Face and Face Mask Detection Using Convolutional Neural Network

Muhammad Mustaqim Zainal[1], Radzi Ambar[1,2(✉)], Mohd Helmy Abd Wahab[1], Hazwaj Mhd Poad[1], Muhammad Mahadi Abd Jamil[1], and Chew Chang Choon[1]

[1] Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia
aradzi@uthm.edu.my

[2] Computational Signal, Image and Intelligence Research Focus Group, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia

Abstract. The COVID-19 outbreak has posed a severe healthcare concern in Malaysia. Wearing a mask is the most effective way to prevent infections. However, some Malaysians refuse to wear a face mask for a variety of reasons. This work proposes a real-time face and face mask detection method using image processing technique to promote wearing face mask. Haar Cascade is used for the face detection to extract the features of the human faces as a method of approach. On the other hand, the face mask detection utilizes convolutional neural network (CNN) to train a model using the MobileNetV2 training model designed using Python, Keras and Tensorflow. OpenCV package was used as the interface for the algorithms to be connected to a web camera. Based on the performance metric calculation of detection rate analysis of the experimental results, the face detection rate is at 90% true and 10% false detection, which shows very good detection rate. Furthermore, the training accuracy and validation accuracy for the face mask detector are efficiently near to 1.0, proving a steady accuracy over the time. Training loss and validation loss are almost near to zero and decreasing over time, reassuring the algorithm performance is accurate and efficient for a datasets of 4000 images.

Keywords: Face mask detection · Face detection · Image processing · Convolutional neural network

1 Introduction

In Malaysia, the government has declared the first standard operating procedures (SOP) to prevent COVID-19 major outbreak in the early of 2020, after the first case was reported in Malaysia of a traveler coming back from China [1]. The government has also declared a Movement Control Order (MCO) to handle and contain the spread of the virus. The move had successfully taken control of the situation in an orderly manner preventing a panic situation from happening among the people [2]. The SOP includes social distancing of 1 m between people and also mandatory face mask wearing in public places as a fine would be issued from any SOP misconduct.

J.-H. Kim et al. (Eds.): IHCI 2021, LNCS 13184, pp. 597–609, 2022.
https://doi.org/10.1007/978-3-030-98404-5_55

In the effort to fight against the virus outbreak, apart from issuing new laws and SOP, multiple efforts has been implemented to cope with the situation by utilizing the technology of the modern civilization. An alternative innovation to combat the spread of COVID-19 is by utilizing image processing technology that has always been part of human daily lives. Image processing technology has enabled human to achieve unthinkable task such as object detections [3], imagery editing [4] and even medicinal usage [5]. The use of image processing technology have also been applied to cope with the current pandemic situation in various stages such as detection, prevention and response stages. For instance, [6] proposed an image processing techniques to classify computed tomography (CT) chest images into normal or infected based on artificial intelligence algorithm. Recently, [7] utilized deep learning technique for computer-assisted screening tool of lung ultrasound imagery to diagnose pulmonary conditions caused by COVID-19 infection. These are examples of image processing application in detection stage. On the other hand, a study in [8] is an example of prevention stage application, where a real-time video stream-based image processing technique using YOLOv3 has been proposed to maintain social distancing in closed environment. Then, authors in [9] introduced a simulation of camera equipped drone-based network system that can capture individual images to measure social distancing and thermal images The work is an example of COVID-19 outbreak prevention stage.

It is well-informed that one of the most important preventive measure to control the outbreak of COVID-19 is by wearing face mask. Several countries have introduced rules and regulations to enforce wearing face mask in public places and closed facilities. However, it is impossible to perfectly impose these regulations because not all individuals willing to wear face mask in public places and the authority do not have enough man power to enforce the laws. Therefore, automatic system utilizing image processing techniques can be utilized to execute face mask detection in real-time. The main aim for this project is to detect human face wearing a face mask and without face mask using image processing technology and artificial intelligence (AI). The project hopefully can be a future preventive measure to warn the public to wear face mask before going to public places, thus, lowering the infection cases of COVID-19.

2 Methodology

Deep learning is an artificial intelligence (AI) function that mimics the workings of a human brain in processing data. It is also a class of deep neural networks using multiple layers of extracting features from an input [10]. In deep learning, image processing is a part of neural networks learning involving computer vision system. The capability of machine learning in computer vision can be used to assist human in object detections [3], face detections [11] and face recognition [11]. This can be implemented to counter the current pandemic by developing a system which can be used to detect and distinct face mask user and non-user.

