# FACE MASK DETECTION SYSTEM

## 

# ABSTRACT

COVID-19 pandemic has rapidly affected our day-to-day life disrupting the world trade and movements. Wearing a protective face mask has become a new normal. In the near future, many public service providers will ask the customers to wear masks correctly to avail of their services. Therefore, face mask detection has become a crucial task to help global society. This paper presents a simplified approach to achieve this purpose using some basic Machine Learning packages like TensorFlow, Keras, OpenCV and python.

The proposed method detects the face from the image correctly and then identifies if it has a mask on it or not. As a surveillance task performer, it can also detect a face along with a mask in motion. The method attains accuracy up to 95.77% and 94.58% respectively on two different datasets. We explore optimized values of parameters using the Sequential Convolutional Neural Network model to detect the presence of masks correctly without causing over-fitting.

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **CONTENTS** | **i**  **ii** |
|  | **LIST OF FIGURES** | **vii** |
|  | **LIST OF SYMBOLS** | **viii** |
|  | **LIST OF ABBREVIATIONS** | **ix** |
| **1** | **INTRODUCTION** | **1** |
|  | 1.1 OBJECTIVE | 3 |
|  | 1.2 PROJECT DESCRIPTION | 3 |
| **2** | **SYSTEM ANALYSIS** | **4** |
|  | 2.1 EXISTING SYSTEM | 4 |
|  | 2.2 PROPOSED SYSTEM | 4 |
|  | 2.3 TECHNOLOGIES USED | 6 |
|  | 2.3.1 MACHINE LEARNING | 6 |
|  | 2.3.2 ARTIFICIAL INTELLIGENCE | 6 |
|  | 2.3.3 COMPUTER VISION | 7 |  |
|  | 2.3.4 DEEP LEARNING | 7 |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **3** | **LITERATURE SURVEY** | **8** |
|  | **3.1 GENERAL** | **8** |
|  | 3.1.1 IMPLEMENTATION OF PRINCIPAL  COMPONENT ANALYSIS ON MASKED  AND NON-MASKED FACE  RECOGNITION | 8 |
|  | 3.1.2 IDENTIFYING FACEMASK – WEARING CONDITION USING IMAGE SUPER - RESOLUTION WITH CLASSIFICTION NETWORK TO PREVENT COVID-19 | 8 |
|  | 3.1.3 FACE DETECTION AND SEGMENTATION WITH GENERALIZED INTERSECTION OVER UNION BASED ON MASK R-CNN | 8 |
|  | 3.1.4 A NOVEL GAN-BASED NETWORK FOR UNMASKING OF MASKED FACE | 8 |
|  | 3.1.5 IMPROVING MASK DETECTION FROM SPEECH USING AUGMENTATION BY CYCLE-CONSISTENT GANs. | 8 |
|  | 3.1.6 RESHAPING INPUTS FOR CONVOLUTIONAL NEURAL NETWORK: SOME COMMON AND UNCOMMON METHODS | 8 |
|  | 3.1.7 ENLARGING SMALLER IMAGES BEFORE INPUTTING INTO CONVOLUTIONAL NEURAL NETWORK | 8 |
|  | 3.1.8 ACTIVATION FUNCTIONS: COMPARISON OF TRENDS IN PRACTICE AND RESEARCH FOR DEEP LEARNING | 9 |
|  | 3.1.9 AN APPROACH TO FACE DETECTION AND RECOGNITION | 9 |
|  | 3.1.10 COLOR-TO-GRAYSCALE: IMAGE RECOGNITION | 9 |
|  | 3.1.11 BIG DATA CONCEPTS AND TECHNIQUES IN DATA PROCESSING | 9 |
|  | 3.1.12 VISUAL ANALYTICS IN DEEP LEARNING: AN INTERROGATIVE SURVEY FOR THE NEXT FRONTIERS | 9 |
|  | 3.1.13 RETINAMASK: A FACE MASK DETECTOR | 9 |
|  | 3.1.14 CONVOLUTIONAL NEURAL NETWORKS: AN OVERVIEW AND APPLICATION IN RADIOLOGY | 9 |
|  | 3.1.15 DETECTING MASKED FACES IN THE WILD WITH LLE-CNNs | 9 |

