Faculty of Informatics and Computer Science

Artificial Intelligence

Face Mask Detection System

By: Mohamed Fawzy

Supervised By

Professor. Andreas Pester

&

Professor. Mostafa Salama

June 2023

Abstract

Covid-19 changed the lives of many people for the worse so an action must be taken such as people should wear masks to reduce the spread of the pandemic. A face mask detection should be made to ensure people are wearing masks everywhere or else this pandemic will not leave us alone. Face mask detection entails creating computer vision algorithms that can reliably determine whether people are wearing face masks. The main goal is to enforce public health regulations by identifying those who do not wear masks according to protocol, allowing for prompt intervention and fostering a safer atmosphere. For face mask detection, a variety of methods have been used, including both conventional computer vision strategies and deep learning techniques. To distinguish between masked and unmasked faces using traditional methods, hand-crafted features are extracted from facial photos and classifiers like Support Vector Machines (SVM) or Random Forests (RF) are used. These methods do, however, frequently suffer with changes in position, poor illumination, and occlusions. Convolutional neural networks (CNNs) have demonstrated outstanding success in recent years in a variety of computer vision applications, including face mask identification. Deep learning models may automatically learn discriminative features from large-scale datasets, improving generalization and resilience. Even under difficult situations, these algorithms are able to classify faces as masked or unmasked accurately. The model managed to score an accuracy of 98% and a validation accuracy of also 98% which is considered a great performance then a face mask detection mobile application was created using this model. The benefit of this application is that it makes the usage of face mask detectors cheaper and more flexible.

Turnitin Report

Add the report

Acknowledge

I would like to thank Professor Andreas Pester and Professor Mostafa Salama for their continuous feedback and guidance.

I would like to thank my lovely family for their endless support and love.

I would like to thank my friends for making my life easier and more fun in many ways.

Table of Contents

Table of Contents

List of Figures

List of Tables

1.Introduction

1 Introduction

1.1 Overview

Why was this topic chosen? It is because of the recent outbreak of the covid-19 virus which changed the lives of many people drastically and took the lives of many and injured quite a few. Therefore, it was decided to make a face mask detection system so that people would always be wearing masks everywhere but why should people be wearing masks? Because it reduces viral transmission (if worn correctly), it prevents others from getting sick (before knowing that you had covid-19), it protects you from catching it and in general it is good hygiene.

1.2 Motivation

The motivation behind this is that the world can not afford for covid-19 or any other major pandemic to spread again because the outbreak of them would cause havoc around the world while countries and even people are still trying to recover from the damages and annihilation that the pandemic has caused so to make sure that the pandemic does not spread is by creating a model that detects face masks so that a person would not enter a building which is crowded with people without wearing a face mask.

1.3 Problem Statement

The problem that not everybody realizes is the sighting of many people not wearing masks in a certain area and that happening will lead to the spreading of the covid-19 pandemic. The pandemic caused an economy crisis as many organizations' businesses were severely damaged, many countries' economy was on a constant decline, lots of curfews were imposed to limit the gathering of people and because of the pandemic the lives of many were lost. The solution to this problem is wearing a mask all the time as wearing a mask lowers the chances of the virus being transmitted to another person.

1.4 Scope and Objectives

The scope of the system includes hospitals, schools, universities and basically any building that is going to be filled with lots of people so before people would enter, security should have the system running to detect face masks. The objectives of the project is the following:

- Preventing Covid-19 from spreading any further.
- Detecting if people have face masks on or not.

- Converting the process of manual face mask checking to automatic face mask checking.
- Scaling down the transmission of viruses and diseases.
- Lowering the usage of humans which leads to the decreasing of the costs.

1.5 Report Organization (Structure)

What is coming in the following sections is the methodology of the work to give us an idea of how this project was done after that a background which tells us the history of face mask detection what old architectures were used back then and what state of the art AI techniques are being used now then a literature review which talks about lots of papers which includes the journey of many people in using lots of different architectures old and new and the results which they have come up with in the end. In the end what is left is the implementation of the model, its evaluation, results, and the work which could be done in the future.

1.6 Work Plan (Gantt chart)

The following figure contains a chart called Gantt chart which is a chart which was created to introduce to us how the work plan is distributed across the year.

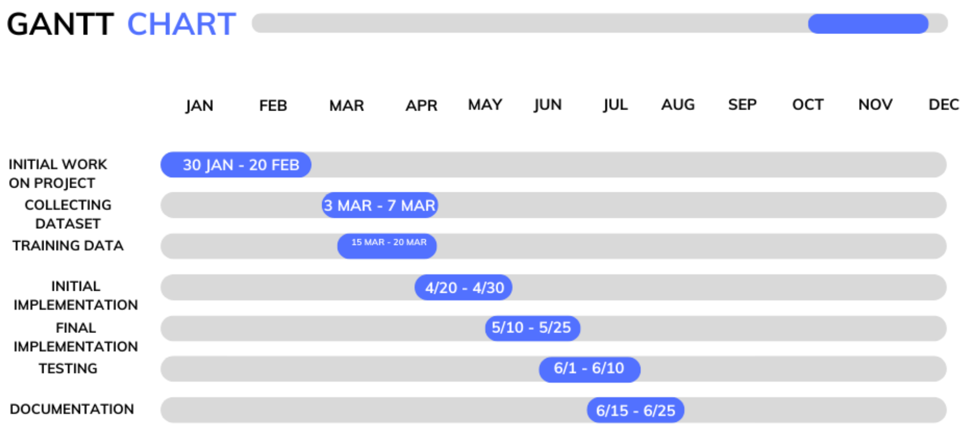


Figure 1: Gantt Chart which showcases the work plan.

2.Related Work (State-of-The-Art)

2 Related Work (State-of-The-Art)

2.1 Background

According to the paper in [1] a facemask detection algorithm is a kind of algorithm for detecting objects which locates facemasks using bounding boxes in an image or video stream. Object detection combines both image classification and localization. Currently, while other facemask detection techniques concentrate on image localisation, some just focus on image classification. Additionally, current object detection algorithms have adopted and trained facemask detection techniques, object detection algorithms such as you look only once, single-shot detector and convolutional neural networks. The majority of facemask identification methods proposed after COVID-19 are based on deep learning (DL) and they are a subset of machine learning (ML) algorithms. The deep learning-based algorithms consist of deep neural network (DNN), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM). The feature extraction capabilities of the deep learning-based algorithms are better than those of conventional machine learning based techniques. The pooling layer reduces the dimensionality while the convolutional aspect of CNN retrieves valuable features utilising filters from an image. In the past a facemask detection model that utilises the Viola-Jones detector was presented to detect masked faces in a medical operating room. By dividing faces into two halves, this technique used colour filter to distinguish between a face with a mask and one without one. Facemask detection using a colour filter was successful, however because the features were hand-coded, the feature extraction was less successful than using DNNs. ResNet and ResNet-50 are both seen being employed in facemask detection algorithms following the Covid-19 outbreak. both networks are of type residual neural network which is a very deep CNN, and it is an effective method in detection of facemasks.

2.2 Literature Survey

Most papers have one thing in common and that is using tools such as TensorFlow, CNN, keras, OpenCV and MobileNetV2 classifier as they are optimal and essential for maximizing the process of face mask detection. Some other papers use some additional tools which is going to be discussed.

2.2.1 TensorFlow, CNN, Keras, OpenCV and MobileNetV2:

Papers from [2], [3], [4], [5], [6], [7], [8] and [9] all use the previous mentioned tools, so we are going to discuss the paper in [2] only. In [2] the authors wanted to compare between different Ann in face mask detection. The different used neural networks are Retinal Connected Analysis Network (RCNN) which is a deep learning framework for object detection that generates region proposals using selective search, applies a pre-trained CNN to extract features from these regions, performs object classification using SVMs, and refines bounding box locations through regression., Principal Component Analysis with ANN which is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional representation and It identifies the directions, called principal components, that capture the most significant variability in the data, Convolutional Neural Network (CNN) which are deep learning models primarily used for analysing visual data like images and They are designed to automatically learn and extract hierarchical patterns and features from the input data. And lastly Fast Neural Network.

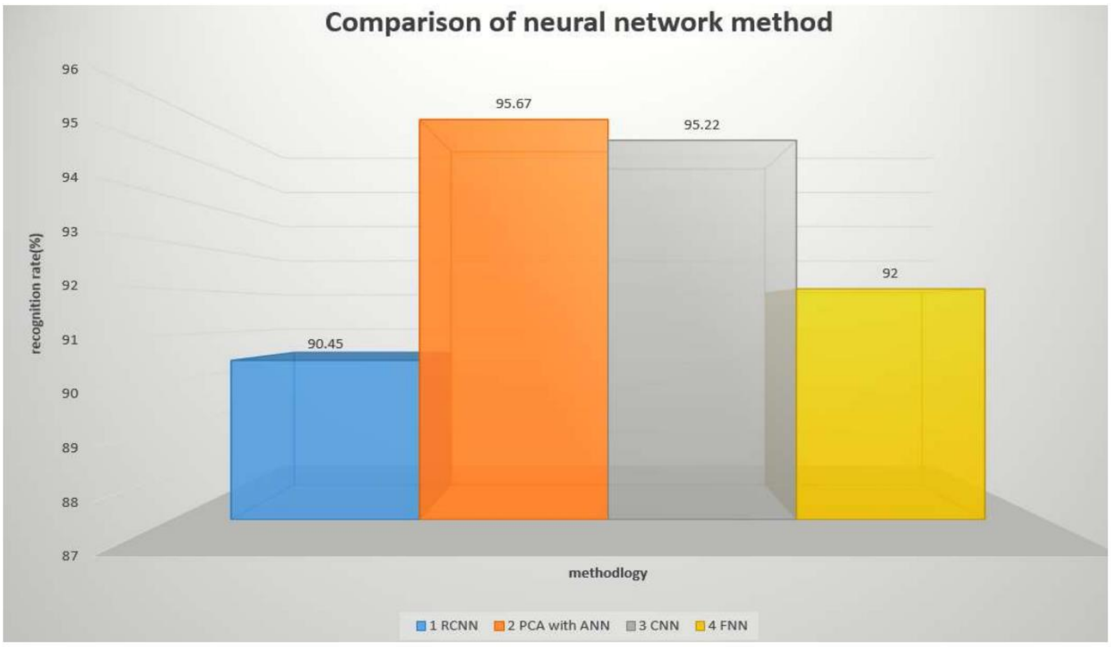


Figure 2:Comparison of Different Neural Network

After a comparison, it was found that the two methods with the best recognition rates were CNN and PCA with ANN. The authors wanted to make a system to detect if the person is wearing a mask or not on the image/video stream with the help of computer vision, deep learning algorithm and the usage of OpenCV, TensorFlow and keras library. Firstly, the deep learning model (MobilNetV2) was trained then face mask detector was applied by using images or live streams.

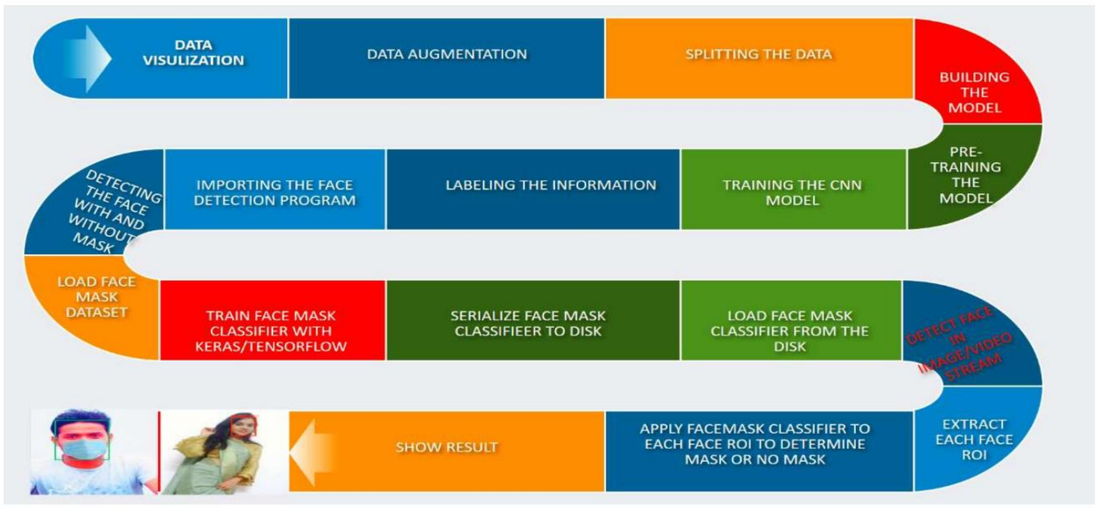

DATA VISULIZATION
DATA AUGMENTATION
SPLITTING THE DATA
BUILDING THE MODEL
PRE-TRAINING THE MODEL
TRAINING THE CNN MODEL
LABELING THE INFORMATION
IMPORTING THE FACE DETECTION PROGRAM
DETECTING THE FACE WITH AND WITHOUT MASK
LOAD FACE MASK DATASET
TRAIN FACE MASK CLASSIFIER WITH KERAS/TENSORFLOW
SERIALIZE FACE MASK CLASSIFIEER TO DISK
LOAD FACE MASK CLASSIFIER FROM THE DISK
DETECT FACE IN IMAGE/VIDEO STREAM
EXTRACT EACH FACE ROI
APPLY FACEMASK CLASSIFIER TO EACH FACE ROI TO DETERMINE MASK OR NO MASK
SHOW RESULT

Figure 3: Flow Diagram for Face Mask Detection

In the previous figure these were the steps which the authors followed in making the system.

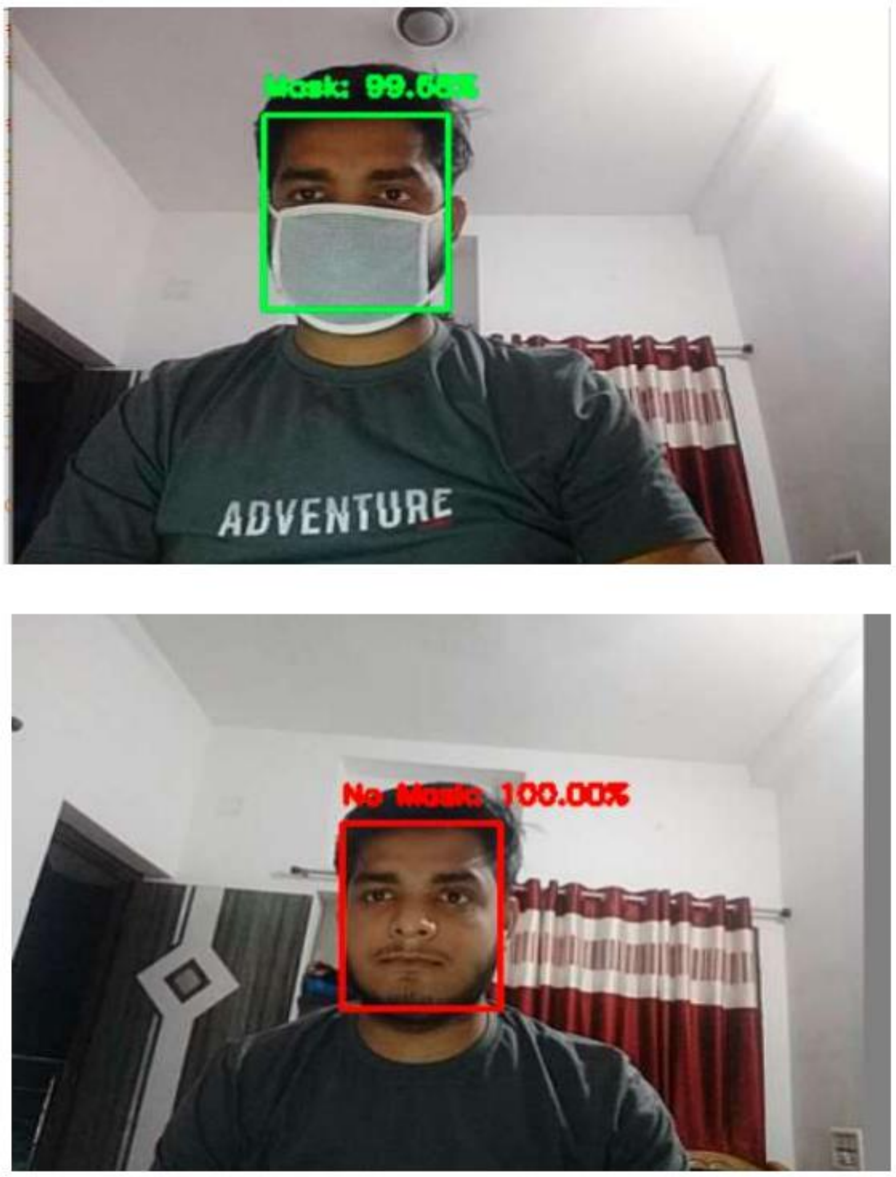


Figure 4: Detecting face with mask or without mask.

The authors concluded that they wanted to also implement in the future social distancing detection which would make the system complete.

2.2.2 TensorFlow, CNN, Keras and OpenCV with addition to (Cascade classifier and the conversion of RGB image to gray image):

The previously mentioned tools are used in papers [10] and [11] so we are going to talk about paper [10] only. The authors in [10] have a goal which is creating a face mask detection system using packages of machine learning like TensorFlow, Keras, OpenCV and Scikit-Learn. Two datasets will be used, the first one contains images with one face in the frame wearing masks and others not wearing masks and all of them wearing the same type of mask and white colour only. The second dataset has multiple faces in the frame with different types of masks and different colours as well. The proposed method which the authors are going to use includes a cascade classifier and a pre trained CNN That has layers of dense neurons attached to two 2D convolution layers. Cascade classifier is an Object Detection Algorithm that is utilized to find faces in pictures or real time videos. The technique makes use of edge or line detecting capabilities. modern descriptor-based image recognition systems frequently use grayscale images, without going into detail about how the colour to grayscale conversion was done. because when using powerful descriptors, the colour-to-grayscale technique is of slight consequence. grayscale is used for the extraction of descriptors rather than working on colour images instantly.