2.1 Convolutional Neural Network (CNN)

In deep learning, CNN is utilized specifically in analysing visual imagery [4, 12]. Its algorithms are able to take an input image and assign importance such as the learnable

weight and biases, and be able to differentiate one from the other. Figure 1 shows the architecture of CNN. It is made up of multiple layers of process through which the information of an input image are filtered, and the importance extracted according to the algorithms. The input takes a 2 dimensional image and process it to be classified into categories. Each layer of the process in the architecture of CNN is described in the next subchapters.

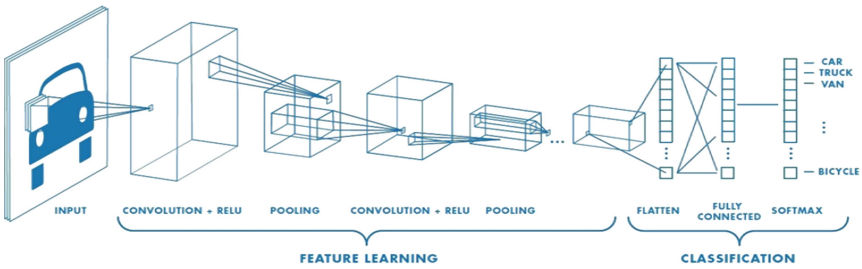


Fig. 1. Convolutional Neural Network [12].

Convolution Layer. The first layer is the convolution layer where features from an input image are extracted, whilst preserving the pixels relationships by using image data (input) from small squares. This is done to reduce the scale of the input image for simplifying the process. The process involves an arithmetic operation that uses two inputs for example an image matrix (volume) of dimension (height x width x depth) and a filter matrix (f_h *x* f_w *x depth*), where f_h is filter for height, f_w is filter for width, respectively.

As shown in Fig. 2, the image matrix will be multiplied with a filter through a process called Feature Mapping. The image matrix with pixels values of 0 and 1 are multiplied to fit the filter. Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters.

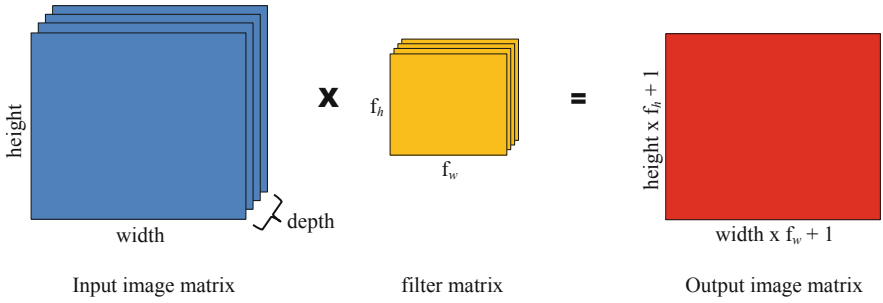


Fig. 2. Feature mapping by multiplying image matrix with a filter matrix.

Strides, Padding and Non-Linearity (ReLU). When convolution is executed, the filter (kernel) is shifted over the input matrix, which is called a stride. Depending on the stride value, the pixels is moved accordingly. In each stride, matrix multiplication operations are performed between the filter matrix and the image matrix part that the filter is hovering, until the entire image is hovered to produce convoluted output as shown in Fig. 2. Padding is done when a filter does not fit an input image perfectly by zero padding or dropping the part of unfit image. Rectified Linear Unit (ReLU) is important as it introduces non-linearity in the convolution network to ensure the network learn non-negative linear values to be able to perceive the real world data.

Pooling Layer. Pooling layer reduces the number of parameter for a large image and reduce the dimensionality of each map whilst retaining the important information. Max Pooling, Sum Pooling and Average Pooling are the three types of spatial pooling. Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

Fully Connected Layer. Figure 3 shows a visualization of a fully connected layer formed by flattening the matrixes into vectors and fed to the fully connected layer like a neural networks [12]. The feature map matrix will be converted to vectors (x1, x2, x3, x4) within the fully connected layer to create a model with an activation function to classify input image into categories (y1, y2, y3).