|  |  |  |
| --- | --- | --- |
| **4** | **SYSTEM DESIGN** | **10** |
|  | 4.1 ARCHITECTURE DIAGRAM | 10 |
|  | 4.2 UML DIAGRAMS | 11 |
|  | 4.2.1 USECASE DIAGRAM | 11 |
|  | 4.2.2 COLLABORATION DIAGRAM | 12 |
|  | 4.3 DATA FLOW DIAGRAMS | 13 |
|  | 4.4 SYSTEM SPECIFICATION | 16 |
|  | 4.4.1 HARDWARE  REQUIREMENTS | 16 |
|  | 4.4.2 SOFTWARE REQUIREMENTS | 16 |
| **5** | **SYSTEM IMPLEMENTATION** | **17** |
|  | 5.1 LIST OF MODULES | 17 |
|  | 5.2 MODULE DESCRIPTION | 17 |
|  | 5.2.1 PYTHON | 17 |
|  | 5.2.2 OPENCV | 17 |
|  | 5.2.3 TENSORFLOW | 18 |
|  | 5.2.4 KERAS | 19 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **6** | **CODING AND TESTING** | **20** |
|  | 6.1 CODING STANDARDS | 20 |
|  | 6.1.1 NAMING CONVENTIONS | 20 |
|  | 6.1.2 VALUE CONVENTIONS | 21 |
|  | 6.1.3 SCRIPT WRITING & | 21 |
|  | COMMENTING STANDARD |  |
|  | 6.2 TEST CASES | 21 |
|  | 6.3 TEST PROCEDURE | 21 |
|  | 6.3.1 SYSTEM TESTING | 22 |
|  | 6.4 UNIT TESTING | 22 |
|  | 6.5 FUNCTIONAL TESTING | 22 |
|  | 6.6 PERFORMANCE TESTING | 23 |
|  | 6.6.1 TESTING TYPES | 23 |
|  | 6.7 TESTING TECHNIQUES | 24 |
|  | 6.7.1 TESTING | 24 |
|  | 6.7.2 SOFTWARE TESTING  STRATIGIES | 26 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | 6.7.3 INTEGRATION  TESTING | 27 |
|  | 6.7.4 VALIDATION  TESTING | 27 |
|  | 6.7.5 OUTPUT TESTING | 28 |
|  | 6.7.6 USER ACCEPTANCE TESTING | 28 |
| **7** | **APPENDICES** | 29 |
|  | 7.1 SAMPLE CODE | 29 |
|  | 7.2 SCREEN SHOTS | 39 |
|  | 7.3 PERFORMANCE ANALYSIS | 41 |
| **8** | **CONCLUSION AND FUTURE** | 42 |
|  | **ENHANCEMENTS** |  |
|  | * 1. CONCLUSION   2. FUTUREENHANCEMENTS | 42  42 |
|  | 8.3 REFERENCES | 43 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO.** |
| 4.1.1 | System Architecture | 10 |
| 4.2.1 | Usecase Diagram | 11 |
| 4.2.3 | Collaboration Diagram | 12 |
| 4.3.1 | Level 0 Data Flow Diagram | 14 |
| 4.3.2 | Level 1 Data Flow Diagram | 14 |
| 4.3.3 | Level 2 Data Flow Diagram | 15 |
| 7.2.1 | COVID-19 face mask detection in images with OpenCV | 39 |
| 7.2.2 | Detecting COVID-19 face masks with OpenCV in real time | 40 |
| 7.2.3 | Detection of mask with multiple faces in live stream | 40 |
| 7.3.1 | Comparison of detection systems | 41 |
| 7.3.2 | Bar Graph Representation | 41 |

**LIST OF SYMBOLS**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **SYMBOLNAME** | **SYMBOL** |
| 1. | USECASE |  |
| 2. | ACTOR |  |
| 3. | PROCESS |  |
| 4. | START |  |
| 5. | DECISION |  |
| 6. | UNIDIRECTIONAL |  |
| 7. | ENTITY SET |  |

|  |  |
| --- | --- |
| 8. | STOP |

# LIST OF ABBREVIATIONS

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVIATION** | **EXPANSIION** |
| 1 | UML | Unified Modeling Language |
| 2 | DFD | Data Flow Diagram |
| 3 | CNN | Convolution Neural Network |
| 4 | AI | Artificial Intelligence |
| 5 | ML | Machine Learning |
| 6 | OPENCV | Open Source Computer Vision Library |

# CHAPTER 1

# INTRODUCTION

The trend of wearing face masks in congregation is rising due to the COVID-19 coronavirus epidemic all over the world. Before Covid-19, People used to wear masks to protect their health from air pollution. While other people are self-conscious about their looks, they hide their emotions from the public by riding their faces. Scientists proved that wearing face masks works on impeding COVID – 19 transmissions. COVID-19 is the latest epidemic virus that hit human health in the last century. In 2020, the rapid spreading of COVID-19 has forced the World Health Organization to declare COVID-19 as a global pandemic. More than five million cases were infected by COVID-19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and overcrowded areas. The coronavirus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight COVID-19 in many ways. Machine learning allows researchers and clinicians evaluate vast quantities of data to forecast the distribution of COVID-19, to serve as an early warning mechanism for potential pandemics, and to classify vulnerable populations. The provision of healthcare needs funding for emerging technology such as artificial intelligence, IoT, big data and machine learning to tackle and predict new diseases. In order to better understand infection rates and to trace and quickly detect infections, the AI’s power is being exploited to address the Covid-19 pandemic. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult. The monitoring process involves the detection of anyone who is not wearing a face mask. Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with OpenCV, TensorFlow and Keras. We have used deep transfer learning for feature extractions and combined it with three classical machine learning algorithms. We introduced a comparison between them to find the most suitable algorithm that achieved the highest accuracy and consumed the least time in the process of training and detection.

**PROBLEM STATEMENT:**

As we all know that there is an ongoing pandemic of coronavirus disease 2019(COVID-19) which is accelerating day by day, self protection is the only way out which can be done by wearing masks due to which the utility of masks nowadays is widely accepted. Though frontline corona warriors are trying their best and ensuring that everyone is wearing mask in congregation but it is not possible for them alone to go to nook and cranny to ensure safety. In view of this current situation our team decided to make face mask detector. The task in hand is to check whether the person is wearing mask or not through Live Stream using Webcam. Besides that, there is also an advantage of detecting face mask through images.