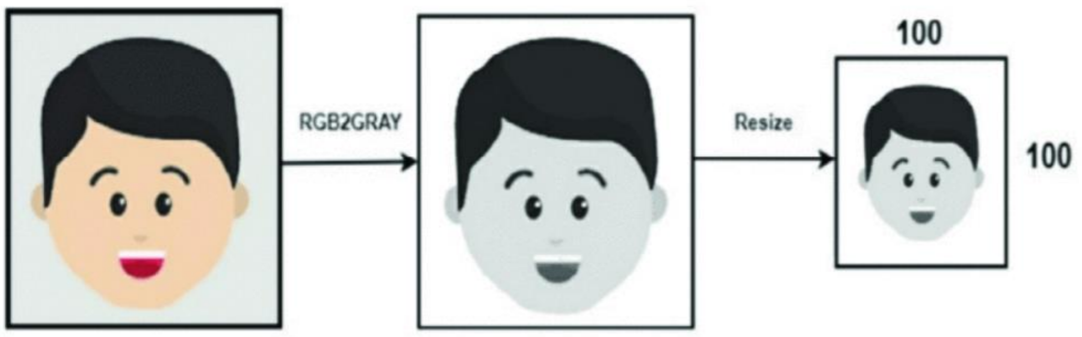


Figure 5: Conversion of a RGB image to a Gray Scale image of 100x100 size

Deep CNNs need an input image that is a constant size. Therefore, we require a constant standard size for each image in the dataset. The grayscale image is enlarged to 100 × 100 using cv2.resize().

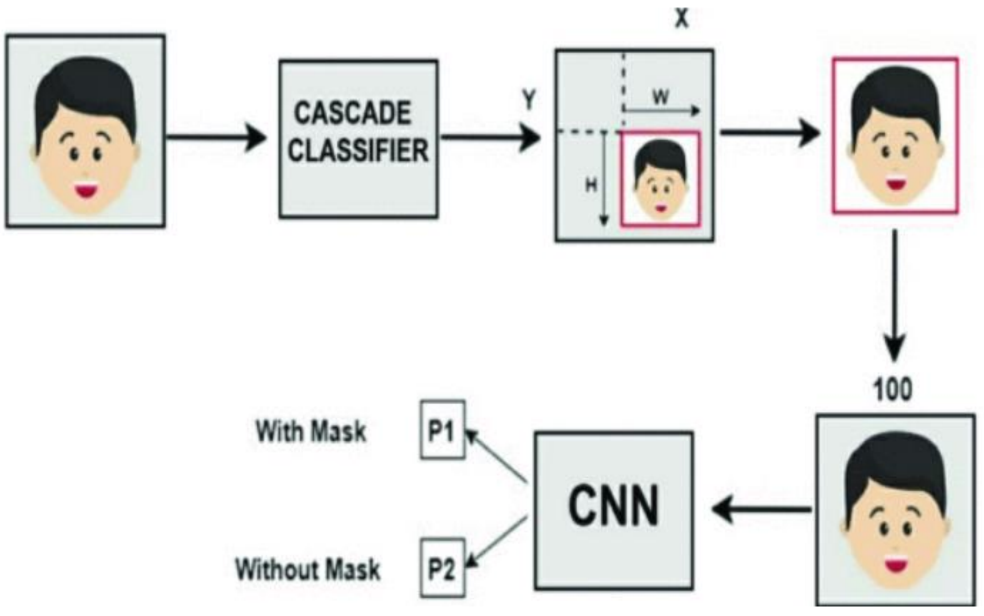


Figure 6: Overview of the Model

The model has been trained, validated, and tested with the help of two datasets.

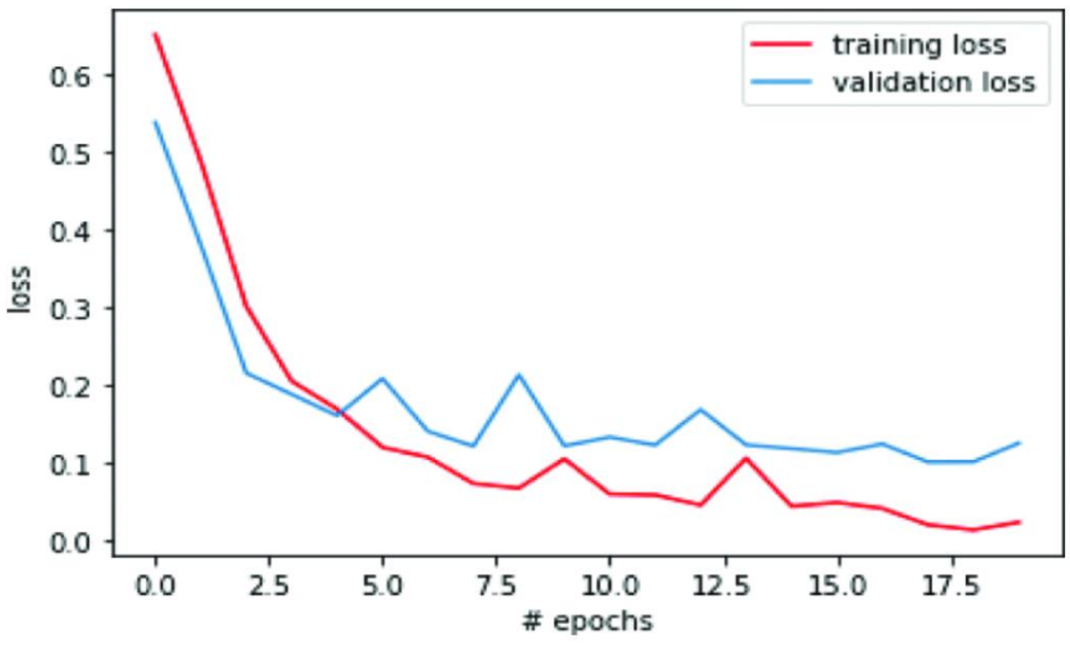


Figure 7: # epochs vs loss corresponding to dataset 1

Figure 6 demonstrates how the cost of error is reduced by this optimized accuracy.

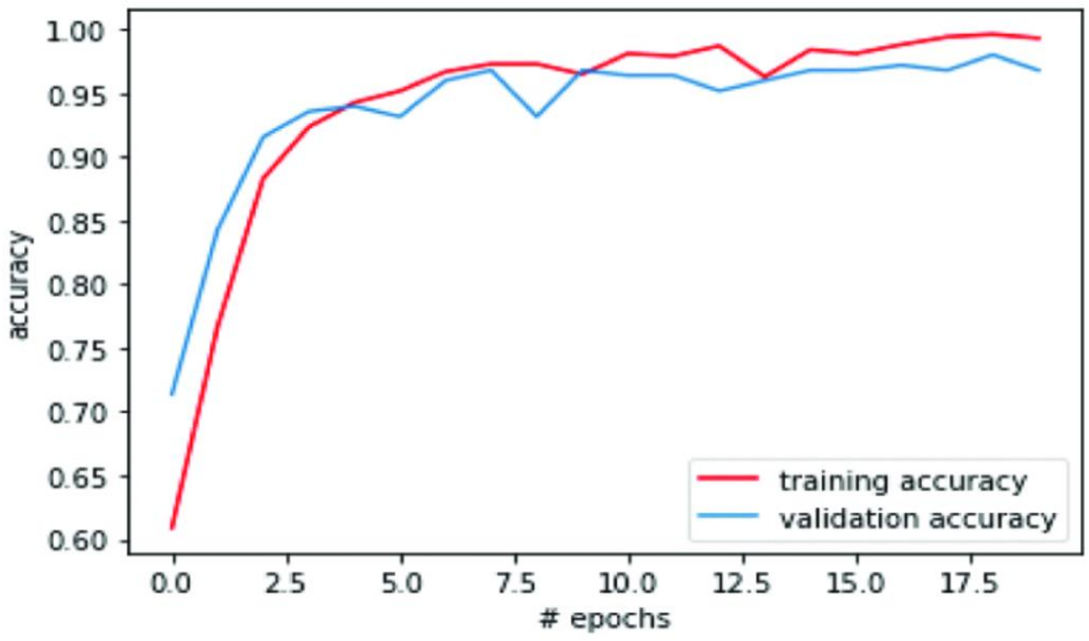


Figure 8: # epochs vs accuracy corresponding to dataset 1

Figure 7 shows us that the technique has a 95.77% accuracy rate.

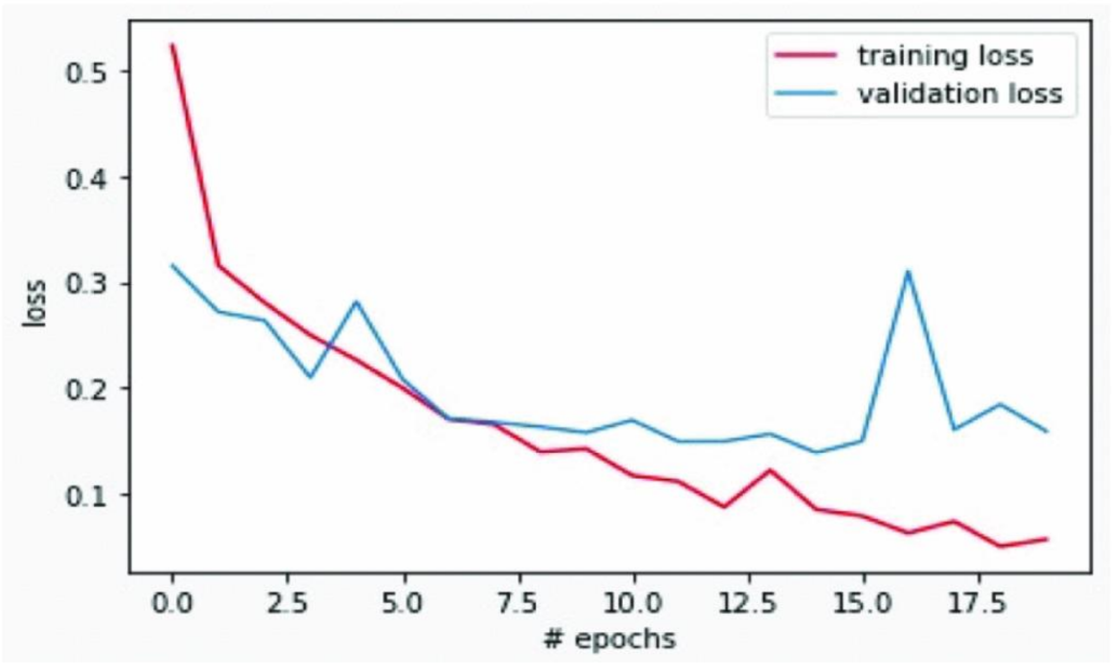


Figure 9: # epochs vs loss corresponding to dataset 2

Figure 8 shows the difference between the loss during training and validation for dataset 2.

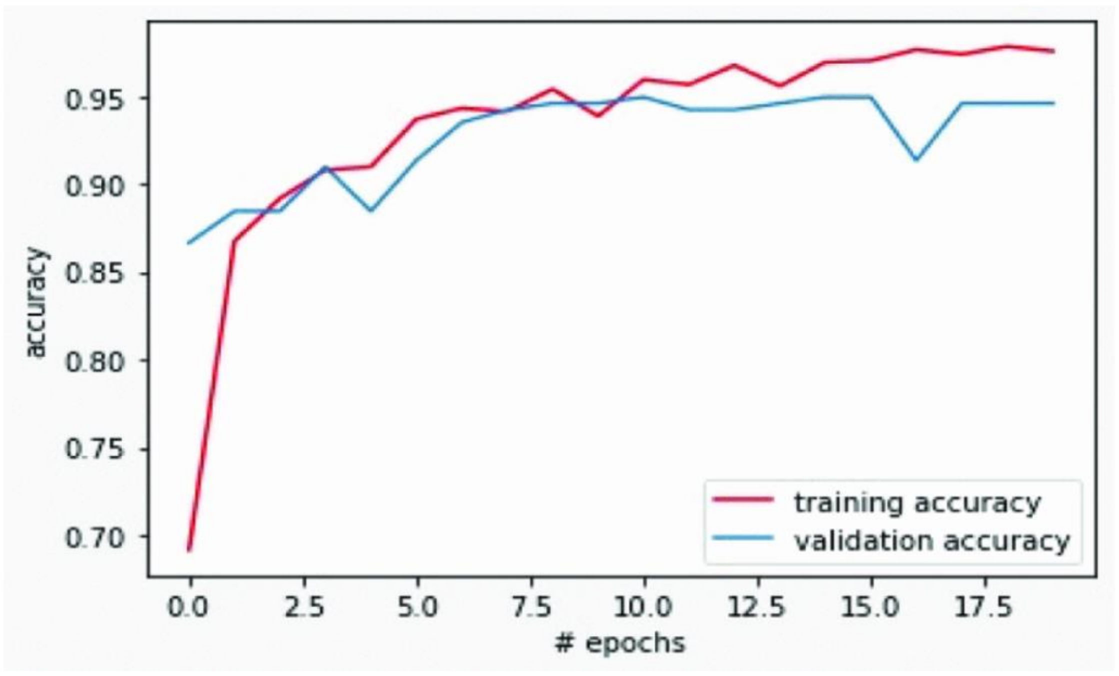


Figure 10: # epochs vs accuracy corresponding to dataset 2

Figure 9 shows that the model returns an accuracy of 94.58%.

From the previous observations and graphs it is concluded that Dataset 2 is more adaptable than Dataset 1 because it includes many faces in the frame and various types of masks with various colors.

2.2.3 Viola Jones Method:

The previous mentioned method is used in papers [12] and [13]. Only paper [12] is going to be discussed. The authors in [12] made an identification system using the Viola Jones method. The Viola Jones approach is a popular framework for object detection, particularly for face detection. It is renowned for both real-time performance and efficiency. An image's local intensity changes are captured by the technique using Haar-like features, which are straightforward rectangular features. A cascade classifier makes use of these characteristics. This research forms the Cascade Classifier and modifies the threshold value using the Viola Jones method to identify the facial region in the image. The Viola Jones approach provides a high detection rate for finding face patterns in pictures with minimal error rates. Viola Jones approach manipulates the image that was inputted by changing it into a new image in the integral image form all of that was done in the face pattern detection process. The integral image that was just created has quite a few numbers of features which will be selected by AdaBoost to be utilized as a classifier component which will be used to classify pictures. the optimal solution can be retrieved by deciding whether specific harr features are present or absent in a picture by choosing certain haar features that will be utilized to adjust the threshold. The cascade classifier identifies the facial region in the image of this method using a mask as the final classification.

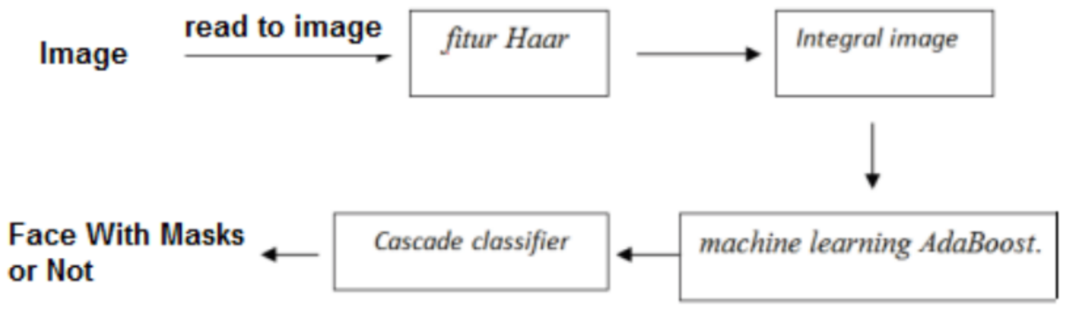
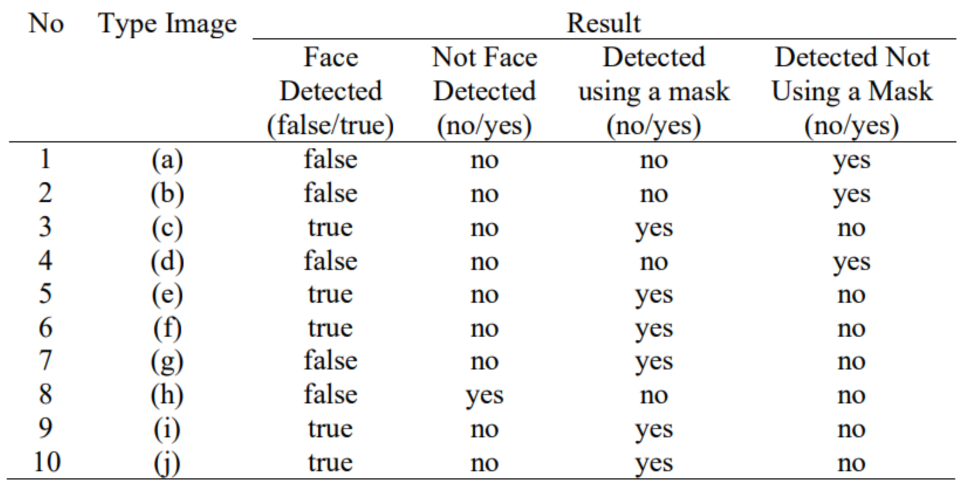


Figure 11: Flow chart of the masked face pattern detection system using the Viola Jones method.

