are 'with' or 'without' PPE. After training is completed, model testing can be done to get the classification probability result in term of percentage.

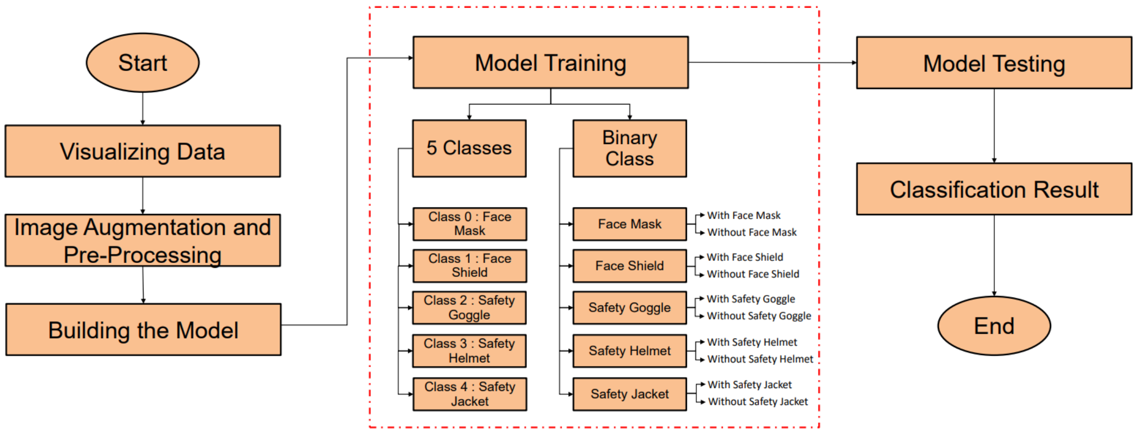


Figure 3.1: Overall Project Flowchart

3.3 Visualising Data

Visualising the data is basically how the datasets are being split-up in different folders before started to write any coding or programming. The dataset which also means the images are sorted according to appropriate folders. This is a crucial first step that needs to be completed correctly so that will be easier for the machine to analyse and understand the datasets especially when they are being called in the programme.

Figure 3.2 shows how the data for 5 classes are being sorted and for 5 classes the dataset consists of 2500 set of images in total. Next, two different folders for training and testing are created in which containing 2000 images and 500 images in the folders respectively. The ratio of images between train and test 80% to 20%. The training has a larger number of pictures compared to the one in testing folders because naturally learning process needs a huge number of inputs it can learn from. In each training and testing folders, there are five folders rename as "FACEMASK", "FACESHIELD", "GOGGLE", "HELMET" and "SAFETYJACKET". In the training folder, each PPE has 400 images while in the testing folder there are 100 images for each of the PPE.

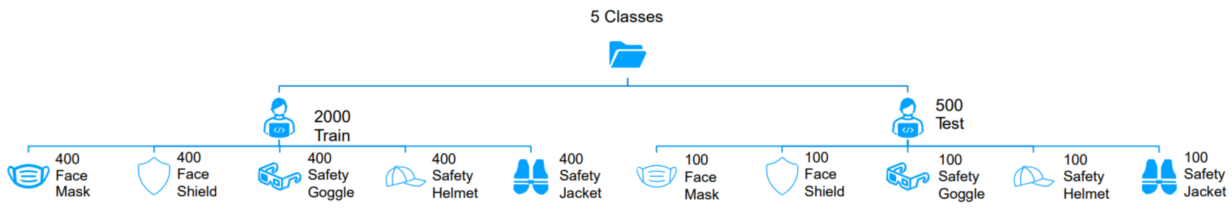


Figure 3.2 : Visualising Data for 5 Classes

Next, Figure 3.3 illustrates the visualising data for binary classification. In total there are 5000 set of images for binary class and the PPE is sorted by having one folder for each of the PPE. Thus, there are five folders which are each one for face mask, face shield, safety goggle, safety helmet and safety jacket. In every folder, there are another two folders called as train and test folder. Inside the train and test folder there are also another two folder that represent 'with' and 'without' PPE respectively. For example, there are 'with face mask' and 'without face mask' folders in each train and test folders of face mask and same condition applied to another four PPE. As for the number of images, there are 400 images in each 'with and without' PPE in train folder and 100 images in each 'with and without' PPE in test folder.

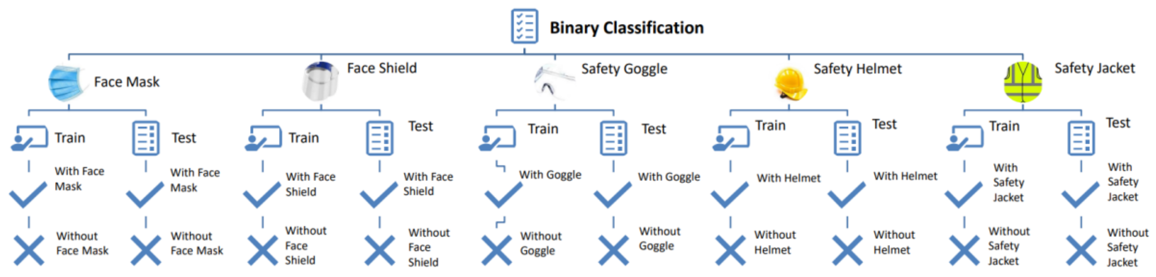


Figure 3.3 : Visualising Data for Binary Class

3.4 Image Augmentation and Pre-Processing

Image augmentation is a familiar technique used in deep learning network especially when a large amount of data is existed. Image augmentation can be defined as a technique of modifying the existing original data to produce a new data that has been enhanced for the model training process. The modification is generated from input data itself therefore it benefits in saving time instead of collecting the data manually. In addition, the deep learning model might encounter difficulties in learning the patterns or desired object from the images due to disturbance such as input image is massive in size, or the machine might not be able to differentiate between foreground and background from the picture. This significantly will impact the performance of the model. Thus, augmentation approach such as appropriate shifting, rotating, zooming, and flipping can be carried out to create variations in the dataset which allows to generalize model and will performed better when dataset is rich and sufficient.

Figure 3.4 shows the block diagram of image augmentation specify with their parameters. Firstly, ImageDataGenerator of Keras library is included in Jupyter Notebook because it provides an easy and fast procedure to augment the input pictures. Keras also provides the access to various augmentation techniques such as the normalization, rotation and many more. Besides it also ensures that the new variations of images are received by the model at each iteration or epoch.

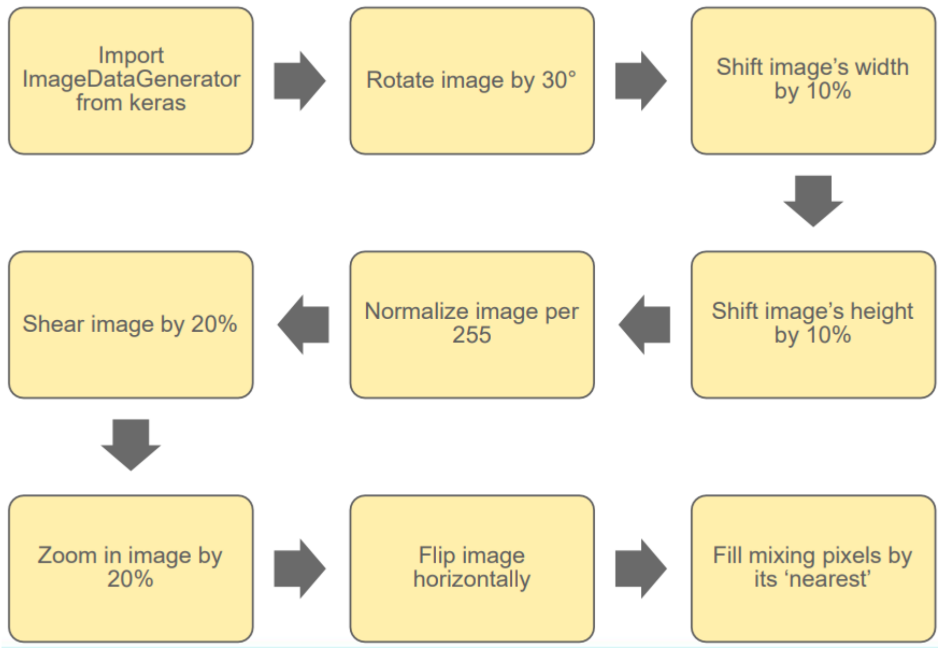

Import ImageDataGenerator from keras
Rotate image by 30°
Shift image's width by 10%
Shift image's height by 10%
Normalize image per 255
Shear image by 20%
Zoom in image by 20%
Flip image horizontally
Fill mixing pixels by its 'nearest'

Figure 3.4 : Image Augmentation and Pre-Processing Block Diagram

Next, ImageDataGenerato*r* library also allows to perform random rotation that generally ranging from 0 until 360 degrees by using *rotation_range* function. Here all the PPE images are being rotated randomly between 0 and 30 degrees. Maximum 30 degree is sufficient to rotate the image as a high degree of rotation will cause even more pixels to be rotated out and some part of the object which is PPE may have been moved outside of the images.

After that, the images are being shifted in two different directions which are horizontally and vertically. Horizontal shifting is when the image's width is shifted, and vertical shifting is when the image's height is shifted. Using *width_shift_range* and *height_shifted_range* function for width shift and height shift respectively, PPE dataset is shifted randomly by 10%. Float number of 0.1 is used to represents that the entire image is being shifted in width along x-axis and height along y-axis by 10% instead of just simply shifting by pixel values.

Other than that, the dataset is rescaled by normalizing it using rescale command. Since the PPE images are coloured images in (Red, Green and Blue) RGB coefficients ranging from 0 to 255 which are quite high to be feed into the model to be processed. Therefore, changing into values between 0 to 1 by scaling with a 1/255 factor. The normalization process helps to change the pixels intensity values so that it becomes smaller and the computation becomes faster and easier (Ponraj, 2021).

Then, shear image using *shear_range* function also by 20%. Shear means cutting away part of the image and slants the shape of the image. In shear transformation, the image is sort of being stretch in certain shear angle while one axis is fixed. Shear is an important step in image augmentation as it gives idea to the computers to see object from different angles like human do (Shankar, 2021).

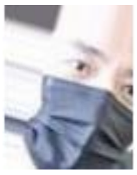
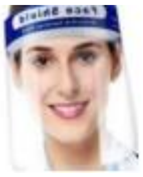
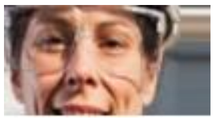
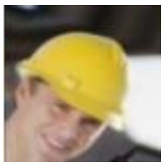
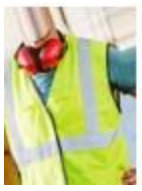
After shearing, the image is then be zoomed randomly by *zoom_range* function. Here the whole image in PPE dataset is zoomed in by maximum of 20%. This processed is called zooming in or also adding some pixels around that image in order to enlarge it. Simply explaining, it magnifies the picture and zooming inside the image randomnly (Random Zoom Image Augmentation - Keras ImageDataGenerator, 2021).

Next ImageDataGenerator from Keras also provides the options for flipping images horizontally and vertically. Here only horizontal flipping is used where the image is allowed to be flipped along x-axis by setting the *horizontal_flip* function to true. Vertical flipping is not used since it can cause the image to be upside down and adding load to the model to learn.

Last but not least, the most important function in image augmentation from ImageDataGenerator is *fill_mode*. When previously the input image is being rotated and shifted, it effects in empty area or left-over space that needs to be filled. Therefore, "nearest" is used which means that the area will be filled with the nearest pixels and stretching it.

Table 3.1 illustrates the results after image augmentation has been done to the PPE dataset. From the example images, it can be seen that each of the images are already went through augmentation process and different comparing to the original images.

Table 3.1: Image Augmentation Result

	Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Original Images					
Augmented Images					

3.5 Proposed Method for PPE Classification

The proposed solution to perform the classification of PPE is by using a deep learning algorithm namely Convolutional Neural Network (CNN). CNN is an adequate commonly used algorithm for object classification as it able to gives a good performance in term of accuracy with acceptable execution time speed. CNN algorithm is good enough to perform classification of the PPE since the typical CNN algorithm is known as classifier in which process like detection that involves localization does not need to be executed in this project and can be focused more in classifying the PPE correctly.

Figure 3.5 illustrates how the PPE images are being feed as input to the Convolutional Neural Network model where it learns features from the images and classifies them into classes which is firstly 5 classes, consists of class Face Mask, class Face Shield, class Safety Goggle, class Safety Helmet and lastly class Safety Jacket. Secondly is the binary classes where the five PPE is being classified into two classes of "with" or "without" PPE. The classification is considered successful if the image is classified into the correct class for example an image with face mask classifies into Face Mask class.

Convolutional Neural Network or CNN learns features of the images based on the existence of various layers and filters in its model architecture. Unlike human, computers see images in term of numbers in which pictures usually represented in 2-dimensional, 2D array of numbers, called as pixels. Thus, in order for the machine to recognise objects, CNN algorithm is used. The layers in CNN are structured in 3-dimensions, 3D which are width, height and depth.

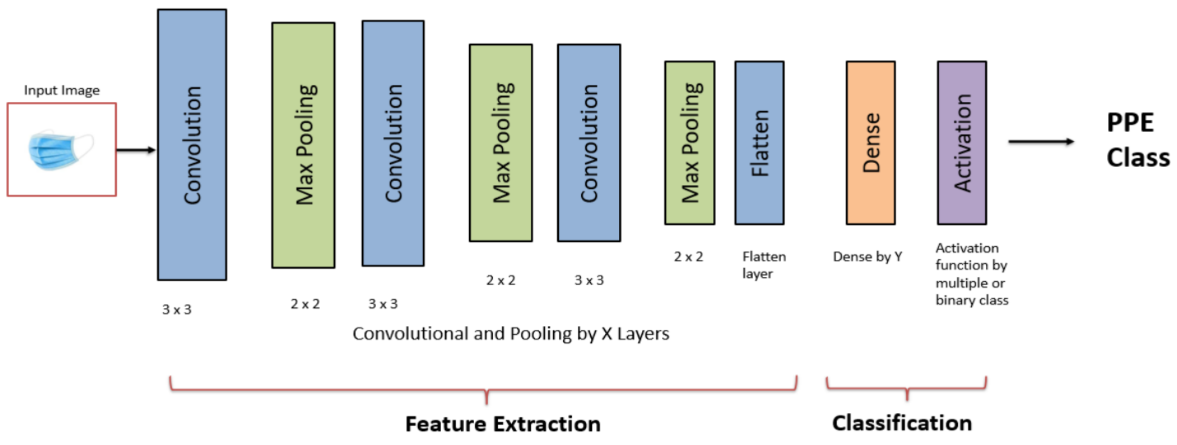


Figure 3.5 : Convolutional Neural Network (CNN) Architecture

There are two crucial components in CNN, firstly is the feature extraction part where all the hidden layers are located. Operations such as convolution is executed such that the features will be detected. Convolution generally can be defined as mathematical operations where two functions are combined to produce an output function. In PPE classification, convolutional layers are according to the number of filter layers in each classification of PPE. Convolution assists by the existence of the filter or kernel and producing a map called feature map. Convolution is performed by sliding the filter over the input image and multiplication of matrix as well as summing the results is performed then updated into the feature map. Figure 3.6 shows how the convolution is performed on an image. The size of the filter used (green square) is a 3 x 3 filter, sliding over the input image (blue square) and mapping the sum of the convolution into the feature map (red square).

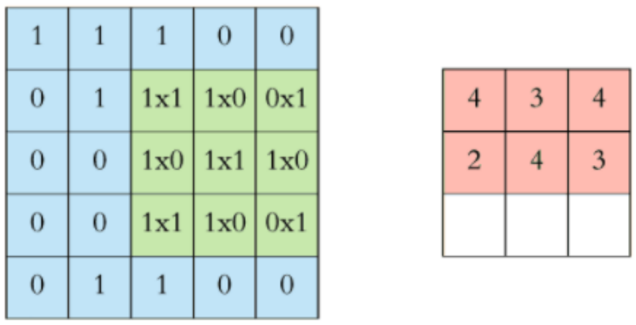


Figure 3.6 : Convolution Operation on an image

After convolutional layer, pooling layer is added to reduce the dimension continuously on the number of parameters as well as computation in the neural network. Pooling layer helps a lot in controlling overfitting where usually overfitting is unavoidable when it comes to data training. The type of pooling layer that commonly used is max pooling layer that considers maximum value in each window. In PPE classification, the number of 2x2 pooling layers are the same of convolutional layers which are depending on the number of layers in each of the PPE. For instance, the multiple classes have 5 layers and for binary classification, face shield, safety goggle, safety helmet and safety jacket have 5 layers while face mask has 4 layers.

The second important component in CNN is the classification where exist layers that are fully connected to each other. This fully connected layer only allows a 1-dimensional, 1D data to pass through. Therefore, flatten layer is used to convert the previous 3-dimensional data into 1-dimensional. Flatten layer is commonly used in the transition from the convolutional layer to the fully connected layer, which is also known as the output layer.

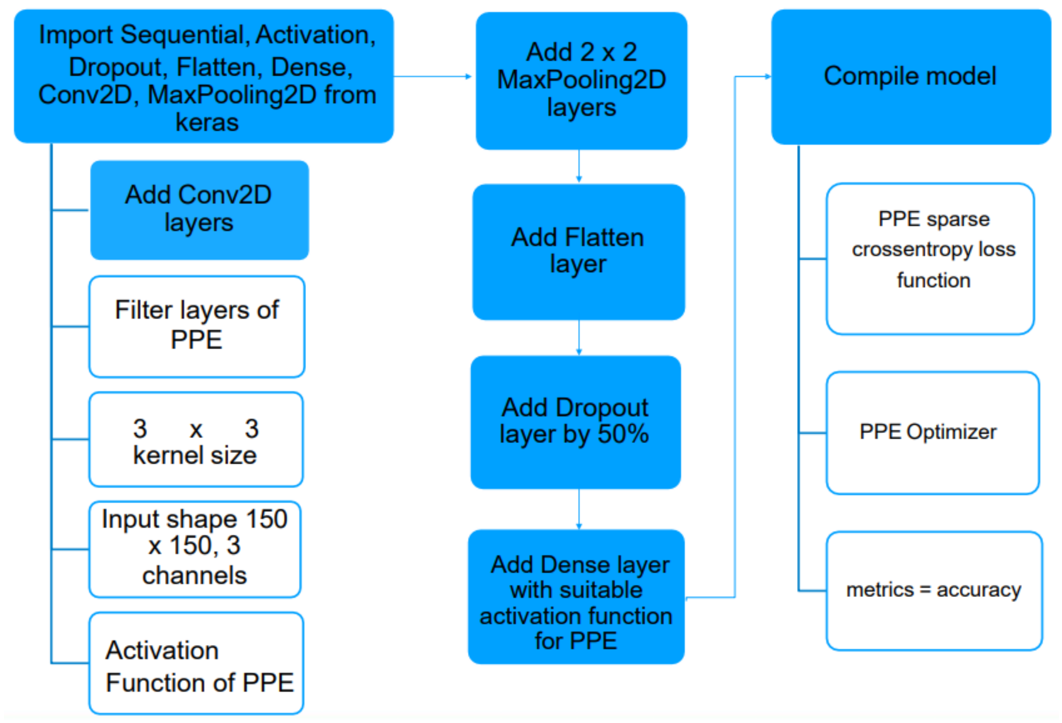
Next, in the classification part from the Figure 3.5Figure 3.5, dense layer is important in order to classify the object based on the output from the previous convolutional layers in hidden layers. Dense layer has neurons that receives information or input from all neurons of the previous layers, (Dumane, 2020). Thus in this PPE classifier, the dense layer depends on whether the classification is binary or multiple classes classification. As for binary class, the dense is set to be 1 while for multiple classes that consists of 5 class of PPE will have the dense layer set to be 5.

Last but not least, activation function in the output layer is also one of the important parameters to complete the classification. In the fully connected layer, the dense layer is matched with suitable activation functions because it is also depending on the type of class. For binary classification with dense layer of 1, sigmoid activation function is used. This is because the characteristics of sigmoid functions that has values ranging only from 0 to 1, (seb, 2021). Therefore, it is very suitable for classification into two classes with values from 0 to 1 as the only potential output values. On the other hand, for multiple classes softmax activation function is used. Softmax function is almost similar to sigmoid function but the difference is that softmax produces output that interrelated to each class in which the probability sum will always equal to 1. Thus, there will be only one class that has higher probability compared to the other classes which means the object is potentially classified into that particular class, (Basta, 2020).

3.6 Building the Model

In deep learning, model building is where the neural network considers inputs and parameters that being processed through hidden layers basically for training purpose. In this project, firstly related libraries such as models Sequential and layers Activation, Dropout, Flatten, Dense, Conv2D and MaxPooling2D. Besides image augmentation, *ImageDataGenerator* class also provides method flow () such as *flow_from_directory()* and *flow_from_dataframe()* that functions to read the pictures from folders containing those images. Here, *flow_from_directory* function is used to called out images from the training and testing folders. As a result, it will find the images that belong to training and images that belong to testing.

Figure 3.7 illustrates building the model block diagram. Sequential model is used because it is suitable for a stack of layers. Sequential model is built by passing several layers with their respective size. The first layer is Conv2D layer where it has filters of 3 x 3 kernel size and rescaling the input shape by 150 x 150 by 3 channels. Activation function in this block diagram is the activation function in the hidden layer, Rectified Linear Unit (ReLU), tangent hyperbolic (tanh) or sigmoid which depends on the PPE. The non-linear activation function helps the network to learn complex pattern. It is also will assists in the convergence process where the learning rate will be decreased when the error has reached its minimum value.