**1.1 OBJECTIVE**

To identify the person on an image/video stream wearing a face mask with the help of computer vision and deep learning algorithm with OpenCV, Keras, TensorFlow. The mask detector that is built in this project could potentially be used to help in ensuring your safety and the safety of others. It is also to detect accuracy of the mask. We will be using python language to implement our project. It is hard to check whether is wearing mask or not in such cases computer vision will be of great help. This system is used in Colleges, Schools, Hospitals and public places.

## PROJECT DESCRIPTION

Face mask detection is an AI based technology that analyzes a video stream to detect and recognize a face mask worn by an individual person or a crowd of people. Our Deep Sight software outputs a confidence value for each detection. Every individual is classified either as ‘wearing a mask’ or flagged as ‘not wearing a mask. Face mask detection had seen significant progress in the domains of Image processing and Computer vision, since the rise of the Covid-19 pandemic. Many face detection models have been created using several algorithms and techniques. The proposed approach uses deep learning, TensorFlow, Keras, and OpenCV to detect face masks. This model can be used for safety purposes since it is very resource efficient to deploy

# CHAPTER 2 SYSTEM ANALYSIS

## EXISTING SYSTEM

Cracks are common pavement surface distresses that affect road performance and periodical road surveys are necessary to assess pavement surface conditions. Traditional manual crack detection methods are very time consuming, labor-intensive, with low-accuracy, and error prone. Pavement crack detection from images is a challenging problem due to intensity inhomogeneity, topology complexity, low contrast, and noisy texture background. Traditional learning based approaches have difficulties in obtaining representative training samples.

## PROPOSED SYSTEM

## The proposed system of face mask detection is to detect the accuracy of the mask. Incase, the person is not wearing a mask, an alert will be generated. Instructions will be given to the users to wear a mask properly. This project uses Machine Learning classification using Python, OpenCV and TensorFlow to detect facemask on people. To predict whether a person has worn a mask correctly, the initial stage would be to train the model using a proper dataset. Details about the Dataset have been discussed. After training the classifier, an accurate face detection model is required to detect faces, so that the SSDMNV2 model can classify whether the person is wearing a mask or not.

## The task in this paper is to raise the Accuracy of mask detection without being too resource-heavy. For doing this task, the DNN module was used from OpenCV, which contains a ‘Single Shot Multibox Detector’ (SSD) (Liu et al., 2016) object detection model with ResNet-10 (Anisimov & Khanova, 2017) as its backbone architecture. This approach helps in detecting faces in real-time, even on embedded devices like Raspberry Pi. The following classifier uses a pre-trained model MobileNetV2 (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018) to predict

## whether the person is wearing a mask or not.

## The other thing is that we have added a voice note to instruct those who are not wearing their mask and also for those who are not wearing it properly. We have also proposed to detect multiple faces at a time in live stream. The proposed method detects the face from the image correctly and then identifies if it has a mask or not.

**PROPOSED** **MKS** **ALGORITHM:**

Step 1: Start

Step 2: Datasets are taken as input.

Step 3: Datasets are images with and without masks.

Step 4: Dataset will be loaded to recognize the faces.

Step 5: Face masks of the persons will be detected.

Step 6: The person will be instructed with a voice note.

Step 7: Face mask detection will be done by live video stream.

Step 8: Multiple faces can be detected at a time.

Step 9: The output is detected with accuracy.

Step 10: Stop

The new improved system gives the gives an edge over the existing system by detecting multiple faces at a time and can detect in live stream. Datasets are loaded to detect the face with mask and without mask.

## TECHNOLOGIES USED

* + 1. **MACHINE LEARNING**

Machine learning (ML)is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as “training data’’, in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering

and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

* + 1. **ARTIFICIAL INTELLIGENCE**

Artificial intelligence is a branch of computer science that works on developing simulations of human intelligence in machines or computer systems. Ultimately, the goal of artificial intelligence is to replicate human intelligence so that machines can carry out tasks that require general human intelligence. AI works by taking in large amounts of data and teaching the machine to recognize patterns. This then enables the machine to be able to perform tasks such as learning, decision making, problem solving, and reasoning. The system detects faces that are not wearing a mask or are not wearing a mask correctly by first detecting the faces of the people in the frame of the video.

## 2.3.3 COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do, Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high dimensional data from the real world in order to produce numerical or symbolic information, e.g, in the forms of decisions, Understanding in this context means the transformation of visual images into descriptions of the world that make sense to thought process and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics and learning theory.

## 2.3.4 DEEP LEARNING

## Deep learning methods aim at learning feature hierarchies with features from higher

## levels of the hierarchy formed by the composition of lower-level features.

## Automatically learning features at multiple levels of abstraction allow a system to

## learn complex functions mapping the input to the output directly from data, without

## depending completely on human-crafted features. Deep learning algorithms seek to

## exploit the unknown structure in the input distribution in order to discover good

## representations, often at multiple levels, with higher-level features defined in terms

## of lower-level features. Deep learning allows computational models that are

## composed of multiple processing layers to learn representations of data with

## multiple levels of abstraction.

# CHAPTER 3

# LITERATURE SURVEY

# 3.1 GENERAL

## M. S. Ejaz et al [1], Principal Component Analysis (PCA) was used by the authors to identify faces with masks. It is necessary for user safety. This method is used to determine whether or not a person is wearing a mask on their face. It demonstrates that using a detection approach, the face mask detection system can help to reduce the spread of Omicron.