The previous figure tells us the steps that were followed to create this face detection system using Viola Jones method. If only one function is utilized, the detection result from Haar is less accurate, hence it is typically used by multiple functions. The results are more accurate the more functions are employed. The purpose of AdaBoost is that it gathers many weak classifiers to form a powerful one. The method's performance was tested using 100 sample images.



No	Type Image	Result			
		Face Detected (false/true)	Not Face Detected (no/yes)	Detected using a mask (no/yes)	Detected Not Using a Mask (no/yes)
1	(a)	false	no	no	yes
2	(b)	false	no	no	yes
3	(c)	true	no	yes	no
4	(d)	false	no	no	yes
5	(e)	true	no	yes	no
6	(f)	true	no	yes	no
7	(g)	false	no	yes	no
8	(h)	false	yes	no	no
9	(i)	true	no	yes	no
10	(j)	true	no	yes	no

Figure 12: Sample of result test system.

The system succeeded in identifying 18 face pattern images with the help of masks from face image samples with an accuracy of 90.9%.

2.2.4 TensorFlow, CNN, Keras, OpenCV, Pytorch and MobileNetV2 in addition to (R CNN):

One paper only uses the previously mentioned tools and its [14]. The authors wanted to identify whether people have their masks on or not, so they are going to use OpenCV, CNN and Pytorch. R CNN is going to be used. it is effective when comparing it with the other machine learning model. RCNN is an object detection model that segments and localizes objects by using bottom-up region proposals and high-capacity CNNs. It employs selective search to find a number of potential bounding-box object regions ("regions of interest") and then independently extracts features for classification from each region.

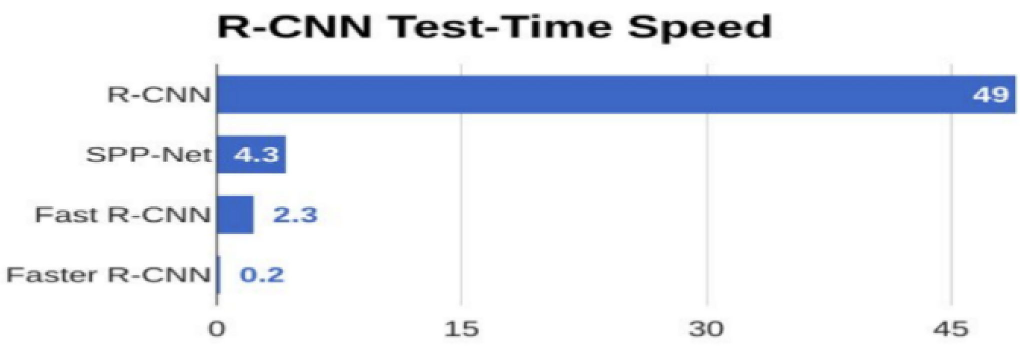


Figure 13: comparison between different object detection algorithms

Although the model's accuracy is about 60%, the process of improving the model is ongoing. A highly accurate solution is being built by tuning the hyperparameters.

2.3 Comparison

This section is going to compare the performance of a paper and the model which is going to be talked about in detail later on in the implementation section.

2.3.1 Viola Jones Method

In paper [15] the authors designed a face detector which utilizes the viola jones method. This detector makes use of extremely basic rectangular features to compare the intensity of rectangular sections. The faces detector employs a cascade of classifiers, where the initial sections of the cascade eliminate the simplest negatives and, as it advances in the cascade, the classifiers get more precise and discriminate the most challenging situations. An additive logistic regression model was developed using a Logit Boost variation of AdaBoost, which optimizes the log-likelihood stage-by-stage. The face detector consists of two phases, the first of which uses grayscale photos and the second of which uses color images. The LFW picture dataset has been used for the training phase of the grayscale case. In the training stage, a number of 15,000 photos were used. The model of this paper scored an accuracy of 95% while the model in the implementation section which uses OpenCV, Caffe framework and MobileNetV2 has scored an accuracy of 98% so it is considered better than the model described in this paper.

2.3.2 Yolo Algorithm

In paper [16] the authors created a face mask detection model using Yolo algorithm. Yolo is one of the quickest object detecting techniques. By enabling quick and precise detection in a single pass through the neural network, it revolutionized object detection. In order to forecast bounding boxes and class probabilities for objects directly at each grid cell, YOLO divides the input image into a grid. This method enables for effective detection while doing away with the necessity for region recommendations. Yolov4 was the version that the authors used. Up to 9000 classes can be predicted by YOLO v4. The foundation of YOLO v4 is a single convolution neural network. An image is divided into regions by the CNN, which also forecasts the boundary boxes and probabilities for each region. YOLO encodes contextual information about classes in addition to their looks because it views the full image during training and testing.

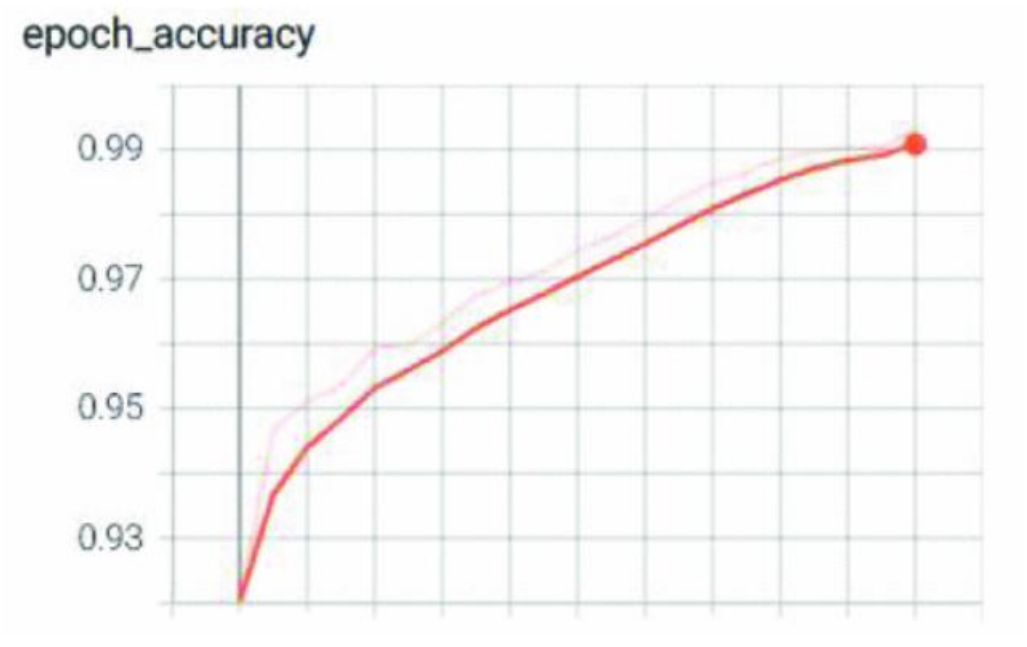


Figure 14:The accuracy of the Yolo algorithm model

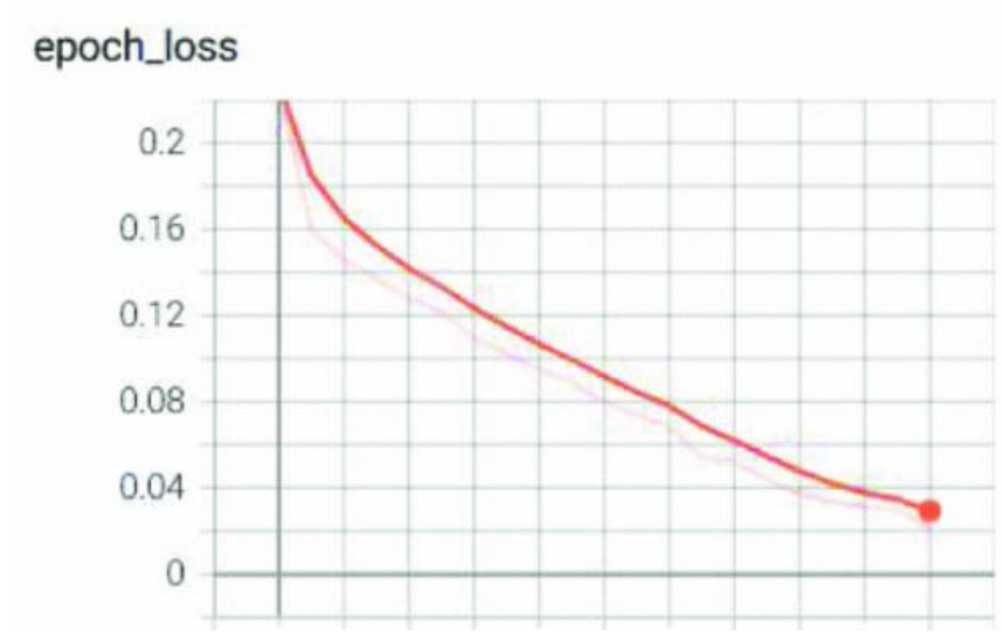


Figure 15: The loss of the yolo algorithm model

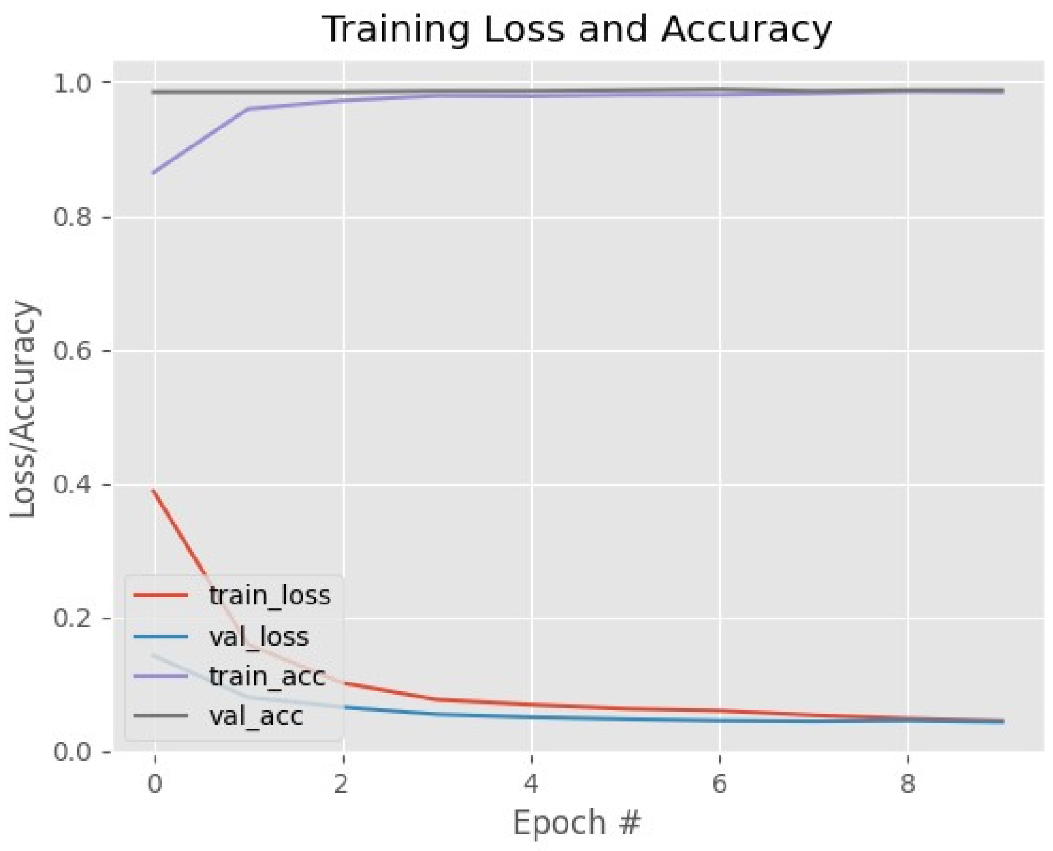


Figure 16: the training loss and accuracy and the validation loss and accuracy of the model in the implementation section

The Yolo model managed to score a fabulous 99.5% accuracy. The model in the implementation section scored an accuracy of 98%.

2.3.3 CNN Algorithm

In paper [17] the authors built a face mask detection model using CNN algorithm. The Convolution Neural Network (CNN) is used to identify if a person is wearing a face mask or not wearing one. This system can be easily connected with any surveillance system installed at a premise. Thus, it makes the process of monitoring easier. The face detection is easily able to detect the correct wearing of a face mask and notifies whether a person is not wearing the face mask in the correct manner as many people tend to do so. It is possible as it is trained in appropriate neural system to achieve accurate results. Every frame of the video has its faces detected. The received frames are transformed from RGB to grayscale pictures. This is accomplished by using the Haar Cascade method to determine whether each frame contains human faces [2]. The detect Multiscale () method of the face Cascade object of the Haar Cascade takes a frame as an argument and applies the classifier to the image [2]. In order to identify faces of various sizes, this method essentially analyzes the frame's subregions at multiple scales.

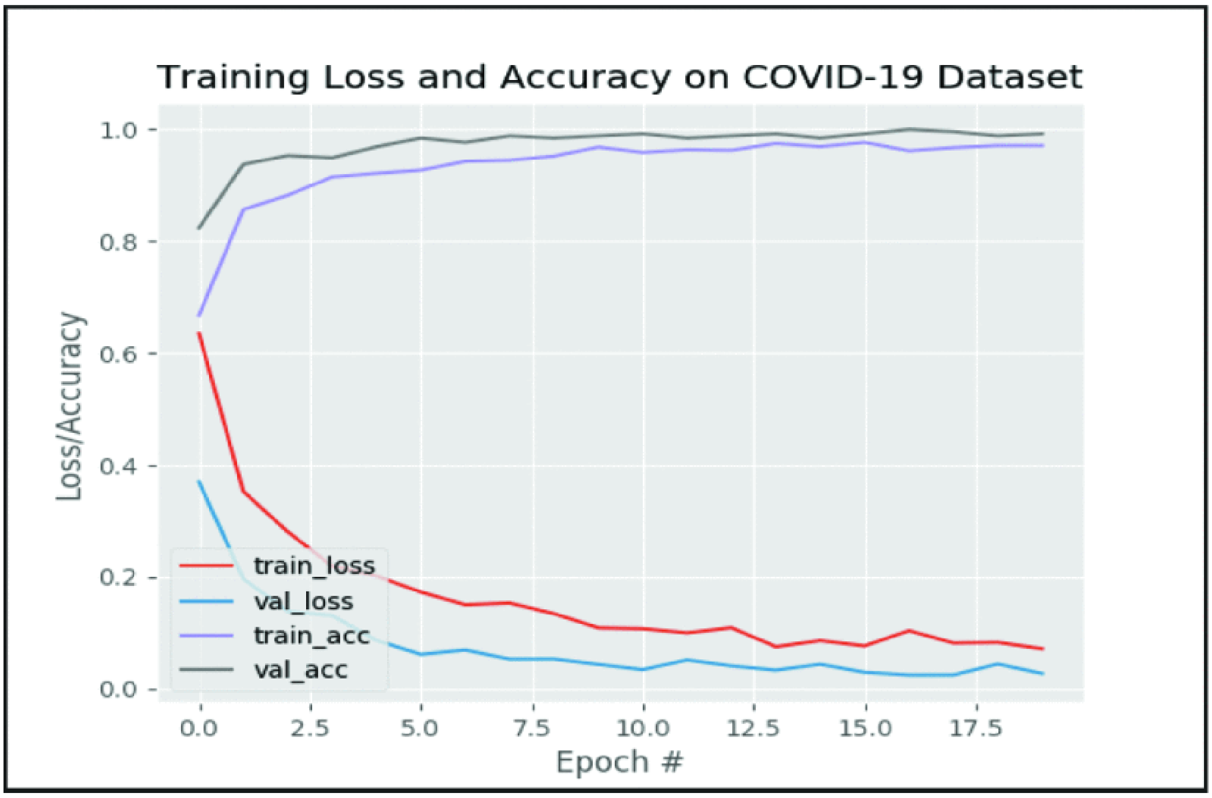


Figure 17: The training loss and accuracy of the paper model

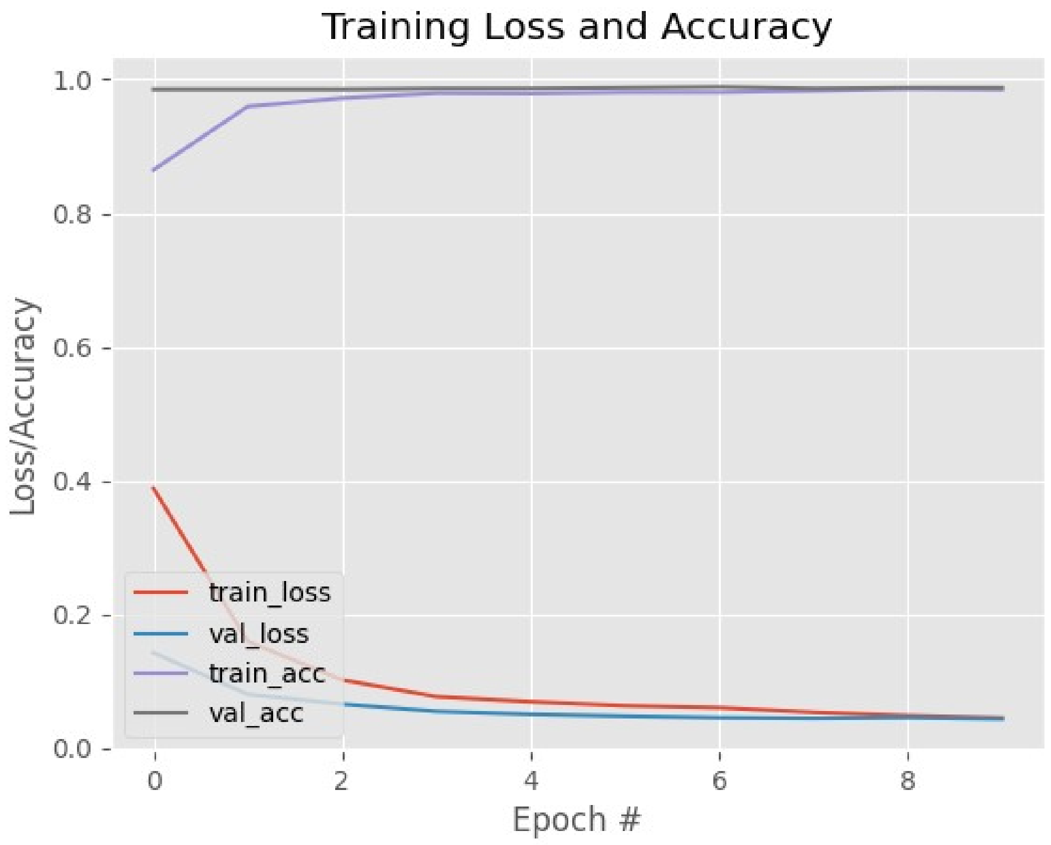


Figure 18: The training loss and accuracy of the model in the implementation section

The accuracy of the model of the paper is a 96% and it scored a validation of 98% while the accuracy of the model in the implementation section scored a 98% and it had a validation of 99%.

