DESIGN AND IMPLEMENTATION OF A FACE MASK DETECTION SYSTEM

BY

OBIDINMA PAUL OZIOMA

(SCI/018/18794)

A PROJECT SUBMITTED

TO

COMPUTER SCIENCE PROGRAMME

COLLEGE OF COMPUTING AND COMMUNICATION STUDIES,

BOWEN UNIVERSITY, IWO, OSUN STATE NIGERIA.

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD

DEGREE OF BACHELOR OF SCIENCE (B.Sc.) IN COMPUTER SCIENCE

JULY, 2022

CERTIFICATION

This is to certify that this project**, DESIGN, AND IMPLEMENTATION OF A FACE MASK DETECTION SYSTEM** was carried out by Obidinma Paul Ozioma (Matriculation Number: SCI/018/18794) of the Computer Science Programme under my supervision

…… ……

Dr. A.O. Akinwunmi Date

(Supervisor)

…… ……

Dr. A.O. Akinwunmi Date

(Programme Coordinator)

DEDICATION

This project work is dedicated to the creator of heaven and the earth who is the source of all wisdom, knowledge, and understanding, the Almighty God.

ACKNOWLEDGEMENT(s)

I would like to use the opportunities to acknowledge and appreciate the sustaining power of the almighty God for wisdom and abundant understanding. I would also like to acknowledge the effort of my supervisor, Dr A.O. Akinwunmi for taking the time, love, and resources to guide me through this work.

With a deep sense of respect, appreciation, and gratitude, I would very much like to acknowledge the efforts of my lecturers in the Programme of Computer Science, Bowen University Dr A.O. Akinwunmi, Dr R.F Famutimi, Dr Mrs O.N. Emuoyibofarhe, Mr B.P Ayaniyi, Mr C. Agbonkhese, Dr D.O. Lanloye, Dr E.K. Olatunji, Dr T.O. Olorunfemi, Dr S. Adebayo, Dr M.O. Oyelami, Dr A.O. Abiodun, Dr A.O. Ibitoye, Dr H.O. Aworinde, Dr O.M. Awoniran, Mrs T.O. Okedigba, Mr A. Adeyemo and Mrs Olaniran for giving me a solid footing in the field of Computer Science and Information Technology.

I also appreciate my parents Mr and Mrs Obidinma for the financial, and emotional support, and prayers. To all my siblings, who were also supportive of me throughout this work. I cannot forget the contribution of all my coursemates and friends in the successful completion of this work, I am forever grateful. Thank you.

TABLE OF CONTENT

LIST OF FIGURES

ABSTRACT

The current scenario of the COVID-19 pandemic demands an efficient face mask detection application. The project's major objective is to develop and test a model that can be used at the entrances to schools, hospitals, places of worship, offices, and other locations where there is a need to control the use of face masks and where there is a higher risk of COVID-19 infection.

The purpose of this project is to develop and put into use a facemask detecting system. This entails gathering the dataset that the model needs in the form of pictures of people wearing and without wearing face masks, respectively. Additionally, it entails preparing the data, training the model, assessing the model's accuracy, and testing the model.

The machine learning and deep learning model used in the training of the model was MobileNetV2. The choose of model was purely based on the fact that the field of computer science aims to execute a given task as fast as possible and the system aims to achieve that with a satisfactory level of accuracy. It is also suitable for embedded vision applications.

The face mask detection system is the least complex in design and provides immediate results; as a result, it may be used in CCTV footage to determine whether a person is correctly wearing a mask so that he does not present a risk to others. This project can ensure that a person wears the face mask by keeping an eye on where it is placed on the face, which helps to limit the spread of the virus.

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

In late 2019, the highly contagious coronavirus disease (COVID-19) was first reported in Wuhan, infecting more than 430 million people and killing more than 5.94 people worldwide. (World Health Organization, 2022). Since then, it has become a public issue in China and even worldwide. This pandemic has devastating effects on societies and economies round the world causing a global health crisis. Therefore, to prevent rapid COVID-19 infection, many solutions such as confinement and lockdowns were suggested by the majority of the World's governments.

Consequently, the World Health Organization (WHO) advised that the wearing of face masks should be mandatory, and Nations, Academic institutions, and Organizations around the world followed the recommendation by making a law to administer this mandate. This brought about the problem with the enforcement of these laws.

Aristotle, the great Greek philosopher made it clear that humans, in general, are classified as social beings basically because they rely on cooperation to survive and thrive. Apart from that, we are also free-spirited beings. This is not an excuse to be rebellious but we need a more conventional way of enforcing the mandatory rule of wearing of face mask.

The COVID-19 pandemic has motivated the research community to aid front-line medical service staff with cutting-edge research for mitigation, detection, and prevention of the virus according to Boccaletti. S *et al.*, (2020). Artificial intelligence is already being applied in different sectors of

combating the COVID-19 pandemic. This is just another instance where the priceless ingenuity of AL can be used.

1.2 Statement of the Problem

Wearing face masks is essential in the fight against COVID-19. Rules and laws were made and enforced to a certain degree but due to various reasons such as human error or public apathy, these rules were not well imposed.

The computing age is meant to improve the efficiency of the current methods and with the introduction of a face mask Detection system, this issue can be solved with ease and with more proficiency.

1.3 Aim of the Study

This project aims to design and implement a system for facemask detection.

1.4 Objectives of the Study

The Objectives of the study are as follows:

i. Collection of the data needed for the model;
ii. train a model for facemask detection;
iii. test a model for facemask detection;
iv. evaluate a model for facemask detection.

1.5 Methodology

An extensive review was done on related topics and existing documented materials such as journals, books, and articles containing related information gathered which were examined and reviewed to retrieve essential data to better understand and know how to improve the system

This project involved the collection of a dataset that consists of 3833 images with 1915 images containing images of people wearing face masks and 1918 images with people without face masks obtained from open-source platforms such as Kaggle, a reputable online resource for data scientists. Kaggle offers a variety of purposes, but when it comes to datasets, it offers a number of formats.

MobileNetV2 was used as deep learning architectures to develop an efficient facemask detection network. The project has been implemented in a Python notebook. Libraries such as Pandas and NumPy were used. To train the model and run the python code for this project the following libraries with the given or higher version were required: TensorFlow 1.15.2, Keras 2.3.1, NumPy 1.18.2, SciPy 1.4.1, etc.

The model evaluation was done immediately after training the model on python notebook. The MathPlotLib library was used to show the training accuracy and loss in the graphical form. The model testing was done at the last stage of the model development and is the most significant part of the project. The libraries required were tensorflow, imutils, and OpenCV-python 4.2.0. It also required a webcam.

1.6 Significance of the Project

With every day that goes by, the world becomes more and more technologically inclined. Because of this, people are taking advantage of this in problem-solving and the issue of wearing face masks needs to be solved in the same manner.

Creating a face mask detection system is therefore very relevant and allows and aims for simplicity and efficiency. If we also consider the cost estimation for implementing the project, it will be almost no cost as most of the infrastructures and organizations around the world already have a camera installed in public places.

1.7 Scope of the Project

Indeed, several case studies have been enrolled to demonstrate the real- time script of the COVID-19 issue but the deployment of the systems in real- time is veritably delicate. Developing a system that's adaptive to all surrounds and surroundings is getting a difficulty. This system can thus be used in real-time operations that demand face mask detection for safety purposes due to the outbreak of Covid- 19. This project can be integrated in systems for operation in airfields, road stations, services, seminaries, and public places to ensure that public Covid-19 safety guidelines are followed.

1.8 Structure of the Project

The Project report is organized into five Chapters. It begins with Chapter one which handles the introduction. After that, comes Chapter two which handles the survey of the literature review. Chapter three handles the project methodology. The chapter focuses on the implementation, result, and testing of the system and lastly, Chapter five handles the conclusion and recommendation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

The goal of this chapter is to provide an extensive explanation of the subject to allow readers to gain a better understanding of a face mask detection system, as well as how existing technologies affected and benefitted my research. The review's goal is to identify trends, identify potential gaps in the literature for the topic, provide a conceptual framework for the project, and demonstrate knowledge of the chosen sector. Journals, websites, papers, thesis/published materials, and blogs are among the sources included in this review.

2.2 Historical Background

2.2.1 Face Masks

A face mask is defined according to the Oxford Learner's Dictionary, as something that you wear over part or all over your face, to protect it or prevent it from diseases. Face masks vary in type and they are used in different aspects of life and used by different people from surgeons, Nurses, and health care professionals to construction workers like carpenters, masons, and plumbers. They are more or less used by everyone on earth right now due to the ongoing COVID-19 pandemic which is a testament to their relevance in our everyday life.

Much awareness has been given to the role of the use of face masks in helping control the COVID-19 pandemic. The concept of universal masking has been debated at length since the early phases of the pandemic. The WHO recommends that every person should wear a mask. Most countries and organizations approve it by setting rules and laws to issue it but others still believe it to be

hearsay. Whether we like it or not, we as a society, have to adhere to this advice or rules for our wellbeing.

Figure 2.1: Diagram of a face mask (lifecycleinitiative.org, 2022)

2.2.2 Origin of Face masks

Although face masks have become a symbol of our times, they have been around for many years. In 1897, Dr Carl Georg Friedrich Wilhelm Flügge, a prominent bacteriologist and hygienist in Germany, developed the droplet theory of infection. His theory revolved around the idea that microorganisms in droplets expelled from the respiratory tract are a means of transmission. That same year, Dr Johann Freiherr von Mikulicz-Radecki, a Polish surgeon, proposed that one layer of gauze could serve as what is now known as a surgical mask. In 1898, Dr W. Huebner recommended masks made of two layers of gauze to be worn during operations, stating this mask was more efficient (ormanagement.net, 2021). Over the last century, medical researchers have continued to experiment with designs and materials (clinicaloncology.com, 2021).

