

# **PREDICTION OF FACE MASK AND SOCIAL DISTANCING**

Submitted in partial fulfillment of the requirements for  
the award of  
Bachelor of Technology degree in Computer Science and Engineering

by  
Gowtham (37110061)  
Sampath ( 37110850)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**SCHOOL OF COMPUTING**

**SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC  
**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,  
CHENNAI - 600 119**

**MARCH - 2021**



# **SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY  
**(DEEMED TO BE UNIVERSITY)**  
Accredited with "A" grade by NAAC  
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600  
119  
[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

## **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of Gowtham (37110061) and Sampath (37110850) who carried out the project entitled "Prediction Of Face Mask And Social Distancing" under our supervision from November 2020 to April 2021.

### **Internal Guide**

Dr. R. Jeberson Retna Raj

### **Head of the Department**

Dr. S. VIGNESHWARI, M.E., Ph.D.

Dr. L. Lakshmanan, M.E., Ph.D.

**Submitted for Viva voce Examination held on**

**Internal Examiner**

**External Examiner**

## **DECLARATION**

I Gowtham (37110061) and Sampath ( 37110850) hereby declare that the Project Report entitled "Prediction Of Face Mask And Social Distancing " done by me under the guidance of **Dr. R.Jeberson Retna Raj .**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

**DATE:**

**PLACE:**

**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing , **Dr.S.Vigneshwari M.E., Ph.D.,** and **Dr.L.Lakshmanan M.E., Ph.D,** **Head** of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. R.Jeberson Retna Raj.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

## ABSTRACT

The COVID-19 pandemic forced governments across the world to impose lockdowns to prevent virus transmissions. Reports indicate that wearing facemasks while at work clearly reduces the risk of transmission. An efficient and economic approach of using AI to create a safe environment in a manufacturing setup. A hybrid model using deep and classical machine learning for face mask detection will be presented. A face mask detection dataset consists of with mask and without mask images , we are going to use OpenCV to do real-time face detection from a live stream via our webcam. We will use the dataset to build a COVID-19 face mask detector with computer vision using Python, OpenCV, and Tensor Flow and Keras. Our goal is to identify whether the person on image/video stream is wearing a face mask or not with the help of computer vision and deep learning. Coronavirus disease 2019 has become a major health problem. It is spreading very widely due to its contact transparent behavior. So WHO declared wearing the mask in crowded areas as a prevention method. In some of the areas, the diseases become widely spread out due to improper wearing of facial masks. So to overcome this problem we required an efficient mask monitoring system. We propose a high- precision and effective MobileFaceMask face-mask detector in this article. The proposed MobileFaceMask is a one-step Pyramid detector, which combines high-quality semantime information, multi-function maps, and a new PC module for facial mask detection. Besides, we propose a new algorithm for the removal of cross-class objects to reject low-confidence predictions and high union intersection. Experiments show that MobileFaceMask delivers good results with 2.3% and 1.5% higher face and mask detection accuracy, while 11.0% and 5.9% more than baselines, respectively, are used in a public face mask dataset. Furthermore, we discuss the possibility of deploying MobileFaceMask for embedded or mobile devices with the lightweight MobileNet neural network.

Keywords: Convolutional Neural Networks, Deep learning, Image Processing, OpenCV, Tensor flow.

# Table Of Contents

|                                                                |    |
|----------------------------------------------------------------|----|
| CHAPTER 1                                                      | 8  |
| INTRODUCTION                                                   | 9  |
| 1.1 MACHINE LEARNING                                           | 10 |
| 1.2 COMPUTER VISION                                            | 12 |
| 1.3 DEEP LEARNING                                              | 14 |
| 1.4 OpenCV                                                     | 15 |
| 1.5 Overview                                                   | 16 |
| 1.6 Applications of Face Mask Detection                        | 16 |
| 1.7 OBJECTIVES                                                 | 19 |
| CHAPTER 2                                                      | 20 |
| LITERATURE REVIEW                                              | 20 |
| 2.1 Bounding Box Image                                         | 24 |
| CHAPTER 3                                                      | 28 |
| METHODOLOGY                                                    | 28 |
| 3.1 Network Architecture                                       | 28 |
| 3.2 Transfer Learning                                          | 29 |
| 3.3 Context Attention Module                                   | 29 |
| 3.4 Loss Function                                              | 30 |
| 3.5 FACE MASK DETECTION WITH DIFFERENT ANN AND<br>COMPARE THEM | 31 |
| 3.6 Dataset                                                    | 34 |
| 3.7 Training                                                   | 34 |
| 3.8 Hyperparameters                                            | 35 |
| 3.9 Face detection using OPEN-CV DNN                           | 36 |
| 3.10. Classification of images using MobileNetV2               | 36 |
| 3.11 Algorithms explaining the complete pipeline               | 37 |
| CHAPTER 4                                                      | 39 |

|                                                           |    |
|-----------------------------------------------------------|----|
| COVID-19: FACE MASK DETECTION USING TENSORFLOW AND OPENCV | 39 |
| <b>Step 1: Data Visualization</b>                         | 40 |
| Step 2: Data Augmentation                                 | 40 |
| Step 3: Splitting the data                                | 40 |
| Step 4: Building the Model                                | 41 |
| Step 5: Pre-Training the CNN model                        | 41 |
| <b>Step 6: Training the CNN model</b>                     | 41 |
| Step 7: Labeling the Information                          | 42 |
| Step 8: Importing the Face detection Program              | 42 |
| Step 9: Detecting the Faces with and without Masks        | 43 |
| 4.1 CORPORATED PACKAGES                                   | 43 |
| 4.2 Pre-processing                                        | 44 |
| 4.3 FLOW DIAGRAM:                                         | 45 |
| 4.4 Data augmentation                                     | 45 |
| CHAPTER 5                                                 | 46 |
| EXPERIMENTAL RESULTS ANALYSIS                             | 46 |
| Chapter 6                                                 | 56 |
| CONCLUSION                                                | 56 |
| 6.1 Sample code                                           | 58 |
| 6.2 Screenshots                                           | 59 |
| 6.3 Plagirism report                                      | 60 |
| 6.4 REFERENCES                                            | 61 |

## **LIST OF ABBREVIATION**

AI – Artificial intelligence

IOT – Internet of things

ML – Machine learning

SARS – Severe acute respiratory syndrome

MERS – Middle east respiratory syndrome

WHO – World health organization

HOG – Histogram oriented gradients

SHIFT – Scale invariant feature transform

SSD – Single shot detector

RCNN – Region based convolutional network

OS – Operating system

FPN – Functional pyramid network

SVR – Support vector regressor

ROI – Region of interest

RPN – Regional personal network

YOLO – You only look once SSH

– Single stage headlines FNN –

Fast neural network

SHD – Single shot detector

BAIR – Berkeley artificial intelligence research

RAM – Random access memory

# CHAPTER 1

## INTRODUCTION

### General study

The trend of wearing face masks in public is rising due to the COVID-19 corona virus 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 hiding their faces. Scientists proofed that wearing face masks works on impeding COVID-19 transmission. COVID19 (known as corona virus) is the latest epidemic virus that hit the 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 corona virus 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, tensor flow 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.