2.4 Analysis of the Related Work

After going through many papers, it can be deduced that lots of tools are common between the papers. After talking about four topics in the literature review the most common tools used in the topics are (TensorFlow, CNN, Keras, OpenCV and MobileNetV2) as 8 papers out of 14 use only these tools specifically which is a big number and it can be deduced that these tools are the optimum tools for maximizing the accuracy of face mask detection systems.

3.Face Mask Detection System

3 Face Mask Detection System

3.1 Solution Methodology

3.1.1 The First Model

This system is going to be implemented using OpenCV which is a computer vision library and keras which is a deep learning framework.

3.1.2 The Last Model

The system is going to be implemented by using OpenCV which is a computer vision library, TensorFlow which is an open-source platform for machine learning, keras which is a deep learning framework and MobileNetV2 which is a convolutional neural network (CNN) architecture. A mobile application for face mask detection which uses the same model which the system created that application is going to be developed using flutter which is a user interface toolkit used for building applications for mobile, web and desktop platforms then using the dart programming language as the primary language for flutter and finally TFLite which is a lightweight version of TensorFlow.

3.2 Detailed Explanation

This section is going to talk about the important algorithms, techniques and hyperparameters used in more details.

3.2.1 The First Model

- **RMSProb**: RMSProp (Root Mean Square Propagation) is an optimization algorithm commonly used for training neural networks. It is an adaptive learning rate method that adjusts the learning rate for each parameter based on the magnitude of recent gradients. RMSProp is particularly effective in dealing with issues such as vanishing or exploding gradients by adapting the learning rate on a per-parameter basis. It reduces the learning rate for parameters with large gradients and increases it for parameters with small gradients. This adaptive behaviour allows for more stable and efficient training.
- **Batch Normalization**: Batch normalization is a technique used in deep neural networks to improve the training process and the overall performance of the model. It aims to address the problem of internal covariate shift, which refers to the change in the distribution of layer

inputs during training. Batch normalization's main principle is to use the mean and variance calculated over a mini batch of training instances to equalize the activations of each layer. Prior to using the activation function, the normalization is used.

- **MaxPooling**: Max pooling is a pooling operation commonly used in convolutional neural networks (CNNs) to down sample feature maps. It reduces the spatial dimensions of the input data while retaining the most salient features by selecting the maximum value within each pooling region. Max pooling's main goal is to isolate the most dominating features from each pooling zone while tossing out less important data. Max pooling assists in achieving translation invariance and robustness to small spatial fluctuations in the input data by choosing the maximum value.

3.2.2 The Last Model

- **OpenCV**: OpenCV (Open-Source Computer Vision Library) is a well-known open-source computer vision and machine learning software library. For developers and academics working with photos and videos, it offers a wide range of features and algorithms. It has interfaces for Python, Java, MATLAB/Octave, and other computer languages, and it is written in the C++ programming language. Over 2,500 efficient algorithms are available in the library, covering topics including object detection and recognition, image processing, video analysis, camera calibration, machine learning, and more. Users may carry out operations including face detection, image stitching, feature extraction, image segmentation, optical flow analysis, and more thanks to these techniques.
- **TensorFlow**: TensorFlow is a complete open-source machine learning platform. Researchers can advance the state-of-the-art in machine learning thanks to its extensive, adaptable ecosystem of tools, libraries, and community resources, while developers can simply create and deploy ML-powered applications. To undertake machine learning and deep neural network research, the Google Brain team, a group of researchers and engineers within Google's Machine Intelligence Research division, created TensorFlow. The system is broad enough to work in a number of different additional domains as well. Stable Python and C++ APIs are offered by TensorFlow, as well as non-guaranteed backward compatibility APIs for other languages. TensorFlow uses dataflow graphs to represent calculations. The nodes in this network correspond to mathematical operations, and the edges to the data arrays—referred to as tensors—that connect the operations. With the aid of TensorFlow, users can effectively define and run these computational graphs on CPUs, GPUs, or other specialized hardware accelerators.
- **Keras**: Python-based Keras is an open-source framework for deep learning. It offers a high-level API for creating and training neural networks, which makes it simpler to design deep

learning models and prototypes. User-friendly, modular, and extendable is how Keras is made. An easy-to-use interface is provided by Keras for defining and customizing neural networks. Users may create model topologies, provide activation functions, and stack layers with ease. A variety of pre-built layers, optimizers, loss functions, and other building elements are available in the Keras framework and can be simply coupled to produce unique neural network topologies. Additionally, it enables the creation of original layers and extensions. Keras provides high-level abstractions for common deep learning tasks, such as image classification, object detection, and natural language processing (NLP). These abstractions simplify the development process and abstract away many implementation details.

- **MobileNetV2**: A convolutional neural network (CNN) architecture called MobileNetV2 was created specifically for effective and portable deep learning on mobile and embedded devices. It aims to deliver higher performance and accuracy while keeping a compact model size and computational efficiency, which is an advance over the original MobileNet architecture. To lower convolutional computation costs while maintaining representational capacity, MobileNetV2 combines depth wise separable convolutions and linear bottlenecks. Depth wise separable convolutions divide the standard convolution process into a pointwise convolution that applies 1x1 filters to combine the input channels and a depth wise convolution that applies a single filter to each input channel. To increase efficiency even more, linear bottlenecks add a second bottleneck layer with smaller dimensions. By utilizing these architectural components, MobileNetV2 achieves a good balance between model size, latency, and accuracy. It is suitable for a range of computer vision tasks such as image classification, object detection, and semantic segmentation, particularly on resource-constrained devices where computational and memory limitations exist.
- **CNN**: A deep learning network design called a convolutional neural network (CNN) learns directly from data. CNNs are very helpful for recognizing objects, classes, and categories in photos by looking for patterns in the images. They can be quite useful for categorizing signal, time-series, and audio data. Tens or even hundreds of layers can be present in a convolutional neural network, and each layer can be trained to recognize various aspects of an image. Each training image is subjected to filters at various resolutions, and the result of each convolved image is utilized as the input to the following layer. Beginning with relatively basic properties like brightness and borders, the filters can get more complicated until they reach characteristics that specifically identify the object.
- **Average Pooling**: Average pooling is a type of pooling operation commonly used in convolutional neural networks (CNNs) for downscaling feature maps. It reduces the spatial dimensions of the input data while retaining essential information by taking the average value within each pooling region. A pooling window or filter with a fixed size and stride is defined

in average pooling. The pooling window moves over the input feature map, computing the average value of the window's components at each place. The appropriate area in the output feature map is replaced by the calculated average value. The main purpose of average pooling is to reduce the spatial resolution of the feature map while preserving the dominant features. By summarizing local information through averaging, average pooling helps to achieve translation invariance and robustness to small spatial variations in the input data.

- **Dropout**: Dropout is a regularization technique used in neural networks to prevent overfitting and improve generalization performance. It works by randomly "dropping out" (setting to zero) a certain proportion of the neurons or connections in a neural network during training. Dropout deactivates a specific proportion of neurons or connections throughout each training cycle with a probability known as the dropout rate. This indicates that at that iteration, the dropped-out neurons have no impact on the network's forward pass or backward pass. Dropout prevents the network from becoming overly dependent on certain neurons or connections by arbitrarily removing neurons or connections. Because it must disperse the learning over various combinations of neurons or connections, this encourages the network to develop more reliable and generalized representations. When several subnetworks are taught with various subsets of neurons or connections dropped out, dropout functions as a type of ensemble learning. The weights of the dropped-out neurons or connections are adjusted to account for the dropout during training, but the entire network is used at inference time without dropout.
- **Adam**: Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used for training deep neural networks. It combines the benefits of two other popular optimization methods, AdaGrad and RMSProp, to provide efficient and adaptive learning rates. Every parameter in the neural network continues to train at an adjustable rate thanks to the Adam optimizer. Based on the gradients' first instant (the mean) and second moment (the uncentered variance), it calculates individual learning rates. The "momentum" and "variance" terms of the algorithm are computed as exponential moving averages of the gradients and their squares. The parameters are then updated during training using these estimates. Additionally, Adam uses bias correction to account for momentum and variance term initialization bias, particularly in the early phases of training when the estimates might be off.
- **Imutils**: Imutils is a Python library that provides convenience functions for working with images and video processing tasks. It is primarily designed to simplify common image processing operations and bridge compatibility gaps between different computer vision libraries, such as OpenCV. Imutils provides functions to resize images while maintaining the aspect ratio, crop images based on specific dimensions or regions of interest, and handle resizing for both width and height. Imutils provides functions to draw shapes like rectangles,

circles, lines, and text on images, which can be useful for visualizing results or annotations. Working with video streams is made easier by Imutils, which makes it simpler to read frames from a video file or a camera, resize frames, and display them.

- **ReLU:** ReLU (Rectified Linear Unit) is an activation function commonly used in neural networks, especially in deep learning architectures. It introduces non-linearity to the network by outputting the input directly if it is positive and zero if it is negative. ReLU gives the network non-linear behaviour, which enables it to simulate more intricate interactions between input and output. ReLU' s linearity for positive values facilitates gradient propagation and quick training. ReLU does not saturate for large positive inputs. This characteristic helps alleviate the vanishing gradient problem, where gradients become extremely small, hindering the learning process.
- **SoftMax:** The SoftMax function is an activation function that transforms a vector of K real values into a vector of K real values that sum to 1. The SoftMax turns the input values, which can be positive, negative, zero, or higher than one, into values between 0 and 1, allowing them to be understood as probabilities. The SoftMax converts little or negative inputs into small probabilities, and big or positive inputs into large probabilities, although it will always fall between 0 and 1. the SoftMax function is often combined with a loss function such as categorical cross-entropy to measure the discrepancy between predicted and actual class probabilities. This loss is then used to update the network's weights and improve its performance.
- **Binary Cross Entropy:** Binary cross entropy is a commonly used loss function in binary classification tasks. It measures the dissimilarity between the predicted probabilities and the true binary labels of a binary classification problem. The loss function compares the anticipated probability (p) with the actual label (y) to determine the logarithmic loss for each sample in the dataset. The loss penalizes larger departures from the true label more severely thanks to the logarithmic terms. Since the binary cross entropy loss function is differentiable, gradient-based optimization algorithms can be employed to change the model's weights during training. The objective is to reduce this loss function as much as possible, which enhances the model's capability to correctly forecast the binary classes.

3.2.3 The Application

- **Flutter:** Flutter is a mobile app development platform created by Google. It allows developers to create web, desktop, and cross-platform apps that run on Android and iOS devices. Flutter uses a reactive programming language called Dart, making development faster and easier than traditional methods. Flutter offers a rich set of customizable and pre-built UI components, known as widgets, that make it easy to create beautiful and consistent user interfaces. These widgets can be easily customized and styled to match specific design requirements.

- **Dart:** Dart is a programming language developed by Google. It is designed to be a versatile and efficient language for building applications across different platforms, including mobile, web, desktop, and server-side applications. Dart combines a modern syntax with features that promote productivity, performance, and scalability. The async/await syntax makes it simple to build asynchronous code in Dart because it contains native support for asynchronous programming. This makes it possible to handle I/O operations and concurrency effectively, leading to applications that are responsive and effective. Both JIT and AOT compilation are supported by Dart. The Dart virtual machine (VM) leverages JIT compilation during development to provide quick iteration and hot reloading. Dart can be AOT compiled into highly optimized native code for production deployments to improve performance.
- **TFLite:** TFLite (TensorFlow Lite) is specifically designed for mobile and embedded devices with limited computational resources. It enables efficient deployment of machine learning models on devices such as smartphones, IoT devices, and microcontrollers. TensorFlow models can be converted using TFLite's tools and converters into a format that is best for inference on devices with limited resources. The model size is decreased during conversion, memory use is optimized, and hardware acceleration capabilities on the target device are taken advantage of. On devices, model inference is performed using the TFLite runtime. It offers an interpreter that enables the low latency and small memory footprint execution of TFLite models. The TFLite runtime is tailored for many platforms and hardware architectures and supports a wide range of operations. TFLite supports a wide range of machine learning tasks, including image classification, object detection, natural language processing, and more. It offers APIs for model loading, inference, and accessing the model outputs. Additionally, TFLite supports hardware-specific optimizations, such as using the Android Neural Networks API or the Edge TPU for accelerated inference on specific devices.

3.3 Functional/ Non-functional Requirements

The following table shows us the functional requirements which is in the first column and in the second column the explanation of each functional requirement is given.

Functional Requirements	**Explanation**
Detect mask	This function tells us whether a person has his mask on or not
Drawing box	This function draws a box around a person's face
Multiple face detection	This function detects multiple faces in the frame at the same time

Table 1: Functional Requirements.

The coming table showcases the non-functional requirements in the first column while the second column explains to us each non-functional requirement.

Non-Functional Requirements	**Explanation**
Reliable	The ability to always perform under any circumstances without shutting down
Performance	Fast response times with no latency
Scalability	The ability to work under immense workload
Recoverability	The ability to repair or replace system components

Table 2: Non-Functional Requirements.

4.Implementation

4 Implementation

4.1 The Dataset

The dataset contains over 4000 images of persons with masks and persons with no masks so with mask and without mask are considered our two classes. The dataset has been gathered from [18] which is Kaggle. Kaggle is an online platform for data science and machine learning competitions, datasets, and a community of data scientists and machine learning practitioners.

4.2 The First Model

4.2.1 Importing

- from keras.optimizers import RMSprop
- from keras.preprocessing.image import ImageDataGenerator
- import cv2
- from keras.models import Sequential
- from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout
- from keras.models import Model, load_model
- from keras.callbacks import TensorBoard, ModelCheckpoint
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import f1_score
- from sklearn.utils import shuffle
- import imutils
- import numpy as np

these lines import the necessary libraries and techniques.

4.2.2 CNN

1- model =Sequential([

2- Conv2D(100, (3,3), activation='relu', input_shape=(150, 150, 3)),

3- MaxPooling2D(2,2),

4- Conv2D(100, (3,3), activation='relu'),

5- MaxPooling2D(2,2),

6- Flatten(),

7- Dropout(0.5),

8- Dense(50, activation='relu'),

9- Dense(2, activation='softmax')])

10- model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

The provided lines of code define and compile a convolutional neural network (CNN) model.

4.2.3 Training

1- model.fit_generator(train_generator,

2- epochs=10,

3- validation_data=validation_generator,

4- callbacks=[checkpoint])

These lines of code initiate the training process of the model using generator functions for data loading and augmentation, specifying the number of epochs, validation data, and callbacks to be utilized during training.

4.2.4 Activating The Webcam

1- webcam = cv2.VideoCapture(0)

2- cv2.imshow('Detecting Face…',   im)

These lines are executed to trigger the webcam to activate.