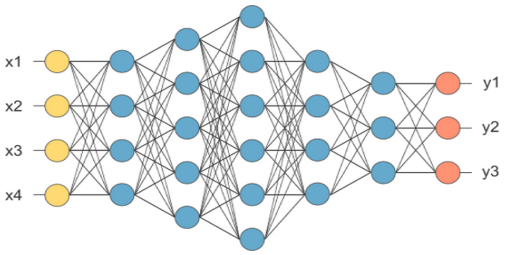


Fig. 3. Fully connected layer visualization [12].

2.2 Face and Face Mask Detection

This subchapter describes the method to detect human face and face mask covering the nose and mouth area of the face. Figure 4 shows the overview of the project that consists of two stages of detection which are (i) face detection, (ii) face mask detection. The functionality of both algorithms are different from each other, and data extraction are processed across both features.

The first stage is face detection. The algorithm starts with raw input of images and the cascade classifier extract the feature of a human face from the input and process it to be passed on to the recognizer where the human faces seen through the computer

vision would be recognized and detected. The second stage is the face mask detection where the object detection algorithms are being used as a reference. The input images are preprocessed to be passed on to the CNN layers using the model trainer architecture of convolution, and the created model will be a recognizer for face mask detection through the computer vision system. The face mask detector will output two types of coloured frames on the video stream in real-time, green frames for faces with face masks, and red frame for non-face mask faces. In order to design the system using Python, a number of software packages are required which are Anaconda software, Keras application programming interface (API) with its libraries and Tensorflow. The training model used is the MobileNetV2, a small sized package that can carry the operations of CNN with a high accuracy and minimal losses. The OpenCV package was used as the interface for the algorithms to be connected to the webcam.

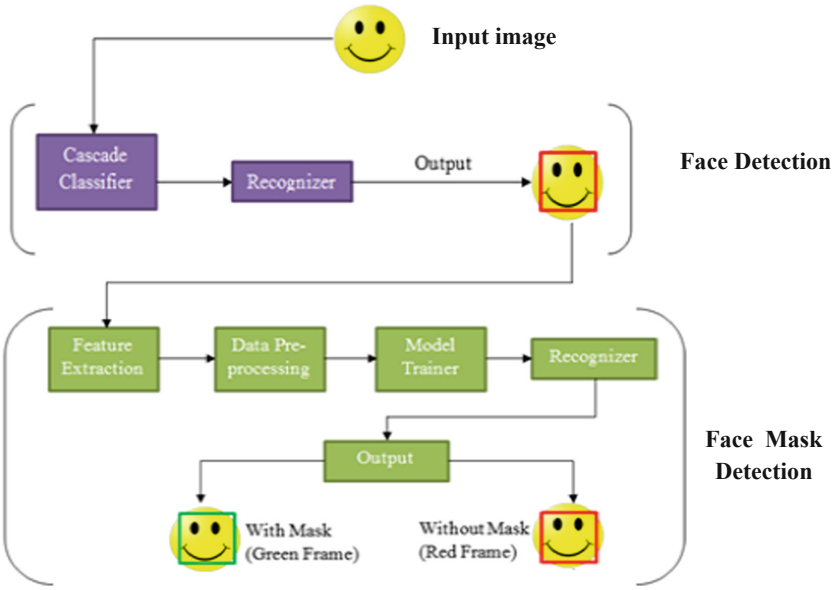


Fig. 4. Overview of the project.

Figure 5 shows the four main dependencies used to develop the algorithms which are the Keras library together with its backend Tensorflow library for deep and machine learning. This allows for construction of the algorithm possible with the wide range of programmable commands of artificial neural networks. Besides that, the training model is MobileNetV2 used for convolution layer during model training process. Lastly, OpenCV is used for its library which enables computer vision interaction and feature extraction. The dataset of people wearing face mask are obtained from Kaggle website.

The next stage is the process stage. Pre-processing is the process where datasets are augmented before being fed to the CNN layers. The image parameters are altered in the preprocessed section to enable feature extraction to be more accurate. During the CNN process, the model training began for a time period pre-set in the algorithm. The

parameters for CNN layer affect the training performance of the model created. The real time video interface is the OpenCV dependencies feature to use the trained model as a recognizer and determine the detection parameter such as frame, prediction of detection and starting the interface for the program.

The interconnected arrows which moves in both ways shows the relationship of the features are interconnected and influence the performance of each feature. The algorithm uses the computer vision system, the built in web camera to begin detection and displays an output on the monitor screen. The expected output are the face detection and face mask detection.