## 1.1 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.

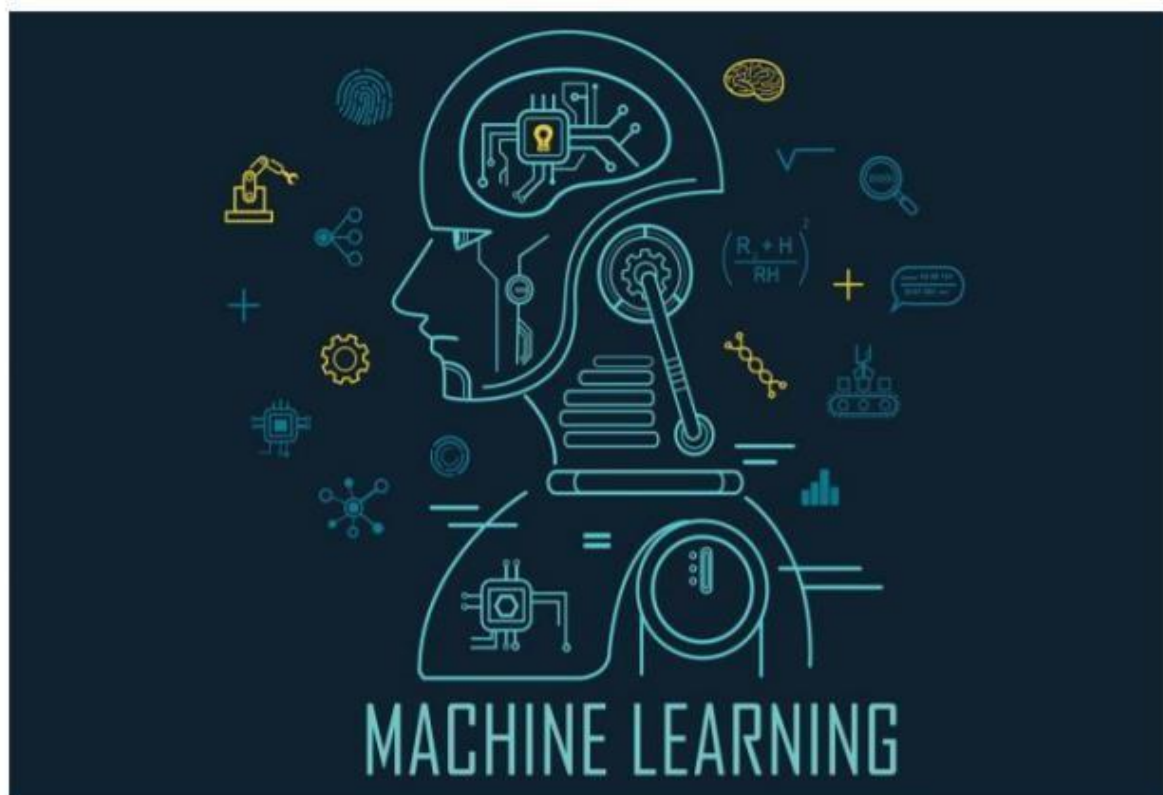


Fig -1: Machine learning outlook.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches have been developed which don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system.

The situation report 96 of world health organization (WHO) [1] presented that coronavirus disease 2019 (COVID-19) has globally infected over 2.7 million people and caused over 180,000 deaths. In addition, there are several similar large scale serious respiratory diseases, such as severe acute respiratory syndrome (SARS) and the Middle East respiratory syndrome (MERS), which occurred in the past few years [2, 3]. Liu et al. [4] reported that the reproductive number of COVID-19 is higher compared to the SARS. Therefore, more and more people are concerned about their health, and public health is considered as the top priority for governments [5]. Fortunately, Leung et al. [6] showed that the surgical face masks could cut the spread of coronavirus. At the moment, WHO recommends that people should wear face masks if they have respiratory symptoms, or they are taking care of the people with symptoms [7]. Furthermore, many public service providers require customers to use the service only if they wear masks [5]. Therefore, face mask detection has become a crucial computer vision task to help the global society, but research related to face mask detection is limited.

Face mask detection refers to detect whether a person wearing a mask or not and what is the location of the face [9]. The problem is closely related to general object detection to detect the classes of objects and face detection is to detect a particular class of objects, i.e. face [10, 11]. Applications of object and face detection can be found in many areas,



Figure 2: Examples of Images in the Face Mask Dataset.

such as autonomous driving [12], education [13], surveillance and so on [10]. Traditional object detectors are usually based on handcrafted feature extractors. Viola Jones detector uses Haar feature with integral image method [14], while other works adopt different feature extractors, such as histogram of oriented gradients (HOG), scale-invariant feature transform (SIFT) and so on [15]. Recently, deep learning based object detectors demonstrated excellent performance and dominate the development of modern object detectors. Without using prior knowledge for forming feature extractors, deep learning allows neural networks to learn features with an end- to-end manner [16]. There are one-stage and two-stage deep learning based object detectors. One-stage detectors use a single neural network to detect objects, such as single shot detector (SSD) [17] and you only look once (YOLO) [18]. In contrast, two-stage detectors utilize two networks to perform a coarse-to-fine detection, such as region-based convolutional neural network (R-CNN) [19] and faster R-CNN [20]. Similarly, face detection adopts similar architecture as general object detector, but adds more face related features, such as facial landmarks in RetinaFace [21], to improve face detection accuracy. However, there is rare research focusing on face mask detection.