4.2.5 The Detection

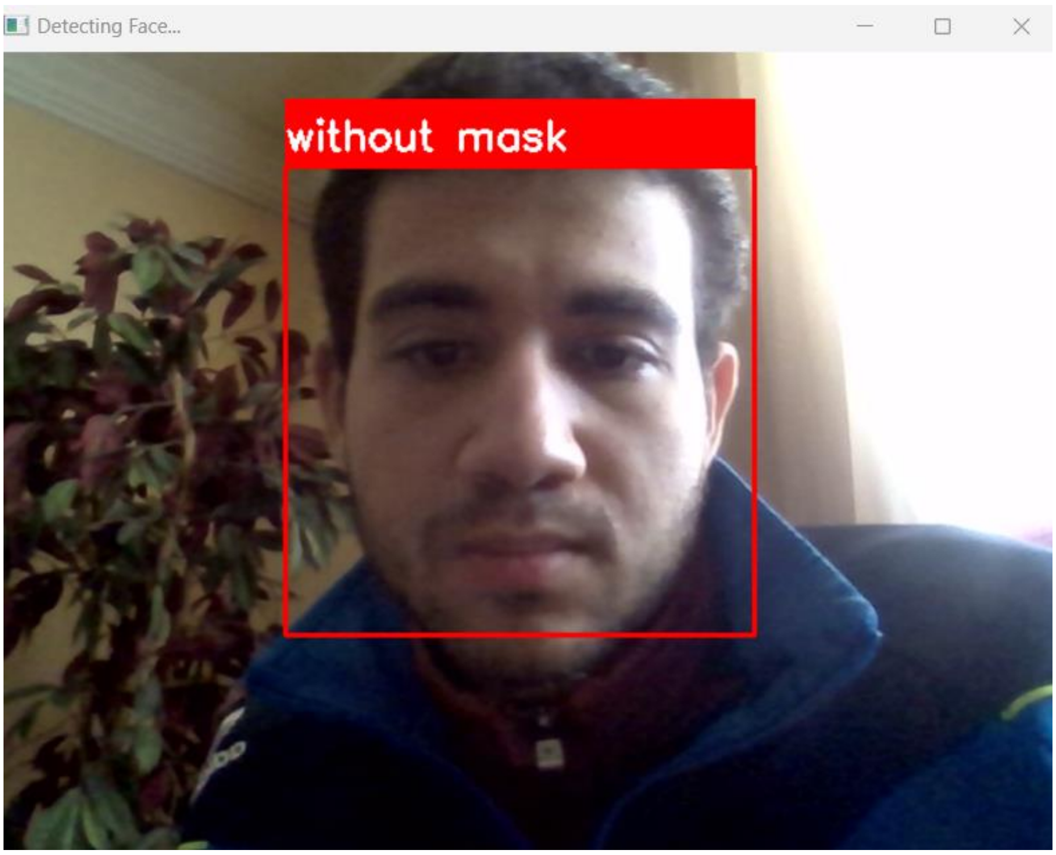


Figure 19: Webcam of first model detecting a person not wearing a face mask.

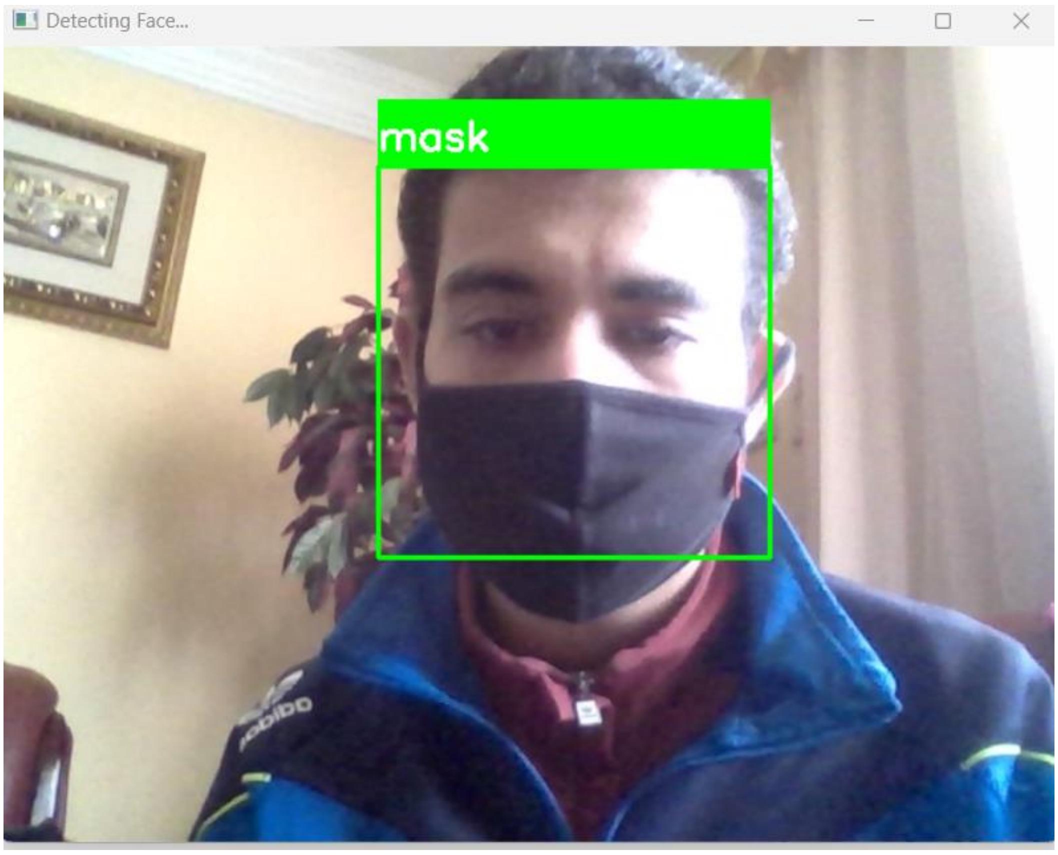


Figure 20: Webcam of first model detecting a person wearing a face mask.

4.3 The Last Model

The Last Model consists of three important parts Training, detecting mask by video and detecting mask by image.

4.3.1 Training

4.3.1.1 Importing

- from tensorflow.keras.preprocessing.image import ImageDataGenerator
- from tensorflow.keras.applications import MobileNetV2
- from tensorflow.keras.layers import AveragePooling2D
- from tensorflow.keras.layers import Dropout
- from tensorflow.keras.layers import Flatten
- from tensorflow.keras.layers import Dense
- from tensorflow.keras.layers import Input

- from tensorflow.keras.models import Model
- from tensorflow.keras.optimizers import Adam
- from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
- from tensorflow.keras.preprocessing.image import img_to_array
- from tensorflow.keras.preprocessing.image import load_img
- from tensorflow.keras.utils import to_categorical
- from sklearn.preprocessing import LabelBinarizer
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import classification_report
- from sklearn.metrics import confusion_matrix
- import itertools
- from imutils import paths
- import matplotlib.pyplot as plt
- import numpy as np
- import argparse
- import os

these lines when executed import the necessary libraries and techniques.

4.3.1.2 Initialization

1- INIT_LR = 1e-4

2- EPOCHS = 10

3- BS = 32

These lines initialize the initial learning rate, number of epochs to train for and batch size.

4.3.1.3 Image Preprocessing

1- image = load_img(imagePath, target_size=(224, 224))

2- image = img_to_array(image)

3- image = preprocess_input(image)

An image is prepared for a deep learning model's input. The image is loaded, scaled to a particular dimension (for example, 224x224 pixels), translated to a numerical array, and then preprocessed to satisfy the input specifications of the selected model. By taking these measures, you can be sure that the image will be correctly structured and work with the deep learning model for processing and inference.

4.3.1.4 Splitting The Data

```
(trainX, testX, trainY, testY) = train_test_split(data, labels,
 test_size=0.20, stratify=labels, random_state=42)
```

partitioning the data into training and testing splits.

4.3.1.5 Data Augmentation

```
aug = ImageDataGenerator(
 rotation_range=20,
 zoom_range=0.15,
 width_shift_range=0.2,
 height_shift_range=0.2,
 shear_range=0.15,
 horizontal_flip=True,
 fill_mode="nearest")
```

constructing the training image generator for data augmentation.

4.3.1.6 MobileNetV2

```
baseModel = MobileNetV2(weights="imagenet", include_top=False,
 input_tensor=Input(shape=(224, 224, 3)))
```

loading the MobileNetV2 network.

4.3.1.7 Neural Network Model

```
1- headModel = baseModel.output
2- headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
3- headModel = Flatten(name="flatten")(headModel)
4- headModel = Dense(128, activation="relu")(headModel)
```

5- headModel = Dropout(0.5)(headModel)

6- headModel = Dense(2, activation="softmax")(headModel)

In the previous lines the following was done which is

- adding an average pooling layer.
- adding a flatten layer.
- adding a fully connected layer using the ReLU activation function.
- adding a dropout layer which helps avoid overfitting.
- adding another fully connected layer this time using SoftMax activation function.

4.3.1.8 Training The Model

1- model.fit(

2- aug.flow(trainX, trainY, batch_size=BS),

3- steps_per_epoch=len(trainX) // BS,

4- validation_data=(testX, testY),

5- validation_steps=len(testX) // BS,

6- epochs=EPOCHS)

These lines train the model with a number of 10 epochs in the process showing the accuracy, loss and the validation of the model.

4.3.2 Detection By Video

4.3.2.1 Importing

- from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
- from tensorflow.keras.preprocessing.image import img_to_array
- from tensorflow.keras.models import load_model
- from imutils.video import VideoStream
- import numpy as np
- import argparse
- import imutils

- import time
- import cv2
- import os

importing important libraries and techniques.

4.3.2.2 Initialization

1- faces = []

2- locs = []

3- preds = []

initializing the list of faces, their corresponding locations, and the list of predictions from the face mask network.

4.3.2.3 boundary box

1- (startX, startY) = (max(0, startX), max(0, startY))

2- (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

ensuring the boundary boxes fall within the dimensions of the frame.

4.3.2.4 Image Preprocessing

1- face = frame[startY:endY, startX:endX]

2- if face.any():

3- face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

4- face = cv2.resize(face, (224, 224))

5- face = img_to_array(face)

6- face = preprocess_input(face)

extracting the face ROI, converting it from BGR to RGB channel, ordering, resizing it to 224x224, and preprocessing it.

4.3.2.5 Adjustments

1- label = "Mask" if mask > withoutMask else "No Mask"

2- color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

3- label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

The previous lines show how the label above the boundary box functions and how the probability is measured which are all going to be illustrated when seeing a figure later on.

4.3.2.6 WebCam

cv2.imshow("Frame", frame)

Opening the webcam.

4.3.2.7 The Detection

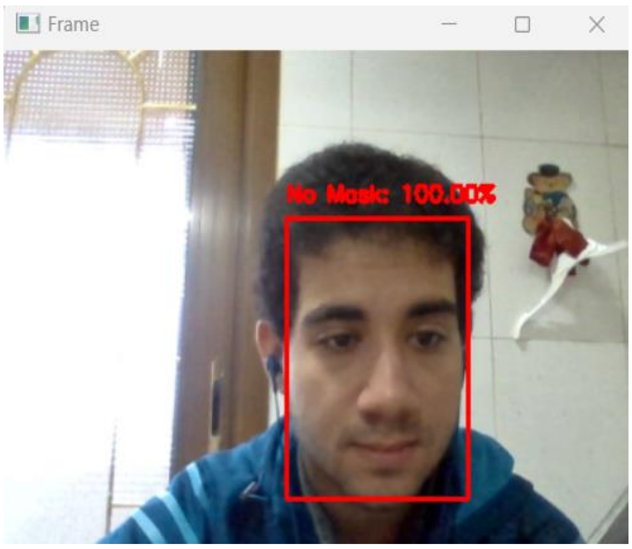


Figure 21: Webcam of last model detecting a person not wearing a face mask.

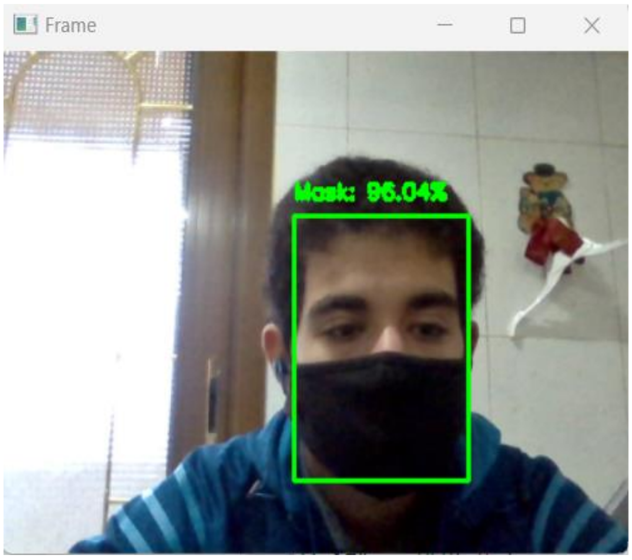


Figure 22: Webcam of last model detecting a person wearing a face mask.

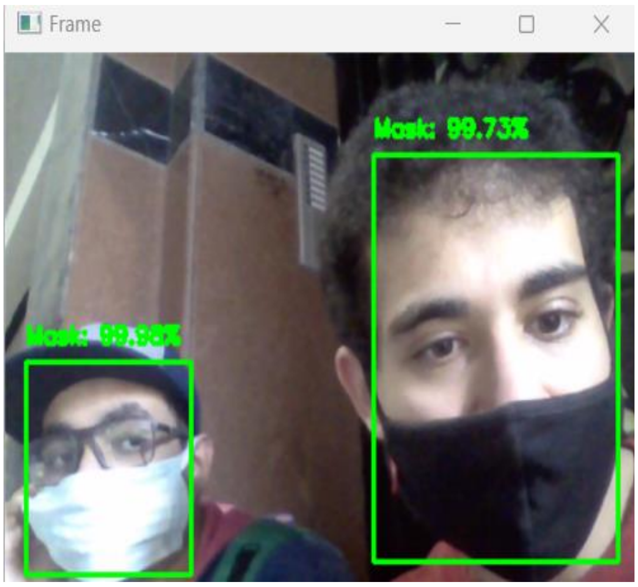


Figure 23: Webcam of last model detecting two persons wearing two different colors of a face mask.

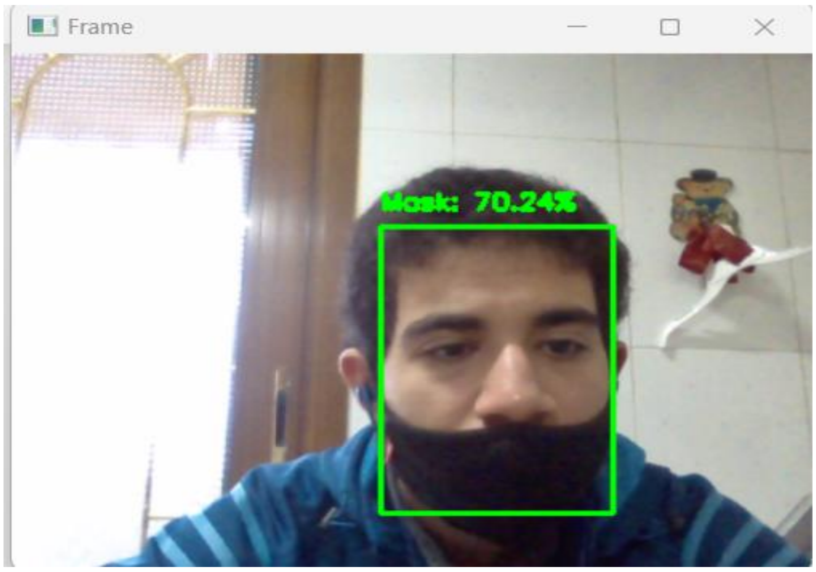


Figure 24: Webcam of last model detecting a person not wearing the face mask properly.

The percentage tends to fluctuate when detecting a face mask not worn properly.

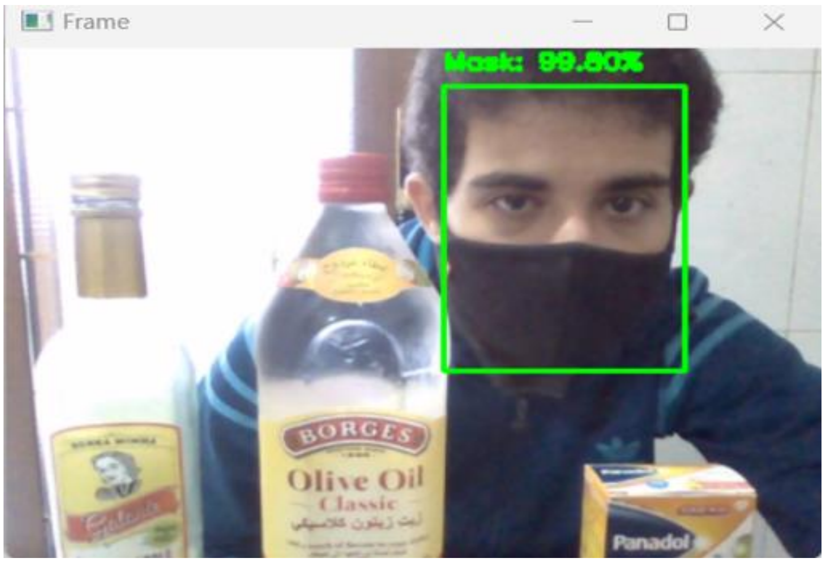


Figure 25:Webcam of last model detecting a person wearing a face mask surrounded by multiple objects.

The model tends to work as intended even when many objects appear in the webcam.

4.3.2.8 Potential Drawback

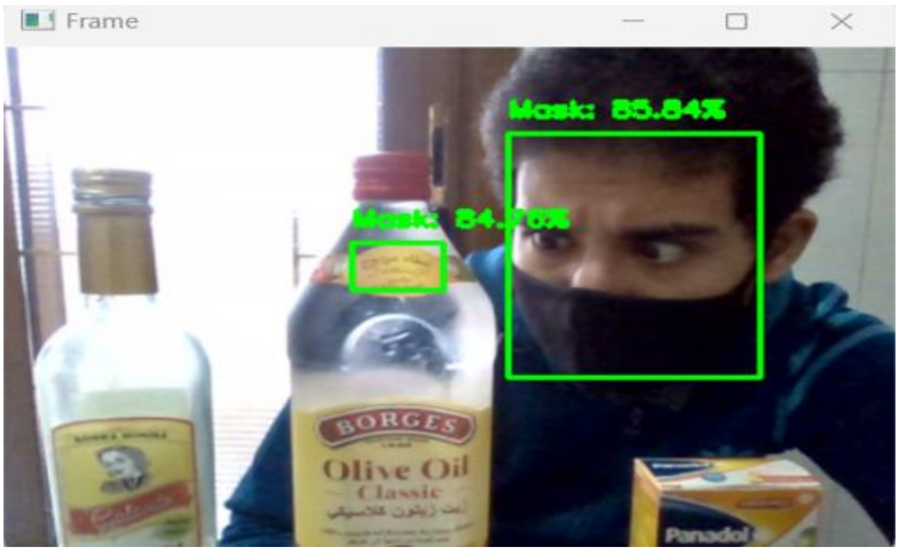


Figure 26: Webcam of last model detecting a person and a sticker of a bottle wearing a face mask.