Today, the COVID-19 pandemic has brought a huge surge in the number of patients, a shortage of health care personnel, and limited bed space, and the contribution of an efficient surgical mask plays a crucial role in helping to prevent COVID-19 transmission. Surgical masks not only help us protect ourselves and others by reducing overall viral transmission, but they function as a visual aid to intensify diligent hand hygiene and social distancing. Surgical masks also represent an act of solidarity, displaying that the citizens of the world are on board with the preventive measures needed to control COVID-19 (clinicaloncology.com, 2021).

2.2.3 Uses/Application of Face masks

Earlier, we saw that face masks are used in different aspects and sections of our everyday life and are used in different jobs/occupations today. This, therefore, means that they come in different

shapes and sizes for the different situation but how can we narrow it down. Here are some scenarios where a face mask is used.

A very easy and common example would be doctors, nurses, surgeons, and other medical practitioners. Anesthesiologists especially use a type of face mask called a gas mask while administrating treatment to patients for surgical operations. They use the gas mask on the patient, delivering anaesthesia through the apparatus to prevent the patient from feeling pain during the surgery and place them into a comatose state. Regular and general surgeons, doctors, and even dentists make use of disposable face masks to decrease the risk of infection and contamination during operations.

Carpenters, masons, and others that work in conditions that cause a lot of dust and debris to fill their air also wear disposable face masks to minimize breathing in potentially dangerous particles. The list goes on from waste removal workers, hazardous materials removal workers, and firefighters to even cooperative workers, civil officers, and students, the relevance of face masks spans throughout our everyday life.

2.3 Conceptual Review

2.3.1 Machine Learning

Machine learning is a branch of artificial intelligence (AI) that allows systems to research and enhance from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can get admission to facts and use them to examine on their own. The studying method begins with observations or data, such as examples, and direct experience, to find patterns in the data and make a higher and independent decision in the future

based totally on the examples provided or given. The primary intention is to allow the computer to analyze robotically without the assistance or interference of humans.

2.3.2 Machine Learning Model

A machine learning model can be a mathematical representation of a real-world process. To generate a machine learning model, you will need to provide training data to a machine-learning algorithm to learn from (medium.com, 2017).

Machine learning models are homogeneous to functions that predict some output for a given input.

Different machine learning architectures are required for each purpose. A car is a car that takes you to work or an expedition, a tractor pulls a plough, and an 18-wheeled vehicle transports a lot of goods. Each machine learning model is used for a variety of purposes. One is used to classify images, the other is used to predict the next item in the sequence, and the other is suitable for sorting the data into groups. Some are suitable for multiple purposes, while others are only one. Machine Learning algorithms can assist computers in playing chess, performing surgery, and improving their intelligence and personalization. We live in an era of ongoing technological advancement, and by looking at how computers have progressed over time, we may forecast what will happen in the future. One of the most notable aspects of this revolution is the democratization of computer tools and processes. Data scientists have constructed sophisticated data-crunching machines in the last five years by effortlessly performing modern procedures. The outcomes have been spectacular (Wakefield, 2022).

There are four types of machine learning algorithms:

i. Supervised Learning

ii. Unsupervised Learning

iii. Semi-supervised Learning

iv. Reinforcement Learning

2.3.3 Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before (medium.com, 2022).

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained using a large set of labelled data and neural network architectures containing many layers (MathWorks.com, 2020). Figure 2.2 illustrates when deep learning falls down in machine learning and in the world of artificial intelligence.

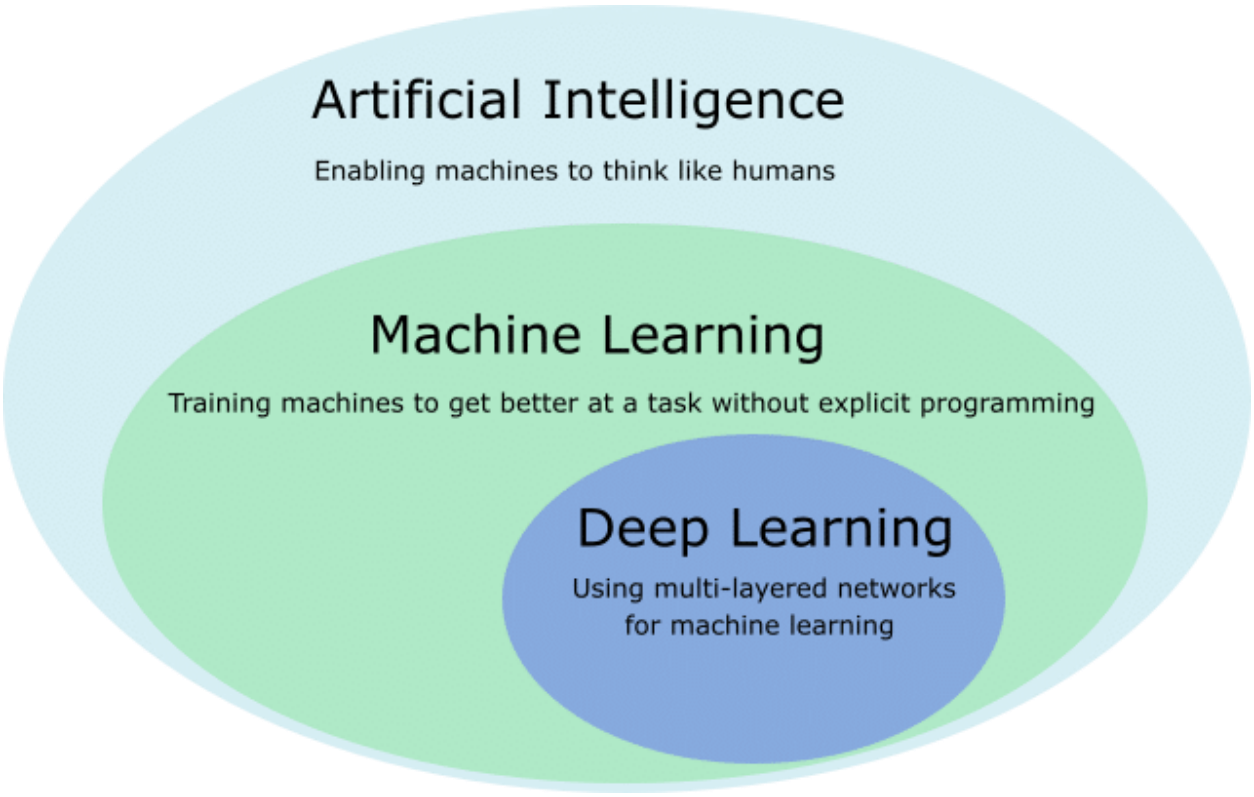


Figure 2.2:Machine learning Illustrated

2.3.4 Machine Learning Methods

Machine learning has four methods which are the supervised machine learning algorithm, semi-supervised machine learning algorithm, reinforcement machine learning algorithm and unsupervised machine learning algorithm.

Deep learning uses supervised learning in situations such as image classification or object detection, as the network is used to predict a label or a number (the input and the output are both known). As the labels of the images are known, the network is used to reduce the error rate, so it is “supervised” (ai-med.io, 2021).

2.3.5 Object detection using Deep Learning

Object detection is the problem of finding and classifying a variable number of objects on an image. It can also be defined as a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results (mathworks.com, 2022). The important difference is the “variable” part. In contrast with problems like object classification, the output of object detection is variable in length, since the number of objects detected may change from image to image (tryolabs.com, 2022).

2.3.6 Overview of Convolutional Neural Network Models (CNN)

Deep Learning – which has emerged as an effective tool for analyzing big data – uses complex algorithms and artificial neural networks to train machines/computers so that they can learn from experience, classify and recognize data/images just like a human brain does. Within Deep

Learning, a Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN. CNNs are playing a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self-driving cars, as well as speech recognition in natural language processing. As CNNs are playing a significant role in these fast-growing and emerging areas, they are very popular in Deep Learning. (Happiest Minds, 2022).

In deep learning, convolutional neural networks (CNN / ConvNet) are the most commonly used class of deep neural networks for the analysis of visual images. When we think of neural networks, we think of matrix multiplication, but not with ConvNet. It uses a special technique called convolution. In mathematics, convolution is a mathematical operation on two functions, producing a third function that describes how one shape is modified by the other. But you don't have to go through the math to understand what a CNN is or how it works. In conclusion, the role of ConvNet is to put the image in an easy-to-process format without losing the essential functionality for proper prediction. (Mandal, 2021).

2.3.7 MobileNet

In Keras, MobileNet resides in the applications module. Keras offers out-of-the-box image classification using MobileNet if the category you want to predict is available in the ImageNet categories. If the category doesn’t exist in ImageNet categories, there is a method called fine-tuning that tunes MobileNet for your dataset and classes (gogul.dev, 2018).

MobileNet offers tons of advantages over other state-of-the-art convolutional neural networks such as VGG16, VGG19, ResNet50, InceptionV3 and Xception.

MobileNets are lightweight deep neural networks best suited for mobile and embedded vision applications.

MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions.

MobileNet uses two simple global hyperparameters that efficiently trade-off between accuracy and latency.

MobileNet could be used in object detection, fine grain classification, face recognition, large-scale geo localization etc. (gogul.dev, 2018).

Following are the advantages of using MobileNet over other state-of-the-art deep learning models.

i. Reduced network size - 17MB.
ii. Reduced number of parameters - 4.2 million.
iii. Faster in performance and are useful for mobile applications.
iv. Small, low-latency convolutional neural network.

Advantages always come up with some disadvantages and with MobileNet, it's the accuracy. Yes! Even though MobileNet has reduced size, reduced parameters and performs faster, it is less accurate than other state-of-the-art networks as discussed in this paper. But don't worry. There is only a slight reduction in accuracy when compared to other networks (medium.com, 2019).

2.3.8 MobileNetV2

The new mobile architecture, MobileNetV2 is the improved version of MobileNetV1 and is released as a part of TensorFlow-Slim Image Classification Library. Developers can even access

it in Collaboratory or can download the notebook and explore it using Jupyter. It is also available as modules on TensorFlow-Hub. The pre-trained checkpoints can be found on the open-source platform GitHub (analyticsindiamag.com, 2018).