## 1.2 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 highdimensional 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 (the input ofthe retina)into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructedwith the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multidimensional data from a 3D scanner or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.



Fig – 3: Computer Vision

Computer vision is concernedwith the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

The image data can take many forms, such as video sequences, views from multiple cameras, or multidimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

### 1.3 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 learned features defined in terms of lower-level features.

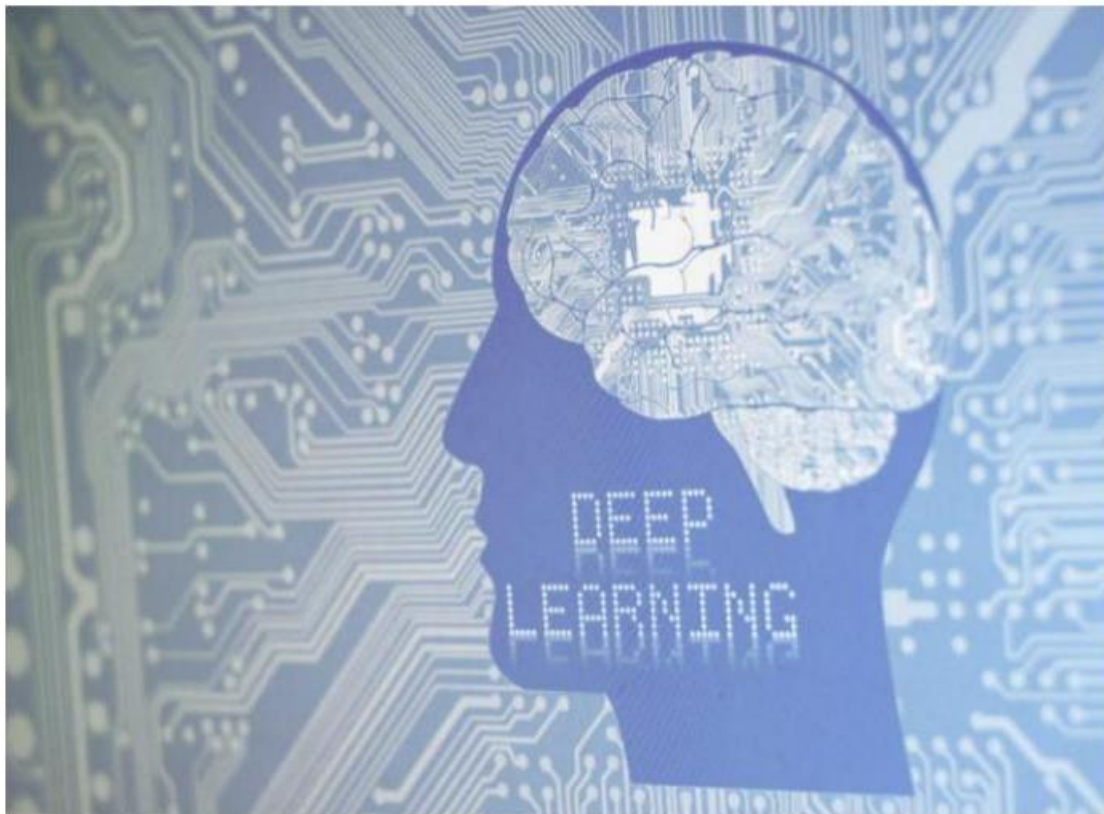


Fig – 4: Deep learning.

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are images of

pixel data, documents of text data or files of audio data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

## 1.4 OpenCV

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 the 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 of user community and 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. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

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.

## 1.5 Overview

The world is fighting with Covid19 pandemic. There are so many essential equipments needed to fight against Corona virus. One of such most essential is Face Mask. Firstly face mask was not mandatory for everyone but as the day progresses scientist and Doctors have recommended everyone to wear face mask. Now To detect whether a person is wearing Face Mask or not, we will use Face Mask Detection Technique. Face Mask Detection Platform utilizes Artificial Network to perceive if a person does/doesn't wear a mask. The application can be associated with any current or new IP cameras to identify individuals with/without a mask. In this blog we will see many important aspect of face mask detection not only for Covid19 cases but also for other regular cases.

### **Types of Face Masks available:**

Basically there are two types of face masks are being used as Corona virus Protecting Gear.

1. N-95 Masks
2. General Surgical Masks

To stop the airborne particles (Corona Virus) from the infected person's sneezing or coughing Face Masks are very much needed. It is applicable for everyone who is Covid19 positive, Covid19 negative, Doctors, Nurse and all other health staffs along with their PPE kit.

## 1.6 Applications of Face Mask Detection

There are several applications of face mask detection. Which includes the applications which are specific to covid19 and others are not related to covid19. So let us take a look at all the applications of Face Mask Detection.

### **1. Hospitals**

Utilizing Face Mask Detection System, Hospitals can screen if their workers are wearing masks during their work day or not. In the event that any health worker is found without a mask, they will get a warning with a suggestion to wear a face mask. Likewise, if quarantine individuals who must wear a mask, the system can keep an eye and distinguish if the mask is available or not and send warning automatically or report to the authorities.

### **2. Airports**

The Face Mask Detection System can be utilized at airports to recognize tourists/individual without veils. Face information of tourists can be caught in the system at the passageway. On the off chance that a tourist is seen as without a face cover, their image is sent to the airport authorities so they could make fast move. If the individual's face is stored in the database, for example the face of any worker of an airport, an alert can be sent to the person's phone directly.

### **3. Offices**

The Face Mask Detection System can be utilized at office area to recognize if employees are keeping up safety standards at work. It screens employees without masks and sends them a suggestion to wear a cover. The reports can be downloaded or sent an email toward the day's end to catch individuals who are not consenting to the guidelines or the prerequisites.

### **4. Educational Institutes**

In educational institutes especially where the number of children is more, it is highly recommended to wear face mask for children. Now sometimes children throw away the masks and that's very dangerous for them as it may tends to viral infection. So if we can use face mask detection in classrooms it will yield a great result.

So via this blog we have seen the different types of application of Face Mask Detection technique. Hope this will help us to understand the importance of wearing masks. We wish you all a good health and hope we will be able to a corona-free world sooner.