## Bosheng Qin et al [2], the authors developed detection of face mask conditions, like how they are wearing the masks. It detects whether the persons mask is properly worn. The output will be wearing-masks, non-wearing mask, incorrectly wearing. This method achieved 85% in accuracy.

## Kaihan Lin et al [3], Using the Generalized Intersection over Union (GIoU) approach, the authors devised a method for identifying mask-wearing and mask-free faces. By identifying faces in the background, this approach helps to decrease background noise. GIoU method is used instead of bounding boxes as they add noise to the detection.

## N. Ud Din et al [4], The authors discovered that data from images was processed for face mask detection. This is done by using spectrograms. The inputs are given for the purpose of detection. One of the components in the detection is the ResNet resource.

## Nicolae-Cătălin Ristea et al [5], The Face Mask Reconstruction method was proposed by the authors. The removal of a face mask from a face and the rebuilding of a face using the CelebA dataset are the two detection procedures used in this method.

## S. Ghosh et al [6], According to this, the authors discovered that Convolutional Neural Networks are used to detect face masks (CNNs). It uses computer vision to aid in the detection of images with greater clarity. It uses computer vision to rebuild the images.

## M. Hashemi [7], For face mask detection, the authors presented a 3D feature tensor. The photos are used as input to the three-dimensional tensor, with each image having its own pixel. The 3D feature approach requires that all photos be of the same size. In 3D feature tensor detection, the size employed is critical.

## C. Nwankpa et al [8], the authors used gradient-descent method for optimization of linear function which contains linear methods. The comparisons with other activation resources shows that the performance and reliability in deep learning is improved.

## D. Meena et al [9], According to this research, the authors proposed face detection method. It includes expression recognition, face tracking and position detection. Face will be identified from a picture. Faces change in size, shape, colour, and other characteristics, making it difficult to recognize them. It is useful for opaque image prevented by confronting the camera.

## C. Kanan et al [10], proposed by the authors Modern computing image recognition technology will be used to perform grayscale recognition. This system only works with grayscale photos and does not expand the process. It's used to transform an image colour to a grayscale version. Its purpose is to avoid negative effects when employing robust computing approaches. The grayscale will simplify the algorithm and reduce the computational requirements. Instead of working with colour photos, extracting computational approaches will be used.

## B. Suvarnamukhi et al [11], The authors devised a method for pre-processing data. It entails converting data from an unfamiliar format to a more user-friendly format. The transformed data format is the one that you want. Images, tables, videos, graphs, and other types of data are all acceptable. This discovers the relationship between several items, which will be utilized to create an organized data set.

## F. Hohman et al [12], For the face mask detection process, the authors presented a Data Visualization approach. It's the process of employing knowledge representations and encodings to transform existing data into meaningful descriptions. The data visualization technique will be based on the dataset's specific pattern.

## M. Jiang et al [13], Face Mask Detection System was developed by the authors. The location of the faces is recognized in this method, as well as whether or not the face has a mask on it. This has something to do with the object detecting approach. Its purpose is to identify the classes of items, such as images. Face recognition is concerned with classifying various items. This system monitors autonomous driving, detection, and surveillance.

## R. Yamashita et al [14], The authors discovered that in computer vision tasks, CNN is the most commonly used approach. This approach was utilized to distinguish between mask-wearing and non-mask-wearing faces.

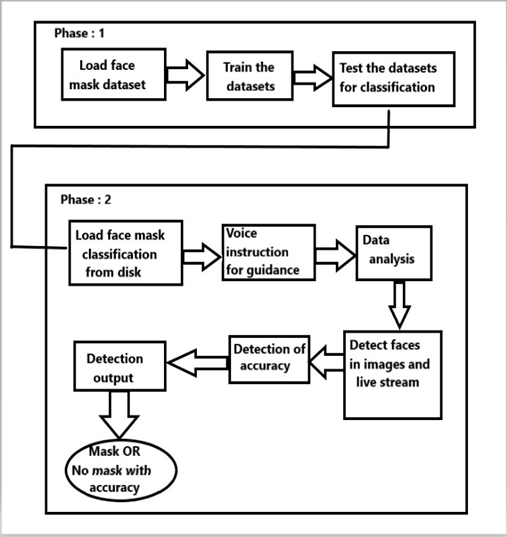
## S. Ge et al [15], The authors found two face detection processes as a result of this research. 1) Masked and unmasked face datasets are both unavailable. 2) Image detection of facial expressions. Face synchronization will be done using the Locally Linear Embedding technique.

# CHAPTER 4

# SYSTEM DESIGN

## ARCHITECTURE DIAGRAM

In the system diagram, all the tasks related to an object recognition problem can be ensembled under three main components. Backbone, Neck and Head as depicted int. Here the backbone corresponds to a baseline convolutional neural network capable of extracting information from images and converting them to a feature map. In the proposed architecture the concept of transfer learning is applied on the backbone to utilize already learned attributes of a powerful pre-trained convolutional neural network in extracting new features for the module.