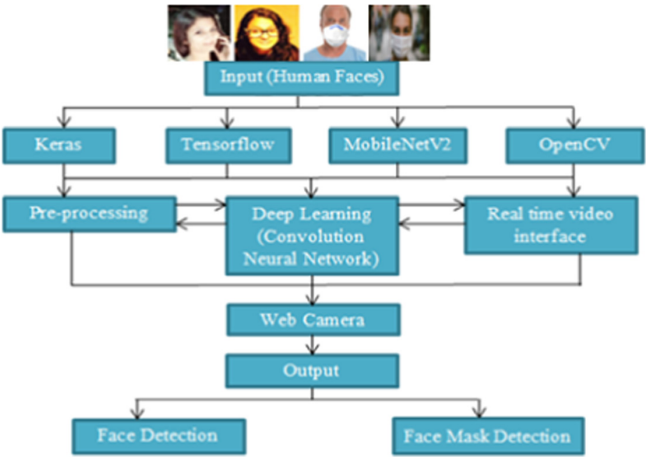


Fig. 5. Dependencies relations with the algorithm structure.

2.3 Face Detection

In developing the face detector, the focus is to detect the characteristics of the human face before proceeding to face mask detection. This work utilizes Haar cascade for the face detector, which is a machine learning based approach where a cascade is trained with the input image data to identify the important features of a human face to be detected. OpenCV is the dependencies required for face detection, as it contains many pre-trained classifier such as the detector for eyes, nose and mouth.

Figure 6 shows the subject in the input image is seen through the web camera, then, the pre-process occurs whilst extracting feature from the subject. The classifier then examines the input data from the feature extractor to be compared with a model database for producing the output. The model database is acquired from the internet to assist in the development of the project. After a successful face detection, a frame (bounding box) is displayed around the subject face on the monitor screen and detects it in real time.

Feature Extraction and Classifier. The feature extraction utilizes the OpenCV built in libraries called Haar cascade. Haar cascade works as a cascade function that is trained

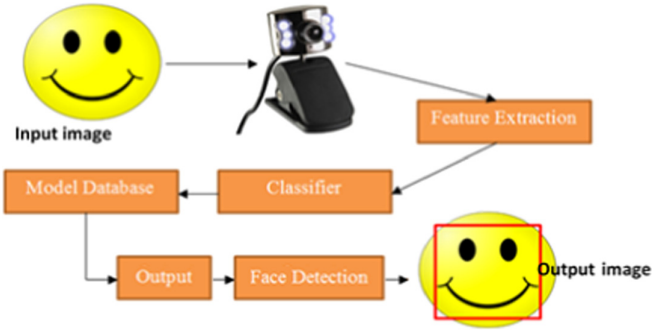


Fig. 6. Flow for face detection algorithm.

to have positive and negative images which is then used for object detection in other images. The cascade takes the input image and cascade the information in black/white squares as feature extraction. The classifier chosen for face detection is the face classifier as the algorithm is written to detect the whole human face area. Figure 7 shows a screen capture of the Python code indicating the Haar cascade function is utilized specifically to detect faces as the line face_cascade indicates. Furthermore, it is used to extract feature related to the human faces.

```
3  # Load the cascade
4  face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Fig. 7. Haar cascade function is loaded into the coding.