Face mask detection is a challenging task. It has been receiving more and more attention in this era due to the spreading of coronavirus disease. Hence many countries following the rule like "No entry without a mask". Face mask detection is a very important issue for security purposes and Covid-19 prevention. In the case of a medical career, the mask decreases the likelihood of an infectious person's future exposure to symptoms. Face mask detection is used in Airports, Hospitals, Offices, and Educational Departments, etc.

So face mask detection is become a very critical and challenging issue. Face recognition without the mask is easier but face recognition with a mask is a critical one because feature extraction of the masked face is very complicated than a normal face. That is so many face features such as nose, mouth, and chin are absent in the masked face. In medicine, masks minimize the possible risk of exposure of an infected individual to symptoms. So many face mask detection can be concentrated in two steps.

## 1) Face Recognition

## 2) Feature Extraction

The first step is to recognize the face; here the face of an image must be detected. Mainly there is a problem such as detecting the multiple masks and unmasked faces in an image.

The 2019 coronaviral disease (COVID-19) has affected around 2.7 million people worldwide and caused 180,000 deaths in the 96 situation survey of the worldwide health organization (WHO). Several similar major respiratory disorders have occurred, For example, in recent years, severe acute respiratory syndrome (SARS) and Middle East respiratory syndrome (MERS)[2, 3]. Liu et al. [4] registered a higher COVID-19 reproductive number than SARS. As a consequence, an increased number of people are cared for and public health is seen as the highest priority of the government[5].

Fortunately, Leung et al. [6] have shown that coronaviral spread can be minimized by surgical masks. WHO is currently advising to wear facial masks while people are suffering from respiratory symptoms or treatment for those with symptoms[7]. Besides, many providers of public services only require consumers to wear masks[5]. Thus facial mask detection is a critical task in computer vision to promote the global community, but there is very little research into facial mask detection.

In this paper, MobileFaceMask has been suggested, a novel facial mask detector capable of identifying face masks and contributing to public health. MobileFaceMask is one of the first facial mask detectors in the best possible way. MobileFaceMask uses several feature maps for the network architecture and uses FPN (Functional Pyramid Network) to merge high-level semantic knowledge into the network.

We suggest a background attention detection head and an algorithm to delete artifacts to improve detection capacity to achieve better detection. Furthermore, We use transfer learning to transfer learning from networks to learning kernels, which we work with for similar facial recognition tasks on a large dataset, Because the mask is a small data set with functions which it is difficult to extract. The approach is assessed on the basis of a mask dataset[8]. The dataset contains several face pictures, masked or unmasked, including masked faces, maskless faces, maskless faces, picture faces, and masks and confusing images. Results from the studies reveal that MobileFaceMask provides cutting-edge results which are 2.3 percent higher and 1.5% higher than the baseline and are 11.0 percent and 5.9 percent higher than the baseline.

## 1.7 OBJECTIVES

- To build a real-time system using Artificial Intelligence algorithms to detect whether the person on the webcam is wearing a mask or not.
- This project can efficiently detect the presence of masks in front of the webcam.

## CHAPTER 2

### LITERATURE REVIEW

Generally, most of the projects specialize in face construction identity recognition when wearing mask. During this projects, the focus is on recognizing the people that wearing mask, or not help in decreasing the transmission and spreading of covid-19. The scientist has proven that wearing a mask help in minimizing the spreading rate of Covid-19.

In [1], the authors developed a face mask wearing condition identification method. They were ready to classify three categories of face mask-wearing. The categories are face maskwearing, incorrect face mask-wearing and no face mask-wearing. Saber et al [2], have applied the principal component analysis on masked and unmasked face recognition to acknowledge the person. Also, PCA was utilized in[3]. The author proposed a way that's used for removing glasses from human frontal faces. In[4], the authors used the YOLOv3 algorithm for face detection. YOLOv3 uses Darknet-53 because the backbone. Nizam et al [5] proposed a completely unique GAN-based network, which will automatically remove mask covering the face area and regenerate the image by building the missing hole. In [6], the authors presented a system for detecting the presence or absence of a compulsory medical mask within the OR. The general is to attenuate the false positive face detection as possible without missing mask detection so as to trigger alarms just for medical staff who don't wear a surgical mask. Shaik et al [7] used deep learning real-time face emotion classification and recognition. They used VGG-16 to classify seven countenance.

In earlier days face detection models are implemented using edge, line and centre near features and patterns are recognized from those feature. These approaches are used to find binary patterns locally. These approaches are very effective to deal with Gray-scale images and the computation effort required also very less.[1]

AdaBoost is a regression based classifier which is going to fit regression function on original data set even some miss classified objects waits also adjusted during back propagation to optimize the results[2]

Viola Jones Detector proposed an real time object model used to detect different classes of objects. It uses 24x24 base window size to evaluate any image with edge ,line and four rectangular features. Harr-like features are like convolutions to check weather given feature is

available in the image or not[3]. This model fail to work in when image brightness varies even it exhibits poor performance when images are in different orientations.

Convolution networks are mainly used for classification problems there are various kinds of CNN architectures such as VGG-16 this architecture consists of 2 convolution layers with input size 224 kernel (64,3x3) followed by max pool with size 2x2 then again two convolution layers followed by max pool then three convolution layers with max pool again three convolution layers and max pool and three fully connected layers final FC is soft max this architecture works fine when compared to AlexNet[4,7].

Google Net architecture fundamentally using inception method by constructing small convolution layers to reduce number of parameters it having around 22 layers with convolution and max pooling etc it can able to work effectively over Alexnet it can able to bring down 60 million features in Alex net to 4 million features[8]. In this paper Deepnueral networks which adopts residual learning to train the models more deeper around 152 layers are used in this which is 8 times more than VGGnet with minimum complexity. This approach achieved relatively better performance in object detection over COCO data set[9].

In this analysis UNet and SEnet are used to perform segmentation of heart ventricular segmentation . This model is arrange the weights in such a way like more weights are given to useful features and less weights are given to unimportant features.[10]Support vector machines are used to perform classification on objects which is going to build an equation for constructing line and classifies the objects based on the values mapping to this line. Semantic segmentation method was used to detect facial mask in this paper they have used VGG net for training and FCN is used to semantically segment the faces available in the image[11] performed experiments on multi parsing human data sets and achieved higher accuracy.