MobileNetV2 is a convolutional neural network architecture aimed at improving performance on mobile devices. It is based on a reverse residual structure where the residual connections are between the bottleneck layers. The intermediate extension layer uses lightweight depth convolution to filter features as a source of non-linearity. Overall, the MobileNetV2 architecture includes the first complete convolution layer with 32 filters, followed by the remaining 19 bottleneck layers.

MobileNetV2 is very similar to the original MobileNet, except that it uses inverted residual blocks with bottlenecking features. It has a drastically lower parameter count than the original MobileNet. MobileNets support any input size greater than 32 x 32, with larger image sizes offering better performance. It is also a very effective feature extractor for object detection and segmentation. For instance, for detection, when paired with Single Shot Detector Lite, MobileNetV2 is about 35 per cent faster with the same accuracy than MobileNetV1. Figure 2.3 shows the overview of MobileNetV2.The blue blocks represent composite convolutional building blocks as shown above (docs.w3cub.com, 2020).

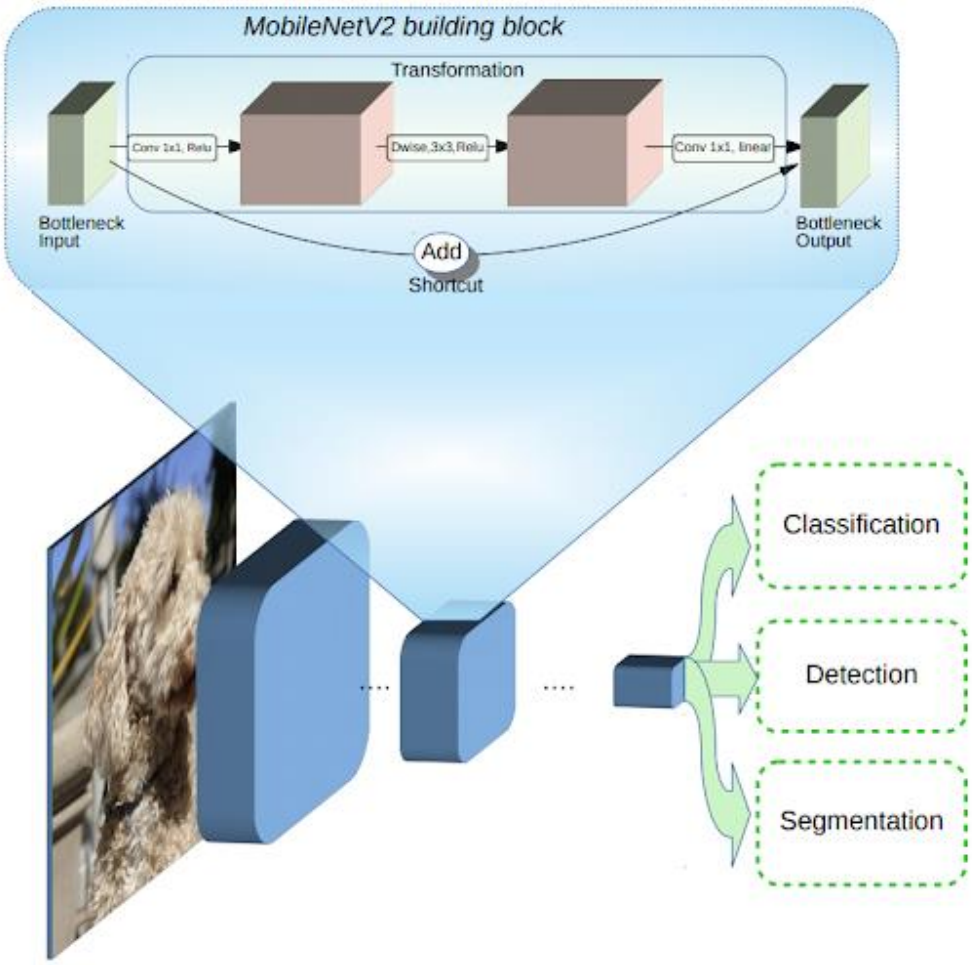


Figure 2.3 Overview of MobileNetV2 Architecture.

2.3.9 Advantages of MobilenetV2

The MobileNetV2 models are much faster in comparison to MobileNetV1. It uses 2 times fewer operations, has higher accuracy and reduced latency as shown on figure 2.4, needs 30 percent fewer parameters and is about 30-40 percent faster on a Google Pixel phone. It allows very memory-efficient inference and utilizes standard operations present in all neural frameworks. For the ImageNet dataset, MobileNetV2 improves the state of the art for a wide range of performance points. For object detection tasks, it outperforms real-time detectors on COCO datasets (analyticsindiamag.com, 2018).

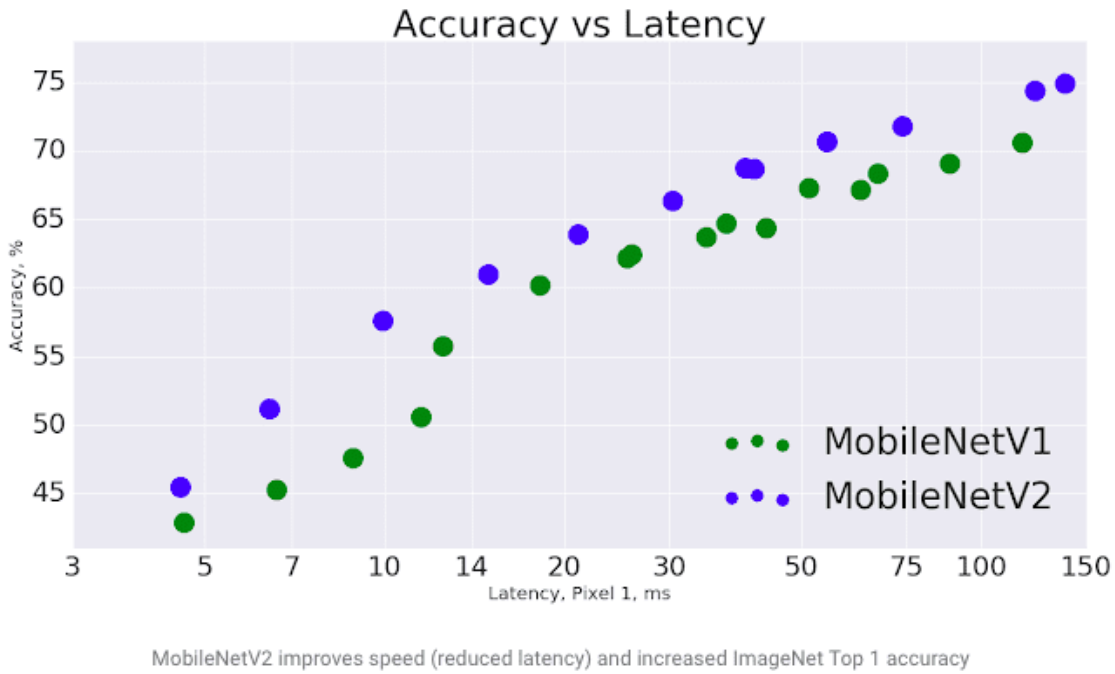


Figure 2.4: MobileNet vs MobileNetV2

2.4 Review of previous works

(Rahman, M. M., Manik, M. M. H., Islam, M. M., Mahmud, S., Kim, J. H., 2020) published a document aimed at developing a system for determining whether a person uses a mask or not and informing the relevant authority in the smart city network. It makes use of real-time filming of various public places of the city to capture facial images. The facial images extracted from this video are being used to identify the masked faces. The convolutional neural network (CNN) learning algorithm extracts features from images, after which those features are learned through multiple hidden layers. Whenever the architecture identifies people without a mask, this information is passed through the city network to the appropriate authority to take the necessary actions. The proposed system assessed promising results based on data collected from various sources. These documents also set out a system that can ensure proper law enforcement against people who do not follow basic health guidelines in this pandemic situation.

(T. Meenpal, A. Balakrishnan, A. Verma, 2019) has introduced semantic segmentation, a model for face detection used in an image by categorizing each pixel into face and non-face. It efficiently creates a binary classifier and then recognizes that fragment into chunks. The design allows you to create accurate face masks for human objects from RGB images containing localized objects. The author demonstrated the results on the Multi Human Parsing Dataset with an average accuracy at the pixel level. In addition, the problem of erroneous predictions is resolved and the correct bounding box is drawn around the segmented area. The proposed network can detect non-frontal faces and multiple faces in one image. The method can be used for complex tasks such as detecting parts of the face.

(Vinitha, V., & Velantina, V., 2020) published one article in which, using a deep learning algorithm and computer vision, they proposed a system that focuses on how to distinguish a person with a masked face in an image/video stream. Libraries like Tensor flow, Open CV, Keras, and PyTorch are being used. The project is being implemented in two stages. Phase one consists of training a deep learning model followed by the second phase where a mask detector is applied to a live image/video stream. OpenCV is the framework used to do real-time face detection from a live stream via a webcam. With computer vision using Python, a COVID-19 face mask detector has been built using a dataset.

(Bhuiyan, Khushbu, Islam ,2020) have published a paper in which the proposed system aims for recognizing the masked and faces are rendered using the advanced YOLOv3 architecture. YOLO (You Only Look Once), uses the learning algorithm Convolution Neural Network (CNN). YOLO establishes a connection with CNN through hidden layers, through research, easy algorithm retrieval, and can detect and locate any type of image. Execution begins by taking 30 unique images of the dataset into the model after combining the results to derive action-level predictions. It gives excellent imaging results and also good detection results. This model is applied to a live video to check the fps rate of the model inside the video and its detection performance with masked/unmasked two layers. Inside the video, our model has impressive outputs with an average fps of 17. This system is more efficient and faster than other methods using their own data set.

CHAPTER THREE
SYSTEM ANALYSIS AND DESIGN

3.1 Overview of System Analysis

This chapter gives details about the various methods, processes and procedures that are adopted by the researcher to achieve both the aims and the objectives and also the conceptual structure in which the research was performed. The methodology of any research work refers to the approach taken up by the researcher to solve a given problem. Since the maintenance and efficiency of any application are dependent on how the designs are worked on. This chapter will give a deep insight into the methods applied to find solutions to the stated objectives of the research.