The model surprisingly detected the sticker of a bottle as a face mask that the bottle is wearing but why is it called potential drawback not actually a drawback is because I tried moving the bottle a lot while experimenting so that this situation would be triggered so I think the possibility of this ever happening to someone is probably like 1%.

4.3.3 Detection By Image

4.3.3.1 Importing

- from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
- from tensorflow.keras.preprocessing.image import img_to_array
- from tensorflow.keras.models import load_model
- import numpy as np
- import argparse
- import cv2
- import os

Importing the essential libraries and techniques.

4.3.3.2 Loading The Model

model = load_model(args["model"])

This line helps in loading the model from the disk.

4.3.3.3 Loading The Image

1- image = cv2.imread(args["image"])

2- orig = image.copy()

3- (h, w) = image.shape[:2]

Loading the input image from the disk and getting the spatial dimensions of the image.

4.3.3.4 Boundary Box

1- (startX, startY) = (max(0, startX), max(0, startY))

2- (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

Adjusting the boundary box.

4.3.3.5 Image Preprocessing

1- face = image[startY:endY, startX:endX]

2- face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

3- face = cv2.resize(face, (224, 224))

4- face = img_to_array(face)

5- face = preprocess_input(face)

6- face = np.expand_dims(face, axis=0)

a face region is extracted from an image and preprocessed for input to a deep learning model. The face image is cropped, color space converted to RGB, resized to a fixed size, converted to a numerical array, preprocessed according to the model's requirements, and expanded to represent a batch of images if needed. These steps ensure that the face region is properly formatted and compatible with the deep learning model for further processing and inference.

4.3.3.6 Mask Prediction

(mask, withoutMask) = model.predict(face)[0]

The model making a decision on if the face is wearing a mask or not.

4.3.3.7 Adjustments

1- label = "Mask" if mask > withoutMask else "No Mask"

2- color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

3- label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

Explaining how the label above the boundary box functions and how the probability is measured.

4.3.3.8 displaying the boundary box and the image

1- cv2.putText(image, label, (startX, startY - 10),

2- cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)

3- cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

4- cv2.imshow("Output", image)

These lines make sure the boundary box and the label are displayed on the image then the image is shown.

4.3.3.9 The Detection

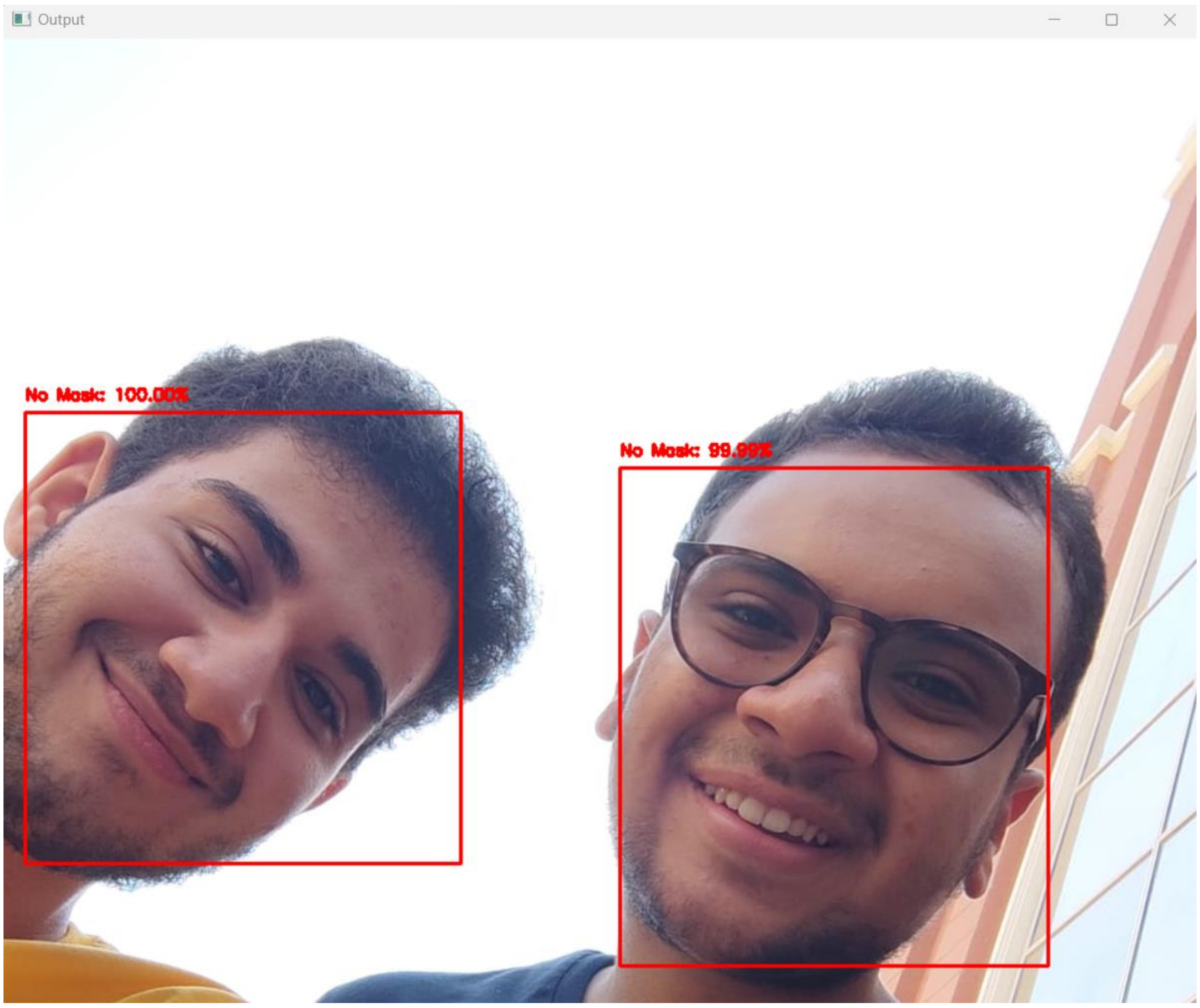


Figure 27: Detection of an image that consists of two persons not wearing a face mask.

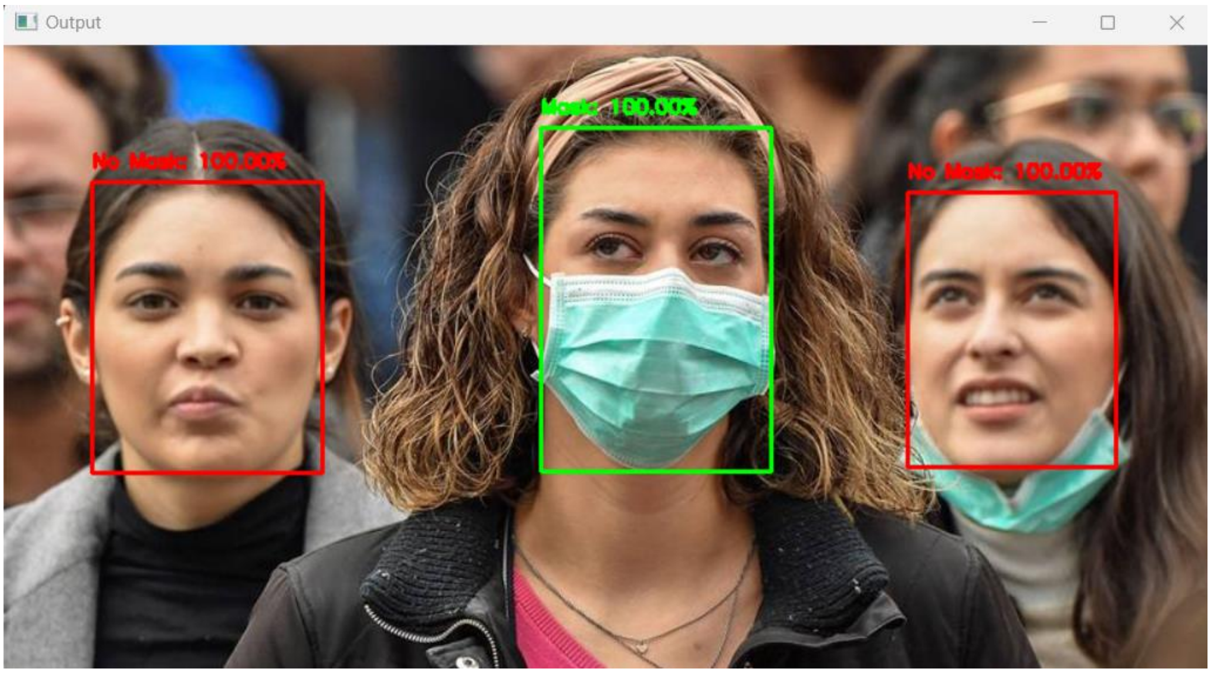


Figure 28:Detection of an image that consists of two persons not wearing a face mask and a person wearing a face mask.

4.4 The Application

When a dart and flutter project is created the following files and directories are automatically created:

- **lib:** This directory is the default location for storing the project's Dart source code files. It contains the main Dart file ('main.dart') that serves as the entry point of the Flutter application. It is where you define the app's functionality, UI, and logic.
- **android:** This directory contains the Android-specific files and configurations for the Flutter project. It includes the Android project structure, such as the 'Main Activity' file, Android manifest file ('AndroidManifest.xml'), and other necessary files for building and running the Flutter app on Android devices.
- **ios:** The Flutter project's iOS-specific files and configurations are located in this directory. It contains the files required to create and execute the Flutter app for iOS devices, including the iOS project structure, the 'AppDelegate' file, the iOS app configuration ('Info.plist'), and other related files.
- **test:** This directory is the default location for storing test files. It is used for writing and running automated tests to verify the correctness of the Flutter app's code.
- **pubspec.yaml:** Similar to a Dart project, this file serves as the project's package specification file. The dependencies, metadata, and other configuration information for the

project are listed. It is used to handle dependencies and create the Flutter app by the package manager 'pub'.

- **pubspec.lock:** This file is automatically generated by the package manager ('pub') and records the exact versions of the dependencies used in the Flutter project. It ensures that subsequent builds use the same dependency versions, providing consistent behavior across different environments.
- **.dart_tool:** This directory is automatically created and contains various tools and artifacts used during development. It typically includes the build system's intermediate files and cache.
- **build:** build is used to specify build settings and customizations for the project's build process. The specific name and format of the build file can vary depending on the platform (Flutter supports both Android and iOS). The build file allows you to enable or disable specific build features, such as optimizations, debug symbols, or specific compiler flags. The build file can be used to specify the exact configurations for each flavor of your app, such as distinct package names, assets, or other settings, if you're dealing with various build variations or flavors of your app (such as development, production, or staging).

The 'lib' directory consists of three dart files a file which adjusts the boundary box, a file which setups the camera and finally the main file which is probably the most important of them all.

4.4.1 Main File

4.4.1.1 Importing

- import 'dart:io';
- import 'package:flutter/material.dart';
- import 'package:mask_detector/face_detection_camera.dart';
- import 'package:tflite/tflite.dart';
- import 'package:image_picker/image_picker.dart';
- import 'package:flutter_svg/flutter_svg.dart';
- import 'package:flutter_share/flutter_share.dart';
- import 'package:url_launcher/url_launcher.dart';
- import 'package:firebase_admob/firebase_admob.dart';

Importing crucial packages and files.

4.4.1.2 Developing The App

```
class MyApp extends StatelessWidget {
@override
Widget build(BuildContext context) {
return MaterialApp(
title: 'Mask detector',
theme: ThemeData(
primarySwatch: Colors.blue,
visualDensity: VisualDensity.adaptivePlatformDensity,
),
home: MyHomePage(),
debugShowCheckedModeBanner: false,
);
}
}
```

defining the overall configuration and structure of the Flutter app. It sets the app's title, theme, initial screen, and debug mode settings.

4.4.1.3 Loading The Model

```
loadModel() async {
await Tflite.loadModel(
model: "assets/model_unquant.tflite",
labels: "assets/labels.txt",
);
}
```

The model (Last Model) which was shown in the last section is being loaded by the app.

4.4.1.4 Getting An Image

```
pickImage() async {
```

```
count++;
print(count);
var image = await ImagePicker.pickImage(source:ImageSource.gallery);
if (image == null) return null;
setState(() {
_loading = true;
_image = image;
});
classifyImage(image);
}
int coun = 0;
```

choosing an image from the gallery and initializing the variable (coun) to track the number of people with masks.

4.4.1.5 incrementing The Variable coun

```
for (var i = 0; i < output.length; i++) {
if (output[i]['label'] == '0 with_mask') {
coun++;
}
}
```

When the output is the class (with_mask) the variable coun is incremented.

4.4.1.6 Detecting With Camera

```
Text(
_outputs[0]["label"]=='0 with_mask'?"Mask detected":"Mask not detected",
style: TextStyle(
color: _outputs[0]["label"]=='0 with_mask'?Colors.green:Colors.red,
fontSize: 25.0,
),
),
```

8- Text("${(_outputs[0]["confidence"]*100).toStringAsFixed(0)}%",style: TextStyle(color:Colors.purpleAccent,fontSize:20),),

The previous lines output a label which decides to output if a mask is detected or not according to the conditions if they were met or not and if it is a mask the text is outputted in green otherwise it is in red. The font size is set to 25. A percentage is also written just like the figures in the last model.

4.4.1.7 Detecting An Image

Text("People with Masks: $peopleWithMasks",style: TextStyle(color: Color.fromARGB(255, 47, 30, 173), fontSize: 20),),

When detecting an image the number of people with masks is outputted alongside if a mask was detected or not and the percentage.

4.4.1.8 Running The App

The format of the model had to be converted to tflite for it to work.

1- import tensorflow as tf

2- model = tf.keras.models.load_model('mask_detector.model')

3- converter = tf.lite.TFLiteConverter.from_keras_model(model)

4- tflite_model = converter.convert()

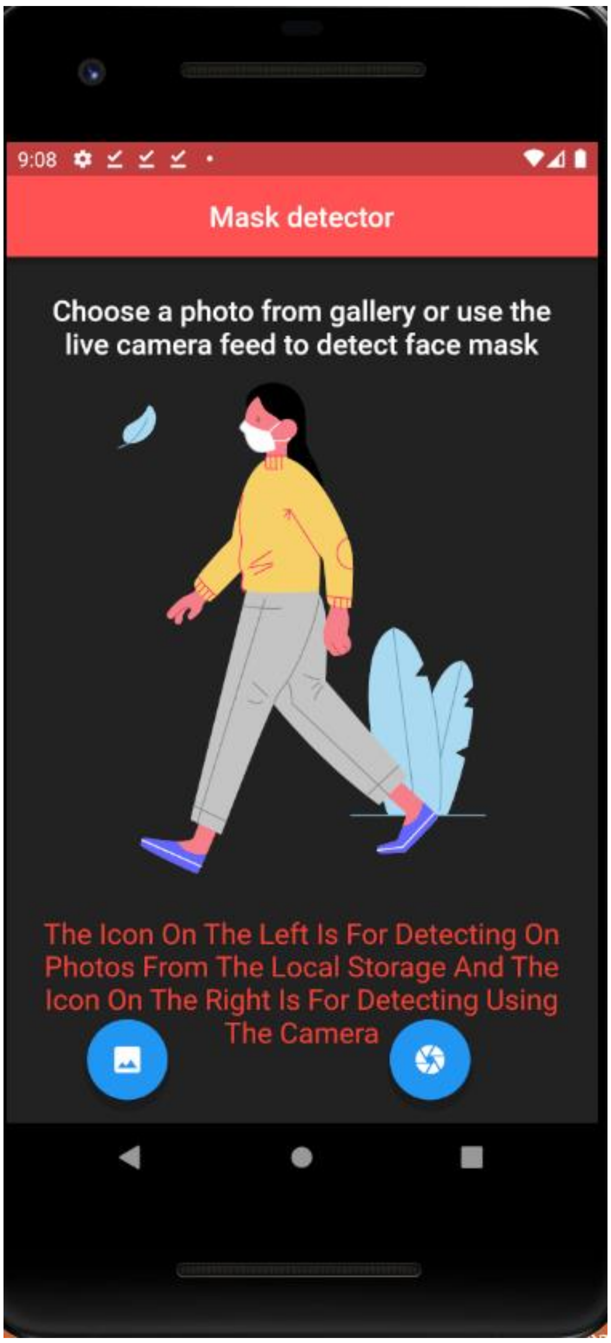
5- with open('my_model.tflite', 'wb') as f:

6- f.write(tflite_model)

These lines were used to load the model then converting it to the preferred format(tflite) and finally saving it to a file.

The app works by running the Main file which allows for an apk to be installed on the android emulator or the mobile phone if one is attached to the pc.

- **apk:** Android Package (also known as Android Package Kit or Android Application Package) is referred to as an APK. Android distributes and installs programs using this file type. As a result, an APK has every component an app requires to successfully install on your device.