In this paper medical image processing was done. They have taken human brain images and are trained by using FCN to identify tumours very effectively in this paper rather than using 2D segmentation for detecting tumour we have used 3D segmentation[12]. Tumuluru,Lakshmi Ramani et. al.[13], used CNN model to detect human face which is used efficiently in security related applications. In this paper they have collected various facial features such as mouth, nose, eyes stored as facial template and used it for detecting difference between faces.

Malathi, J. et. al., [14] mainly focused on identifying forgery images used in different places like in social media, and other publicity required places. In this paper various techniques are

proposed to find out features of a forgery image like image spicing,copy move attack which

can be handled by using correlation analysis to find duplicate features. Patelet. al., [15] proposed a model to find out the quality of the iron ore by extracting the features from sample material in the mining industry. It is very important to assess the quality of the ore. SVR support vector regressor used for online measure of the quality of the ore. In this process they have extracted 280 features extracted for object identification, SFFS was model was developed using SVR.

Object detection become the important area in the field of image investigation there are various techniques are there for image analysis in [15]. In this paper author introduced a wavelet based neural network for feature extraction and learning which is working efficiently in object detection.

Satapathy, Sandeep Kumar, et al., [16] proposed a model to detect number plate which is very important problem helping police to chase many criminal cases. Authors used OCR based approach to detect characters in the number plate and they are stored and processed to client server based model for collecting the details of the owner. Pathak et al., [17] proposed multi dimension biometric authentication system which will work effectively in low lightening conditions here accuracy was improved by using entropy based CNN.

Medical plant detection becoming very important problem which will help ordinary people to detect spices [18]. In this paper authors proposed a model using CNN. It was trained with medical leaf images and it can able to detect medical plants more accurately.

Human pose detection is one of the important research are nowadays drawing lot of attention [19]. In this paper author proposed a model which is going to detect traditional dance based on human pose. To achieve this, they have used CNN and various steps in traditional dance was trained and model learned from it and effectively able to detect traditional dance. Ravi, Sunitha, et al. [20], sign language detection was implemented by training a CNN model which can able to detect signs in the real world video. Which is very much useful in the driver less cars also. Even it is useful in sign language in machine translation.

Joint angular displacement approach was used through CNN for further enhancing the capabilities of CNN to capture 3D motion sign language in real time which can be applied to many real time applications in these days [21]. Patel, Ashok Kumar et. al., [22] proposed a model to find out the quality of the iron ore by extracting the features from sample material in the mining industry. It is very important to assess the quality of the ore. SVR support vector regressor used for online measure of the quality of the ore. In this process they have extracted 280 features extracted for object identification, SFFS was model was developed using SVR.

Under the present Covid-19 lock-in time, this technique is effective in preventing spread in many use cases. Here are some use cases which will benefit from the system.

**Airports:** the proposed system could also be vital for finding travelers at airports. There's no mask. The traveler's data are often captured as a video within the system at the doorway. Any passenger who finds no mask will alert the airport authorities in order that they can act quickly.

**Hospital:** the proposed system are often integrated with CCTV cameras, and therefore the data are often managed to ascertain if its employees are wearing masks. If you discover some doctors

. If they aren't wearing a mask, they're going to receive a reminder to wear a mask.

**Office:** The proposed system can help to take care of safety standards to stop the spread of covid-19 or any such airborne disease. If some employees aren't wearing masks, they're going to receive reminders to wear mask.

The choice of the system must be supported by the simplest performance. So, I'm using the simplest system performance indicators in order that you'll have large-scale implementation.

The system has been used with the MobileNetV2 classifier:

**MobileNetV2:** MobileNetV2 is that the latest technology of mobile visual recognition, including classification, object detection and semantic segmentation [24]. The classifier uses deep intelligent separable convolution, its purpose is to significantly reduce the complexity cost and model size of the network, so it's suitable for mobile devices, or devices with low computing power. In MobileNetV2, another best module introduced is that the reverse residual structure. The nonlinearity within the narrow layer is removed. Maintained because the backbone of feature extraction, MobileNetV2 achieves the simplest performance in object detection and semantic segmentation.

For MobileNetV2 classifier, ADAM optimizer has been applied to see performance:

**ADAM[13]:** Adam, a stochastic optimization algorithm supported step the target function is predicated on an adaptive estimation of low-order moments. In this manner it's computationally efficient and may be executed almost without memory. It's the diagonal of the gradient is rescaled unchanged, which is extremely suitable for the subsequent problems large in terms of knowledge and/or parameters. Hyper parameters are intuitive explain that they typically don't require much adjustment. The empirical results show that Adam it works well in

practice and may be compared with other stochastic optimization method.

Object detection is one of the trending topics in the field of image processing and computer vision. Ranging from small scale personal applications to large scale industrial applications, object detection and recognition is employed in a wide range of industries. Some examples include image retrieval, security and intelligence, OCR, medical imaging and agricultural monitoring.

## 2.1 Bounding Box Image

In object detection, an image is read and one or more objects in that image are categorized. The location of those objects is also specified by a boundary called the bounding box. Traditionally, researchers used pattern recognition to predict faces based on prior face models. A breakthrough face detection technology then was developed named as Viola Jones detector that was an optimized technique of using Haar [1], digital image features used in object recognition. However, it failed because it did not perform well on faces in dark areas and non-frontal faces. Since then, researchers are eager to develop new algorithms based on deep learning to improve the models. Deep learning allows us to learn features with end to end manner and removing the need to use prior knowledge for forming feature extractors. There are various methods of object detection based on deep learning which are divided into two categories: one stage and two stage object detectors.

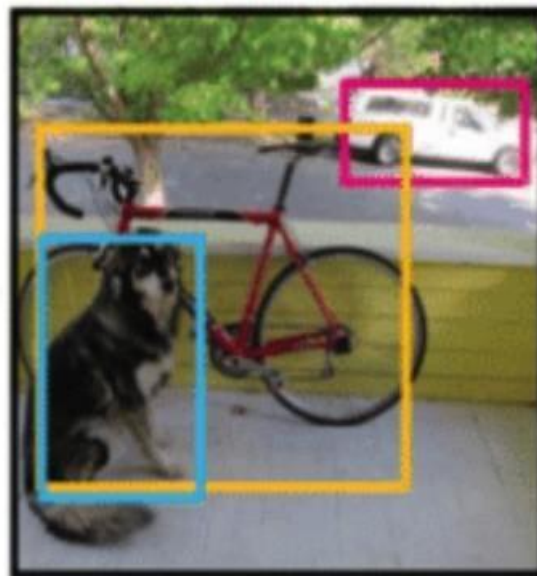


Figure 2.1: Bounding boxes in an image.