3.2 Analysis of Existing System

In the early stages of the covid-19 lockdown, when it was made mandatory to put on a face mask before leaving your house, there was no rigid system to check if the mandate was obeyed. You were simply checked physically before entering a public or social gathering. Some time went by and great minds especially in the field of AI began to put their minds and ingenuity toward creating a solution to this problem before us as we have done since the dawn of time. Sure enough, they did come up with solutions but these solutions had some drawbacks. Some of the models that were developed lacked a bit of accuracy in their detection and others were not efficient and easy to deploy into embedded systems.

3.3 Proposed System

This proposed face mask detection utilizes a machine learning model called MobileNetV2. The model was selected after different comparison-based performances of multiple machine learning algorithms. The model is accurate, and since the MobileNetV2 architecture is used, it’s also computationally efficient, faster and thus making it easier to deploy the model to embedded

systems (Raspberry Pi, Google Coral, etc.). the model undergoes training and testing on a legitimate dataset extracted from trusted sources.

3.4 Benefits of the proposed system

i. This new system will be helpful in reducing costs during development and deployment.
ii. Satisfactory levels of accuracy are achieved.
iii. The system developed can be deployed on multiple platforms including Google mobile platforms.
iv. It will reduce the level of human error.
v. It is faster in comparison to previous systems and fast in general.
vi. It allows very memory-efficient inference.

3.5 Model Development

The model development involves collecting of data/dataset, data pre-processing, model training, running and viewing accuracy and model testing. Figure 3.1 and Figure 3.2 below illustrates the steps involved in developing the system.

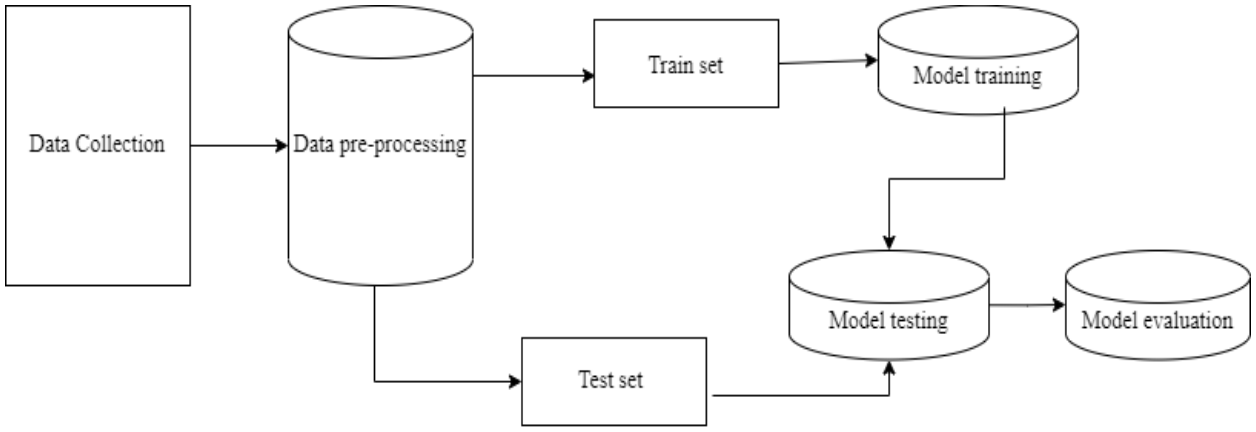


Figure 3.1: Model Development

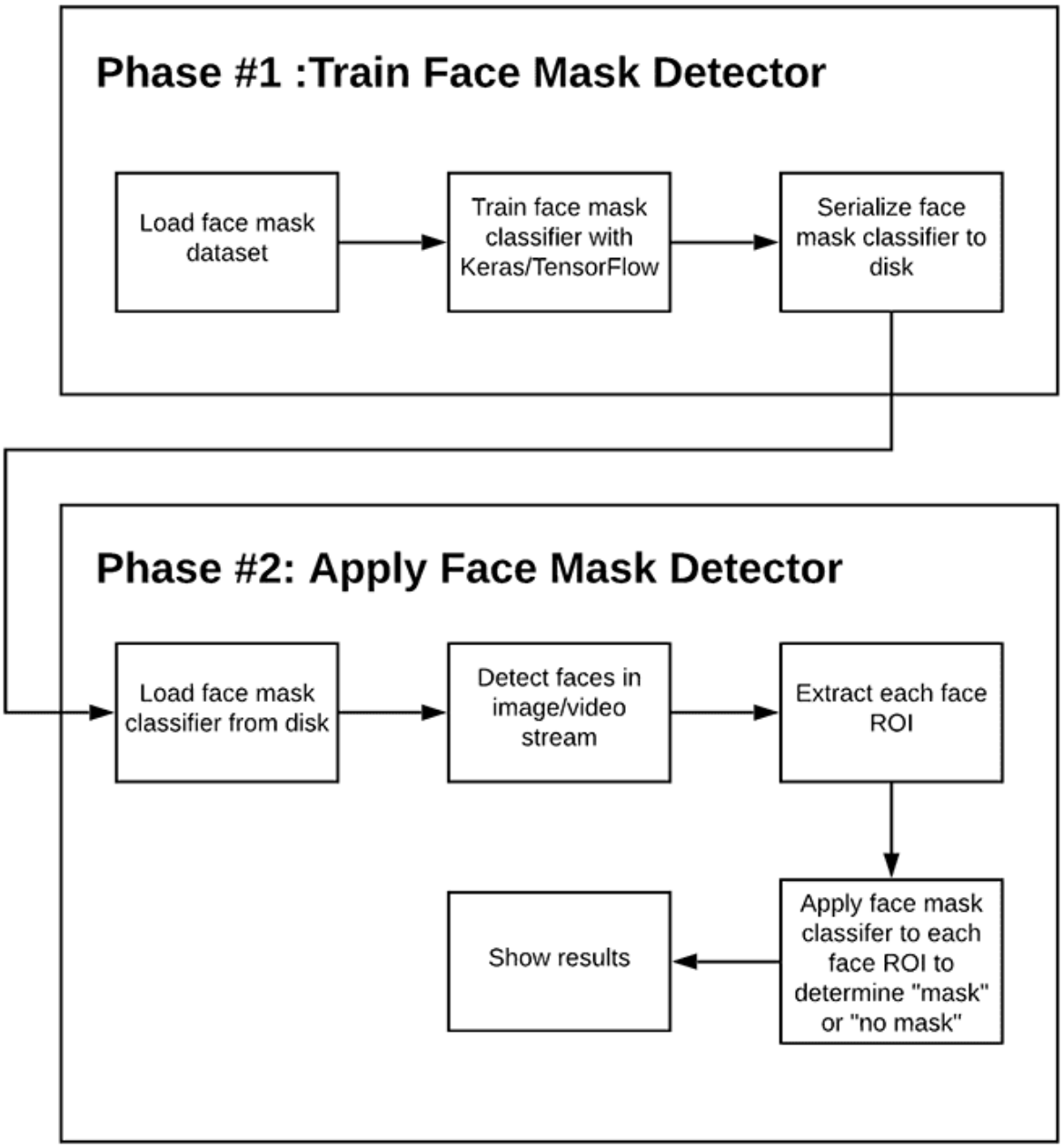


Figure 3.2 The block diagram of required steps for training and testing the proposed approach

3.5.1 Data collection

The data used to generate the datasets on which the models are trained are gotten from an open-source platform. The dataset collection is grouped into images of people with facemasks and images of people without facemasks.

These datasets came from Kaggle, a reputable online resource for data scientists. Kaggle offers a variety of purposes, but when it comes to datasets, it offers a number of formats, however, in this case, the dataset came in form of compressed (zipped) folders containing the categories needed for the system.

3.5.2 Data pre-processing

Data preprocessing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results. The library used in this step is called Keras (tensorflow.keras.preprocessing.Image) found in TensorFlow python. This process involves converting the images from the directory i.e., with mask and without mask into arrays. Two lists were also created namely; data and labels. The data list contains all the images in the array and the label list contains the categories the data can fall under. After that, the target size of the images was chosen and the preprocess module in the Keras library was used to preprocess the data. The next thing done was to initialize the initial learning rate, number of epochs to train for and the batch size. After that, grab the list of images in our dataset directory, then initialize the list of data (i.e., images) and class images. Next, perform one-hot encoding on the labels. One hot encoding is a process by which categorical variables are

converted into a form that could be provided to ML algorithms to do a better job in prediction in this case, binary value of 1s and 0s. After that, training and testing data were separated from the dataset.

3.5.3 Model training

In order to help identify and/or learn qualities from, model training entails providing data to machine learning algorithms. The programming language of choice for this model was python and the necessary packages and libraries were imported and installed. The model was trained using a combination of different tools and frameworks namely; tensorflow, keras, sklearn, imutils and numpy. These variety of python frameworks and libraries were used to minimize development time and to help solve common programming tasks.

By default, 80% of the data is designated for use in training. The next thing to do is to construct the training image generator for data augmentation i.e., Increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data.

The next step involves load the MobileNetV2 network, ensuring the head FC layer sets are left off. Then, construct the head of the model that will be placed on top of the base model and place the head FC model on top of the base model (this will become the actual model to train). After that, loop over all layers in the base model and freeze them so they will not be updated during the first training process. Then, the model is compiled and the head of the network is trained.

After that, make Predictions on the testing set and for each image in the testing set we need to find the index of the label with corresponding largest predicted probability. The last and final step is to serialize the model to disk. This is the process of saving and loading a trained model.