****

***Figure 4.1.1 System architecture***

## 4.2 UML DIAGRAMS

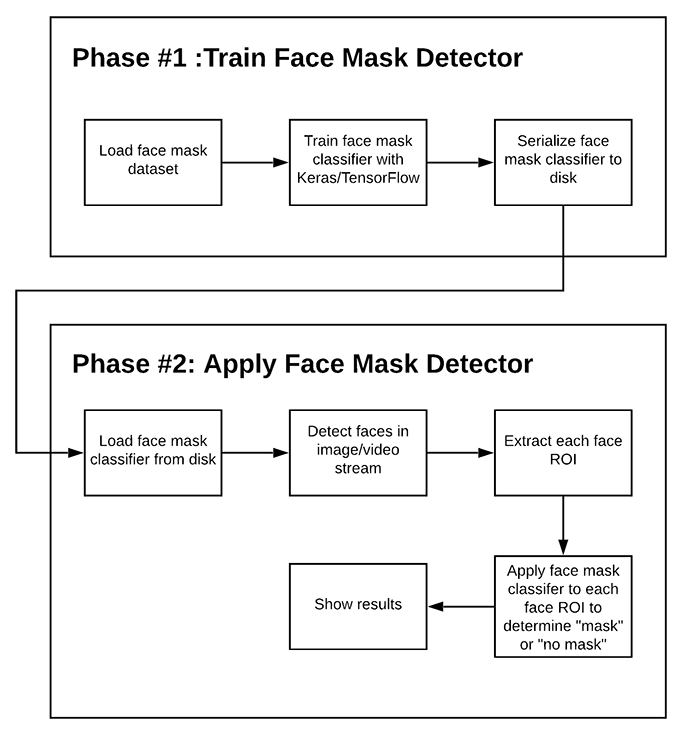
## 4.2.1 USECASE DIAGRAM

A Use Case Diagram is a representation of a user’s interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modeling the dependency between the two use cases. The reuse of an existing use case by using different types of relationships reduces the overall effort required in developing a system.

# 

**4.2.2 COLLABORATIVE DIAGRAM**

In collaborative diagram, there is two classification one is about MobileNetV2 classifier processing using PyTorch and the next part is to apply MobileNetV2 classifier on test data. In that we have to load the face mask data and that will be preprocessing and the datasets are trained for the process. In the second part the masks are loaded from the disk and the faces will be loaded in the video stream for the detection process. This will detect the persons status like “mask” or “no mask” and will show the result.



***Figure 4.2.2 Collaborative diagram***

**4.3 DATAFLOW DIAGRAMS**

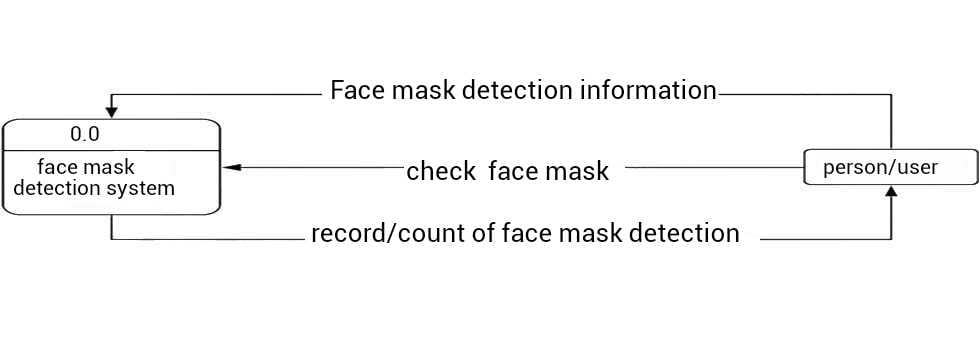
A picture is worth a thousand words. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both.

It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2diagramwhenfurtheranalysisisrequired.Progressiontolevel3,4andsoon is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing particular function really depending on the complexity that function.

## Level 0

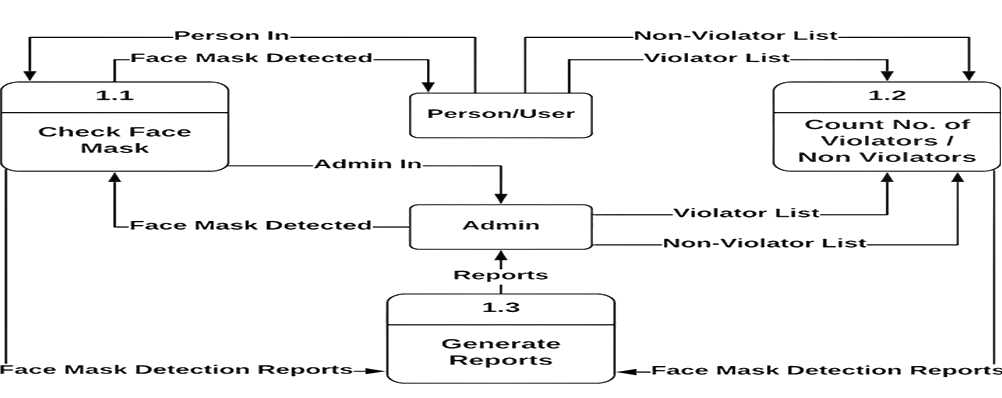
The Face Mask Detection System DFD level 0 is also known as context diagram. It’s supposed to be an abstract view, with the mechanism represented as a single process with external parties. This DFD for the Face Mask Detection System depicts the overall structure as a single bubble. It comes with incoming/outgoing indicators showing input and output data.