Two stage detectors use two neural networks to detect objects, for instance region-based convolutional neural networks (R-CNN) and faster R-CNN. The first neural network is used to generate region proposals and the second one refines these region proposals; performing a

coarse-to-fine detection. This strategy results in high detection performance compromising on speed. The seminal work R-CNN is proposed by R. Girshick et al. [2]. R-CNN uses selective search to propose some candidate regions which may contain objects. After that, the proposals are fed into a CNN model to extract features, and a support vector machine (SVM) is used to recognize classes of objects. However, the second stage of R-CNN is computationally expensive since the network has to detect proposals on a one-by-one manner and uses a separate SVM for final classification.

Fast R-CNN [3] solves this problem by introducing a region of interest (ROI) pooling layer to input all proposal regions at once. Faster RCNN [4] is the evolution of R-CNN and Fast R-CNN, and as the name implies its training and testing speed is greater than those of its predecessors. While R-CNN and Fast R-CNN use selective search algorithms limiting the detection speed, Faster R-CNN learns the proposed object regions itself using a region proposal network (RPN).

On the other hand, a one stage detector utilizes only a single neural network for region proposals and for detection; some primary ones being SSD (Single Shot Detection) [5] and YOLO (You Only Look Once) [6]. To achieve this, the bounding boxes should be predefined. YOLO divides the image into several cells and then matches the bounding boxes to objects for each cell.

This, however, is not good for small sized objects. Thus, multi scale detection is introduced in SSD which can detect objects of varying sizes in an image. Later, in order to improve detection accuracy, Lin et. al [7] proposes Retina Network (RetinaNet) by combining an SSD and FPN (feature pyramid network) to increase detection accuracy and reduce class imbalance. One- stage detectors have higher speed but trades off the detection performance but then only are preferred over two-stage detectors.

Like object detection, face detection adopts the same architectures as one-stage and two-stage detectors, but in order to improve face detection accuracy, more face-like features are being added. However, there is occasional research focusing on face mask detection. Some already existing face mask detectors have been modeled using OpenCV, Pytorch Lightning, MobileNet, RetinaNet and Support Vector Machines. Here, we will be discussing two projects. One project used Real World Masked Face Dataset (RMFD) which contains 5,000 masked faces of 525 people and 90,000 normal faces [8].

These images are 250 x 250 in dimensions and cover all races and ethnicities and are unbalanced. This project took 100 x 100 images as input, and therefore, transformed each sample image when querying it, by resizing it to 100x100. Moreover, this project uses PyTorch then they convert images to Tensors, which is the base data type that PyTorch can work with. RMFD is imbalanced (5,000 masked faces vs 90,000 non-masked faces). Therefore, the ratio of the samples in train/validation while splitting the dataset was kept equal using the train test split function of sklearn. Moreover, to deal with unbalanced data, they passed this information to the loss function to avoid unproportioned step sizes of the optimizer.

Like most of the projects, this project also used Adam optimizer. If any Model has a high rate of learning, it learns faster, but it bounces a lot to reach the global minima and may diverge from the global minima. However, a small learning rate may take considerably lower time to train, but it reaches to the global minima. If the loss of the model declines quickly for any learning rate, then that learning rate would be the best learning rate. However, it seems that this project considered the 0.00001 learning rate would be the best for their model so that it could work efficiently. To train the model they defined a model checkpointing callback where they wanted to save the best accuracy and the lowest loss.

They tried to train the model for 10 epochs and after finding optimal epoch, they saved the model for 8 epochs to test on the real data. To get rid of the problem of occlusions of the face which causes trouble face detectors to detect masks in the images, they used a built-in OpenCV deep learning face detection model. For instance, the Haar-Cascade model could be used but the problem of the Haar-Cascade model is that the detection frame is a rectangle, not a square. That is why, without capturing the portion of the background, the face frame can fit the entirety of the face, which can interfere with the face mask model predictions.

In the second project [9], a dataset was created by Prajna Bhandary using a PyImageSearch reader. This dataset consists of 1,376 images belonging to all races and is balanced. There are 690 images with masks and 686 without masks. Firstly, it took normal images of faces and then created a customized computer vision Python script to add face masks to them.

Thereby, it created a realworld applicable artificial dataset. This method used the facial landmarks which allow them to detect the different parts of the faces such as eyes, eyebrows, nose, mouth, jawline etc. To use the facial landmarks, it takes a picture of a person who is not wearing a mask, and, then, it detects the portion of that person's face. After knowing the location of the face in the image, it extracted the face Region of Interest (ROI).

In this project, embedded devices are used for deployment that could reduce the cost of manufacturing. MobileNetV2 architecture is used as it is a highly efficient architecture to apply on embedded devices with limited computational capacity such as Google Coral, NVIDIA Jetson Nano. This project performed well, however, if a large portion of the face is occluded by the mask, this model could not detect whether a person is wearing a mask or not. The dataset used to train the face detector did not have images of people wearing face masks as a result, if the large portion of faces is occluded, the face detector would probably fail to detect properly. To get rid of this problem, they should gather actual images of people wearing masks rather than artificially generated images.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Network Architecture

The architecture of proposed RetinaFaceMask is shown in Fig. 3.1. In order to design an effective network for face mask detection, we adopt the object detector framework proposed in, which suggests a detection network with a backbone, a neck and heads. The backbone refers to a general feature extractor made up of convolutional neural networks to extract information in images to feature maps. In RetinaFaceMask, we adopt ResNet as a standard backbone, but also include MobileNet as a backbone for comparison and for reducing computation and model size in deployment scenarios with limited computing resources. In terms of the neck, it is an intermediate component between a backbone and heads, and it can enhance or refine original feature maps. In RetinaFaceMask, FPN is applied as a neck, which can extract high-level semantic information and then fuse this information into previous layers' feature maps by adding operation with a coefficient.