Import Sequential, Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D from keras
Add Conv2D layers
Filter layers of PPE
3 x 3 kernel size
Input shape 150 x 150, 3 channels
Activation Function of PPE
Add 2 x 2 MaxPooling2D layers
Add Flatten layer
Add Dropout layer by 50%
Add Dense layer with suitable activation function for PPE
Compile model
PPE sparse crossentropy loss function
PPE Optimizer
metrics = accuracy

Figure 3.7 : Building the model block diagram

Next, a MaxPooling2D layer is added with its pool size 2 x 2 to reduce the spatial dimensions of the output volume since the total of 288 filters for learning. Flatten layer is also added to create a single 1D array to input into the next layer. A single long feature vector will be produced which means the pixel data will be structured in one line and connected with the final layer. Then dropout layer is added by 50%. Dropout helps to reduce overfitting of the model by randomly turning neurons off during training process. So here it will randomly be turning off 50% of the neurons. The last layer is the dense layer with activation function from the output or fully connected layer, that also set according to the type of classification as mentioned previously.

After adding all the necessary layers, the model is then compiled with the suitable sparse cross-entropy loss function for the PPE. For binary class, *binary_crossentropy* is utilized while for the multiple classes *sparse_categorical_crossentropy* is set as the loss function. Sparse categorical cross-entropy is used because the classes are mutually exclusive in which in the multiple classes single image only belong to a single class.

Other than that, another parameter called as optimizer that leads the network to converge faster. In this PPE classifier, there are few optimizers that are being considered such as Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop), Stochastic Gradient Descent (SGD), and another of two variants from Adam which are Adamax and Nesterov-accelerated Adaptive Moment Estimation (Nadam). These optimizers are adaptive type optimizer in which it does not require to tune the learning rate because they have their default learning rate. For example, Adam has 0.001 learning rate which is very fast, therefore the network will learn features in less time. As for the other optimizer, RMSprop, Adamax and Nadam have the same learning rate as Adam which is 0.001 while SGD has a slightly slower learning rate which is 0.01, (Keras Documentation, n.d.).

3.7 Training the Model

Block diagram of training the model is as shown in Figure 3.8. As for training the model, library *fit_generator* is used meaning that Keras is calling the function of generator that previously used in image augmentation. The *fit_generator* function will perform backpropagation on a batch of dataset that it receives and followed by updating the weights in the model. Next, the class_mode argument where 'binary' class mode is used for binary classification and 'sparse' class mode is used for the multiple class classification. As for mutually exclusive dataset and needs to be matched with what has been used previously in with sparse_categorical_crossentropy so that the coding is free from error. Then, the PPE dataset is trained with suitable epochs meaning that the algorithm will perform the training according to the set epochs for each PPE. Overall, after completing until the end of the epochs, the trained model is saved in H5 format. It saves data in Hierarchical Data Format (HDF) and it is a lightweight way to save model that has been trained.

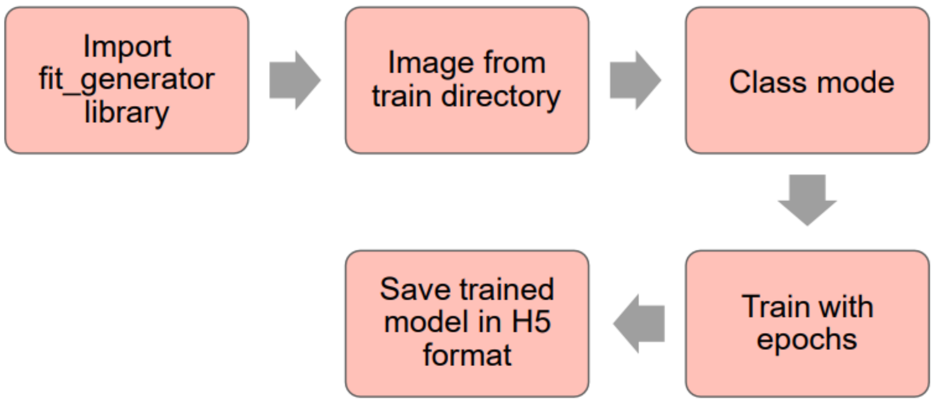

Import fit_generator library
Image from train directory
Class mode
Train with epochs
Save trained model in H5 format

Figure 3.8 : Training The Model Block Diagram

As mentioned in the overall project's flowchart, the training process will be divided into two parts. The first one is training the model with 5 classes. The parameters involved in training these 5 classes are using 100 epochs, ReLU activation function in the hidden layer, Adam optimizer, 5 filter layers that corresponds to the number of convolutional and pooling layer and lastly 20% width and height shift. ReLU activation function is used because it is a well-known non-linear activation function in the deep learning domain due to computationally efficient because it only activates certain neurons at a time. For instance, it activates neurons only if the output of linear transformation is more than 0, (Gupta, 2020). Adam optimizer is used because Adam generally recommended as default optimization algorithm since it has faster running time, requires less tuning and needs only low memory. In term of filter layers, 5 filter layers are added in the training and it is considered as sufficient amounts of layers for the CNN learn the complexity of the PPE features. Table 3.2 shows the parameters involve in training the PPE with 5 classes.

Table 3.2 : Training with 5 classes parameters

Epoch	**Activation Function**	**Optimizer**	**Filter Layers**	**Width Shift**	**Height Shift**
100	ReLU	Adam	5 Layers	20%	20%

The second part of the training process is by training the PPE in binary classification. This is done by firstly changing one parameter each time while the other parameters are maintained. Table 3.3 shows the parameters involves in training the PPE with binary class.

Table 3.3 : Training with binary class parameters

Epoch	**Activation Function**	**Optimizer**	**Filter Layers**
100,150,200,250,300	ReLU	Adam	3 layers
100	ReLU,tanh,sigmoid	Adam	3 layers
100	ReLU	Adam,rmsprop, SGD,Adamax, Nadam	3 layers
100	ReLU	Adam	3,4,5 layers

Based on the table, each parameter is changed for each training until the best accuracy is obtained. The training in binary class is done by changing epochs starting from 100 until training is completed. Then, training the second time by changing epoch to 150 while maintaining activation function, optimizer and filter layers as ReLU, Adam and 3 layers respectively. Execute the third, fourth and fifth training using 200, 250, and 300 epochs. Next continue the training by changing activation function while epoch, optimizer and filter layers are maintained at 100, Adam and 3 layers respectively. This training process is done until all the parameter changes have been trained and their weights in format of H5 files are saved.

After completing the training by changing parameters, the highest and second highest training and testing accuracy of activation function and optimizer are combined to be trained once again.

3.8 Testing the Model

Figure 3.9 shows the block diagram for testing the model using dataset from the testing folder in both 5 classes and binary class PPE. There are in total 500 testing images from 5 classes PPE and 1000 testing images from binary class PPE. Similar like the other process, the first step of including the library is a must to execute the model testing. Here, NumPy library, keras models and pre-processing module from keras are included. NumPy stands for Numerical Python and it is important in order to perform logical and mathematical operations on arrays. In short, here it is used to quickly create and analyse the data. Keras model library if for loading the trained model which is saved in H5 format before and pre-processing module is for importing the wanted image from testing directory.

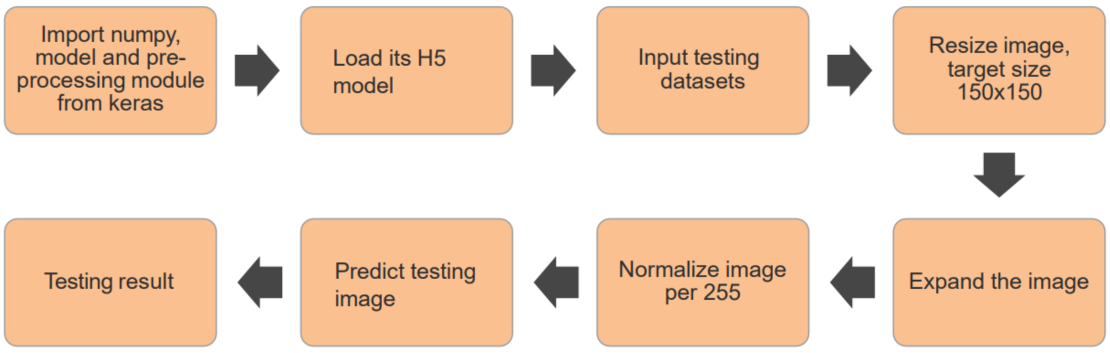


Figure 3.9 : Testing the model block diagram using dataset from each testing folder

Next, the loaded testing image of PPE is resized into 150 x 150 then converting it into an array. Then, NumPy library is used to expand the dimension of the image and significantly changes its shape so that the network learns it is a batch of one image. The command should be only run once because it has the possibility of being wrong and object not detected if the image keep expanding.

Other than that, the testing image also needs to be normalized so that it only falls between 0 and 1 value same as the training dataset. Last but not least the loaded PPE images then predicted using *model.predict* function and result will be shown in term of array that indicates the probability of the image belong to which class. This step is repeated every time predictions is being done to any of the PPE images such as calling out either face mask, face shield, safety goggle, safety helmet or safety jacket from the test directory for 5 classes and for binary classes it results in term of whether the images is with or without PPE with its probability.

The second part of the testing is after combining for the best result using new dataset and the block diagram is shown in Figure 3.10Figure 3.10 . The new dataset consists of only 12 images that are mixed between own captured images and online images. Almost all steps are similar to the previous testing process where it starts with importing the Tenserflow and NumPy library and *load_model* module from Keras. Then, it loads the weight of H5 model of the combination for best result of each five PPE. The 12 new testing images is imported into the testing network by inserting the correct directory. After that, same as previous testing where the images need to be resized, expanded and normalized. The output images will pop-up along with what type of PPE it classifies as present or not present with their respective probability scores.

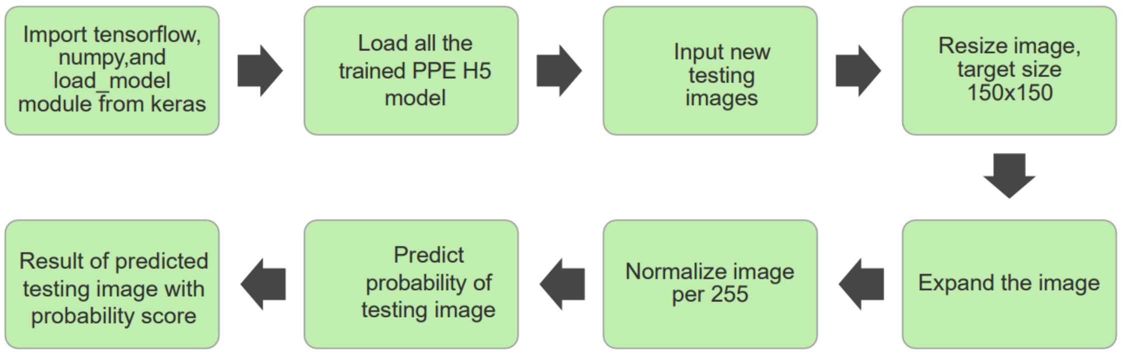

Import tensorflow, numpy,and load_model module from keras
Load all the trained PPE H5 model
Input new testing images
Resize image, target size 150x150
Expand the image
Normalize image per 255
Predict probability of testing image
Result of predicted testing image with probability score

Figure 3.10 : Testing the model block diagram using the new dataset

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this project, the final result is in percentage format that indicates probability of the PPE image belongs to which class between the five classes and between the combined binary class. For 5 classes, the classes have been set into image class indices such that class 0 is Face Mask, class 1 is Face Shield, class 2 is Safety Goggle, class 3 is Safety Helmet and lastly class 4 is Safety Jacket. On the other hand, for binary class it gives in term of two classes which are whether the PPE is classified or not classified in that image.

4.2 Training and Testing Accuracy of 5 Classes PPE

As for the 5 classes of PPE, the model is trained firstly using 2000 set of images and testing using 500 set of images of each PPE. Figure 4.1 and Figure 4.2 show the training and testing model accuracy and loss for the 5 classes of PPE. According to the pattern of the graph, training losses decreasing as accuracy increasing while for the testing, it has increasing losses then decreasing at the end of the epoch. Generally, both graphs of accuracy and testing have fluctuating pattern along the 0 to 100 epochs. This condition may be caused by there are some portions of the images are classified randomly, thus it produces random result between the correct and wrong classification. Besides, it also may be due to the overfitting of the network where the CNN unable to generalize well because model ends up fitting the noise that present along side of the data, (Koehrsen, 2018).

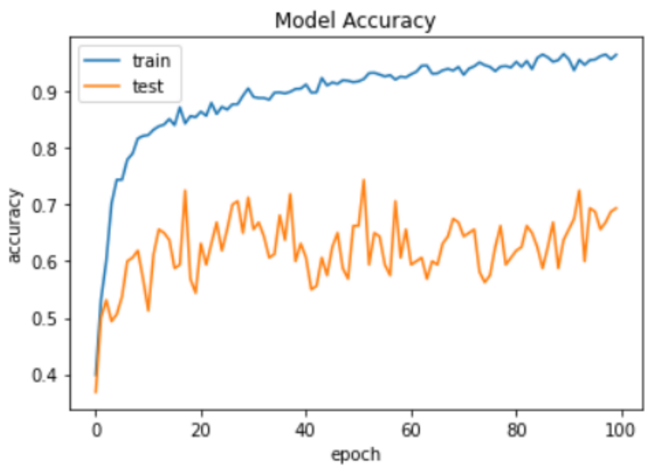


Figure 4.1 : Training and testing model accuracy graph for 5 classes of PPE.

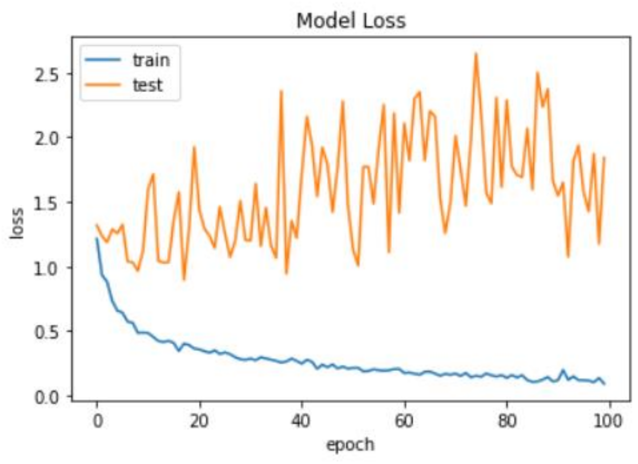
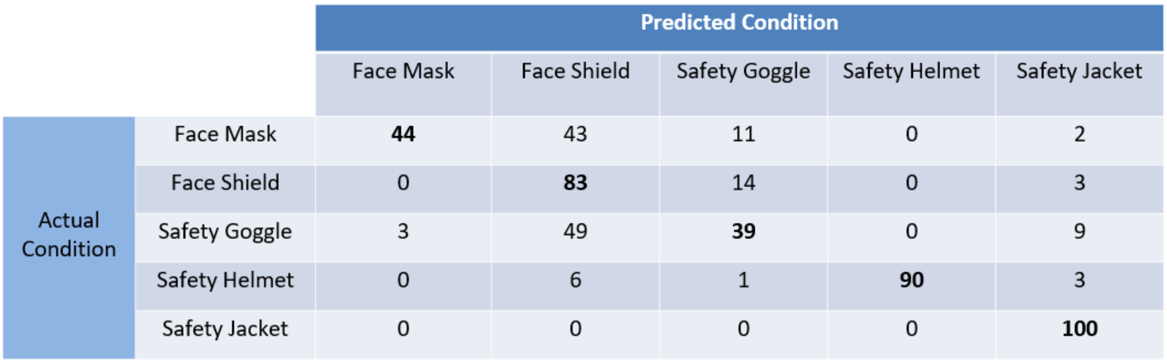


Figure 4.2 : Training and testing model loss graph for 5 classes of PPE.

Table 4.1 shows the summary of testing accuracy of the 5 classes PPE and Figure 4.3 shows confusion matrix of 5 classes PPE. Safety jacket has the highest testing accuracy which is 98.94%, followed by safety helmet 90.46% and safety goggle 83.35%. As for face mask and safety goggle, both recorded low testing accuracy which is 43.66% and 45.91% for each face mask and safety goggle respectively. This condition may be affected by the width and height shifting as much as 20% in augmentation before the training is conducted. By shifting the width and height maximum of 20% of the original images may not a good option and possibly has shifting the PPE in the images that was supposed to be classified as object. Thus, when the testing using new images, the network is unabled to recognize as well as classified the PPE absent in the images. The 20% should be decreased so that the low accuracy can be increased further.

Table 4.1 : Testing accuracy of 5 classes PPE

PPE	Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Total Images	500	500	500	500	500
Testing Accuracy (%)	43.66	83.35	45.91	90.46	98.94



		Predicted Condition				
		Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Actual Condition	Face Mask	**44**	43	11	0	2
	Face Shield	0	**83**	14	0	3
	Safety Goggle	3	49	**39**	0	9
	Safety Helmet	0	6	1	**90**	3
	Safety Jacket	0	0	0	0	**100**

Figure 4.3 : Confusion matrix of 5 classes PPE

4.3 Training and Testing Accuracy of Binary Class PPE

The model is firstly trained using 4000 images in total and another 1000 images are used for testing. In each PPE, there are 200 testing images which divided into 100 images for ‘with’ and 100 images for ‘without’ PPE.

4.3.1 Training by Changing the Epochs

Table 4.2 shows the training and testing accuracy of the binary class PPE when the epochs are changing starting from 100 until 300. Face mask and safety helmet have reached highest accuracy when the epoch is 200. 98.41% training accuracy and 93.31% testing accuracy for face mask while 99.46% training accuracy and 96.97% testing accuracy for safety helmet. Face shield’s accuracy is the highest at 250 epochs which is 98.52% and 89.30% accuracy for training and testing respectively. Both safety goggle and safety jacket achieve highest accuracy during 300 epochs. 99.58% training accuracy and 84.35% testing accuracy for safety goggle while 99.67% training accuracy and 87.34% testing accuracy for safety jacket. All of the highest result in training accuracy recorded good performance which is up to 98% accuracy as well as testing accuracy which is up to 84% accuracy.

Table 4.2 : Training and testing accuracy by changing epoch

Epoch	Classification Accuracy (%)									
	Face Mask		Face Shield		Safety Goggle		Safety Helmet		Safety Jacket	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
100	96.99	88.92	97.00	86.86	96.30	73.09	99.20	95.92	99.69	83.91
150	97.71	90.46	97.69	89.09	97.33	72.04	99.36	96.25	99.47	86.26
200	98.41	93.31	98.30	88.03	97.82	75.72	99.46	96.97	99.58	84.32
250	98.45	93.05	98.52	89.30	98.00	79.50	99.42	96.45	99.65	87.18
300	98.34	91.87	98.58	88.99	99.58	84.35	99.89	96.96	99.67	87.34

4.3.2 Training by Changing the Activation Function

Table 4.3 shows the training and testing accuracy of the binary class PPE when the activation functions are changing from ReLU, tanh and sigmoid. Face mask and safety helmet have reached highest accuracy when the activation function is sigmoid. 97.52% training accuracy and 97.10% testing accuracy for face mask while 99.31% training accuracy and 96.41% testing accuracy for safety helmet. Safety goggle's accuracy is the highest at ReLU activation function which is 96.30% and 73.09% accuracy for training and testing respectively. Both face shield and safety jacket achieve highest accuracy when the activation function is tanh. 97.27% training accuracy and 87.72% testing accuracy for face shield while 99.02% training accuracy and 82.27% testing accuracy for safety jacket. All of the highest result in training accuracy recorded good performance which is up to 96% accuracy as well as testing accuracy which is up to 73% accuracy.

Table 4.3 : Training and testing accuracy by changing activation function

Activa-tion Functi-on	**Classification Accuracy (%)**									
	Face Mask		**Face Shield**		**Safety Goggle**		**Safety Helmet**		**Safety Jacket**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
ReLU	96.99	88.92	97.00	86.86	96.30	73.09	99.20	95.92	99.69	83.91
Tanh	97.20	90.47	92.27	87.72	95.83	65.96	99.35	96.15	99.02	82.27
Sigmoid	97.52	97.10	97.22	87.36	96.31	71.79	99.31	96.41	99.28	84.59

4.3.3 Training by Changing the Optimizer

Table 4.4 shows the training and testing accuracy of the binary class PPE when the optimizers are changing starting from Adam, RMSprop, SGD, Adamax and Nadam. Face mask's accuracy is the highest at Adamax optimizer which is 97.69% and 89.22% accuracy for training and testing respectively. Three PPE have the highest accuracy by using RMSprop as optimizer which are face shield, safety goggle and safety jacket. Face shield has 99.66% training accuracy and 87.78% testing accuracy. As for safety goggle, it has 93.87% training accuracy and 73.59% testing accuracy. As for safety jacket, it has 98.88% training accuracy and 84.86% testing accuracy. Safety helmet's accuracy is the highest at Nadam optimizer which is 99.26% and 96.24% accuracy for training and testing respectively. All of the highest result in training accuracy recorded good performance which is up to 96% accuracy as well as testing accuracy which is up to 73% accuracy overall.

Table 4.4 : Training and testing accuracy by changing optimizer

Epoch	**Classification Accuracy (%)**									
	Face Mask		**Face Shield**		**Safety Goggle**		**Safety Helmet**		**Safety Jacket**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
ReLU	96.99	88.92	97.00	86.86	96.30	73.09	99.20	95.92	99.69	83.91
Rmspro-p	96.83	88.78	96.66	87.78	93.87	73.59	98.91	85.14	98.88	84.86
SGD	95.53	87.36	93.46	80.90	88.75	56.00	98.87	95.13	98.24	83.66
Adamax	97.69	89.22	97.13	85.24	95.65	70.36	99.45	95.57	99.58	80.63
Nadam	97.60	88.56	97.16	87.17	95.84	66.20	99.26	96.24	99.13	82.87

4.3.4 Training by Changing the Filter Layers

Table 4.5 shows the training and testing accuracy of the binary class PPE when different filter layers are used starting from 3 layers, 4 layers and finally 5 layers. All PPE, face shield, safety goggle, safety helmet and safety jacket except for face mask recorded the highest accuracy when the filter layers at its largest, which is 5 layers. Face shield has 99.69% training accuracy and 89.43% testing accuracy. As for safety goggle, it has 96.63% training accuracy and 83.21% testing accuracy. For safety helmet, it has 99.00% training accuracy and 98.63% testing accuracy. As for safety jacket, it has 99.12% training accuracy and 88.35% testing accuracy. The only PPE that has highest accuracy at 4 layers is face mask with 97.52% training accuracy and 92.90% testing accuracy. All of the highest result in training accuracy recorded good performance which is up to 96% accuracy as well as testing accuracy which is up to 83% accuracy overall.