3.5.4 Model testing

Model testing is the procedure through which completely trained data is examined to demonstrate the model's correctness. We only have 20% of the data left over to assess the effectiveness of each of the used models because, by default, 80% of the data is utilized for training. The model testing involved the following steps or processes:

i. import the necessary packages.
ii. grab the dimensions of the frame and then construct a BLOB from it. A BLOB (Binary Large Object) is a collection of binary string that can be up to 2,147,483,647 characters long stored as a single entity.
iii. pass the blob through the network and obtain the face detections.
iv. initialize our list of faces, their corresponding locations, and the list of predictions from our face mask network.
v. loop over the detections.
vi. extract the confidence (i.e., probability) associated with the detection.
vii. filter out weak detections by ensuring the confidence is greater than the minimum confidence.
viii. compute the (x, y)-coordinates of the bounding box for the object.
ix. ensure the bounding boxes fall within the dimensions of the frame.
x. extract the face ROI, convert it from BGR to RGB channel ordering, resize it to 224x224, and preprocess it.
xi. add the face and bounding boxes to their respective lists only make a prediction if at least one face was detected.
xii. only make a prediction if at least one face was detected

xiii. for faster inference we'll make batch predictions on all faces at the same time rather than one-by-one predictions in the above for loop.

xiv. return a 2-tuple of the face locations and their corresponding locations.

xv. load our serialized face detector model from disk.

xvi. load the face mask detector model from disk.

xvii. initialize the video stream and loop over the frames from the video stream.

xviii. grab the frame from the threaded video stream and resize it to have a maximum width of 400 pixels.

xix. detect faces in the frame and determine if they are wearing a face mask or not.

xx. loop over the detected face locations and their corresponding locations.

xxi. unpack the bounding box and predictions.

xxii. determine the class label and color we'll use to draw the bounding box and text

xxiii. include the probability in the label.

xxiv. display the label and bounding box rectangle on the output frame.

xxv. show the output frame.

xxvi. if the `q` key was pressed, break from the loop.

xxvii. do a bit of cleanup.

3.5.5 Model evaluation

Model evaluation helps us to determine if a model works well or not, it is necessary to estimate the generalization accuracy of the model. To test the classification performance and appropriately assess the deployed model, the Sci-kit Learn module was utilized to create a number of scores and utility functions and the mathplotlib module was utilized to plot the training loss and accuracy.

CHAPTER FOUR
SYSTEM IMPLEMENTATION AND RESULTS

4.1 Installation Requirements

The hardware (physical components of a computer system that can be seen, felt or touched) and software (both system and the application software installed and used in the system development) requirements needed to achieve the previously stated project objectives are highlighted below.

4.1.1 Hardware Requirements

The hardware requirements include the following:

i. A laptop or desktop computer (preferably 64-bits)
ii. A webcam
iii. RAM: Eight gigabytes minimum
iv. Processor: Intel Core i5, 2.4 GHz minimum

4.1.2 Software Requirements

The software requirements include the following:

i. Operating system: Windows 10 or higher
ii. Python (Installed)
iii. An open-source integrated development environment (IDE) such as Pycharm, Jupyter notebook etc.

4.2 Model Development

The machine learning model used in this project was MobileNetV2. The development of this model was done in different stages namely; data collection, data pre-processing, model selection, model training, model evaluation and model testing which are shown below. The model detects whether or not a person or a group of people are wearing a facemask or not.

4.2.1 Data Collection

The data used in this project was gotten from Kaggle.com listed in the earlier stated methodology. The dataset is categories into two types namely; images of people wearing facemask and images of people without facemask which are shown below in Figure 4.1 and Figure 4.2 respectively.

4.2.2 Data Pre-processing

This aspect of the model development is illustrated in Figure 4.3 and Figure 4.4.

4.2.3 Model training

This aspect of the model development is illustrated in Figure 4.5 and Figure 4.6.

4.2.4 Model evaluation

This aspect of the model development is illustrated in Figure 4.7 and Figure 4.8.

4.2.5 Model testing

This aspect of the model development is illustrated in Figure 4.9, Figure 4.10 and Figure 4.11.

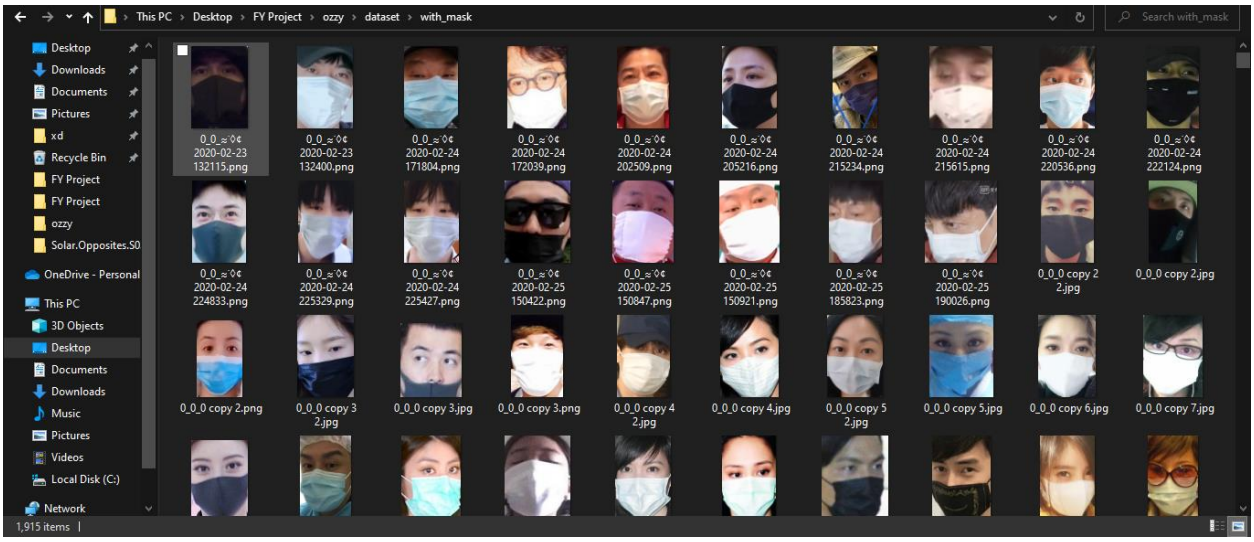


Figure 4.1:Capture of the dataset (with face mask)

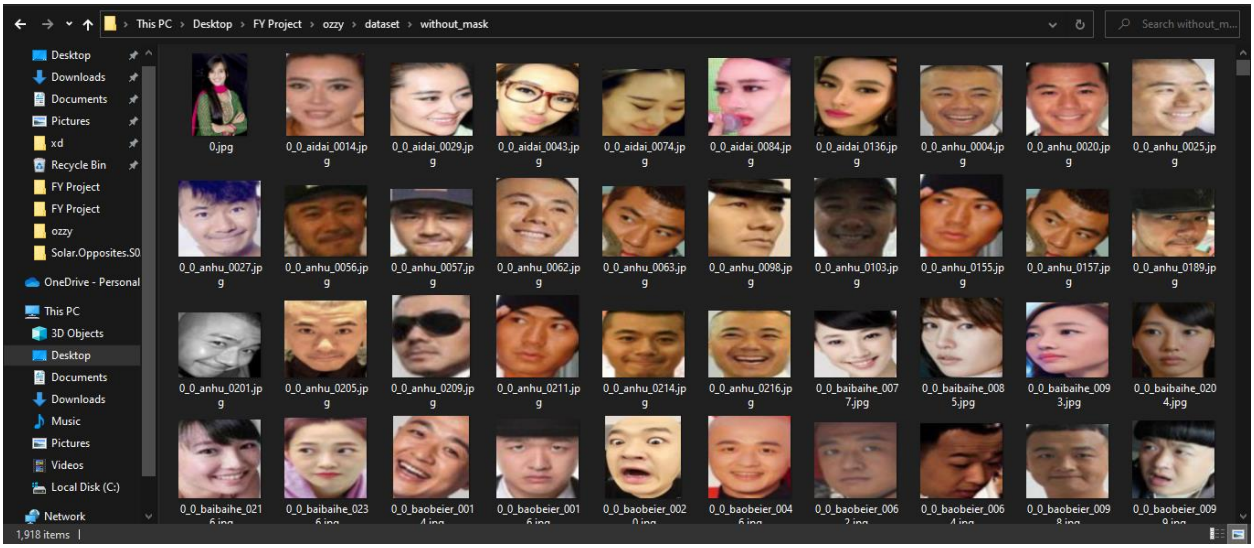


Figure 4.2: Capture of the dataset (without face mask)