****

***Figure 4.3.1 Level 0***

**Level 1**

The content of Face Mask Detection System DFD level 1 must be single process node from the context diagram and is broken down into sub processes. In this level, the system must display or reveal further processing information. These procedures require information such as record of persons with face mask, without face mask and total count of individuals that has entered the firm/establishment from which served as the bases for admin to manage the Face Mask Detection System.

****

***Figure 4.3.2 Level 1***

## Level 2

## 

## The Level 2 DFD for the system should represent the basic modules as well as data flow between them. Since the DFD level 2 is the highest abstraction level, its Face Mask Detection System processes must be detailed that is based on the DFD level 1. This level gives you precise destination of the data that flows in the system.

## 

***Figure 4.3.3 Level 2***

* 1. **SYSTEM SPECIFICATION**

**4.4.1 HARDWARE REQUIREMENTS**

System Core i3, 2.4 GHz

Hard Disk 160 GB of Disk Space

Main Memory 500 MB

Cache Memory 512 KB

Monitor 14/15 inches Colour

Mouse Optical Mouse

Keyboard 108 Keys (Standard Keyboard)

Ram 4 GB

Device Camera

* + 1. **SOFTWARE REQUIREMENTS**

Operating System MacOS, Windows 7, 8,9,10

Programming Language Python

Coding Platform Sublime text

Libraries TensorFlow, Keras, OpenCV, python.

**CHAPTER 5 SYSTEM IMPLEMENTATION**

* 1. **LIST OF MODULES**
     + Python
     + OpenCV
     + TensorFlow
     + Keras

## 5.2 MODULE DESCRIPTION

* + 1. **PYTHON**

Python is an interpreted, high-level, general-purpose programming language. Created by “Guido van Rossum” and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

## OPENCV

All OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera

movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf, and Zeitera that make extensive use of OpenCV.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

**5.2.3 TENSORFLOW**

Data TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, TensorFlow is Google Brain’s second-generation system. Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android

and iOS. Its flexible architecture allows for the easy deployment of computation

across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least Rank Brain in Google search and the fun Deep Dream project. It can run on single CPU systems, GPUs as well as mobile devices and large-scale distributed systems of hundreds of machines.

## 5.2.4 KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

# CHAPTER 6

# CODING AND TESTING

## CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary.

Some of the standard needed to achieve the above-mentioned objectives are as follows:

## NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

## CLASS NAMES

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

## MEMBER FUNCTION AND DATA MEMBER NAME

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

## VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

* + - * Proper default values for the variables.
      * Proper validation of values in the field.
      * Proper documentation of flag values.

## SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is atmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

## TESTCASES

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily *functional* in nature, *non- functional* tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

## TEST PROCEDURE

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of *what* the software is supposed to do but is not aware of *how* it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of *how* the software produces the output in the first place.

## SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware systems. The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called *assemblages*) or between any of the *assemblages* and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

## UNIT TESTING

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure.

## FUNCTIONAL TESTING

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes *what* the system does. Functional testing differs from system testing in that functional testing "*verifies* a program by checking it against the design document or specifications", while system testing "*validate* a program by checking it against the published user or system requirements".

Functional testing typically involves five steps

* The identification of functions that the software is expected to perform.
* The creation of input data based on the function's specifications
* The determination of output based on the function's specifications
* The execution of the testcase
* The comparison of actual and expected outputs

## PERFORMANCE TESTING

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

## TESTING TYPES

* + - * **LOAD TESTING**

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions.

## SPIKE TESTING

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behavior of the system. The goal is to determine whether performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

## STRESS TESTING

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

## ISOLATION TESTING

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

## TESTING TECHNIQUES

* + 1. **TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding any undiscovered error. A successful test is one that uncovers an yet undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete.

Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding.

Any engineering product can be tested in one of the two ways:

## WHITE BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques such as control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment.

## LEVELS

1. **Unit Testing:** White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code.
2. **Integration Testing:** White-box testing at this level are written to test the interactions of each interface with each other. The integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.
3. **Regression Testing:** White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

## BLACK BOX TESTING

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing such as unit, integration, system and acceptance. In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”. It performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. Typical black-box test design techniques include Decision table testing, All pairs testing, State transition tables, Equivalence partitioning, Boundary value analysis.

## SOFTWARE TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

* + - * Testing begins at the module level and works “outward” toward the integration of the entire computer-based system.
      * Different testing techniques are appropriate at different points in time.
      * The developer of the software and an independent test group conducts testing.
      * Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

## INTEGRATION TESTING

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Simulated usage of shared data areas and inter- process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

## VALIDATION

Validation is intended to check the development and verification procedures for a product, service, or system result in a product, service, or system that meets initial requirements. For a new development flow or verification flow, validation procedures may involve modelling either flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system. A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system.

## OUTPUT TESTING

Output of test cases compared with the expected results created during design of test cases. Asking the user about the format required by them tests the output generated or displayed by the system under consideration. Here, the output format is considered into two was, one is on screen and another one is printed format.

The output on the screen is found to be correct as the format was designed in the system design phase according to user needs. The output comes out as the specified requirements as the users hard copy.