Table 4.5 : Training and testing accuracy by changing filter layer

Activa-tion Functi-on	**Classification Accuracy (%)**									
	Face Mask		**Face Shield**		**Safety Goggle**		**Safety Helmet**		**Safety Jacket**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
3	96.99	88.92	97.00	86.86	96.30	73.09	99.20	95.92	99.69	83.91
4	97.52	92.90	97.19	89.22	96.64	79.83	99.26	96.95	99.3	84.25
5	97.26	92.74	96.69	89.43	96.63	83.21	99.00	98.63	99.12	88.35

4.4 Combining The Parameters That Have High Accuracy

This step is the process of combining which parameters that have achieved highest and second highest accuracy from the previously trained by changing the parameter each time. The combination only includes between activation function and optimizer while maintaining epoch and filter layer in which have highest trained accuracy from before.

4.4.1 Combination 1

Table 4.6 is final parameters for the activation function and optimizer based on highest training and testing accuracy. Using these parameters, the model is trained and tested once again to observe in case the accuracy will be further increase or not. Table 4.7 shows the result of training and testing accuracy from combination 1. None of these results have higher accuracy for training and testing compared to the other next three combinations.

Table 4.6 : Combination 1 of activation function and optimizer based on highest training and testing accuracy.

PPE	Parameters			
	Epoch	**Activation function**	**Optimizer**	**Layers**
Face mask	200	Sigmoid	Adamax	4
Face shield	250	Tanh	RMSprop	5
Safety goggle	300	ReLU	RMSprop	5
Safety helmet	200	Sigmoid	Nadam	5
Safety jacket	300	Sigmoid	RMSprop	5

Table 4.7 : Result from the combination 1

PPE	**Face Mask**	**Face Shield**	**Safety Goggle**	**Safety Helmet**	**Safety Jacket**
Training Accuracy (%)	98.22	98.31	97.94	99.00	99.32
Testing Accuracy (%)	91.78	88.50	86.38	98.22	85.75

4.4.2 Combination 2

Table 4.8 shows the result of training and testing accuracy from combination 2. It can be seen that face mask and safety goggle achieve the highest training and testing accuracy when implementing the combination 2. Face mask has 99.4% training accuracy and 91.78% testing accuracy. Safety goggle has 99.69% training accuracy and 89.79% testing accuracy. Both face mask and safety goggle use Adam as the optimizer. As has been discussed previously, Adam is generally known as the best optimizer as it may be helped the face mask and safety goggle to converge faster with high-speed learning rate.

Table 4.8 : Result from the combination 2

PPE	Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Training Accuracy (%)	99.40	99.63	99.69	83.52	86.72
Testing Accuracy (%)	95.60	90.05	89.79	83.04	79.93

4.4.3 Combination 3

Table 4.9 is final parameters for the activation function based and optimizer based on second highest training and testing accuracy. From Table 4.10 shows the result of training and testing accuracy from combination 3, it can be seen that face shield, safety helmet and safety jacket achieve the highest training and testing accuracy when implementing the combination 3. Face shield has 99.58% training accuracy and 94.32% testing accuracy. Safety helmet has 99.39% training accuracy and 98.90% testing accuracy. Safety jacket has 99.53% training accuracy and 88.45% testing accuracy. Same as previously, safety helmet and safety jacket have good accuracy when using Adam optimizer and Nadam optimizer for face shield. Nadam is a variance of Adam optimizer with the same 0.001 learning rate. It applied Nesterov momentum and sometimes is better optimizer than Adam itself in term of optimization algorithm, (Brownlee, 2021). Besides, each face shield, safety helmet and safety jacket are using sigmoid, tanh and ReLU activation function respectively. In which, may be the best matching combination when applying to the Adam and Nadam optimizer.

Table 4.9 : Combination 3 of activation function and optimizer based on second highest training and testing accuracy.

PPE	Parameters			
	Epoch	Activation function	Optimizer	Layers
Face mask	200	Tanh	Adam	4
Face shield	250	Sigmoid	Nadam	5
Safety goggle	300	Sigmoid	Adam	5
Safety helmet	200	Tanh	Adam	5
Safety jacket	300	ReLu	Adam	5

Table 4.10 : Result from the combination 3

PPE	Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Training Accuracy (%)	99.40	99.58	98.06	99.39	99.53
Testing Accuracy (%)	95.60	94.32	80.97	98.90	88.45

4.4.4 Combination 4

Table 4.11 is final parameters for the activation function based on second highest training and testing accuracy and optimizer based on highest training and testing accuracy. Table 4.12 shows the result of training and testing accuracy from combination 4. None of these results have higher accuracy compare to the other three combinations.

Table 4.11: Combination 4 of activation function based on second highest training and testing accuracy and optimizer based on highest and highest training and testing accuracy.

PPE	Parameters			
	Epoch	**Activation function**	**Optimizer**	**Layers**
Face mask	200	Tanh	Adamax	4
Face shield	250	Sigmoid	RMRprop	5
Safety goggle	300	Sigmoid	RMRprop	5
Safety helmet	200	Tanh	Nadam	5
Safety jacket	300	ReLu	RMRprop	5

Table 4.12 : Result from the combination 4

PPE	Face Mask	Face Shield	Safety Goggle	Safety Helmet	Safety Jacket
Training Accuracy (%)	98.80	98.17	98.28	99.19	99.23
Testing Accuracy (%)	92.00	88.69	85.61	98.35	86.37

4.4.5 Final Parameters Based on The Highest Accuracy from The Combination 2 and Combination 3.

Table 4.13 illustrates the final parameters from combination 2 and combination 3 since the PPE recorded highest training and testing accuracy from the combinations. Face mask and safety goggle accuracy are the highest at combination 2 while for face shield, safety helmet and safety jacket achieved highest accuracy at combination 3.

Table 4.13: Final parameters based on the highest training and testing accuracy from combination 2 and combination 3.

PPE	Parameters			
	Epoch	**Activation function**	**Optimizer**	**Layers**
Face mask	200	Sigmoid	Adam	4
Face shield	250	Sigmoid	Nadam	5
Safety goggle	300	ReLU	Adam	5
Safety helmet	200	Tanh	Adam	5
Safety jacket	300	ReLu	Adam	5

Table 4.14 is the final training and testing accuracy after combining the best parameters into combination 2 and combination 3. Face mask records training accuracy as high as 99.40% while its testing records 95.60 %. Face shield also has high as 99.58% training accuracy while testing 94.32%. Safety goggle achieved a high training accuracy, 99.69% but dropped to 89.79% in testing accuracy. As for the safety helmet, its training accuracy also considers as high which is 99.39% and high testing accuracy which is 98.90%. A very close difference between the training and testing result which is good. Lastly, as for the safety jacket, its training accuracy also one of the highest which is 99.53% while has 88.45% testing accuracy. Overall, all PPE have high training accuracy which is up to 99%. On the other hand, for testing face mask, face shield and safety helmet have accuracy up to 94% while for safety goggle and safety jacket have slightly low accuracy but still manage to reach up to 88% accuracy which is kind of good result.

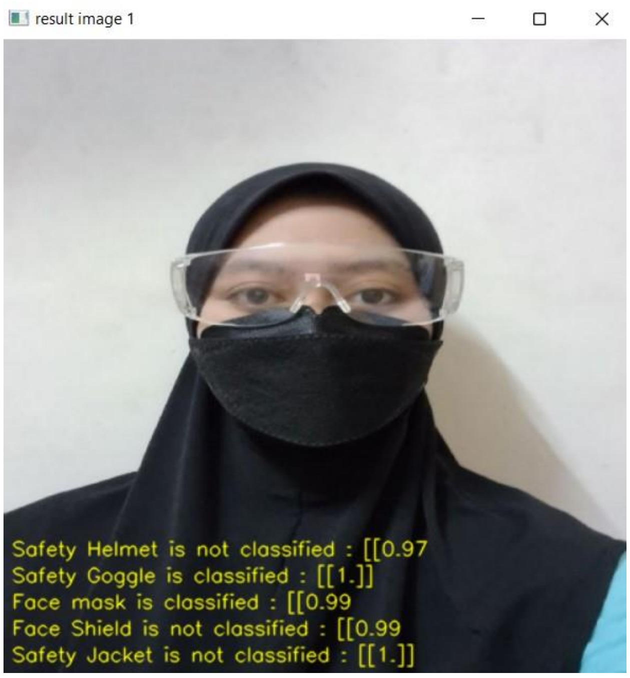
Table 4.14 : Final training and testing accuracy based on Table 4.13.

PPE	**Face Mask**	**Face Shield**	**Safety Goggle**	**Safety Helmet**	**Safety Jacket**
Training Accuracy (%)	99.40	99.58	99.69	99.39	99.53
Testing Accuracy (%)	95.60	94.32	89.79	98.90	88.45

4.5 Testing the Best Combined Parameters Using New Test Dataset.

By comparing the accuracy between both 5 classes and binary class classification result, the binary class obviously scores higher for the other four PPE which are face mask, face shield, safety goggle and safety helmet. The 5 classes classification has the highest classification accuracy for safety jacket but it is the only one it got high result on. Therefore, binary classification's model is chosen as the final classifier model with their best combined parameters to test on new dataset. This dataset only consists of 12 images which are the combination of own captured data and images from online platform.

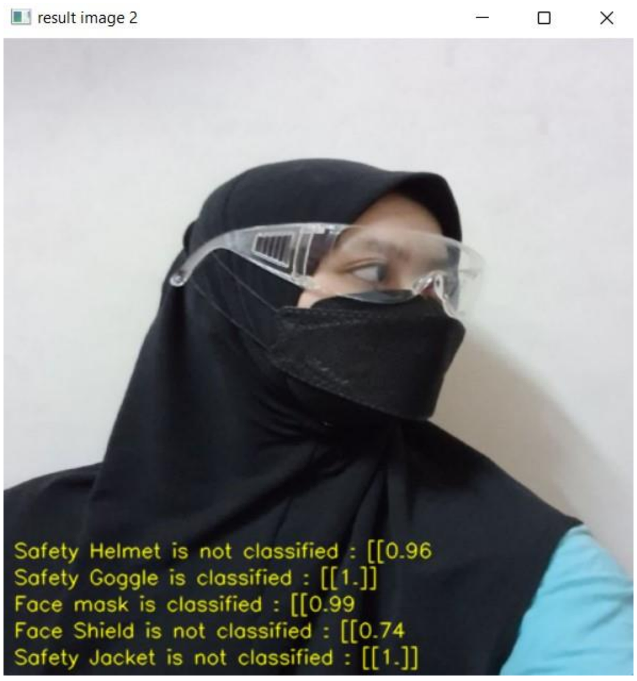
The first image is Figure 4.4, where is shows a front view of a person is wearing face mask and safety goggle. From the output image 1, it is clear that safety goggle and face mask are successfully 'classified' with both have 0.97 and 0.99 probability scores respectively. As for the safety helmet, face shield and safety jacket, they are also successfully 'not classified' with high scores of 0.99 and 1. The PPE classifier is accurately classified present and absent PPE in Image 1.



```
Safety Helmet is not classified : [[0.97
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is not classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.4 : Front view of Image 1

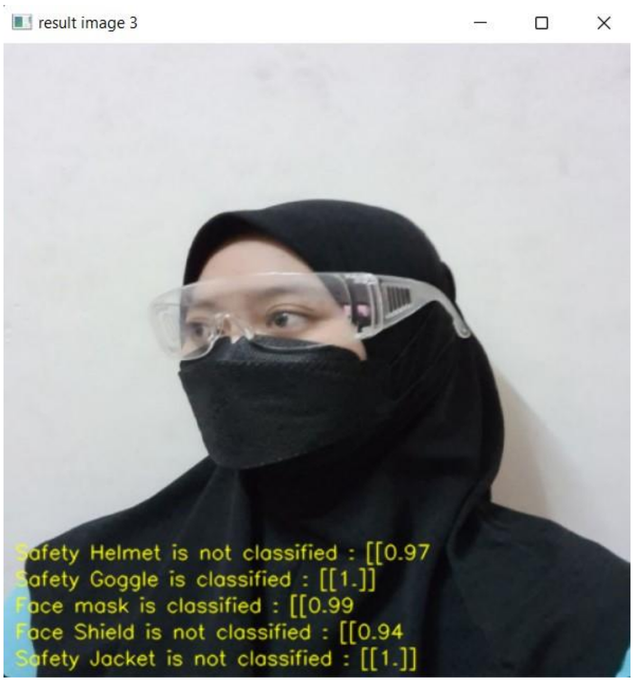
The second image is Figure 4.5, where is shows a left view version of Image 1. From the output image 2, it is clear that safety goggle and face mask are successfully ‘classified’ with both have 1 and 0.99 probability scores respectively. Same as Image 1, the other three PPE, safety helmet, face shield and safety jacket, they are also successfully ‘not classified’ with high scores of 0.96, 0.74 and 1 respectively. The PPE classifier is accurately classified present and absent PPE in Image 2.



```
Safety Helmet is not classified : [[0.96
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is not classified : [[0.74
Safety Jacket is not classified : [[1.]]
```

Figure 4.5 : Left view of Image 2

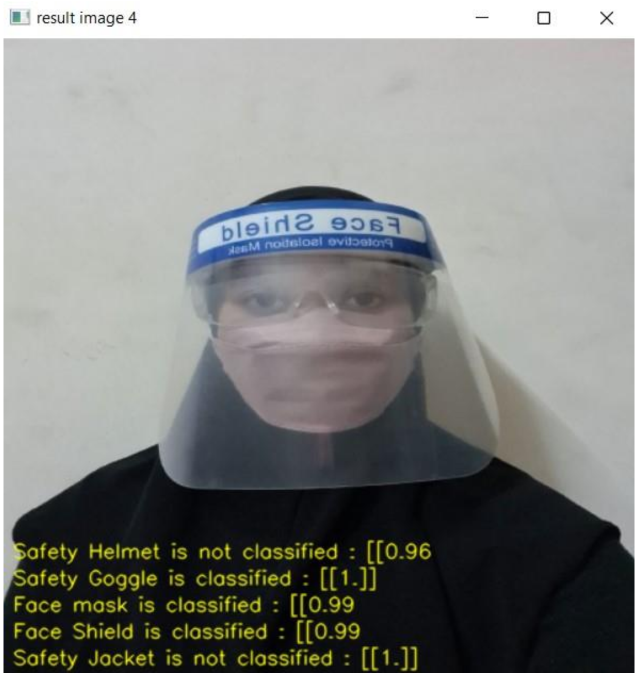
Third, in Figure 4.6 shows a right view version of image 1 and image 2. Similar with image2, image 3 also has successfully ‘classified’ safety goggle and face mask with scores of 1 and 0.99 respectively. As for the other three PPE, safety helmet, face shield and safety jacket, they are also successfully ‘not classified’ with high scores of 0.97, 0.94 and 1 respectively. The PPE classifier is accurately classified present and absent PPE in Image 3.



```
Safety Helmet is not classified : [[0.97
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is not classified : [[0.94
Safety Jacket is not classified : [[1.]]
```

Figure 4.6 : Right view of Image 3

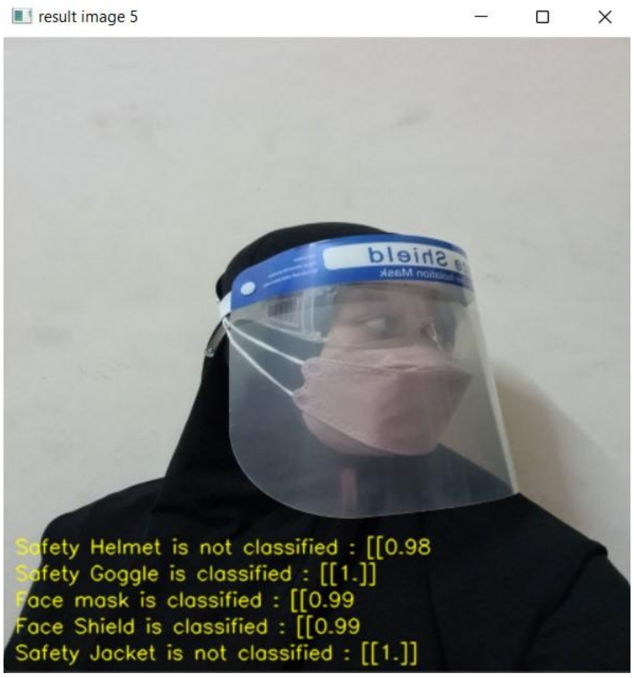
Next image is Figure 4.7 shows the front view version of image 4. From the output image 4, safety goggle, face mask and face shield are successfully 'classified' with having 1, 0.99 and 0.99 probability scores respectively. As for the other two PPE, safety helmet and safety jacket, both are also successfully 'not classified' with high scores of 0.96 and 1 respectively. The PPE classifier is accurately classified present and absent PPE in Image 4.



```
Safety Helmet is not classified : [[0.96
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.7 : Front view of Image 4

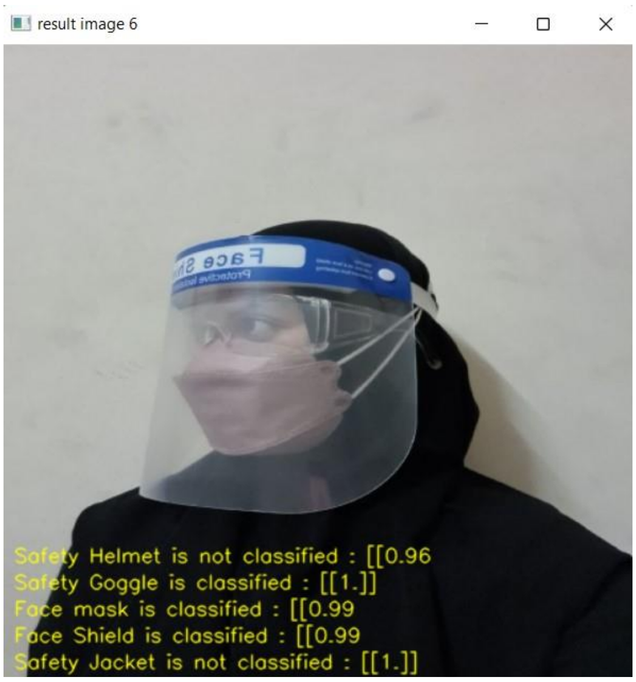
Then, another image is Figure 4.8   Figure 4.8 shows the front view version of image 4. From the output image 5, safety goggle, face mask and face shield are successfully 'classified' with having 1, 0.99 and 0.99 probability scores respectively, similar as image 4. As for the other two PPE, safety helmet and safety jacket, both are also successfully 'not classified' with high scores of 0.98 and 1 respectively. The PPE classifier is once again accurately classified present and absent PPE in Image 5.



```
Safety Helmet is not classified : [[0.98
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.8 : Left view of Image 5

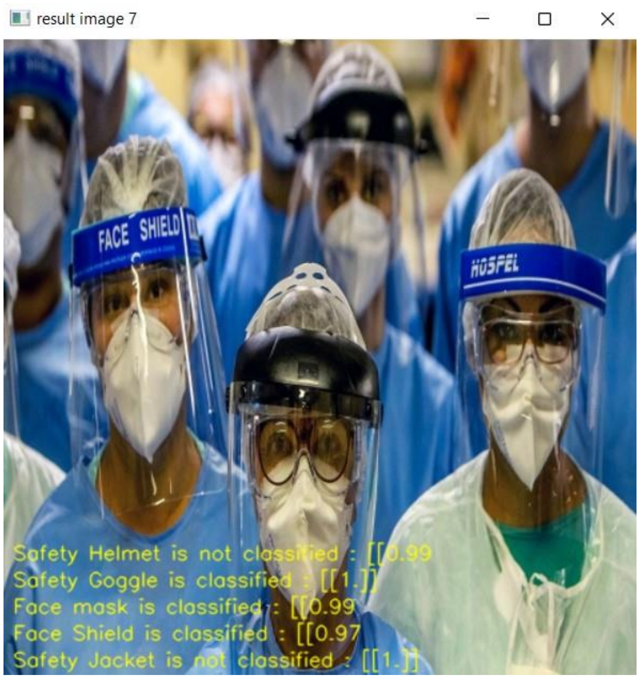
Then, another image as shown in Figure 4.9 which is shows the left view version of image 5. From the output image 5, safety goggle, face mask and face shield are successfully ‘classified’ with having 1, 0.99 and 0.99 probability scores respectively, similar as image 4 and 5. As for the other two PPE, safety helmet and safety jacket, both are also successfully ‘not classified’ with high scores of 0.96 and 1 respectively. The PPE classifier is once again accurately classified present and absent PPE in Image 6.