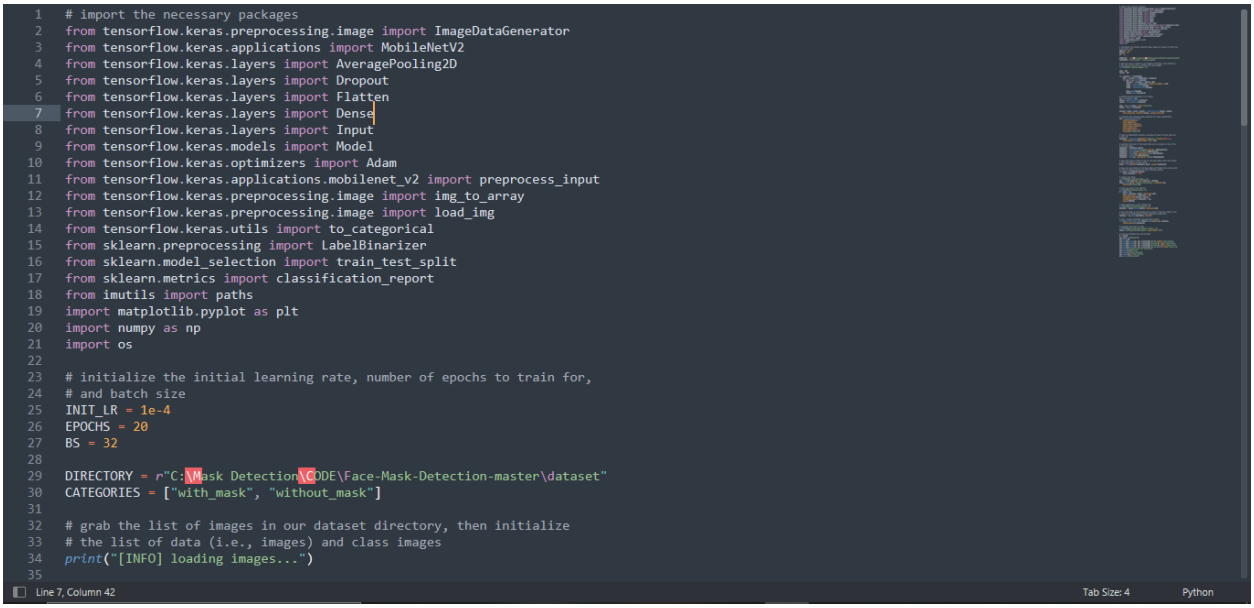
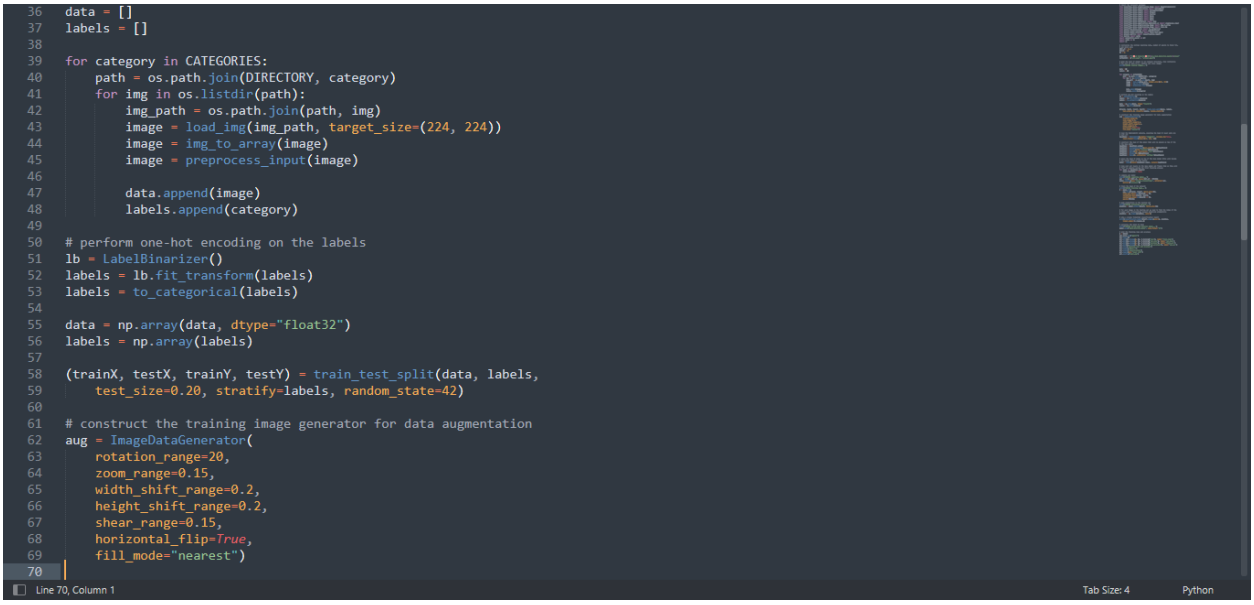


Figure 4.3: Libraries employed



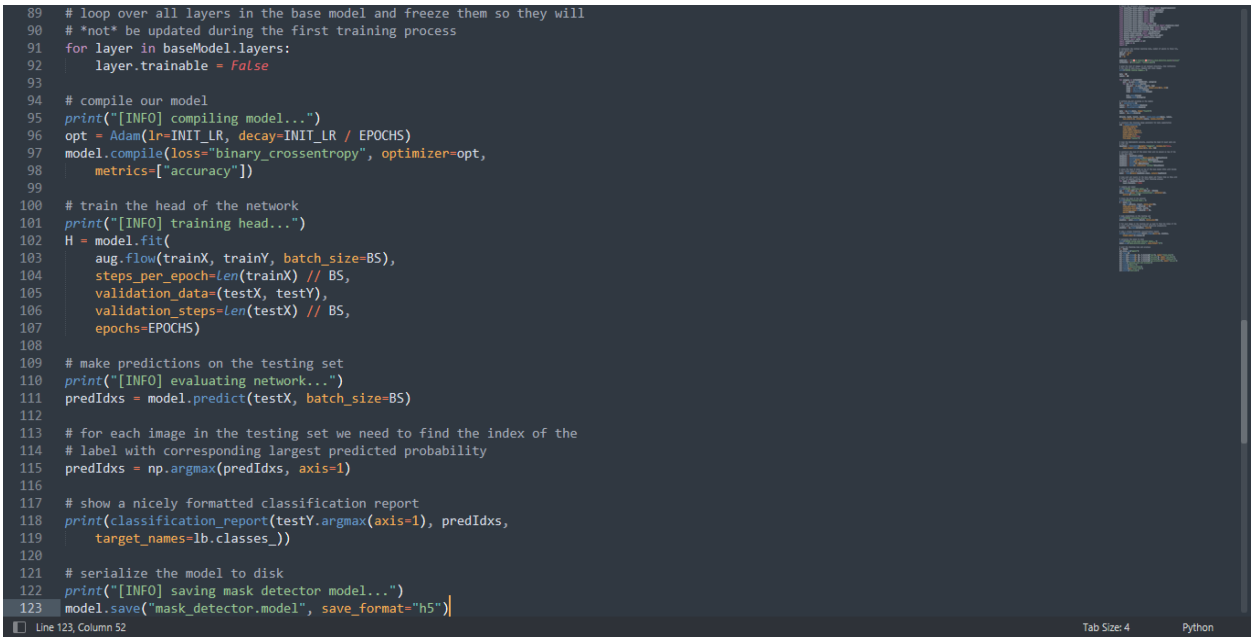
```
data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(224, 224))
        image = img_to_array(image)
        image = preprocess_input(image)

        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

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

# construct the training image generator for 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")
```

Figure 4.4 Data pre-processing

```
# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
```

Line 105, Column 36 Tab Size: 4 Python

Figure 4.5: Model training



```
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")
```