```
5  # Read the input image
6  img = cv2.imread('test.jpg')
7  # Convert into grayscale
8  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
9  # Detect faces
10 faces = face_cascade.detectMultiScale(gray, 1.1, 4)
11 # Draw rectangle around the faces
12 for (x, y, w, h) in faces:
13     cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
14 # Display the output
15 cv2.imshow('img', img)
16 cv2.waitKey()
```

Fig. 8. Parameter setting for face detection.

Model Database and Output. The model database contains all the pre-trained classifier of the OpenCV libraries. The input image information is extracted according to the detection parameters. The output parameter is also determine in the algorithm as the wanted values needed to be set before program execution. Figure 8 shows a screen

capture of Python code indicating that the input image is converted to grayscale as the classifier identifies extraction in grayscale, and a function called cv2.imread is to declare the image file to be tested. The line detectMultiScale is to detect faces in the image and converting the input to grayscale. The frame position and colour are determined in the algorithm using width, height, frame edge and axis to show a successful detection.

2.4 Face Mask Detection

In this subchapter, the face mask detection method is described. Developing a face mask detector using CNN is quite similar to face detection as it is also an image based approach method. A collection of datasets are fed to a model trainer with the exception of having two output. The first output is a face wearing a face mask and the other is a face without a face mask. The algorithm is written to predict the probability of presence of a face mask or without face mask which then will be shown in the monitor.

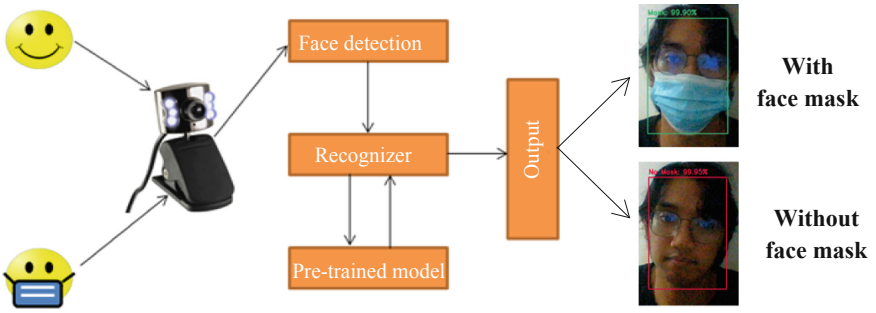


Fig. 9. Face mask detection: without face mask and with face mask.

Figure 9 shows the overview of the proposed face mask detection algorithm. First, a still image or video streams from webcam is converted into single frame that is fed to the algorithm for face detection. From the face detection, the recognizer compares the feature extracted through the face detection with the pre-trained model of face mask detection and yield the output in a prediction values of a face mask detection percentage.

3 Experimental Results and Discussion

In this chapter, the performance of the proposed face and face mask detection algorithms are demonstrated. The obtained experimental results were tested using video and images containing a maximum of six (6) individuals in a household, as performing experiments in larger scale are restricted with the ongoing pandemic situation. The main aim is to ensure the functionality of the algorithms. Experiments were done using both real-time video stream and still images.

3.1 Face Detection Experiment

Figure 10 shows that the algorithm has successfully detected the face area and initiated blue frames around the desired faces area on a still image. The accuracy and consistency of face detection depends on the number of subject and the specification of machine used to run the program. It was found that a slight lag occurred during experiments, maybe due the specification of the laptop used in the experiment was not enough to support a smooth detection.

Figure 11 shows example images on the comparison to determine the efficiency of face detection in bright and dark surroundings. The results shown in Fig. 11(a) shows that detection in dark surrounding were difficult as the algorithm was unable to distinct between the faces and the background. For bright surrounding, it was found that an average amount of brightness is optimal, as over brightness can affect the detection of the subject faces as shown in the Fig. 11(b). Some part of the chin is laminated with bright lighting in the background, thus interrupting with the face classifier to distinguish the subject's face area.

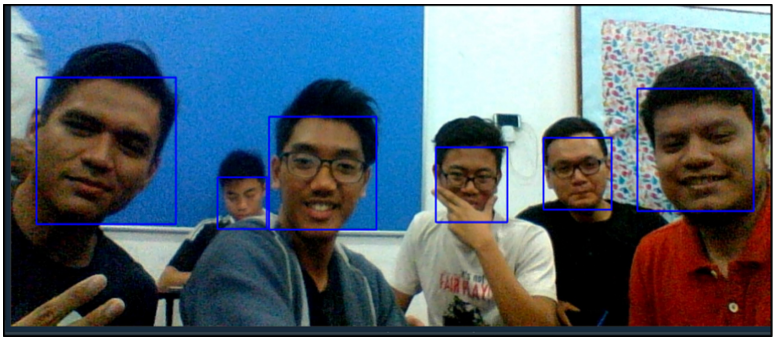