```
Safety Helmet is not classified : [[0.96
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.9 : Right view of Image 6

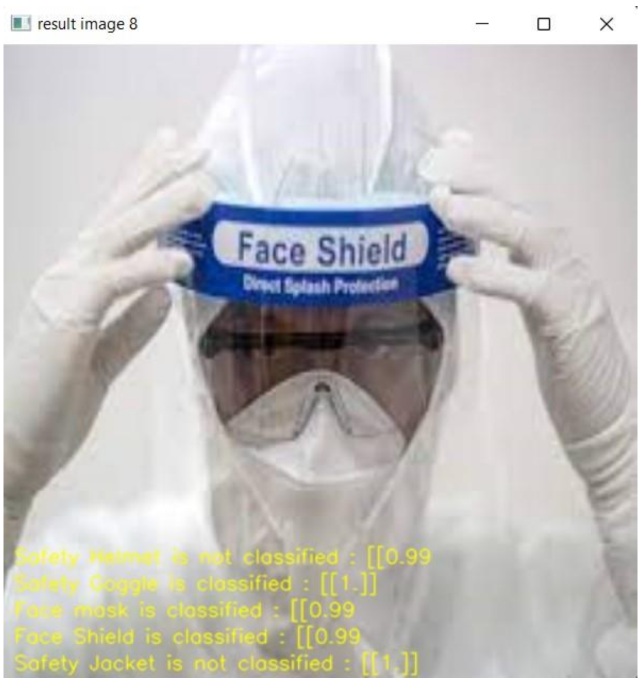
Figure 4.10 shows the multiple persons that wearing a set of PPE including safety goggle, face mask and face shield. From the output image 7, safety goggle, face mask and face shield are successfully ‘classified’ with having 1, 0.99 and 0.97 probability scores respectively. As for the other two PPE, safety helmet and safety jacket, both are also successfully ‘not classified’ with high scores of 0.99 and 1 respectively. The PPE classifier manages to achieve good performance result when testing Image 7.



```
Safety Helmet is not classified : [[0.99
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is classified : [[0.97
Safety Jacket is not classified : [[1.]]
```

Figure 4.10 : Image 7

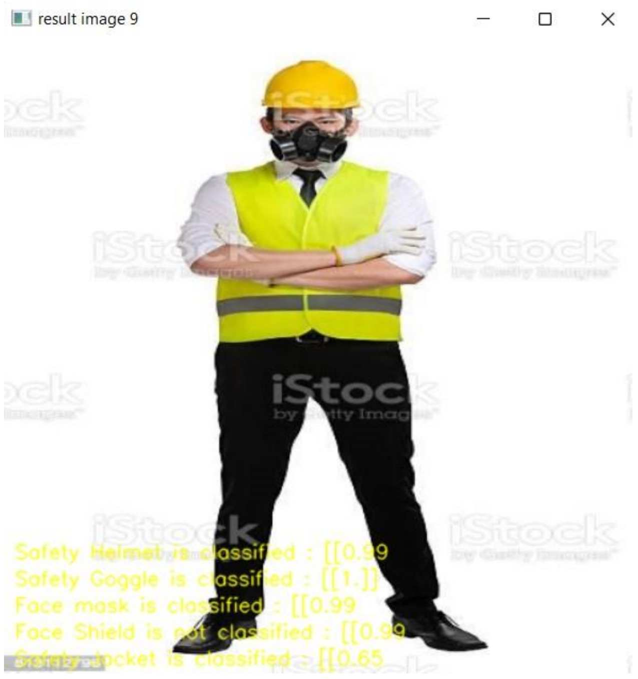
After testing the other image of Figure 4.11, it shows a person that wear a set of PPE including safety goggle, face mask and face shield. From the output image 8, safety goggle, face mask and face shield are successfully ‘classified’ with having 1, 0.99 and 0.99 probability scores respectively. For safety helmet and safety jacket, both are also successfully ‘not classified’ with high scores of 0.99 and 1 respectively. The PPE classifier successfully classified correct PPE accurately.



```
Safety Helmet is not classified : [[0.99
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.11 : Image 8

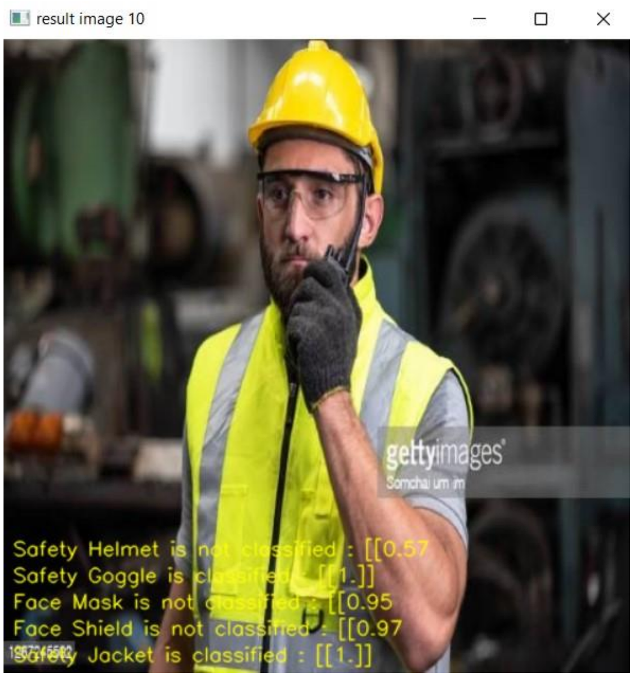
Another image of Figure 4.12, it shows a person that wear a set of PPE including except for safety goggle and face shield while wearing safety helmet, face mask and safety jacket. From the output image 9, safety helmet, face mask and safety jacket successfully 'classified' with scores of 0.99, 0.99 and 0.65 respectively. It also successfully 'not classified' the face shied which is really not present in the picture. However, this classifier inaccurately classifies the absent of safety goggle as present with high scores which is 1. At this time, the classifier is unable to accurately classified a 100% good result due to the misclassification of safety goggle.



```
Safety Helmet is classified : [[0.99
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is not classified : [[0.99
Safety Jacket is classified : [[0.65
```

Figure 4.12 : Front view of Image 9

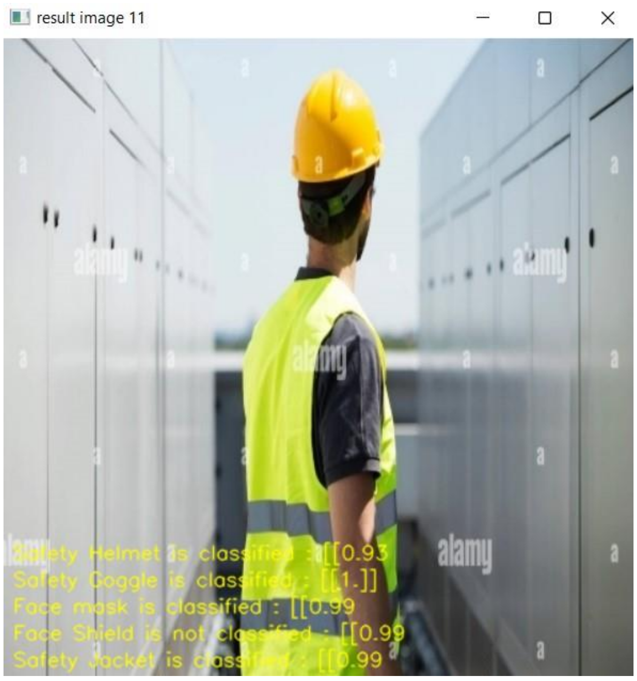
Next, to test image of Figure 4.13 shows a person that wear a safety helmet, safety goggle and safety jacket. From the output image 10, only safety goggle and safety jacket are classified as present in the image with scores of 1 for both PPE. However, the present safety helmet is unable to be classified and recognized as 'not classified' but with only low probability of the misclassification which is only 0.57. This may be caused by the position of the safety helmet in that image that located at the very end of the picture which the network may think it as the background. The other two absent PPE, face mask and face shield are successfully 'not classified' tich scores of 0.95 and 0.97 respectively. The PPE classifier is almost perfect in the classification of the PPE.



```
Safety Helmet is not classified : [[0.57
Safety Goggle is classified : [[1.]]
Face Mask is not classified : [[0.95
Face Shield is not classified : [[0.97
Safety Jacket is classified : [[1.]]
```

Figure 4.13 : Front view of Image 10

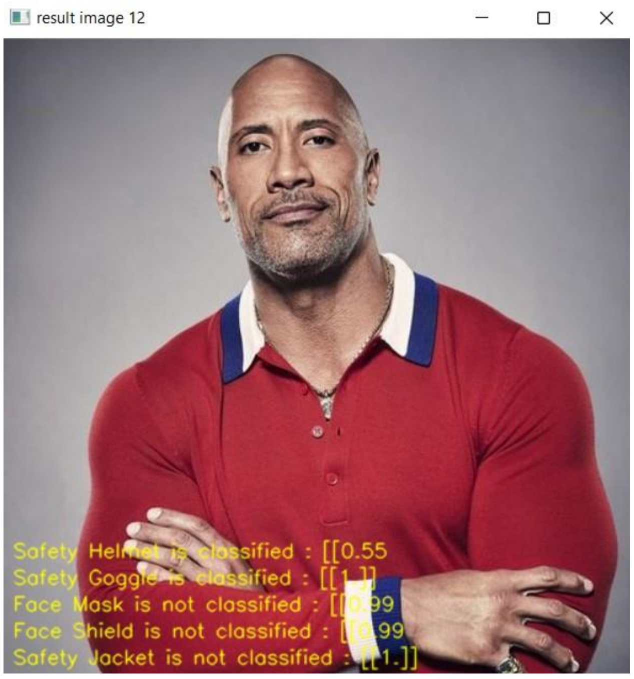
Then, moving to testing other image 11 of Figure 4.14 below, it shows a person that wear a set of PPE but in different view which is from back-view of the person. Only two PPE are visible including safety helmet and safety jacket while others are not. The result shows that from the output image 11, safety helmet and safety jacket are classified successfully with high scores, 0.93 and 0.99 respectively. However, for safety goggle and face mask, both are also classified in the image 11 with high scores 1 and 0.99 respectively. The misclassification may be due to the network may recognize the person'd hair as face mask while inaccurately classify safety goggle. Another PPE which is face shield is 'not classified' successfully with score of 0.99. Observing from image 11, the PPE classifier did not quite successful in classifying the correct present and absent PPE in the image.



```
Safety Helmet is classified : [[0.93
Safety Goggle is classified : [[1.]]
Face mask is classified : [[0.99
Face Shield is not classified : [[0.99
Safety Jacket is classified : [[0.99
```

Figure 4.14 : Front view of Image 11

Lastly, the test image 12 of Figure 4.15, it shows a person that is not wearing any PPE. The result shows good ‘not classified’ result of face mask, face shield and safety jacket with probability scores of 0.99, 0.99 and 1 which are good in term of accuracy performance. However, even though there is no safety helmet and safety goggle but the classifier still classified these two PPE as present, in which occurs misclassification. But the score safety helmet classification is relatively low, which is only 0.55. This may be due to the shape of the person’s head that is almost similar to the shape of safety helmet. As for the safety goggle, the network may sometimes seems to recognize human eyes as the object instead of safety goggle, which is may also be caused by the transparent characteristic of the safety helmet that leads to failure in classifying safety helmet as object.



```
Safety Helmet is classified : [[0.55
Safety Goggle is classified : [[1.]]
Face Mask is not classified : [[0.99
Face Shield is not classified : [[0.99
Safety Jacket is not classified : [[1.]]
```

Figure 4.15 : Front view of Image 12

4.5.1 Confusion Matrix of Binary Class PPE.

Figure 4.16, Figure 4.17, Figure 4.18, Figure 4.19 and Figure 4.20 show the confusion matrix for the 12 test images using the best combined parameters of binary class. The confusion matrix for binary class involves true positive (TP), that correctly indicates present condition, true negative (TN), that correctly indicates absent condition, false positive (FP), that incorrectly indicates present condition and false negative (FN), that incorrectly indicates absent condition.

Figure 4.16 represents the confusion matrix for safety helmet. In actual condition, only 3 images have safety helmet out of 12 test images and it correctly classifies 2 while the other one it classifies as no safety helmet. The other images with absence of safety helmet, it correctly classifies 8 images as without safety helmet while there is one it classifies as with safety helmet.

		Predicted Condition	
	Total : 3 + 9 = 12	Helmet 3	No Helmet 9
Actual Condition	Helmet 3	2	1
	No Helmet 9	1	8

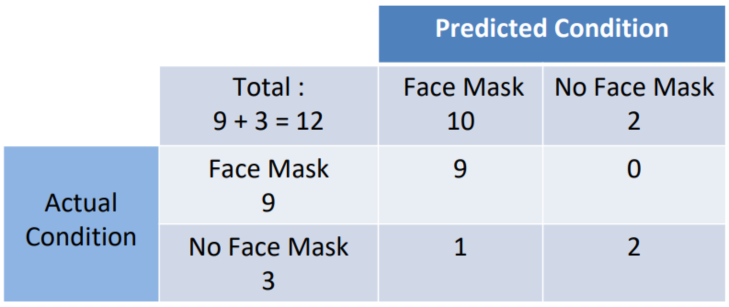
Figure 4.16 : Confusion matrix of safety helmet

Figure 4.17 represents the confusion matrix for safety goggle. In actual condition, there are 9 images that have safety goggle and network manages to classify correctly are all 9 of them. Unfortunately, the classifier classifies all 12 images have safety goggle that leads to a FP value of 3.

		Predicted Condition	
	Total : 9 + 3 = 12	Goggle 12	No Goggle 0
Actual Condition	Goggle 9	9	0
	No Goggle 3	3	0

Figure 4.17 : Confusion matrix of safety goggle

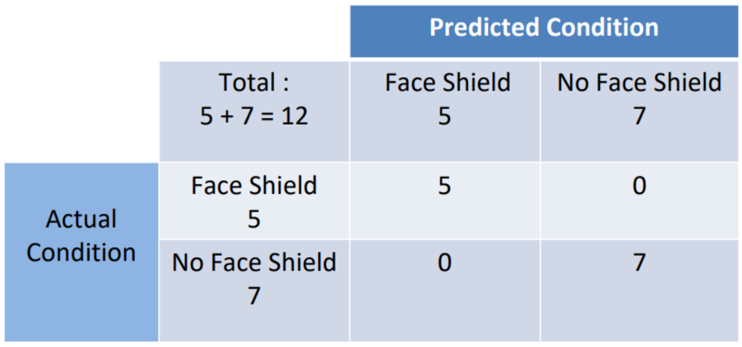
Figure 4.18 represents the confusion matrix for face mask. In actual condition, there are in total of 9 out of 12 images having face mask as one of its objects. All 9 are correctly classified. There are three images are without face mask and two of them correctly classified as absent while there is one considered as having face mask.



		Predicted Condition	
	Total : 9 + 3 = 12	Face Mask 10	No Face Mask 2
Actual Condition	Face Mask 9	9	0
	No Face Mask 3	1	2

Figure 4.18 : Confusion matrix of face mask

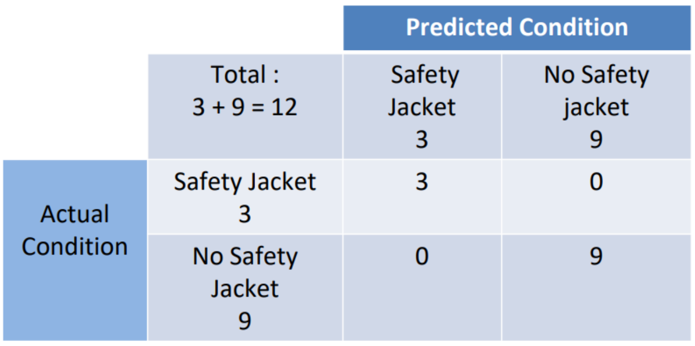
Figure 4.19 represents the confusion matrix for face shield. All present and absent face shield images classified correctly with zero value of FP and FN.



		Predicted Condition	
	Total : 5 + 7 = 12	Face Shield 5	No Face Shield 7
Actual Condition	Face Shield 5	5	0
	No Face Shield 7	0	7

Figure 4.19 : Confusion matrix of face shield

Figure 4.20 represents the confusion matrix for safety jacket. In actual condition, there are only 3 images out of 12 total images containing safety jacket and another 9 have no safety jacket. All 3 and 9 present and absent safety jacket are classified successfully with also no value of FP and FN.



		Predicted Condition	
	Total : 3 + 9 = 12	Safety Jacket 3	No Safety jacket 9
Actual Condition	Safety Jacket 3	3	0
	No Safety Jacket 9	0	9

Figure 4.20 : Confusion matrix of safety jacket

CHAPTER 5

CONCLUSION

5.1 Introduction

In conclusion, PPE classification by using 5 classes has overall accuracy of 89.39% for training and 62.23% in testing. It experienced 27.16% accuracy decrement which is quite a low performance in term of classification accuracy. This decrement may be caused by large amounts of images undergo misclassification and resulting in a bad classification accuracy. Besides, it is also may affected by the unnecessary large percentage of width and height shifting, 20% in which may possibly cause the object in the images have already shifted out of the images. 10% might is already sufficient to shift the images.

As for the PPE classification using the best combined parameters of binary class, it manages to achieved training accuracy which is more than 99% for each PPE. The accuracy is decreasing in testing but it still recorded a good percentage of accuracy which is up to 88% for all type of PPE, face mask, face shield, safety goggle, safety helmet and safety jacket. As for the specific accuracy reading of each of the PPE, Table 5.1 has summarized their accuracy accordingly.

Table 5.1 : Final PPE classifier accuracy

PPE	**Face Mask**	**Face Shield**	**Safety Goggle**	**Safety Helmet**	**Safety Jacket**
Training Accuracy (%)	99.40	99.58	99.69	99.39	99.53
Testing Accuracy (%)	95.60	94.32	89.79	98.90	88.45

Based on the result, Convolutional Neural Network, CNN as one of the deep learning algorithms is almost a good algorithm for the classification of PPE. The previous 12 testing images also have proven a good classification result with high scores in 'classified' and 'not classified' condition. Some incorrect classification also has been recorded and they may also have been caused by the classifier mistakenly took another object as PPE as it recognizes similar features between the object and PPE.

As for the misclassification of the safety goggle such that it shows classification even when there is no safety goggle present. It may possibly classified human eyes as the object instead of the safety goggle since the safety goggle is transparent. The CNN may not recognize there is the transparent safety goggle present before human eyes, that is supposed to be the object, not human eyes. This problem has been addressed and will be improved in future work. Overall, since the result is quite a good result than the method of changing parameters and combining good parameters seem to be a good method to increase the result of CNN accuracy.

5.2 Future Work

As for the future work, a larger dataset can be implemented in the PPE classification application where the number of training and testing images are going to be increased in order to achieve a higher result accuracy. Besides, type of PPE also can be added more which is instead only have five type of PPE, other PPE such as gloves, safety shoes or hearing protection can be added to increase the variety of the PPE. Same goes to the colour of safety helmet and safety jacket, instead of just using yellow colour, other colour can be added since it also gives benefit to the PPE classifier as it can recognizes a lot more variations of PPE.

REFERENCES

123RF. (2021). *123RF*. Retrieved 7 2, 2022, from https://www.123rf.com/

Basta, N. (1 4, 2020). *The Differences between Sigmoid and Softmax Activation Functions*. Retrieved 16 6, 2022, from https://medium.com/arteos-ai/the-differences-between-sigmoid-and-softmax-activation-function-12adee8cf322

Bouaich, S., & Tairi, H. (2019). Vehicle Detection using Road Lines. (p. 5). Fes: IEEE.

Brownlee, J. (19 3, 2021). *Gradient Descent Optimization With Nadam From Scratch*. Retrieved 17 6, 2022, from https://machinelearningmastery.com/gradient-descent-optimization-with-nadam-from-scratch/#:~:text=Nesterov%2Daccelerated%20Adaptive%20Moment%20Estimation,performance%20of%20the%20optimization%20algorithm

COCO. (n.d.). Retrieved from Common Objects in Context: https://cocodataset.org/#home

Deepanshi. (25 5, 2021). *Analytic Vidhya*. Retrieved from All you need to know about your first Machine Learning model – Linear Regression: https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-your-first-machine-learning-model-linear-regression/

Delhi, V. S., & Thomas, R. S. (2020). Detection of Personal Protective Equipment (PPE) Compliance on Construction Site Using Computer Vision Based Deep Learning Techniques. *frontiers in Built Environment, 6:136*, 10.

Dumane, G. (3 3, 2020). *Introduction to Convolutional Neural Network (CNN) using Tensorflow*. (towards Data Science) Retrieved 16 6, 2022, from https://towardsdatascience.com/introduction-to-convolutional-neural-network-cnn-de73f69c5b83

freeCodeCamp. (24 4, 2018). *An intuitive guide to Convolutional Neural Networks*. Retrieved 7 2, 2022, from https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

Gathani, J., & Shah, K. (2020). Detecting Masked Faces using Region-based Convolutional Neural Network. *IEEE.*

George, j., Mary, L., & K S , R. (2013). Vehicle Detection and Classification From Acoustic Signal Using ANN and KNN. *International Conference on Control Communication and Computing (ICCC)* (p. 4). Kottayam: IEEE.

George, T., Potty, S. P., & Jose, S. (2014). Smile Detection from Still Images Using KNN Algorithm. *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)* (p. 5). Kottayam: IEEE.

Geyzel, M. v. (9 3, 2022). *The Malaysian Lawyer.com*. Retrieved from Malaysia's updated workplace COVID SOPs from 1 April 2022 : Capacity and operating hour limits abolished: https://themalaysianlawyer.com/2022/03/09/malaysias-workplace-sops-april-2022/

Gholamy, A., Vladik Kreinovich, & Olga Kosheleva. (2018). Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation. *Departmental Technical Reports (CS)*(1209), 7.

Gupta. (30 1, 2020). *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* (Analytics Vidhya) Retrieved 17 6, 2022, from https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

HexArmor. (11 June, 2019). *The Hard Truth about Safety Helmet Injuries and Statistics*. Retrieved from https://www.hexarmor.com/posts/the-hard-truth-about-safety-helmet-injuries-and-statistics

iStockPhotoLP. (2021). *iStock*. Retrieved 7 2, 2022, from https://www.istockphoto.com/

Jangra, A. (2020). *Kaggle*. Retrieved 7 2, 2022, from https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

Jian, W., & Lang, L. (2021). Face mask detection based on Transfer learning and PP-YOLO. *IEEE.*

Keras Documentation. (n.d.). Retrieved 17 6, 2022, from https://keras.io/api/optimizers/

Koehrsen, W. (28 1, 2018). *Overfitting vs. Underfitting: A Complete Example*. (Towards Data Science) Retrieved 17 6, 2022, from https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM, 60*, 7.

Liu, R., & Ren, Z. (2021). Application of Yolo on Mask Detection Task . *IEEE.*

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector.

Liu, X., Dai, B., & He, H. (2011). Real-time On-Road Vehicle Detection Combining Specific Shadow Segmentation and SVM Classification. *Second International Conference on Digital Manufacturing & Automation* (p. 4). Changsa: IEEE.

Mane, S., & Prof.Supriya Mangale. (2018). Moving object detection and tracking Using Convolutional Neural Networks. *IEEE.*

Mingyuan Xu, S. Y., Wang, H., & Li, R. (2020). Mask Wearing Detection Method Based on SSD-Mask Algorithm. *IEEE.* Shanghai.

Occupational Safety and Health Administration. (2004). Personal Protective Equipment. U.S. Department of Labor.

OSH. (2022). *Unied States Department of Labor*. Retrieved from Personal Protective Equipment: https://www.osha.gov/personal-protective-equipment

Papers With Code. (n.d.). Retrieved from Datasets 6499 machine learning datasets: https://paperswithcode.com/datasets?q=Sequential%20CIFAR-10

Ponraj, A. (20 2, 2021). *A Tip A Day — Python Tip #8: Why should we Normalize image pixel values or divide by 255? | Dev Skrol*. (Analytics Vidhya) Retrieved 16 6, 2022, from https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-8-why-should-we-normalize-image-pixel-values-or-divide-by-255-4608ac5cd26a#:~:text=To%20reduce%20this%20we%20can,range%20from%200%20to%201

Protik, A. A., Rafi, A. H., & Siddique, S. (2021). Real-time Personal Protective Equipment (PPE) Detection Using YOLOv4 and TensorFlow. *IEEE.*

Random Zoom Image Augmentation - Keras ImageDataGenerator. (27 8, 2021). (study tonight) Retrieved 16 6, 2022, from https://www.studytonight.com/post/random-zoom-image-augmentation-keras-imagedatagenerator

Ray, S. (13 9, 2017). *Analytic Vidhya*. Retrieved from Understanding Support Vector Machine(SVM) algorithm from examples (along with code): https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

seb. (3 5, 2021). *The Sigmoid Function and Binary Logistic Regression*. Retrieved 16 6, 2022, from https://programmathically.com/the-sigmoid-function-and-binary-logistic-regression/#:~:text=As%20you%20can%20see%2C%20the,1%20as%20potential%20output%20values

Selamat, M. H., & Rais, H. M. (2015). Image Face Recognition Using Hybrid Multiclass SVM (HM-SVM). *International Conference on Computer, Control, Informatics and Its Applications* (p. 6). Tronoh: IEEE.

Shankar, K. (2021). *What is Shear in data Augmentation*. (Quora) Retrieved 16 6, 2022, from https://www.quora.com/What-is-shear-in-data-augmentation

Srivastava, T. (27 3, 2018). *Analytic Vidhya*. Retrieved from Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R): https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

United States Department of Labor. (n.d.). Retrieved from 1910.133-Eye and face protection: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.133

United States Department of Labor. (n.d.). Retrieved from 1910.135-Head protection: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.135

United States Department of Labor. (n.d.). Retrieved from 1910.135-Head protection: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.135

United States Department of Labor. (n.d.). Retrieved from 1910.134-Respiratory protection: https://www.osha.gov/laws-regs/regulations/standardnumber/1910/1910.134

United States Department of Labor. (29 1, 2021). Retrieved from Protecting Workers: Guidance on Mitigating and Preventing the Spread of COVID-19 in the Workplace: https://www.osha.gov/coronavirus/safework

United States Department of Labor. (2022). Retrieved from Personal Protective Equipment: https://www.osha.gov/personal-protective-equipment/standards

Varma, S., & Das, S. (27 9, 2018). *Chapter 9 Training Neural Network Part 3* . Retrieved from Github: https://srdas.github.io/DLBook/HyperParameterSelection.html

Vinh, T. Q., & Anh, N. T. (2020). Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier. *IEEE.*

Wolfewicz, A. (21 4, 2022). *Levity*. Retrieved from Deep Learning vs. Machine Learning – What's The Difference?: https://levity.ai/blog/difference-machine-learning-deep-learning#:~:text=Machine%20learning%20means%20computers%20learning,as%20documents%2C%20images%20and%20text.

World Health Organization. (1 12, 2020). Mask Use In The Context of COVID-19. *Interim Guidance*, 22.

Yang, Y., Wang, J., & Yang, Y. (2012). IMPROVING SVM CLASSIFIER WITH PRIOR KNOWLEDGE IN MICROCALCIFICATION DETECTION1. *ICIP* (p. 4). Chicago: IEEE.

Yu, H., & Liu, H. (2014). Linear Regression for Head Pose Analysis. *International Joint Conference on Neural Networks (IJCNN)* (p. 6). Beijing: IEEE.

Zhang, J., Wang, H., Davoine, F., & Pan, C. (2012). Skin Detection via Linear Regression Tree. (p. 4). Tsukuba: IEEE.

APPENDIX A

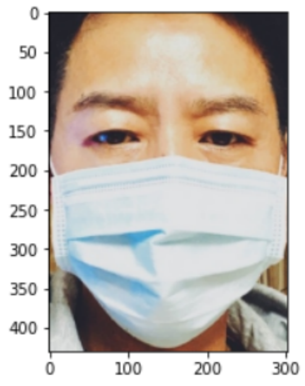
Coding for Training in 5 classes.