Figure 4.6: Model training

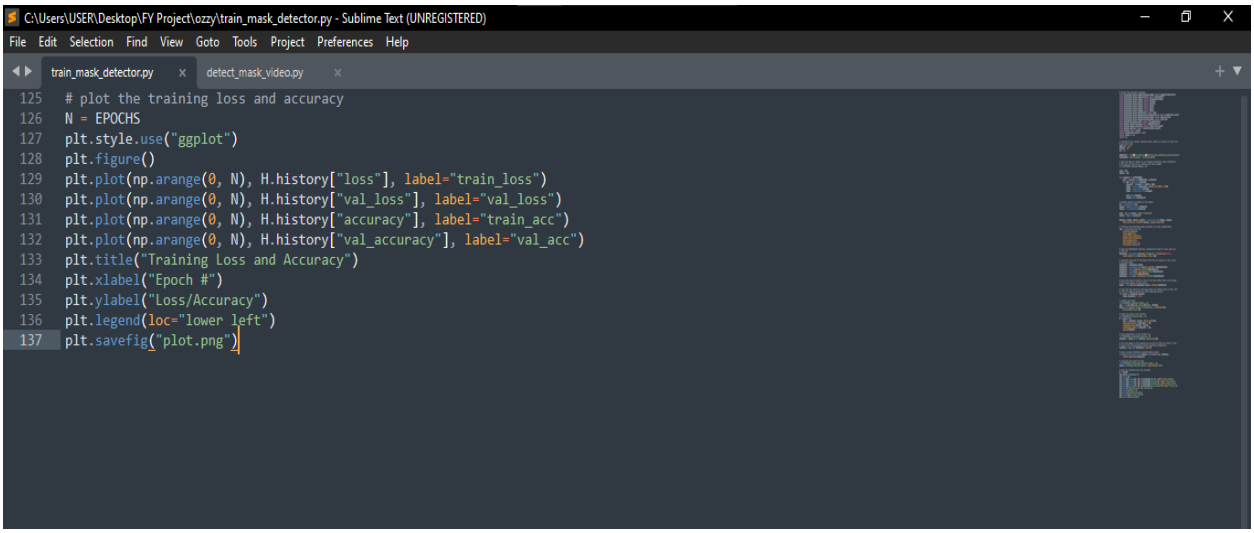


Figure 4.7: Model evaluation

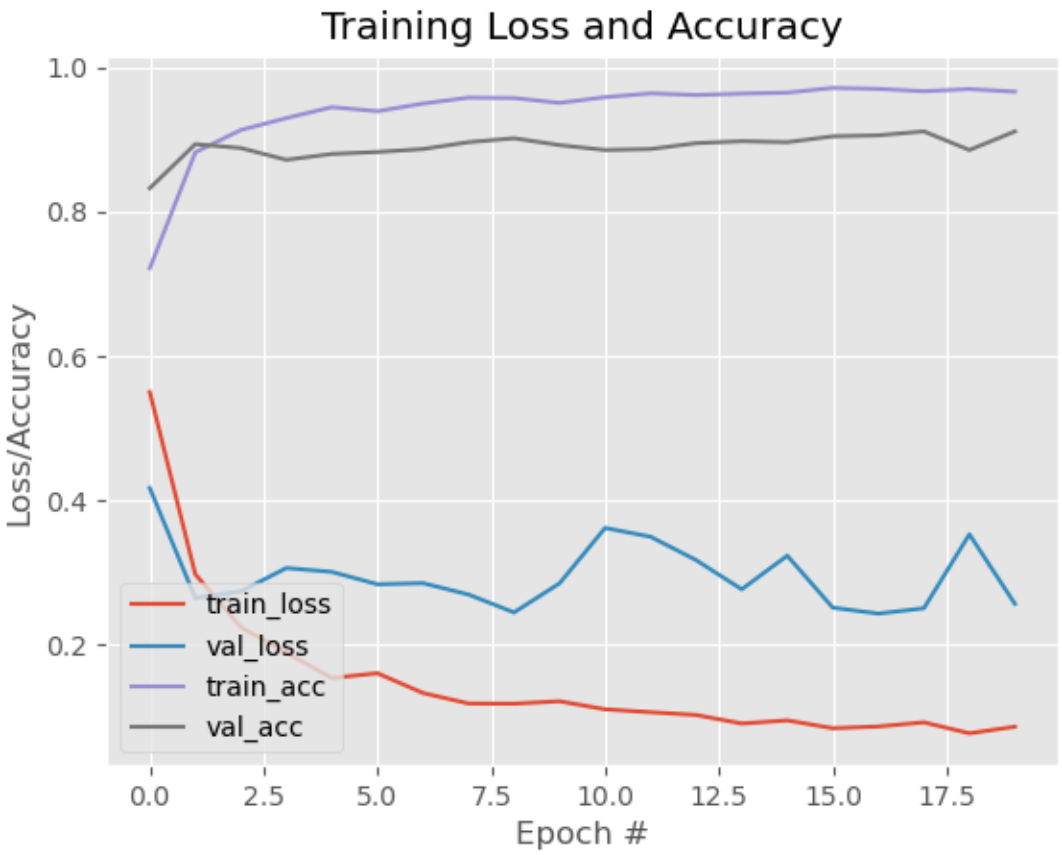


Figure 4.8 Graph showing the training loss and accuracy

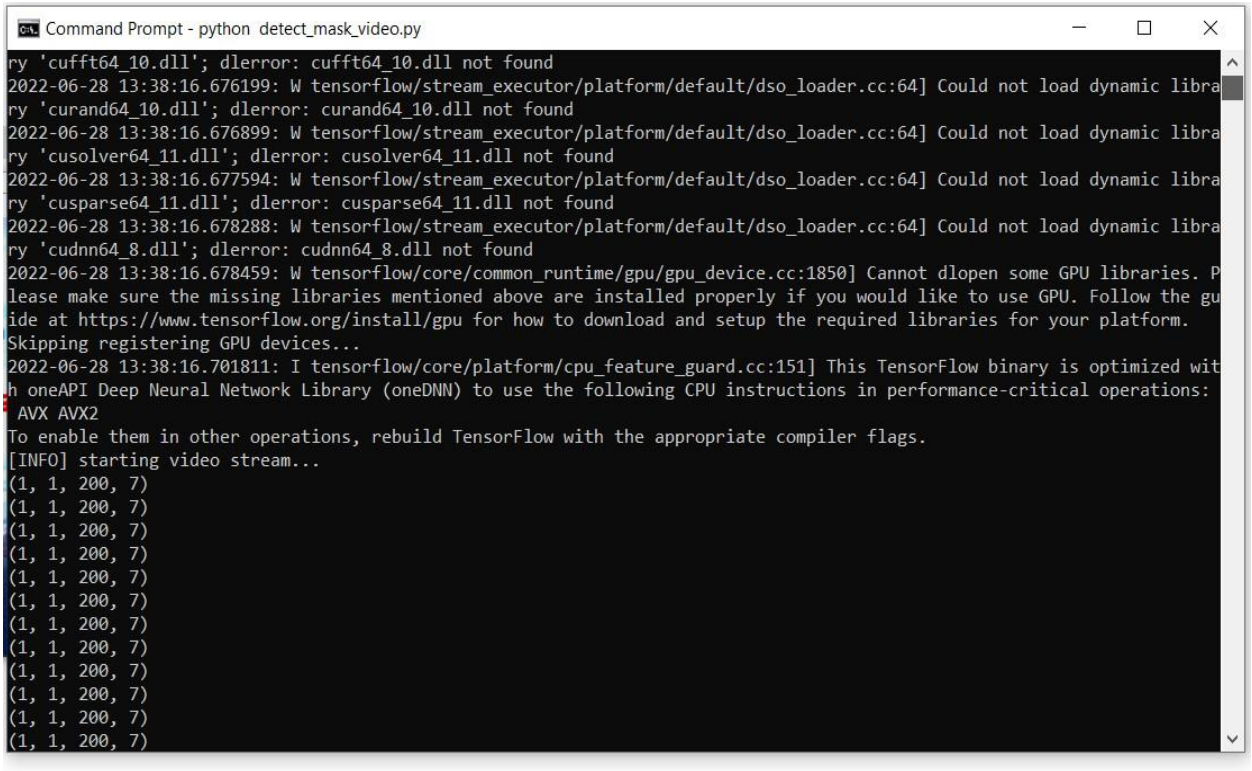


Figure 4.9 Loading the test model

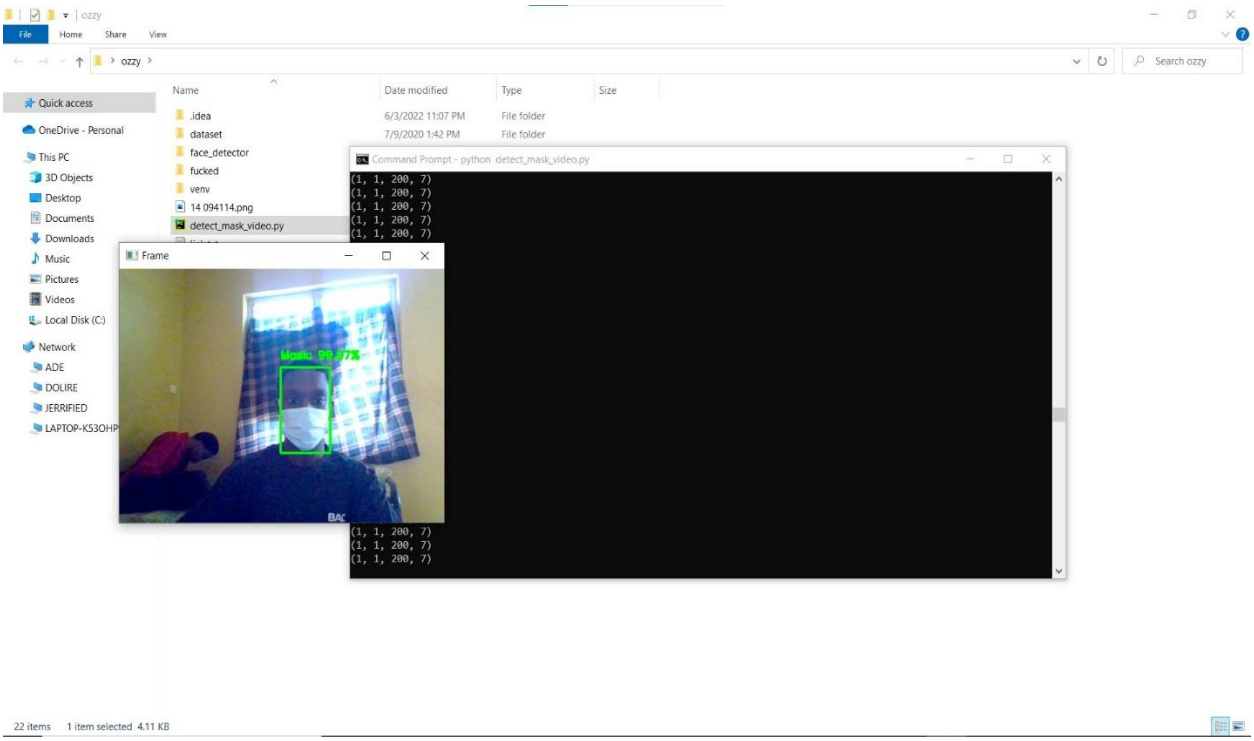


Figure 4.10 Testing the model with one face.

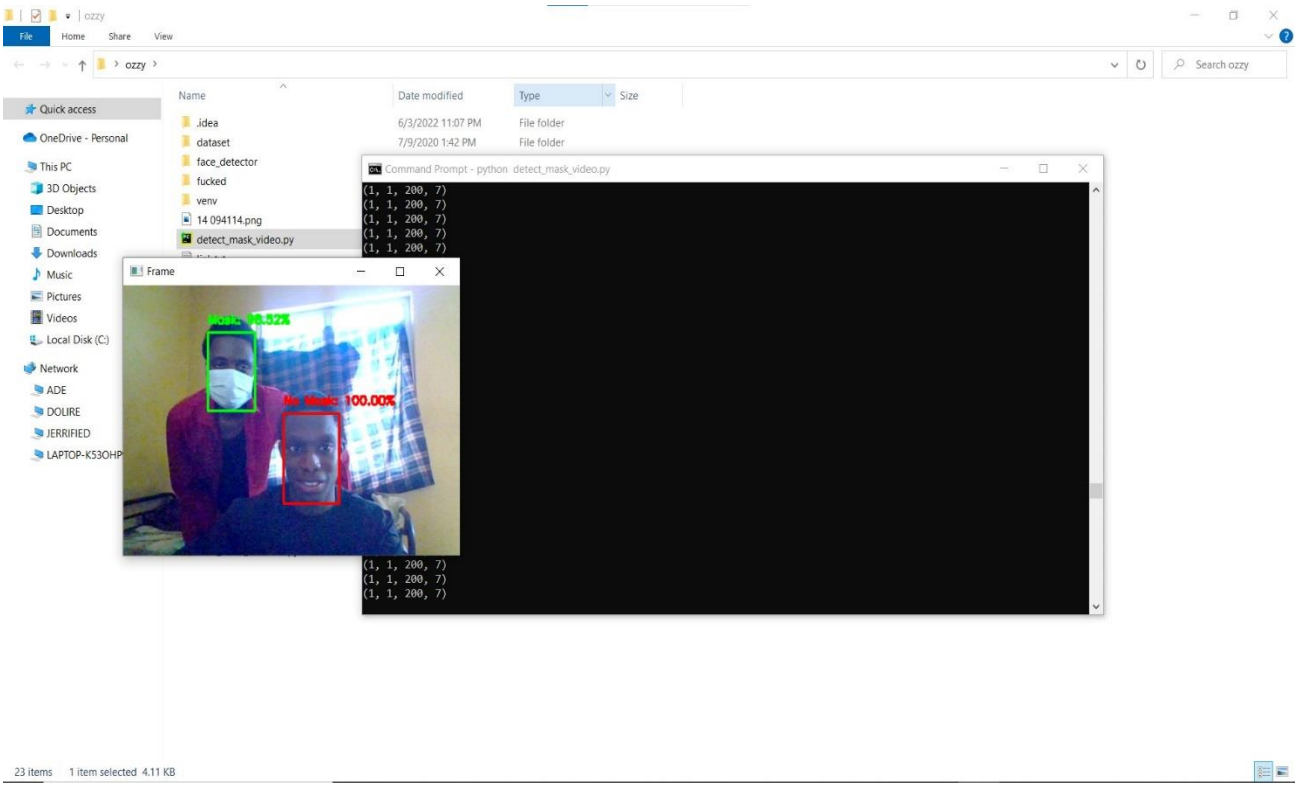


Figure 4.11: Testing the model with multiple faces

CHAPTER FIVE
SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Summary

This paper suggests a contribution to enhance TensorFlow's efficacy. In this study, we investigated the TensorFlow's fundamental idea. there would be a few improvements in the future, though. These include high and low thresholds for an adaptive parameterization of the sample selection parameters. Depending, for instance, on how much each sample contributed compared to the weight updates. Due to the utilization of the MobileNetV2 architecture, the model is accurate. The accuracy of the Tensor is significantly higher and more appropriate when compared to other machine learning techniques. To ensure that the public safety regulations are fulfilled, this project can be combined with embedded systems for use in railway stations, airports, offices, colleges, schools, and public areas.