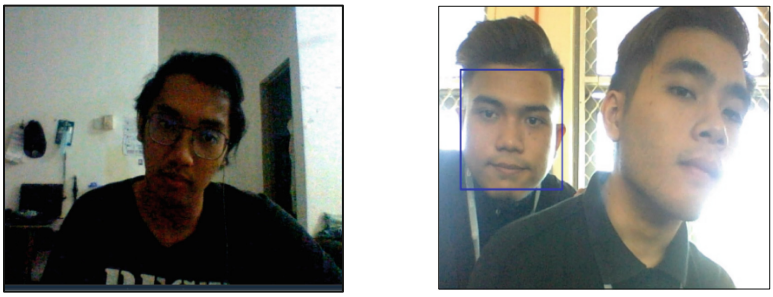


Fig. 10. Face detection on still image.



(a) Dark surrounding (b) Bright surrounding

Fig. 11. Comparison of dark and bright surrounding images.

3.2 Face Mask Detection Experiment

The face mask detection works by initially detecting the face area. By using the function of face detection added with another command of detecting the presence of a face mask, the result produces both output of face detection and face mask detection. The output of face mask detection is in the form of probability percentage displayed on the monitor.

Figure 12 shows the result of a face with and without a face mask. The main distinct different is the frame colour around the face detection. For a non-mask wearer, the frame will be coloured in RED, and for mask wearer it is GREEN. Red symbolizes a warning for the subject without face mask and green for approval as the presence of face mask is detected. Another displayed parameter is the probability percentage of the face mask detection. A high number of percentage indicates the accuracy of detection. A high number of percentage for a face without face mask indicates that there are higher percentage that there are no face mask detected and vice versa.

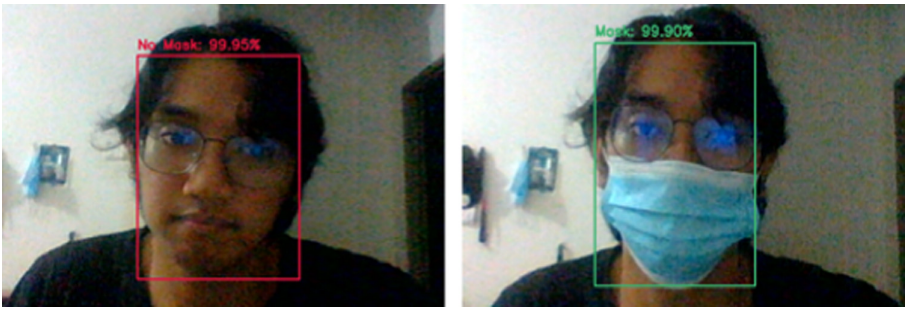


Fig. 12. Face mask detection: without face mask (left), with face mask (right).

3.3 Analysis and Discussion

The performance of an algorithm are measured using the performance matrix which includes accuracy, losses and confusion matrix calculation. Both face and face mask detection algorithms were tested with different metrics to determine the algorithm performances.

Face detection experiment were based on the face detection rate. A set of 10 images of human faces were randomly shown to the web camera and the number of times the algorithm succeeded in detecting the faces were observed and tabulated. The percentage of face detection rate can be calculated using the following formula:

$$\frac{\sum \textit{True Detection}}{\textit{Total Number of Images}} \times 100\% \tag{1}$$

Table 1 shows the number of True Detection and False Detection over three different testing with three different subject images. The face detection rate is at 90% true and 10% false detection based on the performance metric calculation of detection rate. The

rate is considered acceptable as it is close to 100% accuracy. Other outer factors such as surrounding lightings may affect the accuracy of detection, hence the non-perfect score of detections. Besides that, the facial structure of the subject's face may be different from the dataset collection as the angle of the camera is static.

Table 1. Face detection rate.

Test no	True detection	False detection
1	8	2
2	9	1
3	10	0

The face mask detection algorithm performance were calculated using the plotted graph metric of training accuracy against losses over the time period as shown in Fig. 13. The graph were plotted directly from the algorithm model training output and saved in PNG format image. The Training_Accuracy and Validation_Accuracy is efficiently near to 1.0 proving a steady accuracy over the time whilst the Training_Loss and Validation_Loss is almost near to zero and decreasing over time reassuring the algorithm performance is accurate and efficient for a datasets of 4000 images.

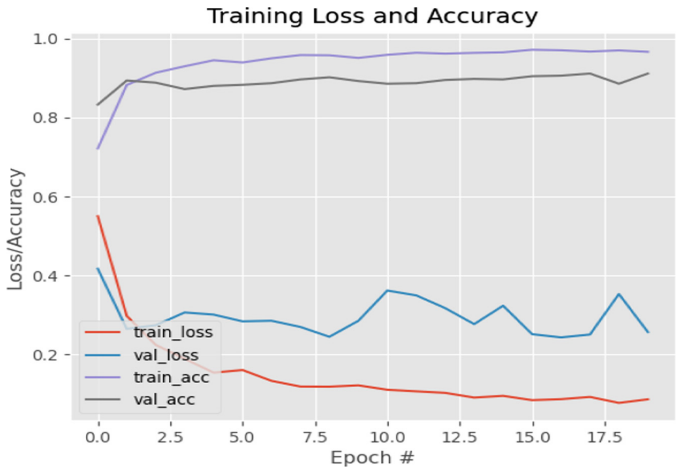


Fig. 13. Training loss and accuracy against time (Epochs).

The face mask detection algorithm performance were calculated using the plotted graph metric of training accuracy against losses over the time period as shown in Fig. 13. The graph were plotted directly from the algorithm model training output and saved in PNG format image. The Training_Accuracy and Validation_Accuracy is efficiently near

to 1.0 proving a steady accuracy over the time whilst the Training_Loss and Validation_Loss is almost near to zero and decreasing over time reassuring the algorithm performance is accurate and efficient for a datasets of 4000 images.