```
In [1]:
import matplotlib.pyplot as plt
import cv2
# Technically not necessary in newest versions of jupyter
%matplotlib inline
```

```
In [3]:
facemask1 = cv2.imread('PPE400_1/train/FACEMASK/with_mask_1.jpg') #read facemask ima
facemask1 = cv2.cvtColor(facemask1,cv2.COLOR_BGR2RGB)  #for correct color display
```

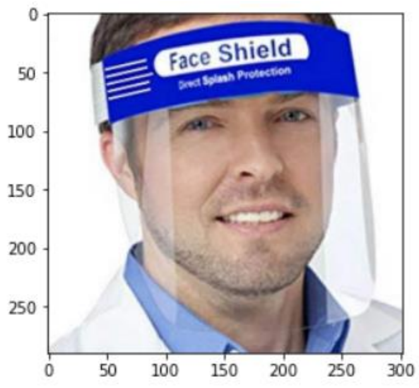
```
In [4]:
plt.imshow(facemask1)
```

```
Out[4]: <matplotlib.image.AxesImage at 0x2da8dce1bb0>
```



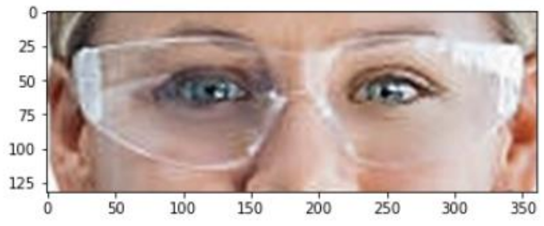
```
In [12]:
faceshield = cv2.imread('PPE400_1/train/FACESHIELD/s26.jpg') #read faceshield image
faceshield = cv2.cvtColor(faceshield,cv2.COLOR_BGR2RGB)
```

```
In [13]:
plt.imshow(faceshield)
```



```
In [14]:
goggle = cv2.imread('PPE400_1/train/GOGGLE/g13.jpg') #read faceshield image
goggle = cv2.cvtColor(goggle,cv2.COLOR_BGR2RGB)
```

```
Out[15]: <matplotlib.image.AxesImage at 0x2da8dfa4880>
```



```
In [16]: helmet = cv2.imread('PPE400_1/train/HELMET/h1.jpg') #read helmet image
         helmet = cv2.cvtColor(helmet,cv2.COLOR_BGR2RGB)
```

```
In [17]: plt.imshow(helmet)
```

```
Out[17]: <matplotlib.image.AxesImage at 0x2da8e011df0>
```

```
In [18]: jacket = cv2.imread('PPE400_1/train/SAFETYJACKET/s1.jpg') #read safety jacket image
         jacket = cv2.cvtColor(jacket,cv2.COLOR_BGR2RGB)
```

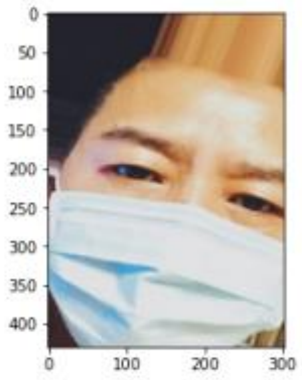
```
In [19]: plt.imshow(jacket)
```

```
Out[19]: <matplotlib.image.AxesImage at 0x2da8e07cc10>
```

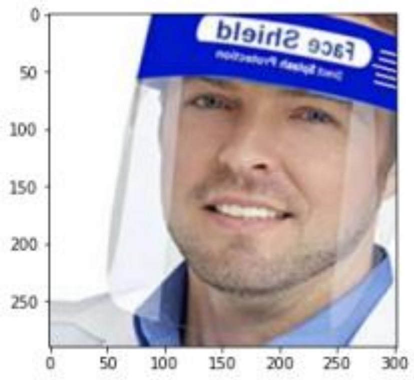
```
In [20]:  #shape for all images initially not same, adjust later
          #IMAGE DATA GENERATOR for rescaling

          from keras.preprocessing.image import ImageDataGenerator
          #from skimage import io
```

```
In [21]:  image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                                         width_shift_range=0.2, # Shift the pic width by a max
                                         height_shift_range=0.2, # Shift the pic height by a m
                                         rescale=1/255, # Rescale the image by normalzing it.
                                         shear_range=0.2, # Shear means cutting away part of t
                                         zoom_range=0.2, # Zoom in by 40% max
                                         horizontal_flip=True, # Allo horizontal flipping
                                         fill_mode='nearest' # Fill in missing pixels with the
                                        )
```

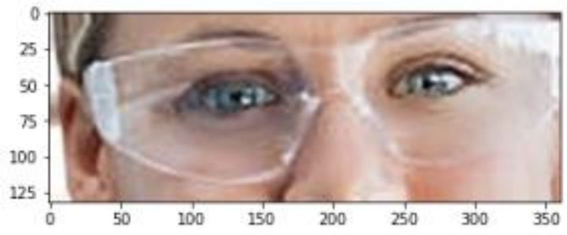
```
In [22]:  plt.imshow(image_gen.random_transform(facemask1))

Out[22]:  <matplotlib.image.AxesImage at 0x2da971e9490>
```



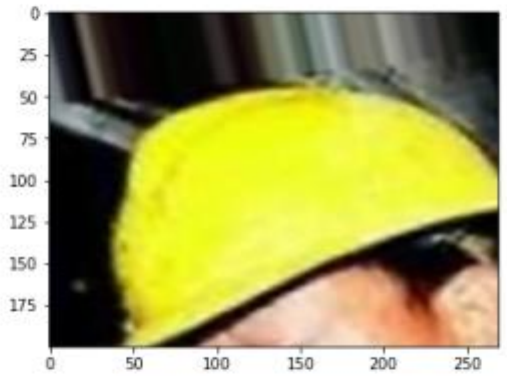
```
In [23]:  plt.imshow(image_gen.random_transform(faceshield))

Out[23]:  <matplotlib.image.AxesImage at 0x2da9822c430>
```



```
In [24]:  plt.imshow(image_gen.random_transform(goggle))

Out[24]:  <matplotlib.image.AxesImage at 0x2da9828cc70>
```



```
In [25]:  plt.imshow(image_gen.random_transform(helmet))

Out[25]:  <matplotlib.image.AxesImage at 0x2da982f7fd0>
```



```
In [26]:  plt.imshow(image_gen.random_transform(jacket))

Out[26]:  <matplotlib.image.AxesImage at 0x2da98360ee0>
```

```
In [28]:  #GENERATE MANY MANIPULATED IMAGES FROM THE DIRECTORY
          image_gen.flow_from_directory('PPE400_1/train')

          Found 2000 images belonging to 5 classes.
Out[28]:  <keras.preprocessing.image.DirectoryIterator at 0x2da984f7190>

In [29]:  image_gen.flow_from_directory('PPE400_1/test')

          Found 500 images belonging to 5 classes.
Out[29]:  <keras.preprocessing.image.DirectoryIterator at 0x2da98277f70>

In [30]:  #RESIZING IMAGES
          image_shape = (150,150,3)  #150 pixels by 150 pixels

In [31]:  #CREATING MODEL
          from keras.models import Sequential
          from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,D

In [32]:  model = Sequential()

          model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='
          model.add(MaxPooling2D(pool_size=(2, 2)))

          model.add(Flatten())

          model.add(Dense(128))
          model.add(Activation('relu'))

          # Dropouts help reduce overfitting by randomly turning neurons off during training.
# Here we say randomly turn off 50% of neurons.
model.add(Dropout(0.5))

model.add(Dense(5))
model.add(Activation('softmax'))

model.compile(loss='sparse_categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])
```

In [33]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 5, 5, 64)          36928

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 activation (Activation)     (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 5)                 645

 activation_1 (Activation)   (None, 5)                 0

=================================================================
Total params: 163,717
Trainable params: 163,717
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
batch_size = 16   #bole change nnt

train_image_gen = image_gen.flow_from_directory('PPE400_1/train',
                                                    target_size=image_shape[:2],
                                                    batch_size=batch_size,
                                                    class_mode='sparse')
```

```
Found 2000 images belonging to 5 classes.
```

In [35]:
```
test_image_gen = image_gen.flow_from_directory('PPE400_1/test',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='sparse')
```

```
Found 500 images belonging to 5 classes.
```

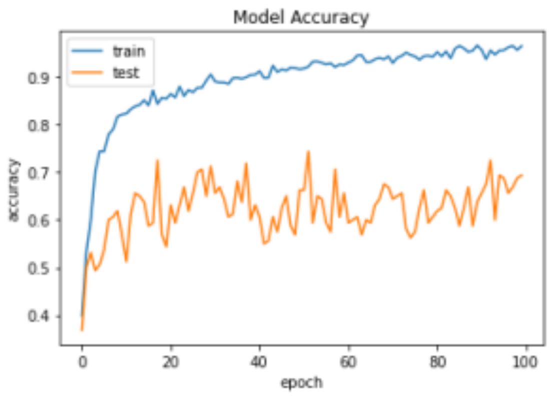
In [36]:
```
train_image_gen.class_indices   #what index belong to what class
```

Out[36]:
```
{'FACEMASK': 0, 'FACESHIELD': 1, 'GOGGLE': 2, 'HELMET': 3, 'SAFETYJACKET': 4}
```

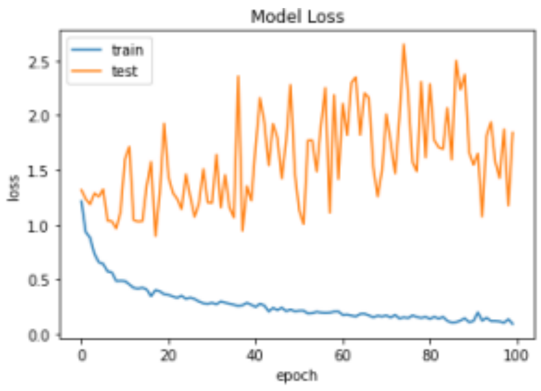
In [37]:
```
#TRAIN AND FIT
results = model.fit_generator(train_image_gen,epochs=100,
                              #steps_per_epoch=100,
                              validation_data=test_image_gen,
                              validation_steps=10)
```

In [38]:
```
#import warnings
#warnings.filterwarnings
#('ignore')

#**NI SIMPAN, RECORD!!!

model.save('ppe400_1_1.h5')
```

In [39]:
```
#kalau nak plot accuracy graph
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [40]:
```
#kalau nak plot loss graph
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
In [42]: #PREDICTING NEW IMAGES
         import numpy as np
         from keras.models import load_model
         #from keras.preprocessing import image
```

```
In [43]: new_model = load_model('ppe400_1_1.h5')
```

```
In [44]: facemask_file2 = 'PPE400_1/test/FACEMASK/f1.jpg'

         from keras.preprocessing import image

         #resize the image that nn expected in, tadi train 150,150, so sama juga
         facemask_img = image.load_img(facemask_file2,target_size=(150,150))

         #tukar kpd array
         facemask_img = image.img_to_array(facemask_img)

         #change above shape so that nn thinks that its a batch of one image
         #so need 1,150,150,3
         #import numpy as np  #sudah kat atas
         facemask_img = np.expand_dims(facemask_img,axis=0)

         #atas ni run sekali ja, nnt jadi salah keep expanding
         #make sure all values btwn 0 and 1
         facemask_img = facemask_img/255

         #TO PREDICT
         import numpy as np
         predict_x=model.predict(facemask_img)
         classes_x=np.argmax(predict_x,axis=1)

         prediction_prob = model.predict(facemask_img)
         # Output prediction

         print(f'Probability that image is facemask is: {prediction_prob} ')
```

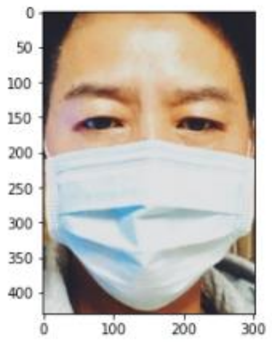
APPENDIX B

Coding for Training Binary Class, Combination 2 Face Mask.

```
In [1]:  import matplotlib.pyplot as plt
         import cv2
         # Technically not necessary in newest versions of jupyter
         %matplotlib inline
```

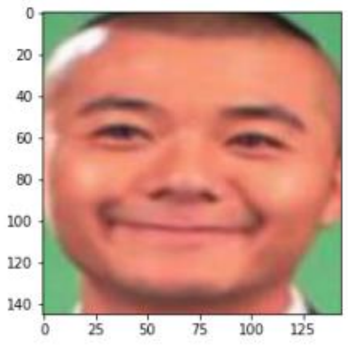
```
In [2]:  facemask = cv2.imread('FACEMASK/train/with_mask/with_mask_1.jpg') #read facemask image
         facemask = cv2.cvtColor(facemask,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [3]:  plt.imshow(facemask)

Out[3]: <matplotlib.image.AxesImage at 0x1b0fc7e1790>
```



```
In [4]:  nofacemask = cv2.imread('FACEMASK/train/without_mask/without_mask_5.jpg') #read facemas
         nofacemask = cv2.cvtColor(nofacemask,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [5]:  plt.imshow(nofacemask)

Out[5]: <matplotlib.image.AxesImage at 0x1b0fc8fe250>
```



```
from keras.preprocessing.image import ImageDataGenerator
```

In [7]:

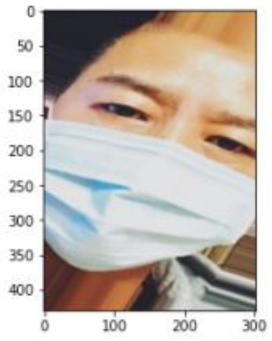
```
image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                               width_shift_range=0.1, # Shift the pic width by a max of
                               height_shift_range=0.1, # Shift the pic height by a max
                               rescale=1/255, # Rescale the image by normalzing it.
                               shear_range=0.2, # Shear means cutting away part of the
                               zoom_range=0.2, # Zoom in by 20% max
                               horizontal_flip=True, # Allo horizontal flipping
                               fill_mode='nearest' # Fill in missing pixels with the ne
                              )
```

In [8]:

```
plt.imshow(image_gen.random_transform(facemask))
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x1b085cf1f70>
```

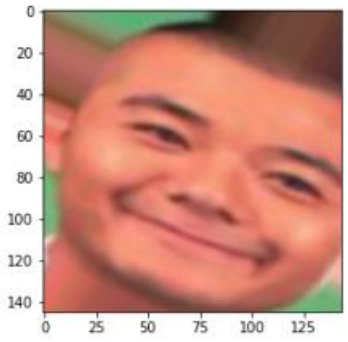


In [9]:

```
plt.imshow(image_gen.random_transform(nofacemask))
```

Out[9]:

```
<matplotlib.image.AxesImage at 0x1b085d6f2b0>
```



```
image_gen.flow_from_directory('FACEMASK/train')
```

```
Found 800 images belonging to 2 classes.
Out[10]: <keras.preprocessing.image.DirectoryIterator at 0x1b0fc92bca0>
```

```
In [11]: image_gen.flow_from_directory('FACEMASK/test')
```

```
Found 200 images belonging to 2 classes.
Out[11]: <keras.preprocessing.image.DirectoryIterator at 0x1b085ef4be0>
```

```
In [12]: #RESIZING IMAGES
         image_shape = (150,150,3)  #150 pixels by 150 pixels
```

```
In [13]: #CREATING MODEL
         from keras.models import Sequential
         from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,Dens
```

```
In [15]: model = Sequential()

         #Highest 4 layers
         model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Flatten())

         model.add(Dense(128))
         model.add(Activation('sigmoid'))   #highest sigmoid

         # Dropouts help reduce overfitting by randomly turning neurons off during training.
         # Here we say randomly turn off 50% of neurons.
         model.add(Dropout(0.5))

         model.add(Dense(1))
         model.add(Activation('sigmoid'))
         #sigmoid so that o/p btwn 0 to 1

         model.compile(loss='binary_crossentropy',
                       optimizer='Adam',          #2nd highest Adam
                       metrics=['accuracy'])
```

```
In [16]: model.summary()
```

```
Model: "sequential_1"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 74, 74, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_6 (Conv2D)           (None, 34, 34, 64)        36928

 max_pooling2d_6 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_7 (Conv2D)           (None, 15, 15, 64)        36928

 max_pooling2d_7 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 3136)              0

 dense_2 (Dense)             (None, 128)               401536

 activation_2 (Activation)   (None, 128)               0

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 1)                 129

 activation_3 (Activation)   (None, 1)                 0

=================================================================
Total params: 494,913
Trainable params: 494,913
Non-trainable params: 0
_________________________________________________________________
```

In [17]:
```
batch_size = 16    #depends, 16 ok

train_image_gen = image_gen.flow_from_directory('FACEMASK/train',
                                                target_size=image_shape[:2],
                                                batch_size=batch_size,
                                                class_mode='binary')
```