## USER ACCEPTANCE TESTING

Final Stage, before handling over to the customer which is usually carried out by the customer where the test cases are executed with actual data. The system under consideration is tested for user acceptance and constantly keeping touch with the prospective system user at the time of developing and making changes whenever required. It involves planning and execution of various types of test in order to demonstrate that the implemented software system satisfies the requirements stated in the requirement document. Two set of acceptance test to be run as follows

* + - * Those developed by quality assurance group.
      * Those developed by customer.

# CHAPTER 7

# APPENDICES

## 7.1 SAMPLE CODE

## 

## ****1. detect\_mask\_video.py****

# 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")

#Voice instruction to the user

import pyttsx3

engine = pyttsx3.init()

engine.say("Please wear mask and start this project . It will show green when you wear the mask and it will show red when mask is not ")

engine.runAndWait()

# 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()

**2. train\_mask\_detector.py**

# 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")

# 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")

## 7.2 SCREENSHOTS

## 7.2.1 COVID-19 face mask detection in images with OpenCV:

## 

**7.2.2 Detecting COVID-19 face masks with OpenCV in real-time:**

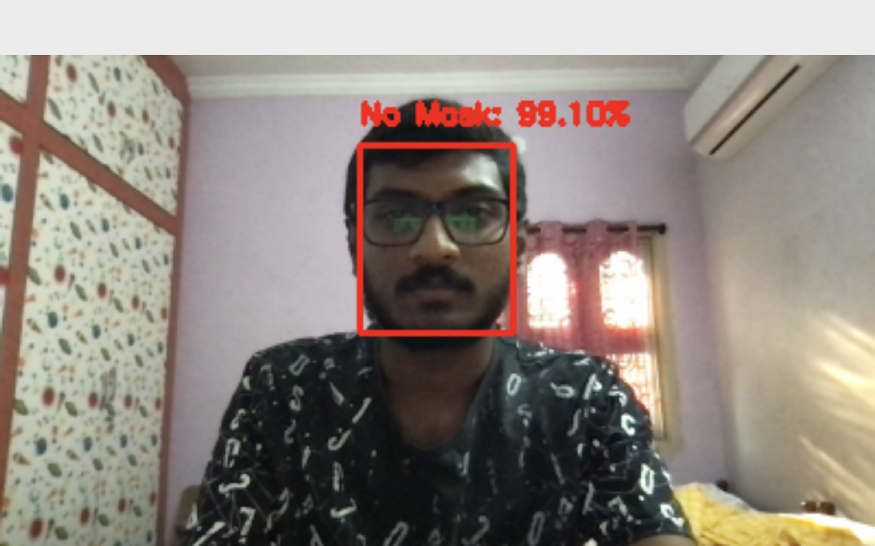


Figure 7.2.2.1: Using Python, OpenCV, and TensorFlow/Keras, our system has correctly detected

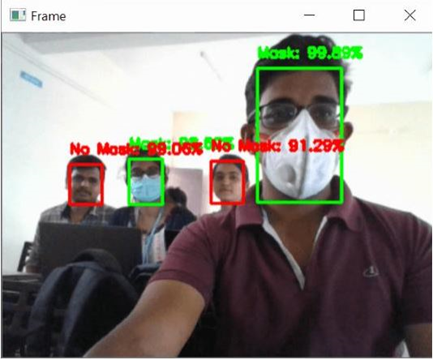
“No Mask” for my face.



Figure 7.2.2.2: Using Python, OpenCV, and TensorFlow/Keras, our system has correctly detected “Mask” for my face.