5.2 Conclusion

Due to new trends in technology, a revolutionary face mask detector has been developed that may benefit public health. A real dataset is used to train the model. In order to determine whether or not people were wearing face masks, we employed OpenCV, tensor flow, keras, and CNN. Images and live video were used to evaluate the models. The model's correctness has been attained, and it is continually optimized as we construct an accurate answer by adjusting the hyperparameters. This particular model may be used as an example of an edge analytics use case. The development of a technology that can determine whether someone is wearing a face mask and permit their admission would be very beneficial to the society.

5.3 Recommendation

This project can be fine-tuned and tweaked to get higher and improved speed, accuracy and compatibility and it can also be deployed into some embedded systems and on other platforms like mobile apps, CCTV systems, web apps, etc.

5.4 Contribution of Knowledge

The notable contribution of this research work shows how an aspect of Computer science (artificial intelligence) can be used in different aspects of our lives. With the model like this, we can curb the spread of the coronavirus disease in public areas substantially and use it to regulate the wearing of face masks where it is needed.

REFERENCES

- Origin of mask https://www.clinicaloncology.com/COVID-19/Article/03-21/Uncovering-the-History-of-Medical-Face-Masks-In-the-Time-of-COVID-19.
- Why Google's MobileNetV2 Is A Revolutionary Next Gen On-Device Computer Vision Network By Smita Sinha https://analyticsindiamag.com/why-googles-mobilenetv2-is-a-revolutionary-next-gen-on-device-computer-vision-network/
- Real-Time Implementation of AI-Based Face Mask Detection.- https://www.hindawi.com/journals/sp/2021/8340779
- Uncovering the History of Medical Face Masks in the Time of COVID-19 - https://www.clinicaloncology.com/Article/PrintArticle?articleID=62804
- Uncovering the History of Medical Face Masks In the Time of COVID-19 https://www.ormanagement.net/COVID-19/Article/03-21/Uncovering-the-History-of-Medical-Face-Masks-In-the-Time-of-COVID-19/63447
- What Is Deep Learning? - MATLAB & Simulink - MathWorks https://www.mathworks.com/discovery/deep-learning.html
- 3 Things You Need to Know About Deep Learning - Medium https://medium.com/mathworks/3-things-you-need-to-know-about-deep-learning-342b71ba118
- Introduction Machine Learning 101 documentation- https://machinelearning101.readthedocs.io/en/latest/_pages/01_introduction.html
- List Of Top 5 Algorithms In Computer Science And Machine Learning - https://www.nytoday.org/list-of-top-5-algorithms-in-computer-science-and-machine-learning

- A guide to the types of machine learning algorithms by Katrina Wakefield | SAS UK - https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html
- ai-med.io › ac-observations › is-deep-learningIs deep learning supervised or unsupervised? - AIMed- https://ai-med.io/ac-observations/is-deep-learning-supervised-or-unsupervised
- Detection with Deep Learning: The Definitive Guide - https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning
- Classifying images using Keras MobileNet and TensorFlow.js - https://gogul.dev/software/mobile-net-tensorflow-js
- Image Classification with client-side neural network using ... - https://medium.com/agara-labs/image-classification-with-the-client-side-neural-network-using-tensorflow-js-8f94d3dc7c5c
- why-googles-mobilenetv2-isWhy Google's MobileNetV2 Is a Revolutionary Next Gen On ... - https://analyticsindiamag.com/why-googles-mobilenetv2-is-a-revolutionary-next-gen-on-device-computer-vision-network
- docs.w3cub.com › tensorflow~2 › kerastf.keras.applications.mobilenet_v2 - TensorFlow 2.3-W3cub- https://docs.w3cub.com/tensorflow~2.3/keras/applications/mobilenet_v2.html
- https://www.bmc.com/blogs/machine-learning-architecture/
- Convolutional Neutral Networks (CNN) with Deep Learning https://www.happiestminds.com/insights/convolutional-neural-networks-cnns

- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H, arXiv:1704.04861, 2017.
- MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. arXiv preprint. arXiv:1801.04381, 2018.
- Rethinking Atrous Convolution for Semantic Image Segmentation, Chen LC, Papandreou G, Schroff F, Adam H. arXiv:1706.05587, 2017.
- Speed/accuracy trade-offs for modern convolutional object detectors, Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017.
- Vinitha.V and Velantina.V, International Research Journal of Engineering and Technology (IRJET), "Covid-19 facemask detection with deep learning and computer vision," Aug, 2020.
- Adrian Rosebrock, "COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning," May 4, 2020. [Online].
- Deep Residual Learning for Image Recognition, He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. arXiv:1512.03385,2015
- Face mask recognition system using CNN model - PMC - NCBI https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8656214
- Face Mask Detection System Using Deep Learning - ProQuest https://search.proquest.com/openview/fbc09cba451d2d6607be72c9f3f7f44f/1?pq-origsite=gscholar

APPENDIX

Phase one of the model development (model training)

```
# import the necessary packages
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 imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os
```

```
# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
            img_path = os.path.join(path, img)
            image = load_img(img_path, target_size=(224, 224))
            image = img_to_array(image)
            image = preprocess_input(image)
```

```
        data.append(image)
        labels.append(category)

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

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

# construct the training image generator for 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")
```

```
# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
        input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
        layer.trainable = False

# compile our model
```

```
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
        metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
        aug.flow(trainX, trainY, batch_size=BS),
        steps_per_epoch=len(trainX) // BS,
        validation_data=(testX, testY),
        validation_steps=len(testX) // BS,
        epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
```

```
        target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")
```

phase two of the model development (model evaluation)

```
# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")
```

phase two of the model development(model testing)

```
# import the necessary packages
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 imutils
import time
import cv2
import os

def detect_and_predict_mask(frame, faceNet, maskNet):
	# grab the dimensions of the frame and then construct a blob
	# from it
	(h, w) = frame.shape[:2]
	blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
		(104.0, 177.0, 123.0))

	# pass the blob through the network and obtain the face detections
	faceNet.setInput(blob)
	detections = faceNet.forward()
	print(detections.shape)

	# initialize our list of faces, their corresponding locations,
	# and the list of predictions from our face mask network
	faces = []
```

```
locs = []
preds = []

# loop over the detections
for i in range(0, detections.shape[2]):
	# extract the confidence (i.e., probability) associated with
	# the detection
	confidence = detections[0, 0, i, 2]

	# filter out weak detections by ensuring the confidence is
	# greater than the minimum confidence
	if confidence > 0.5:
		# compute the (x, y)-coordinates of the bounding box for
		# the object
		box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
		(startX, startY, endX, endY) = box.astype("int")

		# ensure the bounding boxes fall within the dimensions of
		# the frame
		(startX, startY) = (max(0, startX), max(0, startY))
		(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

		# extract the face ROI, convert it from BGR to RGB channel
		# ordering, resize it to 224x224, and preprocess it
```

```
                face = frame[startY:endY, startX:endX]
                face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
                face = cv2.resize(face, (224, 224))
                face = img_to_array(face)
                face = preprocess_input(face)

                # add the face and bounding boxes to their respective
                # lists
                faces.append(face)
                locs.append((startX, startY, endX, endY))

        # only make a predictions if at least one face was detected
        if len(faces) > 0:
                # for faster inference we'll make batch predictions on *all*
                # faces at the same time rather than one-by-one predictions
                # in the above `for` loop
                faces = np.array(faces, dtype="float32")
                preds = maskNet.predict(faces, batch_size=32)

        # return a 2-tuple of the face locations and their corresponding
        # locations
        return (locs, preds)

# load our serialized face detector model from disk
```

```
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk
maskNet = load_model("mask_detector.model")

# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

# loop over the frames from the video stream
while True:
	# grab the frame from the threaded video stream and resize it
	# to have a maximum width of 400 pixels
	frame = vs.read()
	frame = imutils.resize(frame, width=400)

	# detect faces in the frame and determine if they are wearing a
	# face mask or not
	(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)

	# loop over the detected face locations and their corresponding
	# locations
```

```
for (box, pred) in zip(locs, preds):
	# unpack the bounding box and predictions
	(startX, startY, endX, endY) = box
	(mask, withoutMask) = pred

	# determine the class label and color we'll use to draw
	# the bounding box and text
	label = "Mask" if mask > withoutMask else "No Mask"
	color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

	# include the probability in the label
	label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

	# display the label and bounding box rectangle on the output
	# frame
	cv2.putText(frame, label, (startX, startY - 10),
		cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
	cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

# show the output frame
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF

# if the `q` key was pressed, break from the loop
```

```
        if key == ord("q"):
                break


# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```