```
Found 800 images belonging to 2 classes.
```

In [18]:
```
batch_size = 16    #depends, 16 ok

test_image_gen = image_gen.flow_from_directory('FACEMASK/test',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 200 images belonging to 2 classes.
```

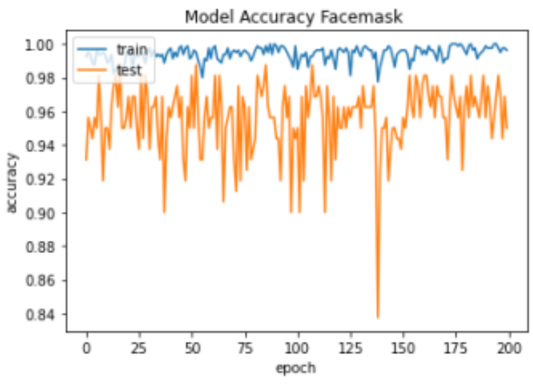
In [19]:
```
train_image_gen.class_indices    #what index belong to what class
```

Out[19]:
```
{'with_mask': 0, 'without_mask': 1}
```

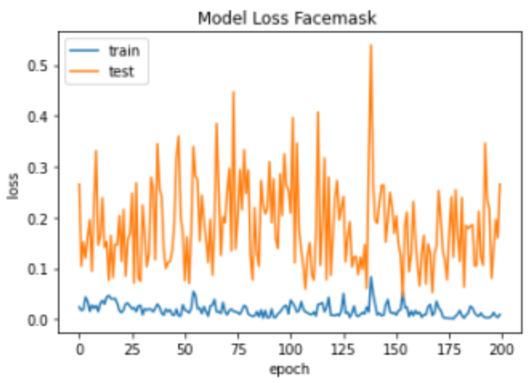
In [21]:
```
#TRAIN AND FIT
results = model.fit_generator(train_image_gen,epochs=200,    #highest 200 epochs
                              #steps_per_epoch=100,
                              validation_data=test_image_gen,
                              validation_steps=10)
```

In [22]:
```
model.save('facemask_C2Adam.h5')
```

```
In [25]: import numpy as np
         np.average(results.history['accuracy'])

Out[25]: 0.9940499901771546

In [26]: np.average(results.history['val_accuracy'])

Out[26]: 0.9559999975562096

In [27]: np.average(results.history['loss'])

Out[27]: 0.018041613011155276

In [28]: np.average(results.history['val_loss'])

Out[28]: 0.18403605287894606

In [29]: #kalau nak plot accuracy graph
         plt.plot(results.history['accuracy'])
         plt.plot(results.history['val_accuracy'])
         plt.title('Model Accuracy Facemask ')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()
```



```
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('Model Loss Facemask ')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
In [31]: #PREDICTING NEW IMAGES
         import numpy as np
         from keras.models import load_model
         #from keras.preprocessing import image

In [32]: new_model = load_model('facemask_C2Adam.h5')

In [33]: train_image_gen.class_indices

Out[33]: {'with_mask': 0, 'without_mask': 1}

In [34]: import numpy as np
         from keras.preprocessing import image

         facemask_file = 'FACEMASK/test/with_mask/f1.jpg'

         #resize the image that nn expected in, tadi train 150,150, so sama juga
         facemask_img = image.load_img(facemask_file,target_size=(150,150))

         #tukar kpd array
         facemask_img = image.img_to_array(facemask_img)

         #change above shape so that nn thinks that its a batch of one image
         #so need 1,150,150,3
         facemask_img = np.expand_dims(facemask_img,axis=0)

         #atas ni run sekali ja, nnt jadi salah keep expanding
         #make sure all values btwn 0 and 1
         facemask_img = facemask_img/255

         #TO PREDICT
         import numpy as np
         predict_x=model.predict(facemask_img)
         classes_x=np.argmax(predict_x,axis=1)

         prediction_prob = model.predict(facemask_img)
         print(f'Probability that image is with facemask is: {(1-prediction_prob)*100}% ')
```

APPENDIX C

Coding for Training Binary Class, Combination 3 Face Shield.

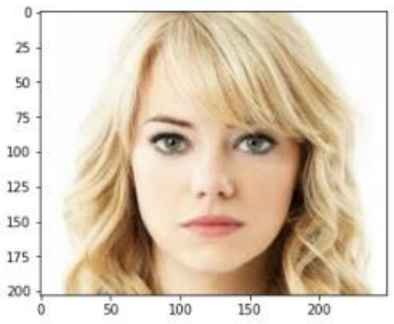
```
In [1]:  import matplotlib.pyplot as plt
         import cv2
         # Technically not necessary in newest versions of jupyter
         %matplotlib inline
```

```
In [2]:  faceshield = cv2.imread('FACESHIELD/train/with_fs/s19.jpg') #read facemask image
         faceshield = cv2.cvtColor(faceshield,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [3]:  plt.imshow(faceshield)

Out[3]:  <matplotlib.image.AxesImage at 0x1cda9492250>
```

```
In [4]:  nofaceshield = cv2.imread('FACESHIELD/train/without_fs/without_mask_640.jpg') #read fac
         nofaceshield = cv2.cvtColor(nofaceshield,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [5]:  plt.imshow(nofaceshield)

Out[5]:  <matplotlib.image.AxesImage at 0x1cda957aeb0>
```

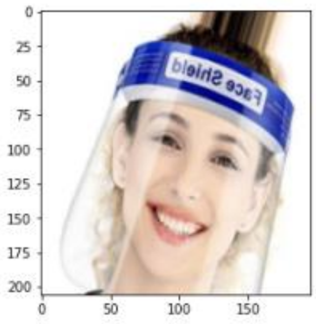


```
from keras.preprocessing.image import ImageDataGenerator
```

In [7]:
```
image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                               width_shift_range=0.1, # Shift the pic width by a max of
                               height_shift_range=0.1, # Shift the pic height by a max
                               rescale=1/255, # Rescale the image by normalzing it.
                               shear_range=0.2, # Shear means cutting away part of the
                               zoom_range=0.2, # Zoom in by 20% max
                               horizontal_flip=True, # Allo horizontal flipping
                               fill_mode='nearest' # Fill in missing pixels with the ne
                              )
```

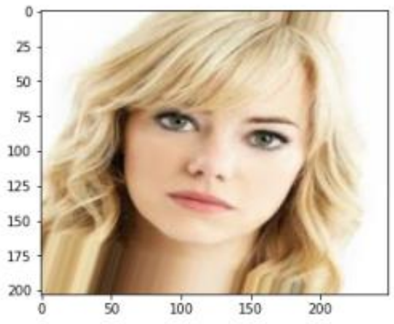
In [8]:
```
plt.imshow(image_gen.random_transform(faceshield))
```

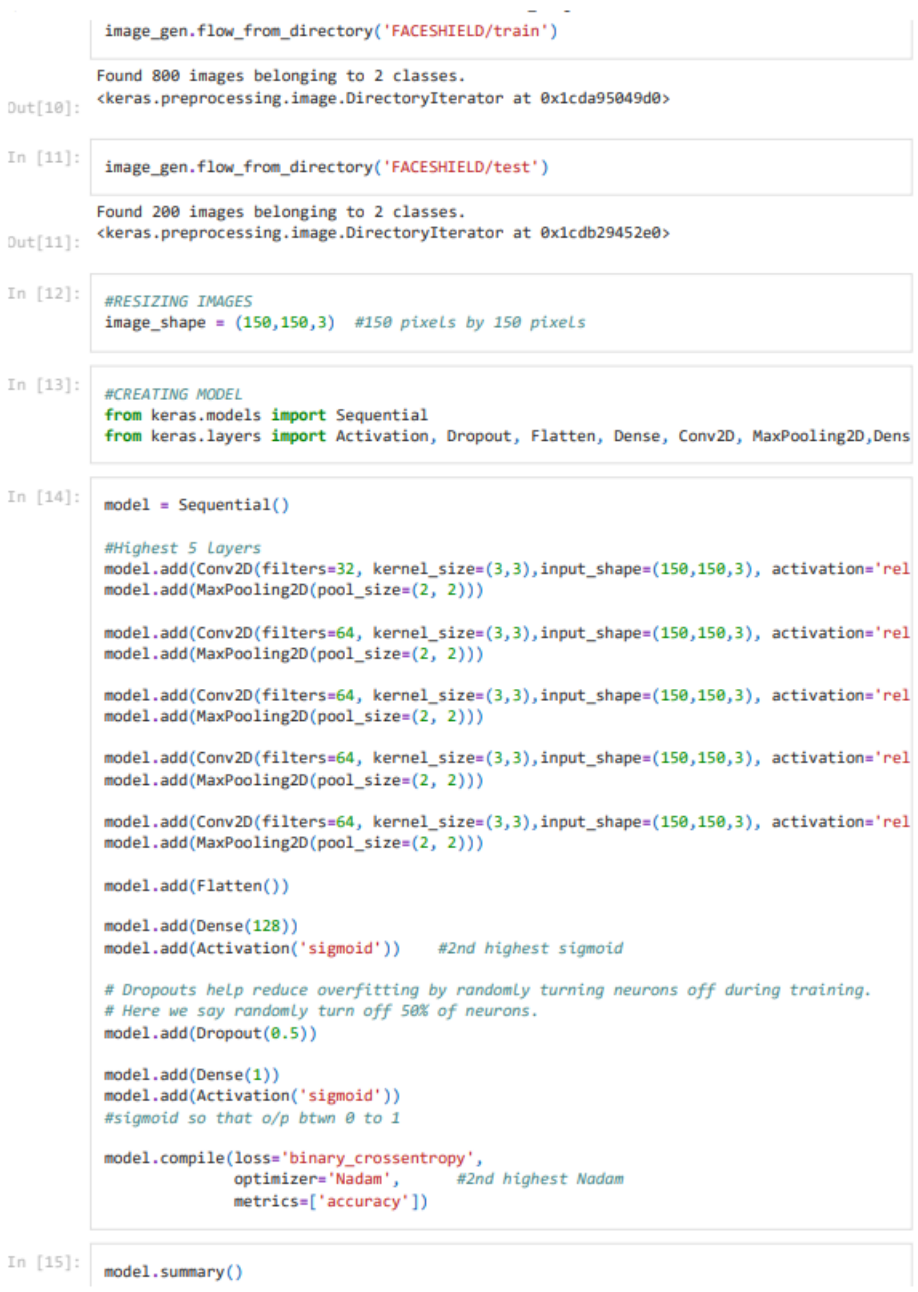
Out[8]:
```
<matplotlib.image.AxesImage at 0x1cdb2740a90>
```



In [9]:
```
plt.imshow(image_gen.random_transform(nofaceshield))
```

Out[9]:
```
<matplotlib.image.AxesImage at 0x1cdb27c0250>
```



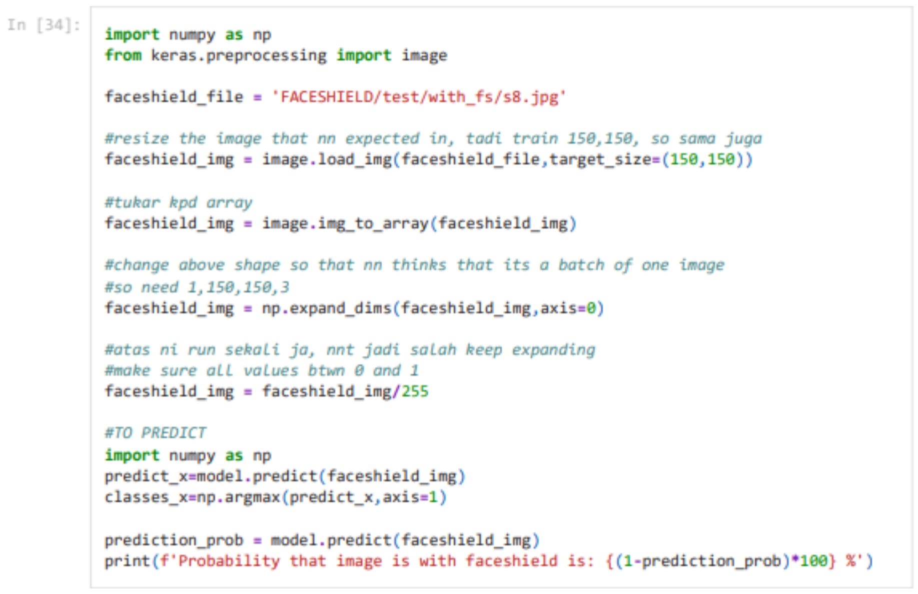


```
        image_gen.flow_from_directory('FACESHIELD/train')

        Found 800 images belonging to 2 classes.
Out[10]: <keras.preprocessing.image.DirectoryIterator at 0x1cda95049d0>

In [11]: image_gen.flow_from_directory('FACESHIELD/test')

        Found 200 images belonging to 2 classes.
Out[11]: <keras.preprocessing.image.DirectoryIterator at 0x1cdb29452e0>

In [12]: #RESIZING IMAGES
         image_shape = (150,150,3)  #150 pixels by 150 pixels

In [13]: #CREATING MODEL
         from keras.models import Sequential
         from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,Dens

In [14]: model = Sequential()

         #Highest 5 layers
         model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Flatten())

         model.add(Dense(128))
         model.add(Activation('sigmoid'))    #2nd highest sigmoid

         # Dropouts help reduce overfitting by randomly turning neurons off during training.
         # Here we say randomly turn off 50% of neurons.
         model.add(Dropout(0.5))

         model.add(Dense(1))
         model.add(Activation('sigmoid'))
         #sigmoid so that o/p btwn 0 to 1

         model.compile(loss='binary_crossentropy',
                       optimizer='Nadam',      #2nd highest Nadam
                       metrics=['accuracy'])

In [15]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 5, 5, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 256)               0         
                                                                 
 dense (Dense)               (None, 128)               32896     
                                                                 
 activation (Activation)     (None, 128)               0         
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 1)                 129       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 163,201
Trainable params: 163,201
Non-trainable params: 0
_________________________________________________________________
```

```
In [16]: batch_size = 16    #depends, 16 ok

         train_image_gen = image_gen.flow_from_directory('FACESHIELD/train',
                                                        target_size=image_shape[:2],
                                                        batch_size=batch_size,
                                                        class_mode='binary')

         Found 800 images belonging to 2 classes.

In [17]: batch_size = 16    #depends, 16 ok

         test_image_gen = image_gen.flow_from_directory('FACESHIELD/test',
                                                       target_size=image_shape[:2],
                                                       batch_size=batch_size,
                                                       class_mode='binary')

         Found 200 images belonging to 2 classes.

In [18]: train_image_gen.class_indices    #what index belong to what class

Out[18]: {'with_fs': 0, 'without_fs': 1}

In [21]: #TRAIN AND FIT
         results = model.fit_generator(train_image_gen,epochs=250,    #highest 250 epochs
                                       #steps_per_epoch=100,
                                       validation_data=test_image_gen,
                                       validation_steps=10)
```

```
In [22]: model.save('faceshield_C3sigmoid.h5')

In [23]: import numpy as np

         np.average(results.history['accuracy'])

Out[23]: 0.9958299901485443

In [24]: np.average(results.history['val_accuracy'])

Out[24]: 0.913224999666214

In [25]: np.average(results.history['loss'])

Out[25]: 0.012754233307554387

In [26]: np.average(results.history['val_loss'])

Out[26]: 0.4125162590146065

In [27]: results.history['accuracy']

In [29]: #kalau nak plot accuracy graph
         plt.plot(results.history['accuracy'])
         plt.plot(results.history['val_accuracy'])
         plt.title('Model Accuracy Faceshield')
         plt.ylabel('accuracy')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()

In [30]: #kalau nak plot loss graph
         plt.plot(results.history['loss'])
         plt.plot(results.history['val_loss'])
         plt.title('Model Loss Faceshield')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['train', 'test'], loc='upper left')
         plt.show()

In [31]: #PREDICTING NEW IMAGES
         import numpy as np
         from keras.models import load_model
         #from keras.preprocessing import image

In [32]: new_model = load_model('faceshield_C3sigmoid.h5')

In [33]: train_image_gen.class_indices

Out[33]: {'with_fs': 0, 'without_fs': 1}
```



```
In [34]:
import numpy as np
from keras.preprocessing import image

faceshield_file = 'FACESHIELD/test/with_fs/s8.jpg'

#resize the image that nn expected in, tadi train 150,150, so sama juga
faceshield_img = image.load_img(faceshield_file,target_size=(150,150))

#tukar kpd array
faceshield_img = image.img_to_array(faceshield_img)

#change above shape so that nn thinks that its a batch of one image
#so need 1,150,150,3
faceshield_img = np.expand_dims(faceshield_img,axis=0)

#atas ni run sekali ja, nnt jadi salah keep expanding
#make sure all values btwn 0 and 1
faceshield_img = faceshield_img/255

#TO PREDICT
import numpy as np
predict_x=model.predict(faceshield_img)
classes_x=np.argmax(predict_x,axis=1)

prediction_prob = model.predict(faceshield_img)
print(f'Probability that image is with faceshield is: {(1-prediction_prob)*100} %')
```

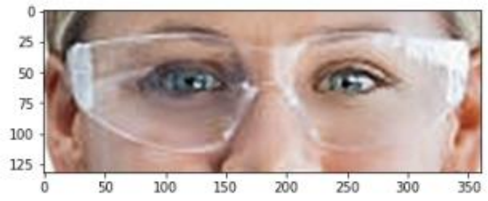
APPENDIX D

Coding for Training Binary Class, Combination 2 Safety Goggle.

```
In [1]:
import matplotlib.pyplot as plt
import cv2
# Technically not necessary in newest versions of jupyter
%matplotlib inline
```

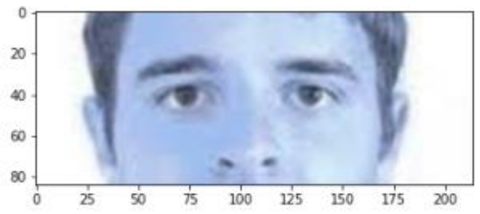
```
In [2]:
goggle = cv2.imread('GOGGLE/train/with_goggle/g13.jpg') #read facemask image
goggle = cv2.cvtColor(goggle,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [3]:
plt.imshow(goggle)

Out[3]: <matplotlib.image.AxesImage at 0x2c1bd7c5a00>
```



```
In [4]:
nogoggle = cv2.imread('GOGGLE/train/without_goggle/ng6.jpg') #read facemask image
nogoggled = cv2.cvtColor(nogoggle,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [5]:
plt.imshow(nogoggle)

Out[5]: <matplotlib.image.AxesImage at 0x2c1bd8dd9d0>
```

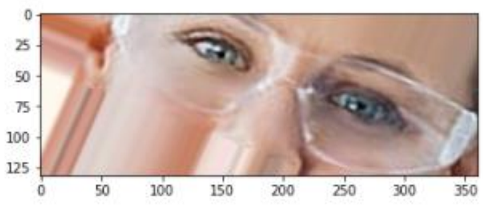


```
In [6]:
#shape for all images initially not same, adjust later
#IMAGE DATA GENERATOR for rescaling

from keras.preprocessing.image import ImageDataGenerator
```

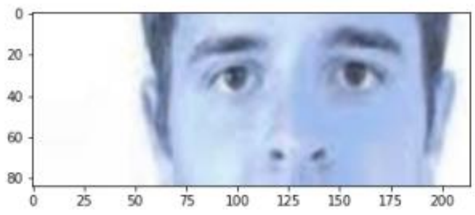
```
In [7]:
image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                               width_shift_range=0.1, # Shift the pic width by a max of
                               height_shift_range=0.1, # Shift the pic height by a max
                               rescale=1/255, # Rescale the image by normalzing it.
                               shear_range=0.2, # Shear means cutting away part of the
```

```
                                zoom_range=0.2, # Zoom in by 20% max
                                horizontal_flip=True, # Allo horizontal flipping
                                fill_mode='nearest' # Fill in missing pixels with the ne
                               )
```

```
In [8]: plt.imshow(image_gen.random_transform(goggle))

Out[8]: <matplotlib.image.AxesImage at 0x2c1c6a4cd00>
```



```
In [9]: plt.imshow(image_gen.random_transform(nogoggle))

Out[9]: <matplotlib.image.AxesImage at 0x2c1c6abfca0>
```

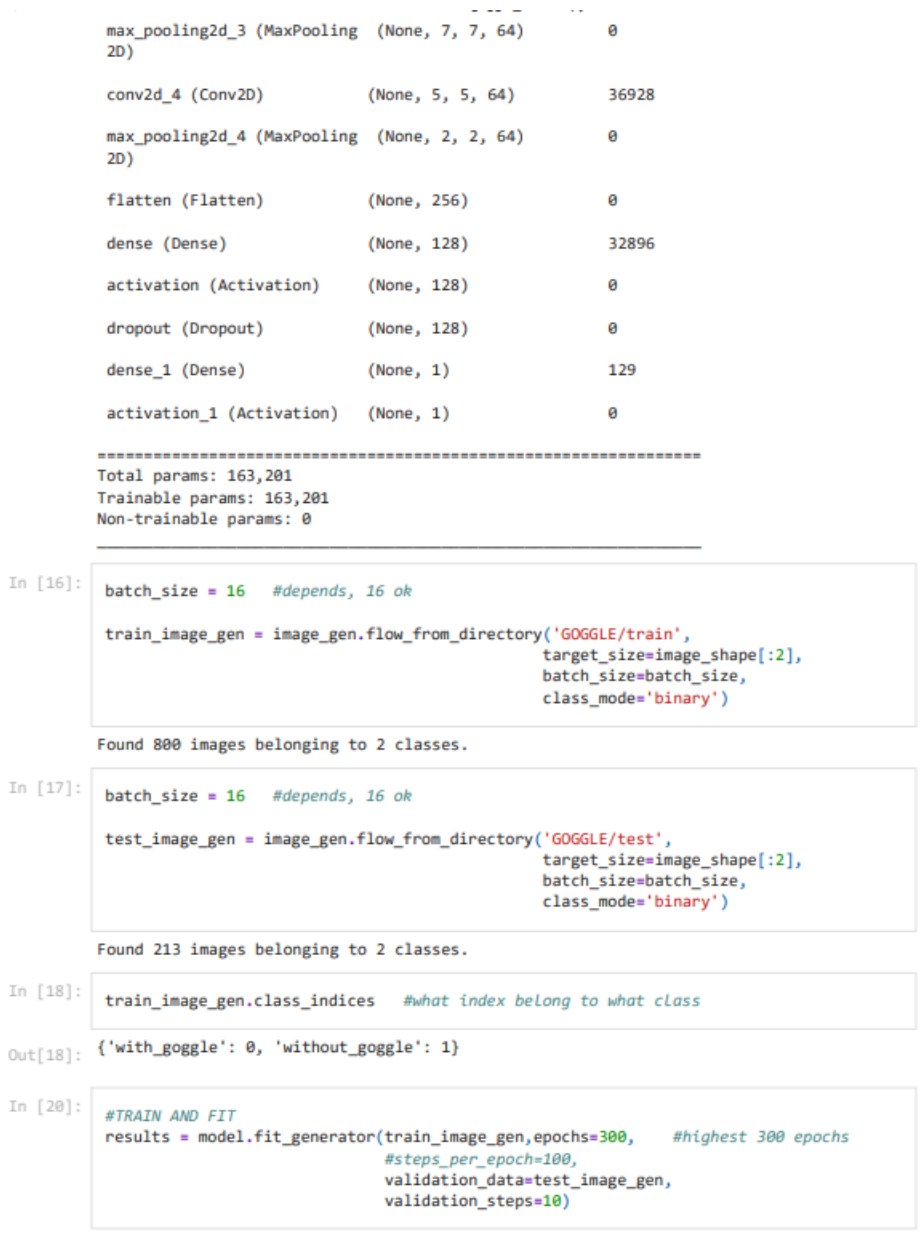