4 Conclusions

Initially, a face shape is detected using a Haar cascade classifier which extracts the feature of a human face from the raw input images. Then, the algorithm process it to be passed on to the recognizer where the human faces seen through the computer vision would be recognized and detected. Based on the performance metric calculation of detection rate analysis of the experimental results, the face detection rate is at 90% True and 10% False detection, which shows very good detection rate. Once a face is detected, the image is processed to determine the face is with or without a face mask. The input images are preprocessed to be passed on to the CNN layers using the model trainer architecture of convolution and the created model will be a recognizer for face mask detection through the computer vision system. The training accuracy and validation accuracy are efficiently near to 1.0, proving a steady accuracy over the time. Furthermore, the training loss and validation loss are almost near to zero and decreasing over time, reassuring the algorithm performance is accurate and efficient for a datasets of 4000 images. The preliminary experimental results obtained proves the algorithm can be improvised to increase the accuracy of recognition. Through the efficiency of the proposed face mask detection algorithm which includes also face recognition algorithm, both the real time video and still images can be detected. For future works, artificially illuminated colour images based on surface gradient information of the images will be used to enhance the capability of the system to detect face mask in various backlight conditions. Furthermore, individual face recognition system will also be implemented with the developed system that can recognize the exact individual without a face mask.

Acknowledgements. This research was supported by Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme for Research Acculturation of Early Career (FRGS-RACER) (RACER/1/2019/TK04/UTHM//5).

References

1. The Star Online. https://www.thestar.com.my/news/nation/2020/03/16/malaysia-announces-restricted-movement-measure-after-spike-in-covid-19-cases. Accessed 5 Oct 2021
2. Bernama. https://www.bernama.com/en/general/news_covid-19.php?id=1890211. Accessed 5 Oct 2021
3. Dhillon, A., Verma, G.K.: Convolutional neural network: a review of models, methodologies and applications to object detection. Progr. Artif. Intell. **9**(2), 85–112 (2019). https://doi.org/10.1007/s13748-019-00203-0
4. Pak, M., Kim, S.: A review of deep learning in image recognition. In: International Conference on Computer Applications and Information Processing Technology (CAIPT), pp. 1–3 (2017)
5. Cai, L., Gao, J., Zhao, D.: A review of the application of deep learning in medical image classification and segmentation. Ann. Transl. Med. **8**(11), 713 (2020)

6. Kaheel, H., Hussein, A., Chehab, A.: AI-based image processing for COVID-19 detection in chest CT scan images. Front. Comms. Net **2**, 645040 (2021)
7. Diaz-Escobar, J., et al.: Deep-learning based detection of COVID-19 using lung ultrasound imagery. PLoS ONE **16**(8), e0255886 (2021)
8. Melenli, S., Topkaya, A.: Real-time maintaining of social distance in Covid-19 environment using image processing and big data. In: 2020 Innovations in Intelligent Systems and Applications Conference (ASYU), pp. 1–5 (2020)
9. Kumar, A., Sharma, K., Singh, H., Naugriya, S.G., Gill, S.S., Buyya, R.: A drone-based networked system and methods for combating coronavirus disease (COVID-19) pandemic. Future Gener. Comput. Syst.: FGCS **115**, 1–19 (2021)
10. Schmidhuber, J.: Deep Learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
11. Wu, W., Yin, Y., Wang, X.: Xu, D: face detection with different scales based on fasterR-CNN. IEEE Trans. Cybern. **49**(11), 4017–4028 (2019)
12. Medium.com Homepage. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. Accessed 2 Oct 2021