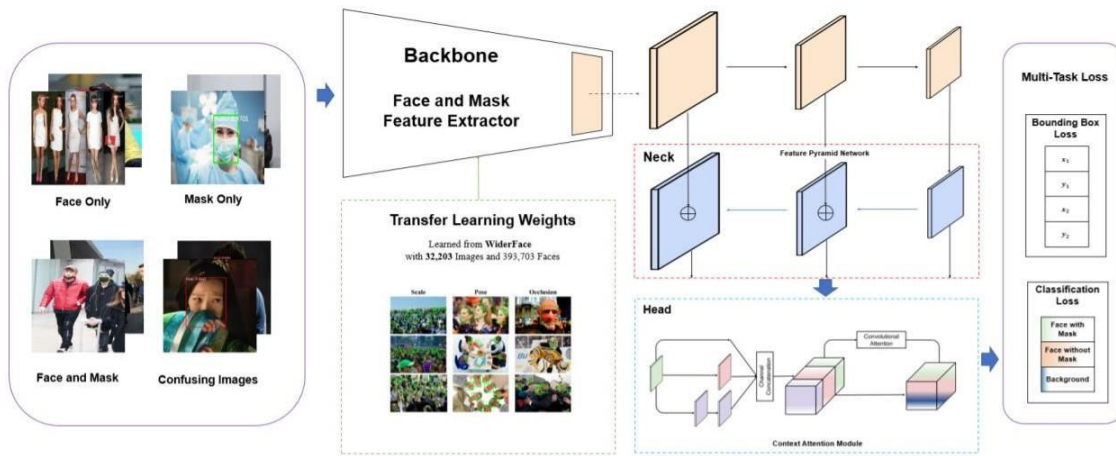


Figure 3.1: Architecture of RetinaFaceMask

Finally, heads stand for classifiers, predictors, estimators, etc., which are able to achieve the final objectives of the network. In RetinaFaceMask, we adopt a similar multi-scale detection strategy as SSD to make a prediction with multiple FPN feature maps, because it can have different receptive fields to detect various sizes of objects. Particularly, RetinaFaceMask utilizes three feature maps, and each of them is fed into a detection head. Inside each detection head, we further add a context attention module to adjust the size of receptive fields and focus on specific areas, which is similar to Single Stage Headless (SSH) but with an attention

mechanism. The output of the detection head is through a fully convolutional network rather than a fully connected network to further reduce the number of parameters in the network. We name the detector as RetinaFaceMask, since it follows the architecture of RetinaNet, which consists of a SSD and a FPN, and able to detect small face masks as well.

### 3.2 Transfer Learning

Due to the limited size of the face mask dataset, it is difficult for learning algorithms to learn better features. As deep learning based methods often require larger dataset, transfer learning is proposed to transfer learned knowledge from a source task to a related target task. According to [34], transfer learning has helped with the learning in a significant way as long as it has a close relationship. In our work, we use parts of the network pretrained on a large scale face detection dataset - Wider Face, which consists of 32,203 images and 393,703 annotated faces. In RetinaFaceMask, only the parameters of the backbone and neck are transferred from Wider Face, the heads are initialized by Kaiming's method. In addition, pretrained Imagenet [26] weights are considered as a standard initialization of our backbones in basic cases.

### 3.3 Context Attention Module

To improve the detection performance for face masks, RetinaFaceMask proposes a novel context attention module as its detection heads (Fig. 3.2). Similar to the context module in SSH, we utilize different sizes of kernels to form an Inception-like block. It is able to obtain different receptive fields from the same feature map, so it would be able to incorporate more different sizes of objects through concatenation operations. However, the original context module does not take into faces or masks into account, so we simply cascade an attention module CBAM after the original context module to allow RetinaFaceMask to focus on face and masks features. The context-aware part has three subbranches, including one  $3 \times 3$ , two  $3 \times 3$ , and three  $3 \times 3$  kernels in each branch individually. Then, the concatenated feature maps are fed into CBAM through a channel attention to select useful channels by a multi-layer perceptron and then a spatial attention to focus on important regions.

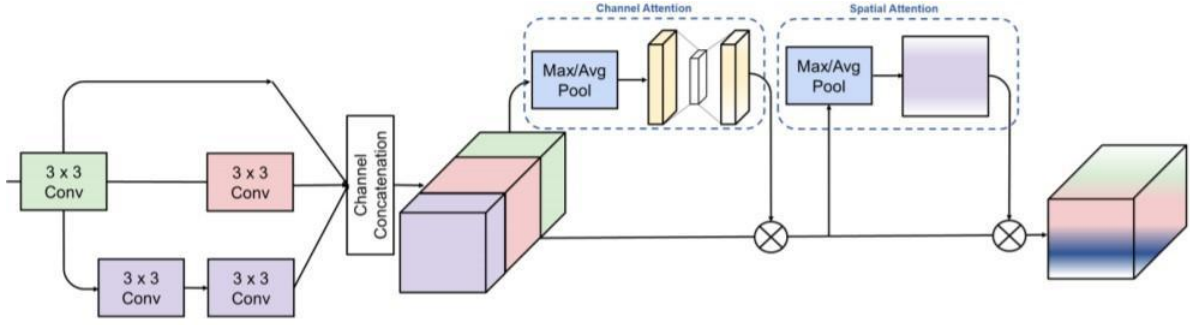


Figure 3.2: Context Attention Detection Head

### 3.4 Loss Function

RetinaFaceMask yields two outputs for each input image, localization offset prediction,  $Y_{loc} \in \mathbb{R}^{p \times 4}$  and classification prediction,  $Y_c \in \mathbb{R}^{p \times c}$ , where  $p$  and  $c$  denote the number of generated anchors and the number of classes. We also have the default anchors  $P \in \mathbb{R}^{p \times 4}$ , the ground truth boxes,  $Y_{loc} \in \mathbb{R}^{o \times 4}$  and the classification label  $Y_c \in \mathbb{R}^{o \times 1}$ , where  $o$  refers to the number of object.

The loss of the overall detection problem can be broken into two main subsets: localization loss and confidence loss. The localization loss is just the difference between the default predicted bounding box and ground truth bounding box ( $g$ ). For a given center ( $cx, cy$ ), we try to alter the width and height of the box such as to decrease the loss. Respectively, the equations for the localization and confidence losses can be defined as:

$$L_{loc}(x, l, g) = \sum_{i \in Pos} \sum_{m \in \{cx, cy, w, h\}} x_{ij}^k \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \quad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

And

$$L_{locf}(x, c) = - \sum_{i \in Pos} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

The notations used in the equations are as follows:

- g: ground truth bounding box
- l: predicted box
- d: default bounding box
- $x_{p_{ij}}$  : matching  $i$  th predicted box with  $j$  th default box with “p” category.
- $c_x, c_y$ : distance from centre of box in both x and y direction
- w, h = width and height with respect to image size
- c: confidence of presence of object or not.