**7.2.3 DETECTION OF MASK WITH MULTIPLE FACES IN LIVE**

**STREAM**

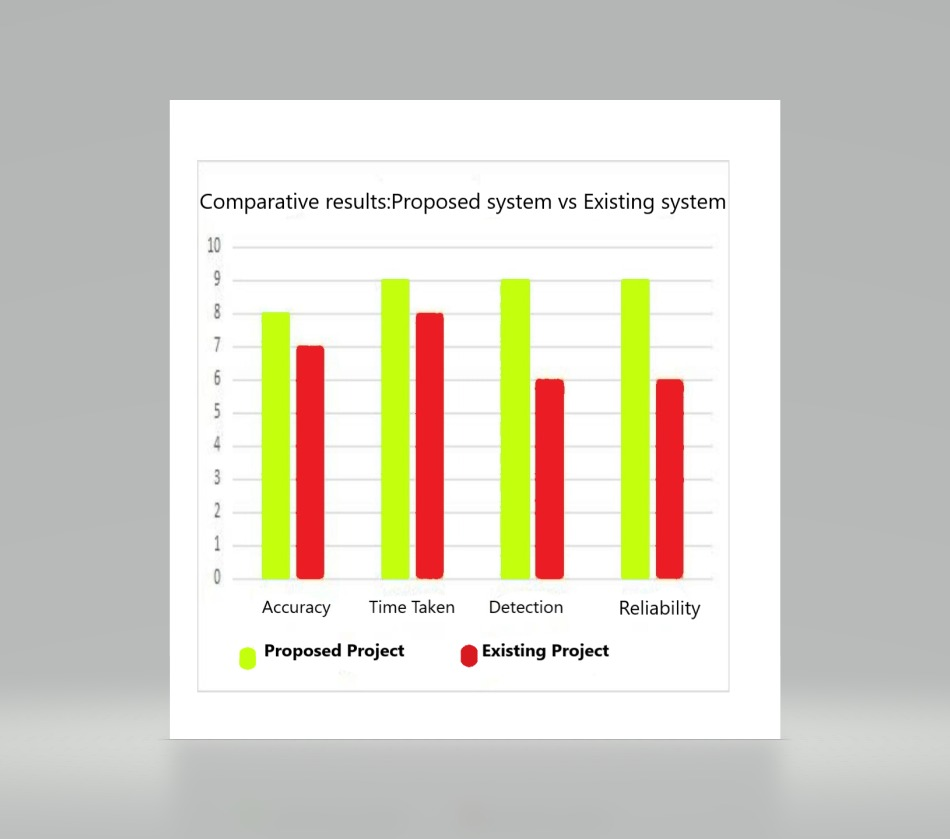


**7.3 PERFORMANCE ANALYSIS**

**7.3.1 COMPARISON OF DETECTION SYSTEM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SI.NO** | **SYSTEM** | **ACCURACY OF IMAGES** | **NOOF SAMPLES** | **TIME**  **TAKEN** |
| 1. | MIN system | 91.3% | WM=500  WOM=500 | 6m30s |
| 2. | ACS system | 88% | WM=700  WOM=700 | 4m20s |
| 3. | SVM system | 77% | WM=900  WOM=900 | 3m41s |
| 4. | Proposed  MKS system | 95% | WM=1500  WOM=1500 | 1m25s |

**7.3.2 BAR GRAPH REPRESENTATION**

****

**CHAPTER 8 CONCLUSION AND FUTURE WORK**

* 1. **CONCLUSION**

## As the technology is blooming with emerging trends the availability so we have novel face mask detectors which can possibly contribute to public healthcare. The architecture consists of MobileNet as the backbone and it can be used for high and low computation scenarios. In order to extract more robust features, we utilize transfer learning to adopt weights from a similar task face detection, which is trained on a very large dataset. We used OpenCV, tensor flow, Keras and CNN to detect whether people were wearing face masks or not. The models were tested with images and real-time video streams. The accuracy of the model is achieved and the optimization of the model is a continuous process and we are building a highly accurate solution by tuning the MobileNet V2. This specific model could be used as a use case for edge analytics. Furthermore, the proposed method achieves state-of-the-art results on a public face mask dataset. By the development of face mask detection, we can detect if the person is wearing a face mask and allow their entry would be of great help to the society. Further we can add heat sensing, social distancing, sending alerts via messages and application whether a person is wearing a mask or not.

## FUTURE ENHANCEMENTS

In future we hope to increase the efficiency of our system. The proposed model can also be enhanced by means of including various parameters like peoples count, social distance and temperature measurement. This also includes the mobile notification and email as a warning message for the person who don’t wear masks. This system will be very helpful and can be implemented in hospitals, airports, schools, colleges, shops, malls, theatres, apartments etc. and can be also be implemented for Covid free event management.

.

## REFERENCES

1. M. S. Ejaz, M. R. Islam, M. Sifatullah and A. Sarker, "Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition", 2019 1st International Conference on Advances in Science Engineering and Robotics Technology (ICASERT), pp. 1-5, 2019.
2. BOSHENG QIN and DONGXIAO LI, Identifying Facemask-wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID- 19, May 2020, [online] Available: <https://doi.org/10.21203/rs.3.rs-28668/v1>+].
3. Kaihan Lin, Huimin Zhao, Jujian Ly (&) Jin Zhan, Xiaoyong Liu, Rongjun Chen, Canyao Li, Zhihui Huang et al., "Face Detection and Segmentation with Generalized Intersection over Union Based on Mask R- CNN", Advances in Brain Inspired Cognitive System, pp. 106-116, 2020.
4. N. Ud Din, K. Javed, S. Bae and J. Yi, "A Novel GAN-Based Network for Unmasking of Masked Face", in IEEE Access, vol. 8, pp. 44276-44287, 2020.
5. Nicolae-Cătălin Ristea and Radu Tudor Ionescu, Are you wearing a mask? Improving mask detection from speech using augmentation by cycle-consistent GANs.
6. S. Ghosh, N. Das and M. Nasipuri, "Reshaping inputs for convolutional neural network: Some common and uncommon methods", Pattern Recognition, vol. 93, pp. 79-94, 2019.
7. M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation", Journal of Big Data, vol. 6, no. 1, pp. 2019.
8. C. Nwankpa, W. Ijomah, A. Gachagan and S. Marshall, "Activation Functions: Comparison Of Trends In Practice And Research For Deep Learning", 2020, [online] Available: https://arxiv.org/abs/1811.03378.2020.
9. D. Meena and R. Sharan, "An approach to face detection and recognition", 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), pp. 1-6, 2016
10. C. Kanan and G. Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?", PLoS ONE, vol. 7, no. 1, pp. e29740, 2012.
11. B. Suvarnamukhi and M. Seshashayee, "Big Data Concepts and Techniques in Data Processing", International Journal of Computer Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018.
12. F. Hohman, M. Kahng, R. Pienta and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers", IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, Aug. 2019.
13. M. Jiang, X. Fan and H. Yan, "RetinaMask: A Face Mask detector", 2020, [online] Available: arXiv.org.
14. R. Yamashita, M. Nishio, R. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018.
15. S. Ge, J. Li, Q. Ye and Z. Luo, "Detecting Masked Faces in the Wild with LLE-CNNs", 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 426-434, 2017.