```
In [10]: #GENERATE MANY MANIPULATED IMAGES FROM THE DIRECTORY
         image_gen.flow_from_directory('GOGGLE/train')

         Found 800 images belonging to 2 classes.
Out[10]: <keras.preprocessing.image.DirectoryIterator at 0x2c1c6aaafd0>

In [11]: image_gen.flow_from_directory('GOGGLE/test')

         Found 213 images belonging to 2 classes.
Out[11]: <keras.preprocessing.image.DirectoryIterator at 0x2c1c6c4fe80>

In [12]: #RESIZING IMAGES
         image_shape = (150,150,3)  #150 pixels by 150 pixels

In [13]: #CREATING MODEL
         from keras.models import Sequential
         from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,Dens
```

```
In [14]: model = Sequential()

         #Highest 5 layers
         model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
         model.add(MaxPooling2D(pool_size=(2, 2)))

         model.add(Flatten())

         model.add(Dense(128))
         model.add(Activation('relu'))     #highest relu

         # Dropouts help reduce overfitting by randomly turning neurons off during training.
         # Here we say randomly turn off 50% of neurons.
         model.add(Dropout(0.5))

         model.add(Dense(1))
         model.add(Activation('sigmoid'))
         #sigmoid so that o/p btwn 0 to 1

         model.compile(loss='binary_crossentropy',
                       optimizer='Adam',          #2nd highest rmsprop
                       metrics=['accuracy'])
```

```
In [15]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928
```



```
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 5, 5, 64)          36928

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 128)               32896

 activation (Activation)     (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 163,201
Trainable params: 163,201
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
batch_size = 16   #depends, 16 ok

train_image_gen = image_gen.flow_from_directory('GOGGLE/train',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 800 images belonging to 2 classes.
```

In [17]:
```
batch_size = 16   #depends, 16 ok

test_image_gen = image_gen.flow_from_directory('GOGGLE/test',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

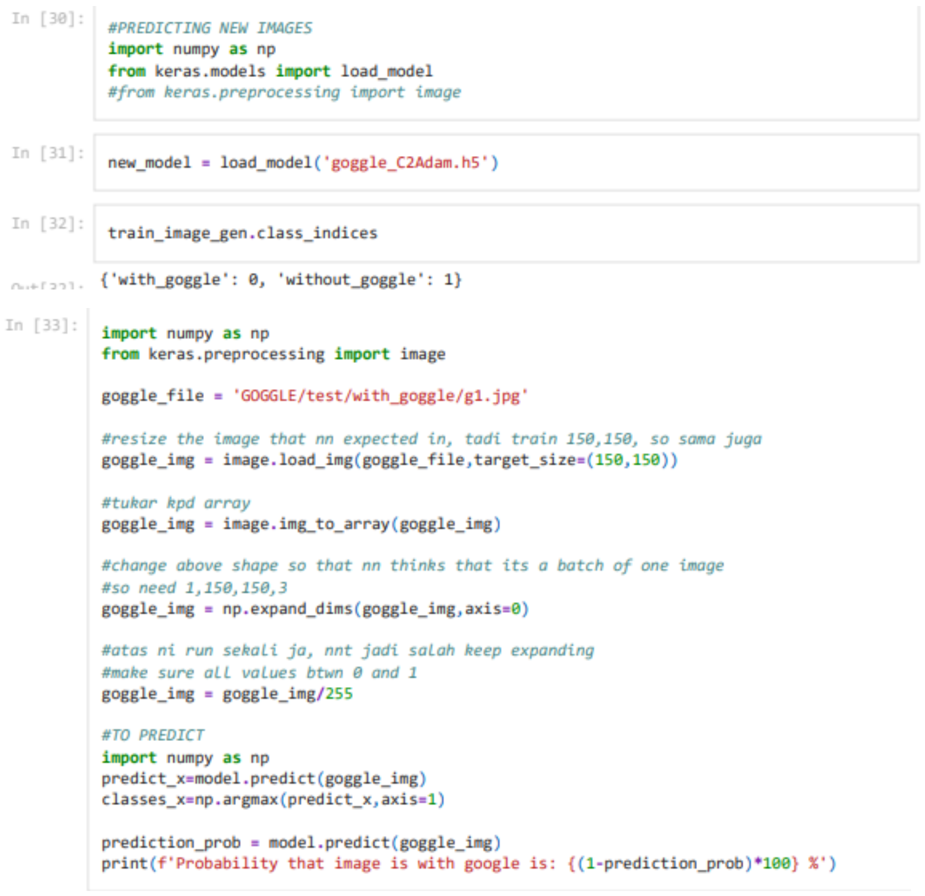
```
Found 213 images belonging to 2 classes.
```

In [18]:
```
train_image_gen.class_indices    #what index belong to what class
```

Out[18]:
```
{'with_goggle': 0, 'without_goggle': 1}
```

In [20]:
```
#TRAIN AND FIT
results = model.fit_generator(train_image_gen,epochs=300,     #highest 300 epochs
                              #steps_per_epoch=100,
                              validation_data=test_image_gen,
                              validation_steps=10)
```

```
In [21]:  model.save('goggle_C2Adam.h5')

In [22]:  import numpy as np
          np.average(results.history['accuracy'])

Out[22]:  0.996949989994367

In [23]:  np.average(results.history['val_accuracy'])

Out[23]:  0.8911874985694885

In [24]:  np.average(results.history['loss'])

Out[24]:  0.010405723120523664

In [25]:  np.average(results.history['val_loss'])

Out[25]:  0.9209939935803413

In [26]:  results.history['accuracy']

In [28]:  #kalau nak plot accuracy graph
          plt.plot(results.history['accuracy'])
          plt.plot(results.history['val_accuracy'])
          plt.title('Model Accuracy Goggle')
          plt.ylabel('accuracy')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()

In [29]:  #kalau nak plot loss graph
          plt.plot(results.history['loss'])
          plt.plot(results.history['val_loss'])
          plt.title('Model Loss Goggle')
          plt.ylabel('loss')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```



```
In [30]:  #PREDICTING NEW IMAGES
          import numpy as np
          from keras.models import load_model
          #from keras.preprocessing import image

In [31]:  new_model = load_model('goggle_C2Adam.h5')

In [32]:  train_image_gen.class_indices

Out[32]:  {'with_goggle': 0, 'without_goggle': 1}

In [33]:  import numpy as np
          from keras.preprocessing import image

          goggle_file = 'GOGGLE/test/with_goggle/g1.jpg'

          #resize the image that nn expected in, tadi train 150,150, so sama juga
          goggle_img = image.load_img(goggle_file,target_size=(150,150))

          #tukar kpd array
          goggle_img = image.img_to_array(goggle_img)

          #change above shape so that nn thinks that its a batch of one image
          #so need 1,150,150,3
          goggle_img = np.expand_dims(goggle_img,axis=0)

          #atas ni run sekali ja, nnt jadi salah keep expanding
          #make sure all values btwn 0 and 1
          goggle_img = goggle_img/255

          #TO PREDICT
          import numpy as np
          predict_x=model.predict(goggle_img)
          classes_x=np.argmax(predict_x,axis=1)

          prediction_prob = model.predict(goggle_img)
          print(f'Probability that image is with google is: {(1-prediction_prob)*100} %')
```

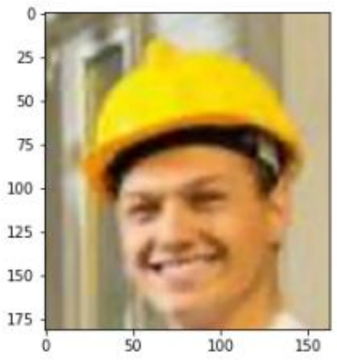
APPENDIX E

Coding for Training Binary Class, Combination 3 Safety Helmet.

```
In [1]:  import matplotlib.pyplot as plt
         import cv2
         # Technically not necessary in newest versions of jupyter
         %matplotlib inline
```

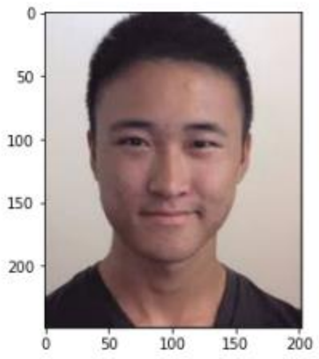
```
In [2]:  helmet = cv2.imread('HELMET/train/with_helmet/h141.jpg') #read facemask image
         helmet = cv2.cvtColor(helmet,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [3]:  plt.imshow(helmet)

Out[3]:  <matplotlib.image.AxesImage at 0x13f4aa47d00>
```



```
In [4]:  nohelmet = cv2.imread('HELMET/train/without_helmet/without_mask_2186.jpg') #read facema
         nohelmet = cv2.cvtColor(nohelmet,cv2.COLOR_BGR2RGB)  #for correct color display
```

```
In [5]:  plt.imshow(nohelmet)

Out[5]:  <matplotlib.image.AxesImage at 0x13f4ab653d0>
```

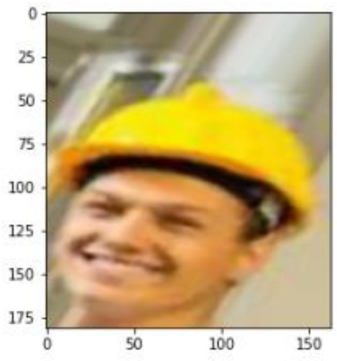


```
from keras.preprocessing.image import ImageDataGenerator
```

```
In [7]: image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                                       width_shift_range=0.1, # Shift the pic width by a max of
                                       height_shift_range=0.1, # Shift the pic height by a max
                                       rescale=1/255, # Rescale the image by normalzing it.
                                       shear_range=0.2, # Shear means cutting away part of the
                                       zoom_range=0.2, # Zoom in by 40% max
                                       horizontal_flip=True, # Allo horizontal flipping
                                       fill_mode='nearest' # Fill in missing pixels with the ne
                                      )
```

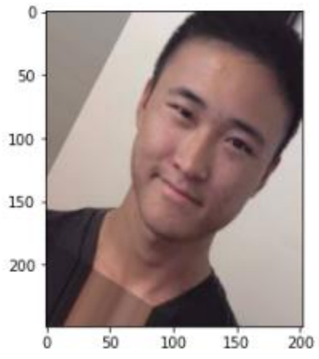
```
In [8]: plt.imshow(image_gen.random_transform(helmet))
```

```
Out[8]: <matplotlib.image.AxesImage at 0x13f53ce3e50>
```



```
In [9]: plt.imshow(image_gen.random_transform(nohelmet))
```

```
Out[9]: <matplotlib.image.AxesImage at 0x13f53d608b0>
```



```
image_gen.flow_from_directory('HELMET/train')
```

```
Found 808 images belonging to 2 classes.
```

Out[10]:
```
<keras.preprocessing.image.DirectoryIterator at 0x13f4ab73fa0>
```

In [11]:
```
image_gen.flow_from_directory('HELMET/test')
```

```
Found 207 images belonging to 2 classes.
```

Out[11]:
```
<keras.preprocessing.image.DirectoryIterator at 0x13f53edbc10>
```

In [12]:
```
#RESIZING IMAGES
image_shape = (150,150,3)  #150 pixels by 150 pixels
```

In [13]:
```
#CREATING MODEL
from keras.models import Sequential
from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,Dens
```

In [14]:
```
model = Sequential()

#highest 5 layers
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('tanh'))      #2nd highest tanh

# Dropouts help reduce overfitting by randomly turning neurons off during training.
# Here we say randomly turn off 50% of neurons.
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))
#sigmoid so that o/p btwn 0 to 1

model.compile(loss='binary_crossentropy',
              optimizer='Adam',          #2nd highest Adam
              metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 5, 5, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 256)               0         
                                                                 
 dense (Dense)               (None, 128)               32896     
                                                                 
 activation (Activation)     (None, 128)               0         
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 1)                 129       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 163,201
Trainable params: 163,201
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

```
batch_size = 16   #depends, 16 ok

train_image_gen = image_gen.flow_from_directory('HELMET/train',
                                                target_size=image_shape[:2],
                                                batch_size=batch_size,
                                                class_mode='binary')
```

```
test_image_gen = image_gen.flow_from_directory('HELMET/test',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 207 images belonging to 2 classes.
```

In [18]:
```
train_image_gen.class_indices    #what index belong to what class
```

Out[18]:
```
{'with_helmet': 0, 'without_helmet': 1}
```

In [20]:
```
#TRAIN AND FIT
results = model.fit_generator(train_image_gen,epochs=200,     #highest 200 epochs
                              #steps_per_epoch=100,
                              validation_data=test_image_gen,
                              validation_steps=10)
```

In [21]:
```
model.save('helmet_C3tanh.h5')
```

In [22]:
```
import numpy as np
np.average(results.history['accuracy'])
```

Out[22]:
```
0.9938923186063766
```

In [23]:
```
np.average(results.history['val_accuracy'])
```

Out[23]:
```
0.989093744456768
```

In [24]:
```
np.average(results.history['loss'])
```

Out[24]:
```
0.02397972063627094
```

In [25]:
```
np.average(results.history['val_loss'])
```

Out[25]:
```
0.03295112756604794
```

In [26]:
```
results.history['accuracy']
```

In [28]:
```
#kalau nak plot accuracy graph
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('Model Accuracy Helmet')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

In [29]:
```
#kalau nak plot loss graph
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('Model Loss Helmet')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

In [30]:
```
#PREDICTING NEW IMAGES
import numpy as np
from keras.models import load_model
#from keras.preprocessing import image
```

In [31]:
```
new_model = load_model('helmet_C3tanh.h5')
```

In [32]:
```
train_image_gen.class_indices
```

Out[32]:
```
{'with_helmet': 0, 'without_helmet': 1}
```

In [33]:
```
import numpy as np
from keras.preprocessing import image

helmet_file = 'HELMET/test/with_helmet/h10.jpg'

#resize the image that nn expected in, tadi train 150,150, so sama juga
helmet_img = image.load_img(helmet_file,target_size=(150,150))

#tukar kpd array
helmet_img = image.img_to_array(helmet_img)

#change above shape so that nn thinks that its a batch of one image
#so need 1,150,150,3
helmet_img = np.expand_dims(helmet_img,axis=0)

#atas ni run sekali ja, nnt jadi salah keep expanding
#make sure all values btwn 0 and 1
helmet_img = helmet_img/255

#TO PREDICT
import numpy as np
predict_x=model.predict(helmet_img)
classes_x=np.argmax(predict_x,axis=1)

prediction_prob = model.predict(helmet_img)
print(f'Probability that image is with helmet is: {(1-prediction_prob)*100} %')
```

APPENDIX F

Coding for Training Binary Class, Combination 3 Safety Jacket.

```
In [1]:  import matplotlib.pyplot as plt
         import cv2
         # Technically not necessary in newest versions of jupyter
         %matplotlib inline

In [2]:  safety_jacket = cv2.imread('SAFETY_JACKET/train/with_sj/s1.jpg') #read facemask image
         safety_jacket = cv2.cvtColor(safety_jacket,cv2.COLOR_BGR2RGB)  #for correct color displ

In [3]:  plt.imshow(safety_jacket)

Out[3]:  <matplotlib.image.AxesImage at 0x247a0747a30>
```

```
In [4]:  nosafety_jacket = cv2.imread('SAFETY_JACKET/train/without_sj/download (8).jpg') #read f
         nosafety_jacket = cv2.cvtColor(nosafety_jacket,cv2.COLOR_BGR2RGB)  #for correct color d

In [5]:  plt.imshow(nosafety_jacket)

Out[5]:  <matplotlib.image.AxesImage at 0x247a085f730>
```

```
#IMAGE DATA GENERATOR for rescaling

from keras.preprocessing.image import ImageDataGenerator
```

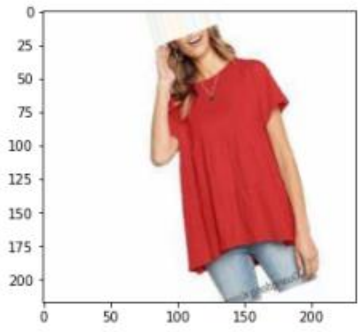
```
In [7]: image_gen = ImageDataGenerator(rotation_range=30, # rotate the image 30 degrees
                                       width_shift_range=0.1, # Shift the pic width by a max of
                                       height_shift_range=0.1, # Shift the pic height by a max
                                       rescale=1/255, # Rescale the image by normalzing it.
                                       shear_range=0.2, # Shear means cutting away part of the
                                       zoom_range=0.2, # Zoom in by 40% max
                                       horizontal_flip=True, # Allo horizontal flipping
                                       fill_mode='nearest' # Fill in missing pixels with the ne
                                      )
```

```
In [8]: plt.imshow(image_gen.random_transform(safety_jacket))
```

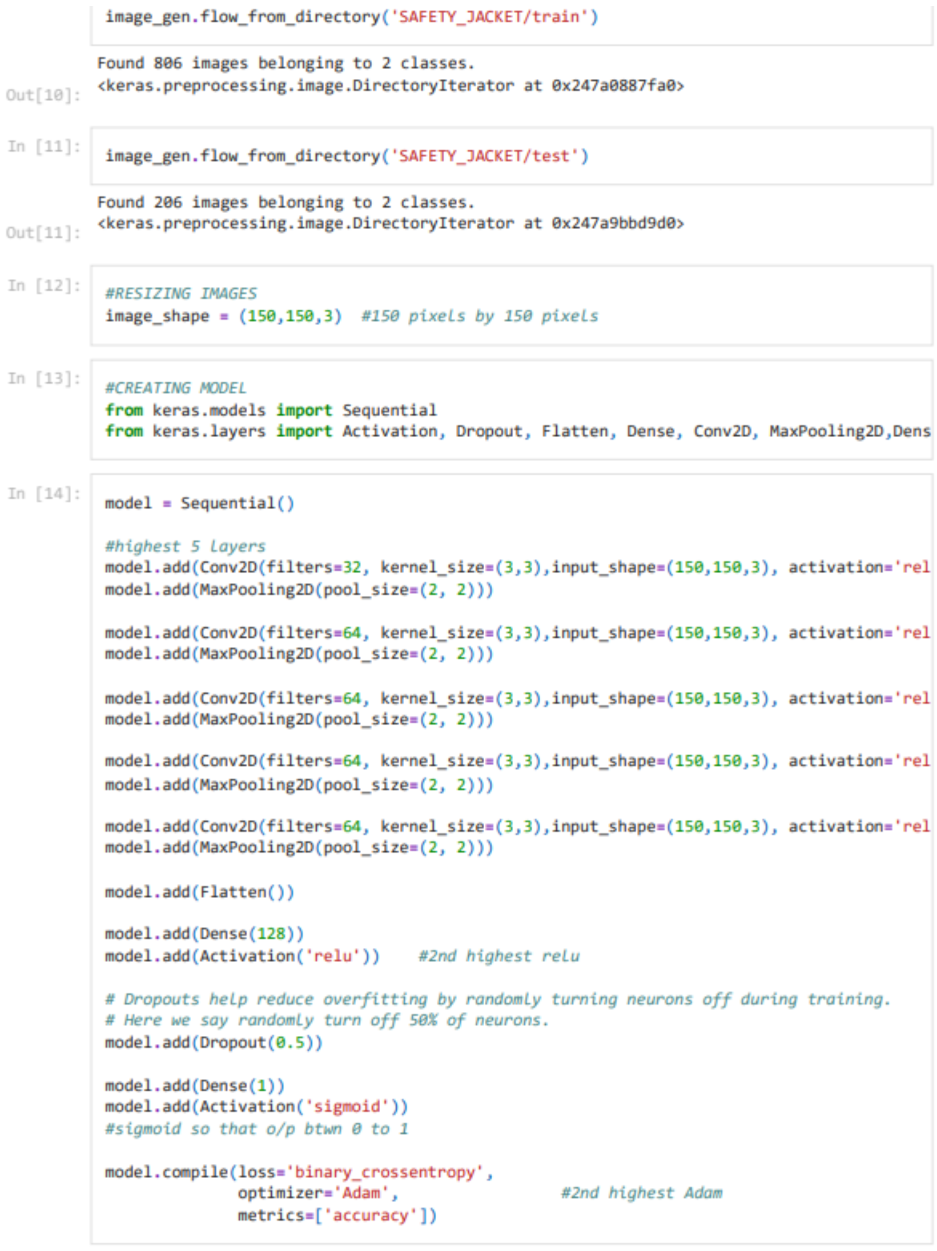
```
Out[8]: <matplotlib.image.AxesImage at 0x247a99c0f10>
```

```
In [9]: plt.imshow(image_gen.random_transform(nosafety_jacket))
```

```
Out[9]: <matplotlib.image.AxesImage at 0x247a9a33b80>
```



```
In [10]: #GENERATE MANY MANIPULATED IMAGES FROM THE DIRECTORY
```



```
image_gen.flow_from_directory('SAFETY_JACKET/train')
```

```
Found 806 images belonging to 2 classes.
```