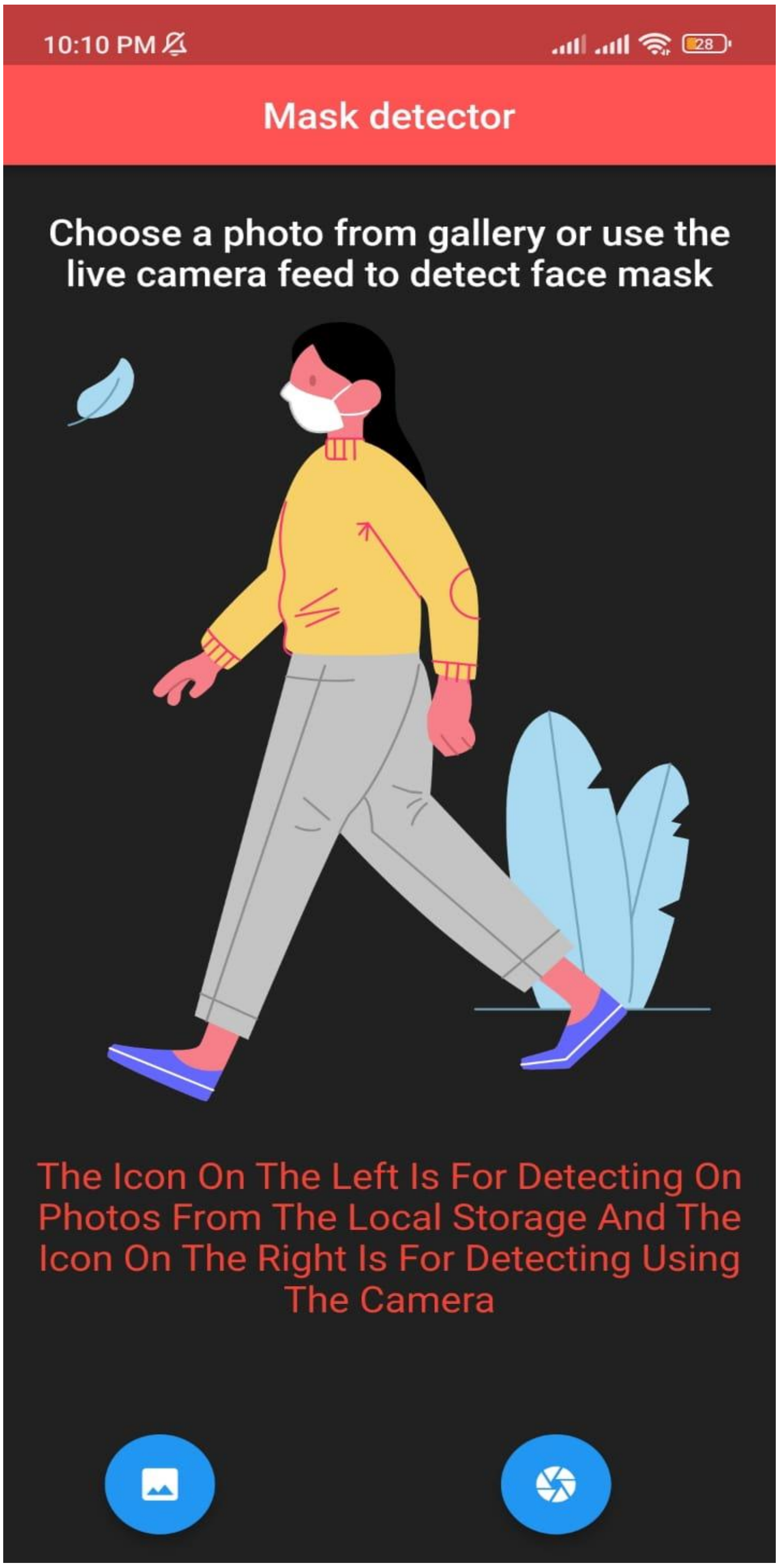
9:08

Mask detector

Choose a photo from gallery or use the live camera feed to detect face mask

The Icon On The Left Is For Detecting On Photos From The Local Storage And The Icon On The Right Is For Detecting Using The Camera

Figure 29: The application working on the android emulator.



Mask detector

Choose a photo from gallery or use the live camera feed to detect face mask

The Icon On The Left Is For Detecting On Photos From The Local Storage And The Icon On The Right Is For Detecting Using The Camera

Figure 30: The application working on a mobile phone.

4.4.1.9 The Detection

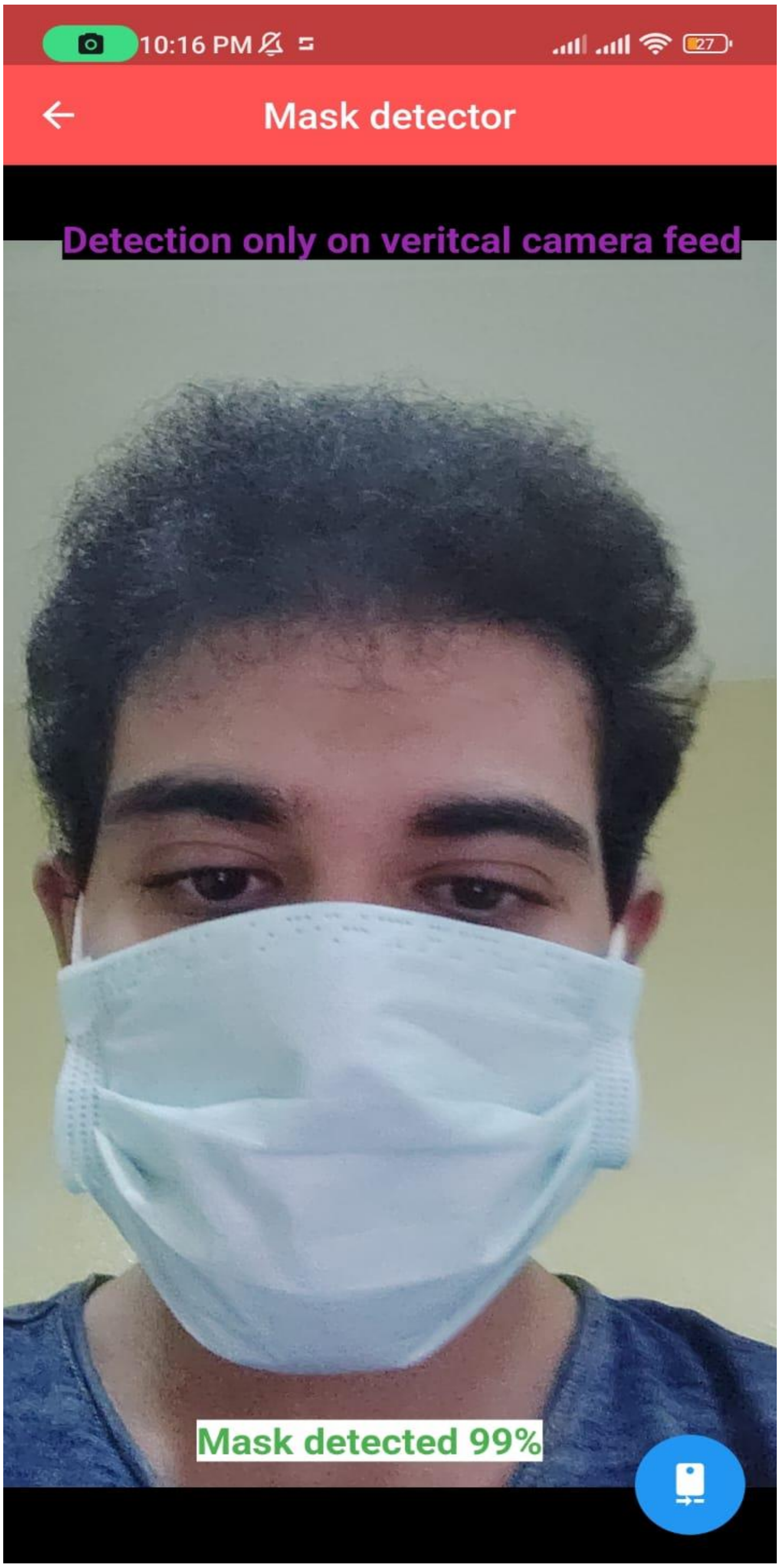


Figure 31: Detecting through camera using the mobile phone with mask on.

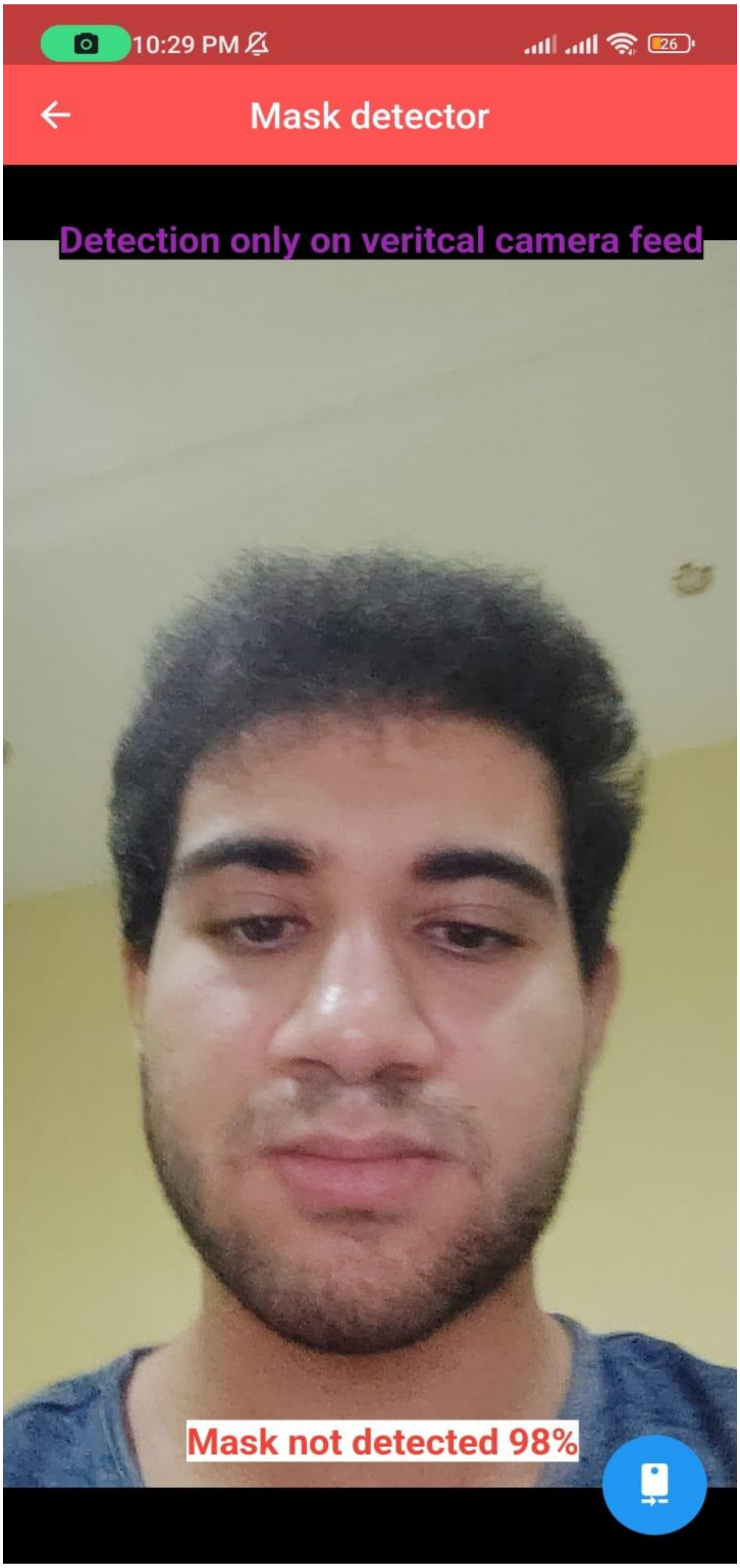


Figure 32: Detecting through camera using the mobile phone without mask.

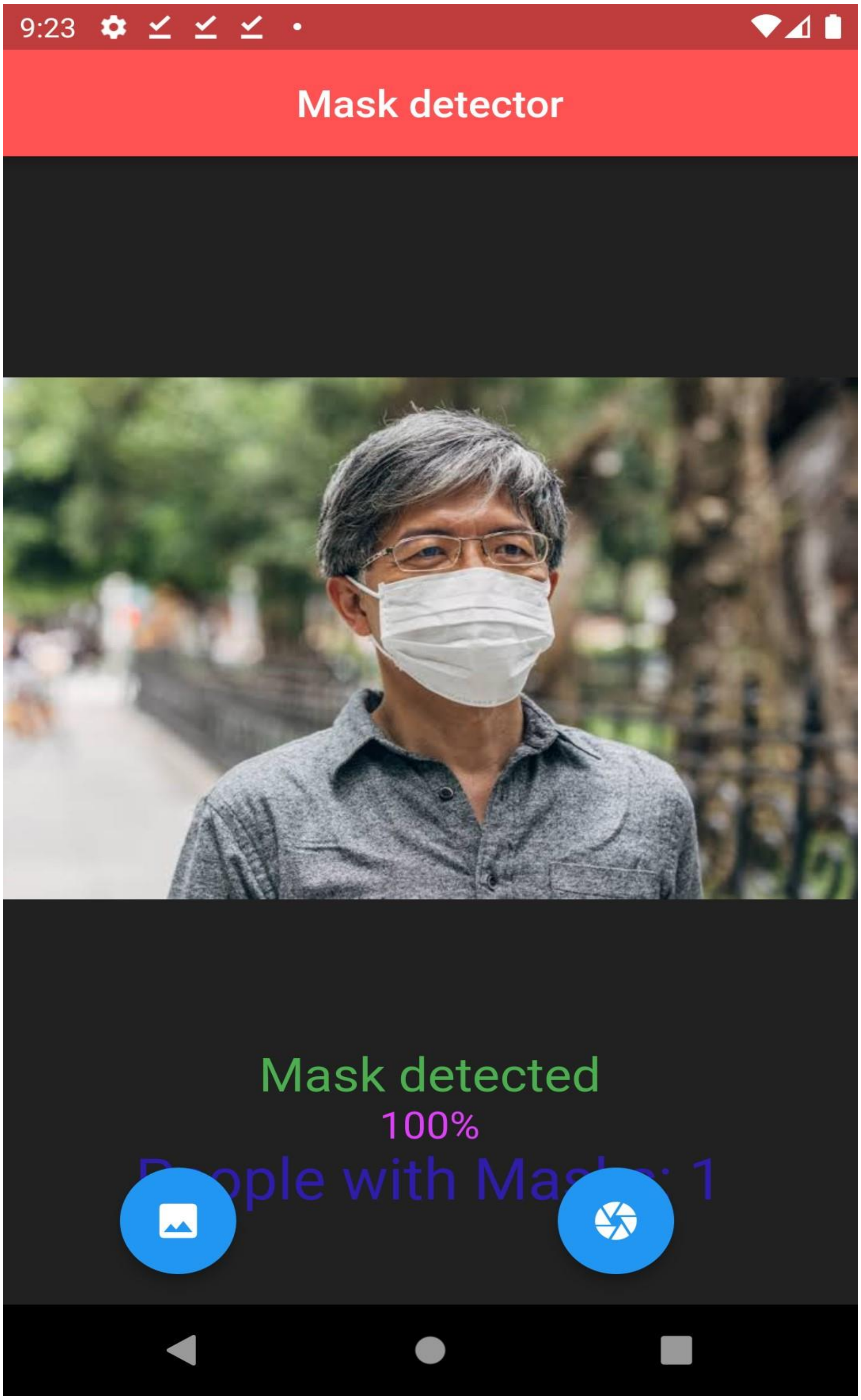
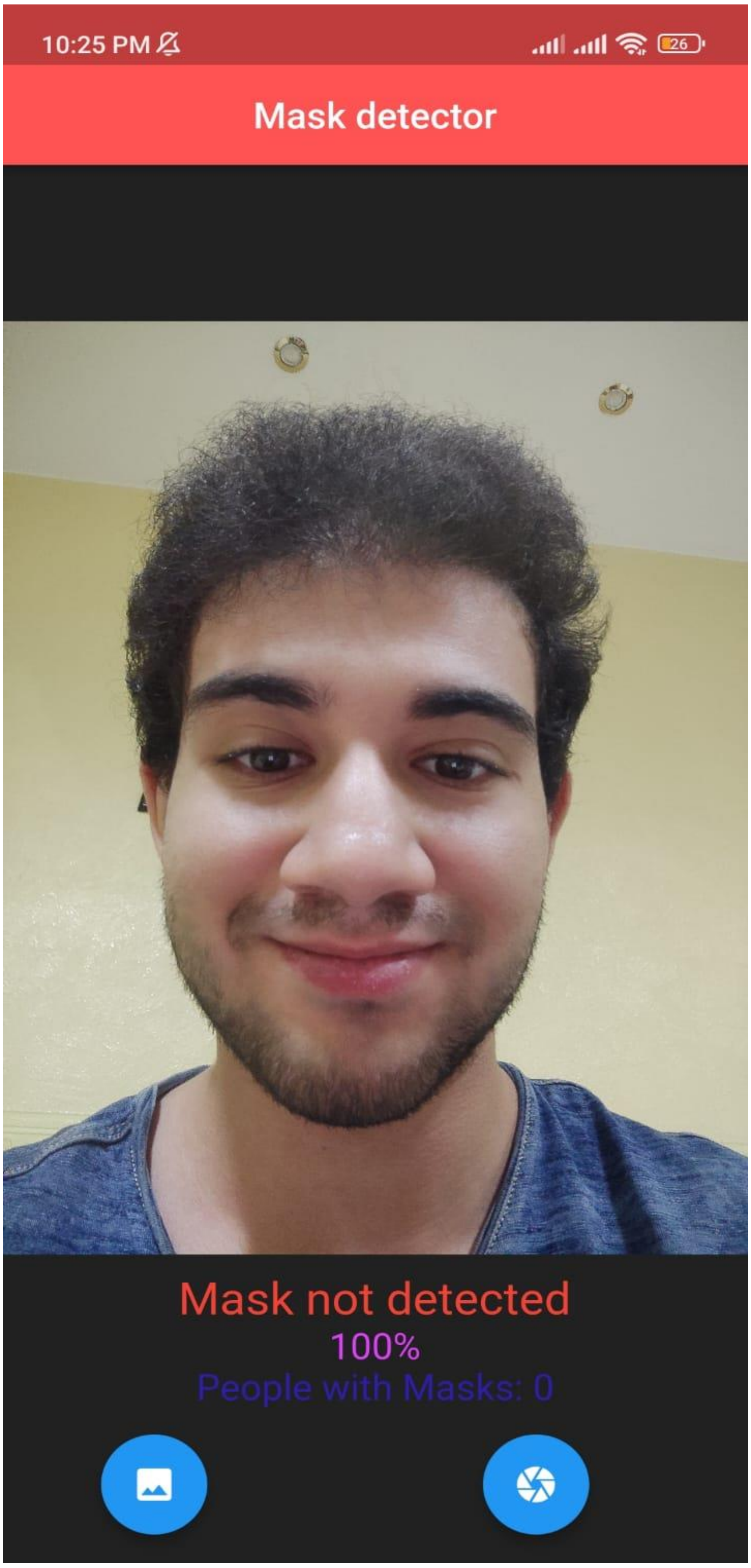


Figure 33: Detecting through an image from the gallery using the android emulator.