Out[10]:

```
<keras.preprocessing.image.DirectoryIterator at 0x247a0887fa0>
```

In [11]:

```
image_gen.flow_from_directory('SAFETY_JACKET/test')
```

```
Found 206 images belonging to 2 classes.
```

Out[11]:

```
<keras.preprocessing.image.DirectoryIterator at 0x247a9bbd9d0>
```

In [12]:

```
#RESIZING IMAGES
image_shape = (150,150,3)  #150 pixels by 150 pixels
```

In [13]:

```
#CREATING MODEL
from keras.models import Sequential
from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D,Dens
```

In [14]:

```
model = Sequential()

#highest 5 layers
model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3),input_shape=(150,150,3), activation='rel
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))     #2nd highest relu

# Dropouts help reduce overfitting by randomly turning neurons off during training.
# Here we say randomly turn off 50% of neurons.
model.add(Dropout(0.5))

model.add(Dense(1))
model.add(Activation('sigmoid'))
#sigmoid so that o/p btwn 0 to 1

model.compile(loss='binary_crossentropy',
              optimizer='Adam',               #2nd highest Adam
              metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 34, 34, 64)        36928     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 17, 17, 64)       0         
 2D)                                                             
                                                                 
 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928     
                                                                 
 max_pooling2d_3 (MaxPooling  (None, 7, 7, 64)         0         
 2D)                                                             
                                                                 
 conv2d_4 (Conv2D)           (None, 5, 5, 64)          36928     
                                                                 
 max_pooling2d_4 (MaxPooling  (None, 2, 2, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 256)               0         
                                                                 
 dense (Dense)               (None, 128)               32896     
                                                                 
 activation (Activation)     (None, 128)               0         
                                                                 
 dropout (Dropout)           (None, 128)               0         
                                                                 
 dense_1 (Dense)             (None, 1)                 129       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 163,201
Trainable params: 163,201
Non-trainable params: 0
_________________________________________________________________
```

In [16]:

```
batch_size = 16   #depends, 16 ok

train_image_gen = image_gen.flow_from_directory('SAFETY_JACKET/train',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 806 images belonging to 2 classes.
```

```
test_image_gen = image_gen.flow_from_directory('SAFETY_JACKET/test',
                                               target_size=image_shape[:2],
                                               batch_size=batch_size,
                                               class_mode='binary')
```

```
Found 206 images belonging to 2 classes.
```

In [18]:

```
train_image_gen.class_indices    #what index belong to what class
```

Out[18]:

```
{'with_sj': 0, 'without_sj': 1}
```

In [19]:

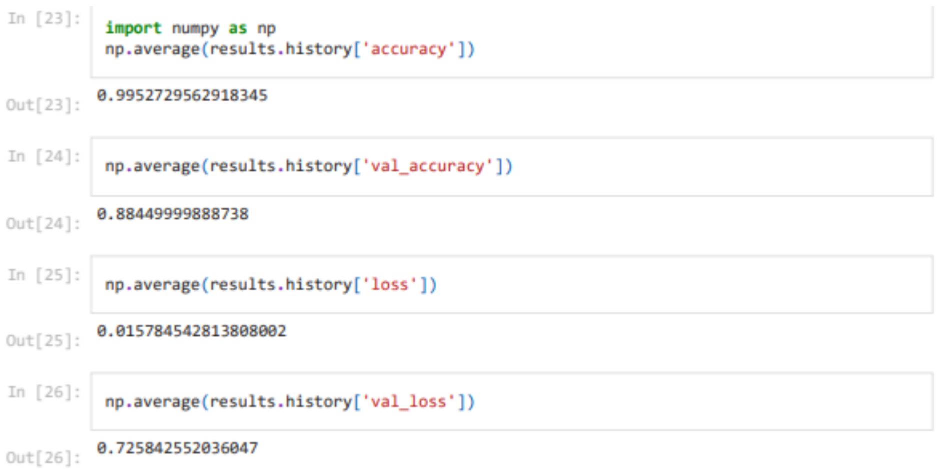
```
#TRAIN AND FIT
results = model.fit_generator(train_image_gen,epochs=300,        #highest 300 epochs
                              #steps_per_epoch=100,
                              validation_data=test_image_gen,
                              validation_steps=10)
```

In [20]:

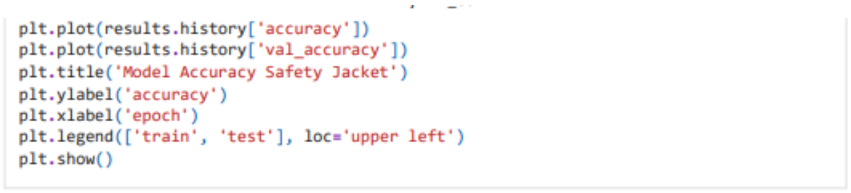
```
model.save('jacket_C3relu.h5')
```

In [21]:

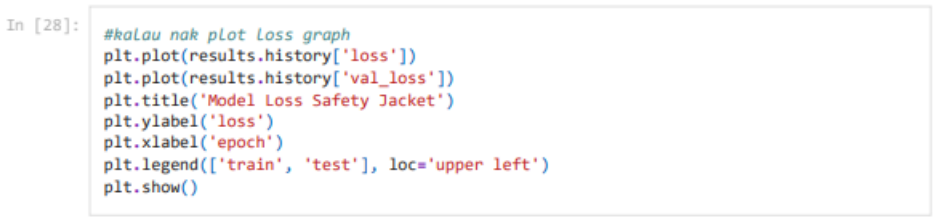
```
results.history['accuracy']
```



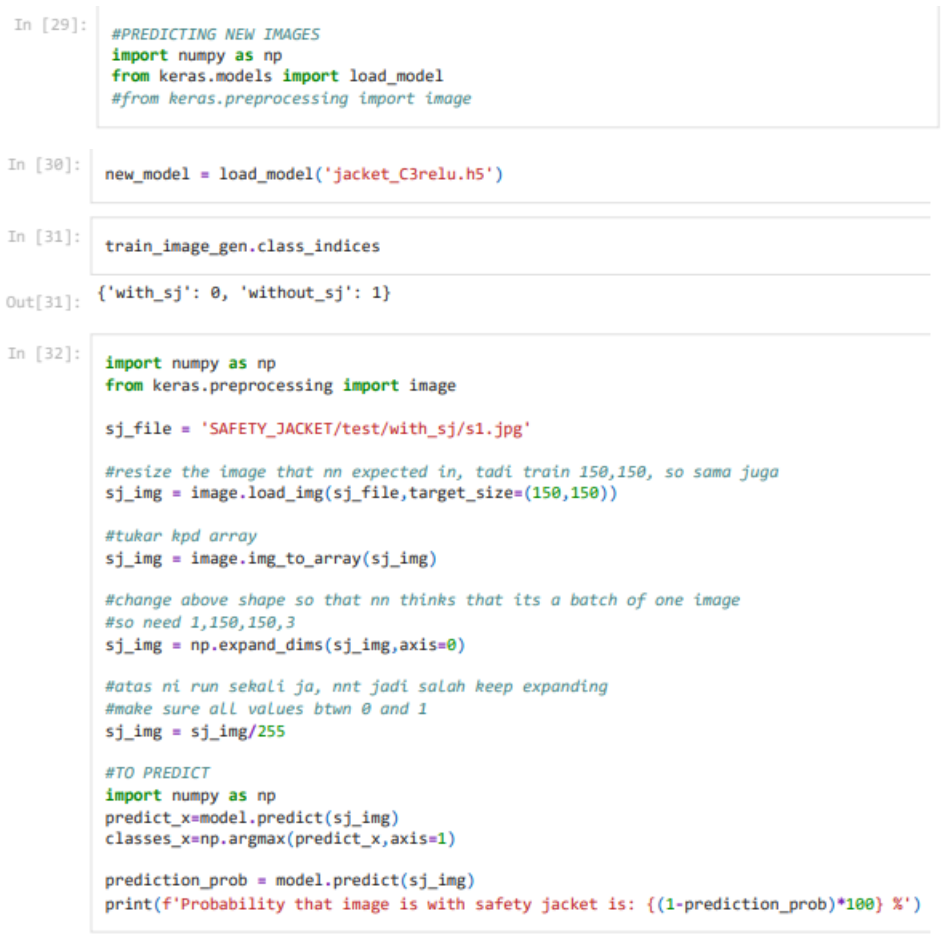
```
In [23]:  import numpy as np
          np.average(results.history['accuracy'])

Out[23]:  0.9952729562918345

In [24]:  np.average(results.history['val_accuracy'])

Out[24]:  0.88449999888738

In [25]:  np.average(results.history['loss'])

Out[25]:  0.015784542813808002

In [26]:  np.average(results.history['val_loss'])

Out[26]:  0.725842552036047
```



```
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('Model Accuracy Safety Jacket')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
In [28]:  #kalau nak plot loss graph
          plt.plot(results.history['loss'])
          plt.plot(results.history['val_loss'])
          plt.title('Model Loss Safety Jacket')
          plt.ylabel('loss')
          plt.xlabel('epoch')
          plt.legend(['train', 'test'], loc='upper left')
          plt.show()
```



```
In [29]:  #PREDICTING NEW IMAGES
          import numpy as np
          from keras.models import load_model
          #from keras.preprocessing import image
```

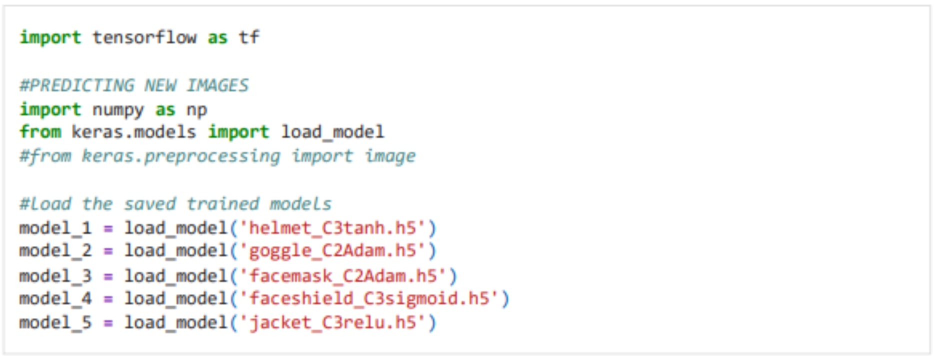
```
In [30]:  new_model = load_model('jacket_C3relu.h5')
```

```
In [31]:  train_image_gen.class_indices

Out[31]: {'with_sj': 0, 'without_sj': 1}
```

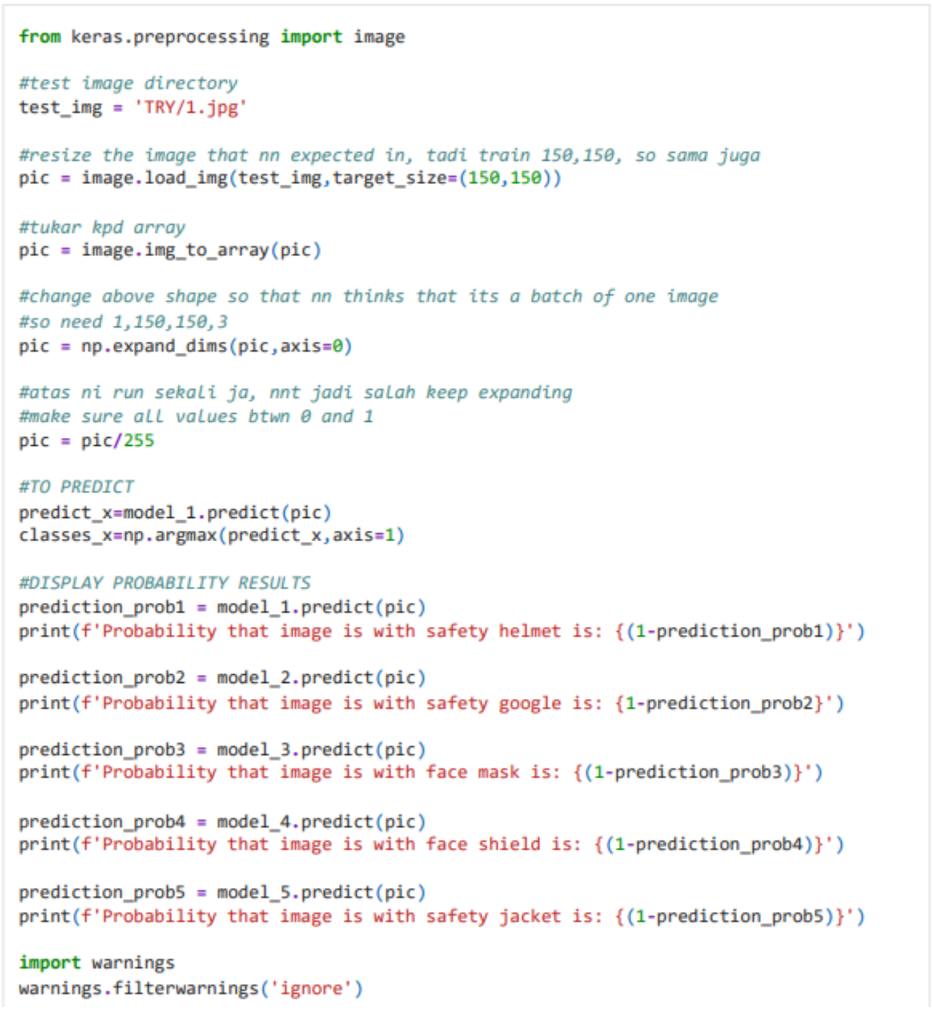
```
In [32]:  import numpy as np
          from keras.preprocessing import image

          sj_file = 'SAFETY_JACKET/test/with_sj/s1.jpg'

          #resize the image that nn expected in, tadi train 150,150, so sama juga
          sj_img = image.load_img(sj_file,target_size=(150,150))

          #tukar kpd array
          sj_img = image.img_to_array(sj_img)

          #change above shape so that nn thinks that its a batch of one image
          #so need 1,150,150,3
          sj_img = np.expand_dims(sj_img,axis=0)

          #atas ni run sekali ja, nnt jadi salah keep expanding
          #make sure all values btwn 0 and 1
          sj_img = sj_img/255

          #TO PREDICT
          import numpy as np
          predict_x=model.predict(sj_img)
          classes_x=np.argmax(predict_x,axis=1)

          prediction_prob = model.predict(sj_img)
          print(f'Probability that image is with safety jacket is: {(1-prediction_prob)*100} %')
```

APPENDIX G

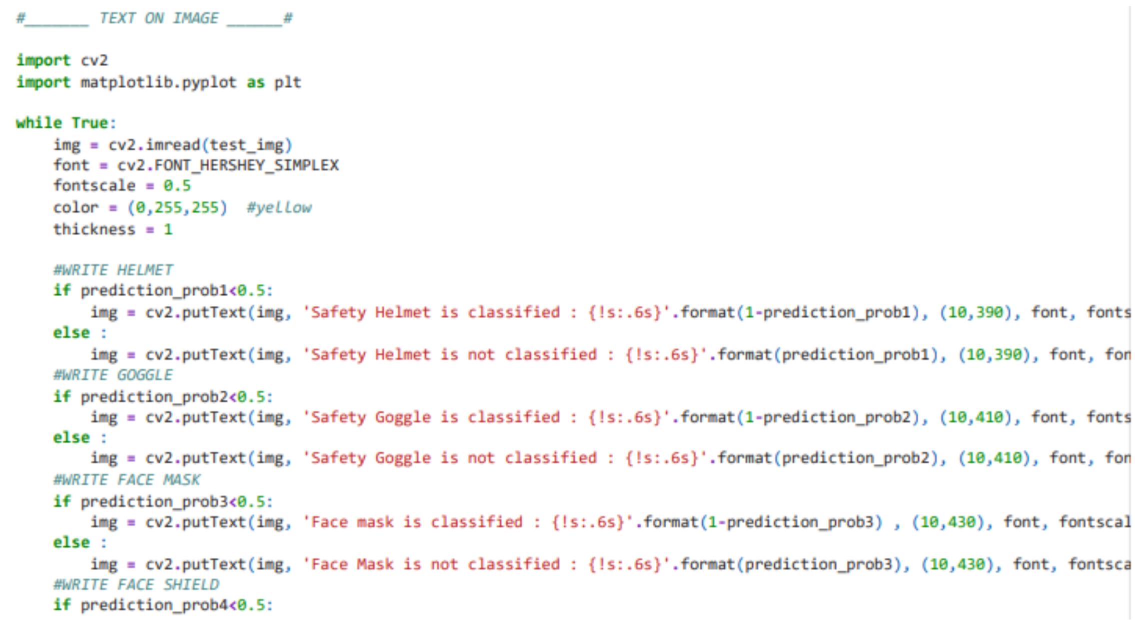
Coding for Testing 12 new images for Binary Class.



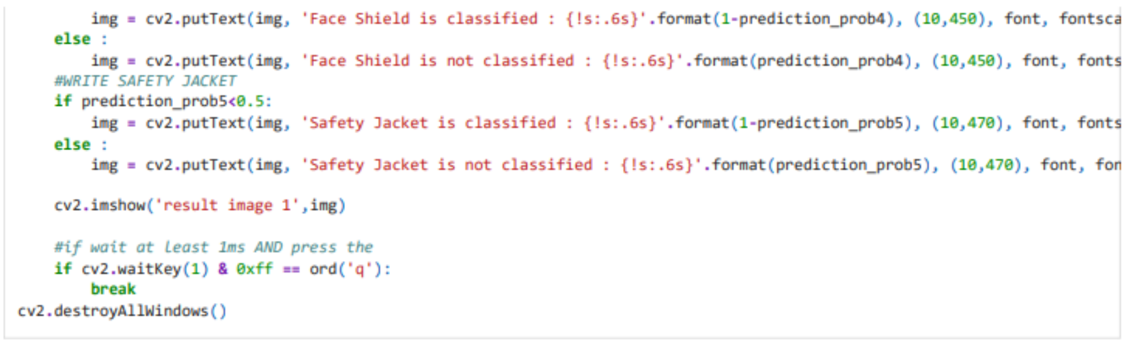
```python
import tensorflow as tf

#PREDICTING NEW IMAGES
import numpy as np
from keras.models import load_model
#from keras.preprocessing import image

#load the saved trained models
model_1 = load_model('helmet_C3tanh.h5')
model_2 = load_model('goggle_C2Adam.h5')
model_3 = load_model('facemask_C2Adam.h5')
model_4 = load_model('faceshield_C3sigmoid.h5')
model_5 = load_model('jacket_C3relu.h5')
```



```python
from keras.preprocessing import image

#test image directory
test_img = 'TRY/1.jpg'

#resize the image that nn expected in, tadi train 150,150, so sama juga
pic = image.load_img(test_img,target_size=(150,150))

#tukar kpd array
pic = image.img_to_array(pic)

#change above shape so that nn thinks that its a batch of one image
#so need 1,150,150,3
pic = np.expand_dims(pic,axis=0)

#atas ni run sekali ja, nnt jadi salah keep expanding
#make sure all values btwn 0 and 1
pic = pic/255

#TO PREDICT
predict_x=model_1.predict(pic)
classes_x=np.argmax(predict_x,axis=1)

#DISPLAY PROBABILITY RESULTS
prediction_prob1 = model_1.predict(pic)
print(f'Probability that image is with safety helmet is: {(1-prediction_prob1)}')

prediction_prob2 = model_2.predict(pic)
print(f'Probability that image is with safety google is: {1-prediction_prob2}')

prediction_prob3 = model_3.predict(pic)
print(f'Probability that image is with face mask is: {(1-prediction_prob3)}')

prediction_prob4 = model_4.predict(pic)
print(f'Probability that image is with face shield is: {(1-prediction_prob4)}')

prediction_prob5 = model_5.predict(pic)
print(f'Probability that image is with safety jacket is: {(1-prediction_prob5)}')

import warnings
warnings.filterwarnings('ignore')
```



```
#______ TEXT ON IMAGE ______#

import cv2
import matplotlib.pyplot as plt

while True:
    img = cv2.imread(test_img)
    font = cv2.FONT_HERSHEY_SIMPLEX
    fontscale = 0.5
    color = (0,255,255)  #yellow
    thickness = 1

    #WRITE HELMET
    if prediction_prob1<0.5:
        img = cv2.putText(img, 'Safety Helmet is classified : {!s:.6s}'.format(1-prediction_prob1), (10,390), font, fonts
    else :
        img = cv2.putText(img, 'Safety Helmet is not classified : {!s:.6s}'.format(prediction_prob1), (10,390), font, fon
    #WRITE GOGGLE
    if prediction_prob2<0.5:
        img = cv2.putText(img, 'Safety Goggle is classified : {!s:.6s}'.format(1-prediction_prob2), (10,410), font, fonts
    else :
        img = cv2.putText(img, 'Safety Goggle is not classified : {!s:.6s}'.format(prediction_prob2), (10,410), font, fon
    #WRITE FACE MASK
    if prediction_prob3<0.5:
        img = cv2.putText(img, 'Face mask is classified : {!s:.6s}'.format(1-prediction_prob3) , (10,430), font, fontscal
    else :
        img = cv2.putText(img, 'Face Mask is not classified : {!s:.6s}'.format(prediction_prob3), (10,430), font, fontsca
    #WRITE FACE SHIELD
    if prediction_prob4<0.5:
```



```
        img = cv2.putText(img, 'Face Shield is classified : {!s:.6s}'.format(1-prediction_prob4), (10,450), font, fontsca
    else :
        img = cv2.putText(img, 'Face Shield is not classified : {!s:.6s}'.format(prediction_prob4), (10,450), font, fonts
    #WRITE SAFETY JACKET
    if prediction_prob5<0.5:
        img = cv2.putText(img, 'Safety Jacket is classified : {!s:.6s}'.format(1-prediction_prob5), (10,470), font, fonts
    else :
        img = cv2.putText(img, 'Safety Jacket is not classified : {!s:.6s}'.format(prediction_prob5), (10,470), font, fon

    cv2.imshow('result image 1',img)

    #if wait at least 1ms AND press the
    if cv2.waitKey(1) & 0xff == ord('q'):
        break
cv2.destroyAllWindows()
```