Mask not detected
100%
People with Masks: 0

Figure 34:Detecting through an image from the gallery using the mobile phone.

4.4.1.10 Drawback

The people with masks variable only outputs 0 or 1 it can’t output more than that. I tried fixing this problem by trying to implement a line that crosses through the faces in the image and detecting every person that has a mask, but it only resulted in an infinite loop of loading the image, so it did not work out.

4.4.1.11 Difficulties

Working with flutter was hard as a lot of errors occurred because of having outdated versions of packages, plugins, etc. so making sure everything is compatible with one another in flutter is essential. It is also crucial to install an android emulator which is done from installing android studio to test the app every time a change happens. Also, there is a lot of stuff that needs to be installed which was kind of time consuming. Installing flutter was hard as it was not working as intended when it was installed as it kept giving errors because of not detecting specific important items after searching in the internet for a long time and watching tutorials which led to doing fixes that managed to get flutter working in the end. It was also discovered that there was a specific command for flutter in the windows PowerShell to make sure that the important dependencies, packages and files were installed in order to get flutter working.

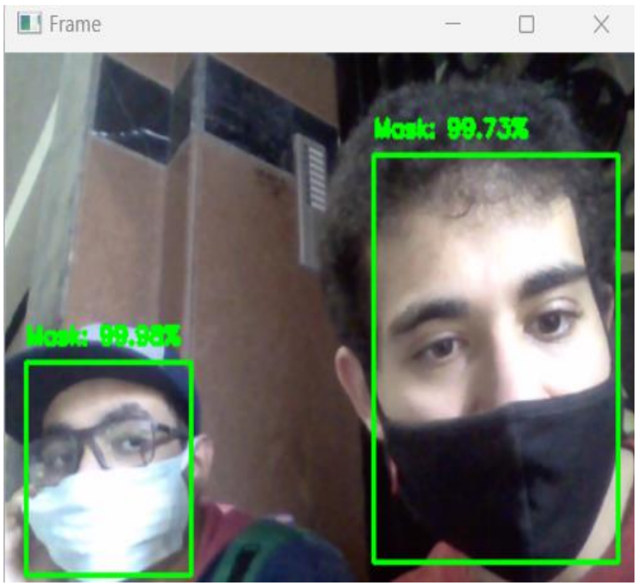
5.Testing and evaluation

5 Testing and evaluation

5.1 Testing

Testing the model is going to be through if it can achieve the functional requirements in the following table.

Functional Requirements	Explanation
Detect mask	This function tells us whether a person has his mask on or not
Drawing box	This function draws a box around a person's face
Multiple face detection	This function detects multiple faces in the frame at the same time



In the previous figure the model tells us if the mask is on or not so the first function is achieved, the model does indeed draw a box around a person's face, so the second function is done and lastly the model is detecting two faces in the frame at the same time so the last and third function is accomplished.

5.2 Evaluation

Imagine a person wearing a mask entering a building and security uses the face mask detection system which draws a box around the person’s face and flashes green with a label that says mask with the accuracy of how the person is wearing the mask. Security allows the person to pass.

6.Results and Discussions

6 Results and Discussions

6.1 The First Model

- The accuracy of this model is 95%.
- The Loss is 0.15.
- The Validation loss is 0.074.
- The Validation accuracy is 97%

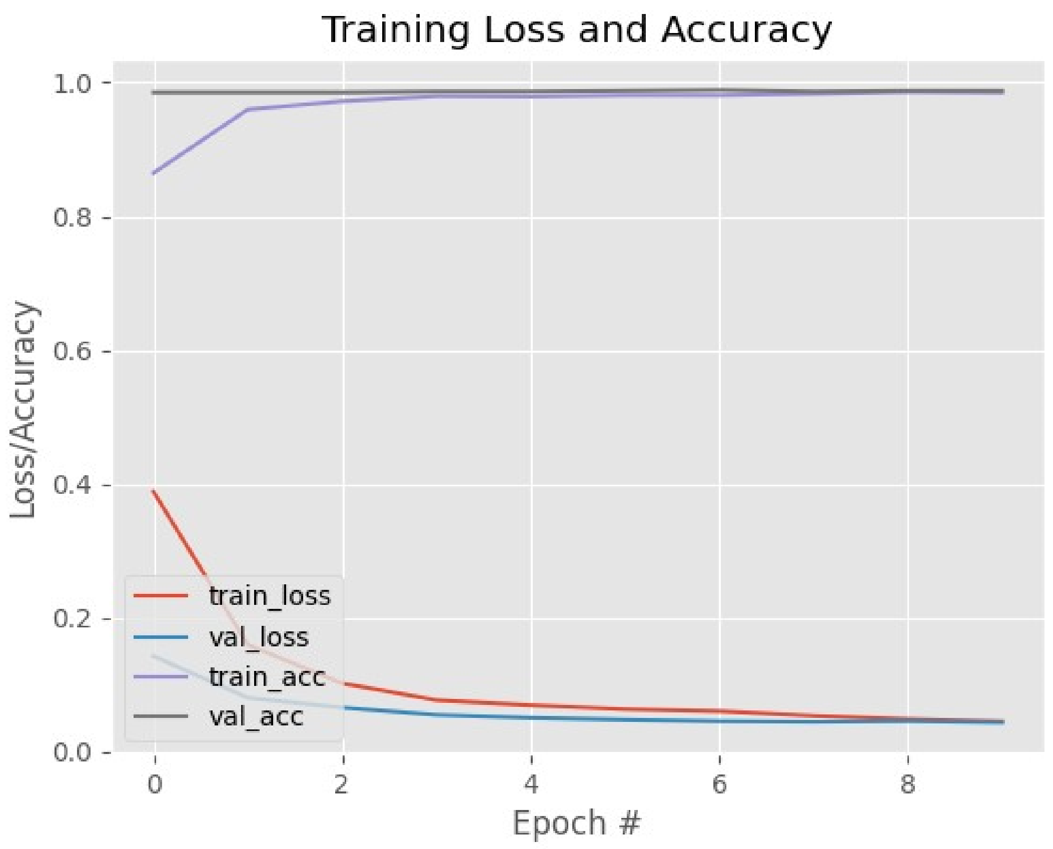
6.2 The Last Model

- The accuracy of this model is 98%.
- The Loss is 0.05.
- The Validation loss is 0.04.
- The Validation accuracy is 98%.

6.2.1 Visualization

The performance of the last model is going to be shown using different visualization methods.

6.2.1.1 Line Plot



This is a line plot showing the following:

- Training Loss
- Validation Loss
- Training Accuracy
- Validation Accuracy

6.2.1.2 Confusion Matrix

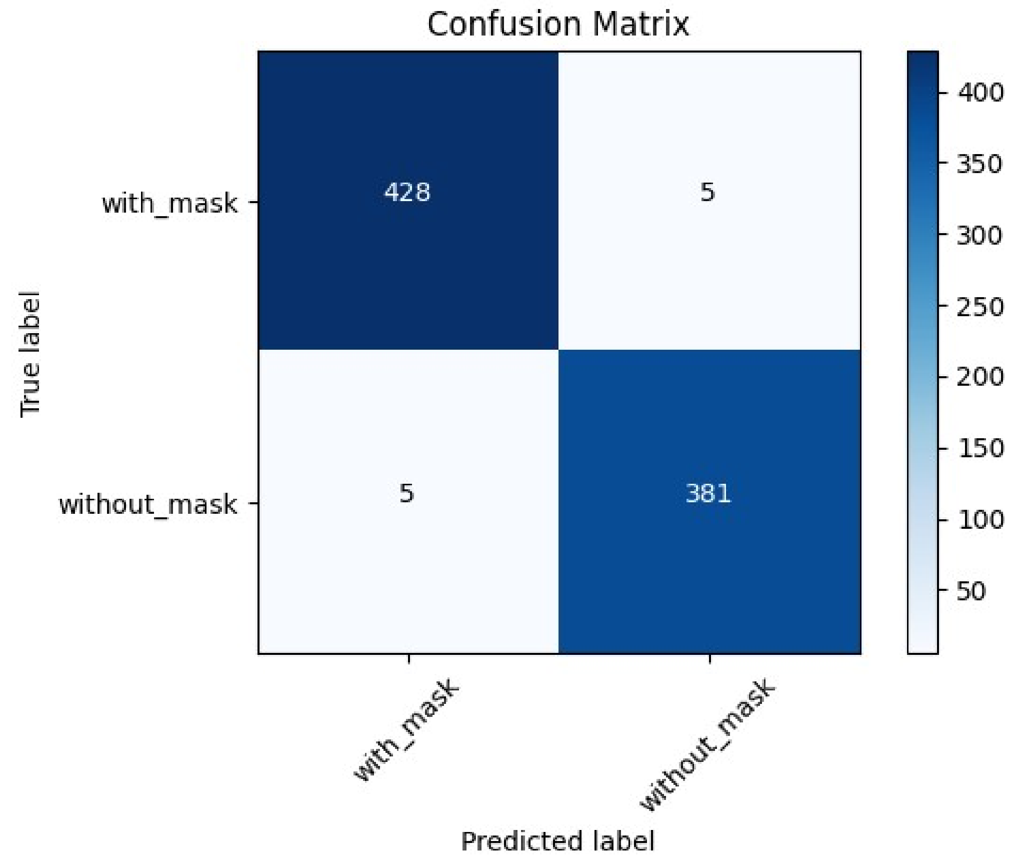
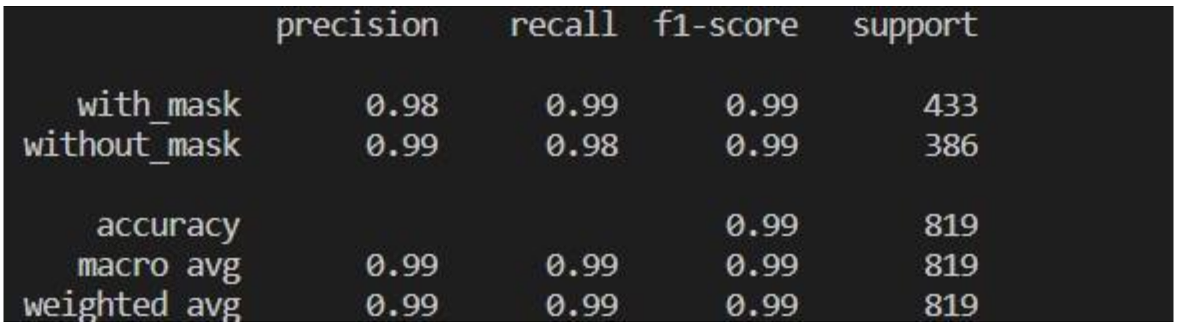


Figure 35: Confusion Matrix for the last model.

- The number of true positives is 428.
- The number of false positives is 5.
- The number of false negatives is 5.
- The number of true negatives is 381.

6.2.2 Classification Report



```
              precision    recall  f1-score   support

   with_mask       0.98      0.99      0.99       433
without_mask       0.99      0.98      0.99       386

    accuracy                           0.99       819
   macro avg       0.99      0.99      0.99       819
weighted avg       0.99      0.99      0.99       819
```

Figure 36: Classification Report of last model.

7.Conclusions and Future Work

7 Conclusions and Future Work

7.1 Summary

To conclude, the report talked about the problem, a background was given about the topic, a literature review was written about the efforts of the authors to make a decent model to detect face masks, the methodology on how the models were made was told accompanied by an explanation about every technique and how it works and lastly the implementation was provided with the results. In the end a model was built with an impressive 98% which was used in the creation of the mobile application.

7.2 Future Work

The model can be improved further by making it have the ability to classify masks. What this means is that it can differentiate between masks like N95, Kn95, ffp2 and KF94. Why is this important because a company might only want people with KN95 masks to enter the building not people with other types of masks. What can be improved in the application is the number of people with masks feature as it has a limited usage because it only outputs 0 and 1.

References

[1] A. Nowrin, S. Afroz, M. S. Rahman, I. Mahmud and Y. -Z. Cho, "Comprehensive Review on Facemask Detection Techniques in the Context of Covid-19," in IEEE Access, vol. 9, pp. 106839-106864, 2021, doi: 10.1109/ACCESS.2021.3100070.

[2] A.K.Bhadani, A.Sinha, "A FACEMASK DETECTOR USING MACHINE LEARNING AND IMAGE PROCESSING TECHNIQUES.,”.

[3] S. Sakshi, A. K. Gupta, S. Singh Yadav and U. Kumar, "Face Mask Detection System using CNN," 2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2021, pp. 212-216, doi: 10.1109/ICACITE51222.2021.9404731.

[4] K. R. Kavitha, S. Vijayalakshmi, A. Annakkili, T. Aravindhan, K. Jayasurya, “Face Mask Detector Using Convolutional Neural Network”, Annals of RSCB, pp. 1979–1985, May 2021.

[5] Kaur, G. et al. (2022) “Face mask recognition system using CNN model,” Neuroscience Informatics, 2(3), p. 100035. Available at: https://doi.org/10.1016/j.neuri.2021.100035.

[6] Pooja, S. and Preeti, S. (2021) “Face mask detection using AI,” Algorithms for Intelligent Systems, pp. 293–305. Available at: https://doi.org/10.1007/978-981-33-4236-1_16.

[7] A.VijiAmuthaMary, PatapatiMouryaVarma, P.A.Khan, A.Jesudoss, L.K.JoshilaGrace, "Real time face mask detection Using python,".

[8] Patil, M., Tiwari, P. and Sonkamble, R. (2022) “Covid-19 face mask recognition using live camera and face mask detection using tensorflow and keras,” International Journal for Research in Applied Science and Engineering Technology, 10(4), pp. 2778–2782. Available at: https://doi.org/10.22214/ijraset.2022.41909.

[9] A. Z. Azman, I. Zulaikha Saiful Bahri, S. Saon, A. K. Mahamad, M. Anuaruddin bin Ahmadon and S. Yamaguchi, "Masked Face Detection System Using Convolutional Neural Networks," 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), 2022, pp. 366-369, doi: 10.1109/LifeTech53646.2022.9754884.

[10] A. Das, M. Wasif Ansari and R. Basak, "Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV," 2020 IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342585.

[11] P.Maurya, S.Nayak, S.Vijayvargiya, M.Patidar, "COVID-19 Face Mask Detection,".

[12] Jauhari, A., Anamisa, D.R. and Negara, Y.D. (2021) “Detection system of facial patterns with masks in new normal based on the viola jones method,” Journal of Physics: Conference Series, 1836(1), p. 012035. Available at: https://doi.org/10.1088/1742-6596/1836/1/012035.

[13] Nieto-Rodríguez, A., Mucientes, M. and Brea, V.M. (2015) “System for medical mask detection in the operating room through facial attributes,” Pattern Recognition and Image Analysis, pp. 138–145. Available at: https://doi.org/10.1007/978-3-319-19390-8_16.

[14] M.Mohan, Sojasomanan, M.Kaleeswari, "Automatic Face Mask Detection Using Python,", vol.2 , issue.1, 2021.

[15] Nieto-Rodríguez, A., Mucientes, M. and Brea, V.M. (2015) ‘System for medical mask detection in the operating room through facial attributes’, *Pattern Recognition and Image Analysis*, pp. 138–145. doi:10.1007/978-3-319-19390-8_16.

[16] S. Abbasi, H. Abdi and A. Ahmadi, "A Face-Mask Detection Approach based on YOLO Applied for a New Collected Dataset," 2021 26th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, 2021, pp. 1-6, doi: 10.1109/CSICC52343.2021.9420599.

[17] G. Saranya, D. Sarkar, S. Ghosh, L. Basu, K. Kumaran and N. Ananthi, "Face Mask Detection using CNN," 2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT), Bhopal, India, 2021, pp. 426-431, doi: 10.1109/CSNT51715.2021.9509556.

[18] *Your machine learning and Data Science Community* (no date) *Kaggle*. Available at: https://www.kaggle.com/.