Confidence loss is the just measure of how high the probability of the presence of an object is, when there exists an object. Similarly, the localization loss is the measure of how much a predicted box differs from the ground truth box in dimensions. Our model will try to push down both losses by predicting the presence of object and then correctly classifying it to the right class.

### 3.5 FACE MASK DETECTION WITH DIFFERENT ANN AND COMPARE THEM :

#### **Artificial neural network**

I am using different architecture and models of ANN were used for face mask detection. ANN can be used in face mask detection because these models can simulate the way neurons work in human brain. I comparison between different neural network for face mask recognition system and lastly we use those model which have better accuracy.

#### **Retinal connected of neural network (RCNN)**

I am presented face mask detection system based on a RCNN that examine small windows of an image to check each window contain face with or without mask. First, a preprocessing step, adapted from, is applied to window of the image. Then window is passed through the neural network, which decides whether the window contain face with or without mask. They used the two training dataset of image. In first dataset with mask images collected by me consist of total 800 images. The second dataset without mask consist of 750 images, The recognition face with mask and without mask equal to 80% accuracy rate.

#### **Principal Component Analysis with ANN:**

I am using PCA with class specific linear projection to detect or recognized face with or without mask in a real time video stream. The system steps to search for face with or without mask in an image:

1. Select every 20x20 region of input image
2. Use intensity values of its pixel as 400 inputs to ANN
3. Feed values is forward through ANN and
4. If the value is above 0.5 the region represent a face
5. Repeat steps several times, each time on a resized version of the original input image to search for faces at different scales.

### **Convolution Neural Network:**

In this planned method, the mask detection model is constructed victimization the successive API of the keras, library. this permits us to make the new layers for our model step by step. the assorted layers used for our CNN model is represented below. The 1st layer is that the Conv2D layer with one hundred filters and therefore the filter size or the kernel size of 3X3. During this first step, the activation operate used is the 'ReLU'. This ReLu function stands for corrected linear measure which is able to output the input directly if is positive, otherwise, it'll output zero. The input size is also initialized as 150X150X3 for all the photographs to be trained and tested victimization this model In the second layer, the MaxPooling2D is employed with the pool size of 2X2 The next layer is once more a Conv2D layer with another one hundred filters of constant filter size 3X3 and {also the} activation operate used is that the 'ReLU'. This Conv2D layer is followed by a MaxPooling3=2D layer with pool size 2X2.

In consecutive step, we have a tendency to use the Flatten () layer to flatten all the layers into one 1D layer. After the Flatten layer, we use the Dropout (0.5) layer to forestall the model from overfitting. Finally, towards the end, we have a tendency to use the Dense layer with fifty units and therefore the activation operate as 'ReLU'.

The last layer of our model are going to be another Dense Layer, with solely 2 units and the activation function used will be the 'Softmax' function. The softmax function outputs a vector which is able to represent the chance distributions of every of the input units. Here, two input units are used. The softmax function will output a vector with two probability distribution value.

## Fast Neural Networks (FNN)

The proposed FNN for the face mask detection. A FNN approach to reduce the computational time for locating human faces with or without mask. Each image is divided into small sub images and then each one is tested separately using a fast ANN. The experimental result of comparison with conventional neural network showed that high speed achieved when applying FNN.

Table 3.1: Comparison of neural network method

| s.no | Methodlogy   | Recognition rate(%) |
|------|--------------|---------------------|
| 1    | RCNN         | 90.45               |
| 2    | PCA with ANN | 95.67               |
| 3    | CNN          | 95.22               |
| 4    | FNN          | 94                  |

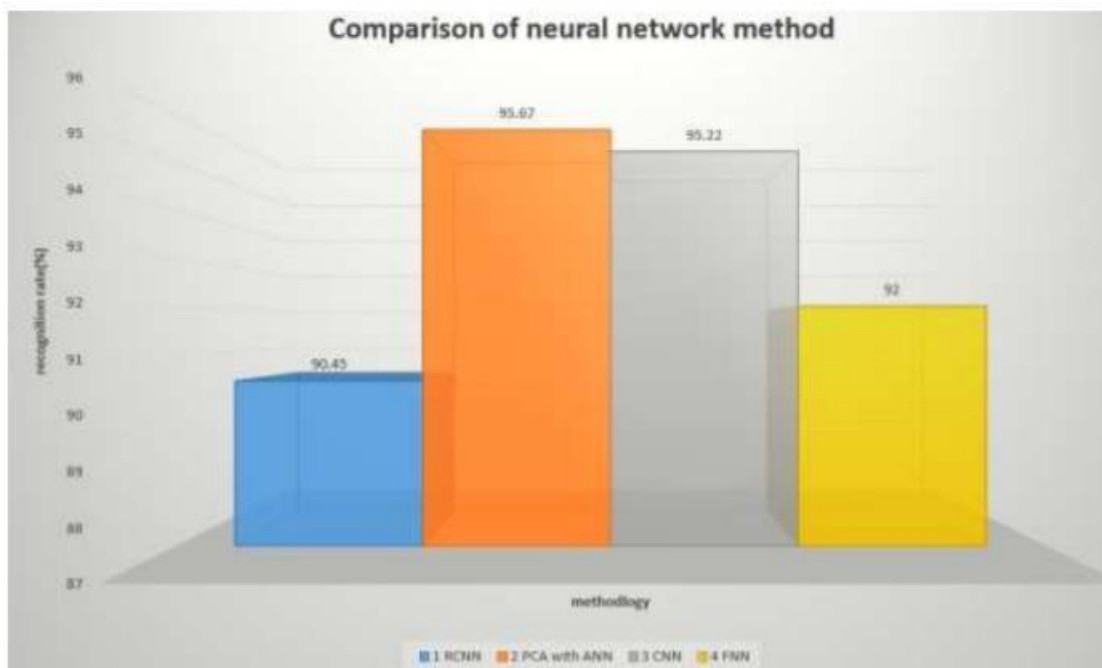


Figure 3.3: visualization of neural network method comparison.

After the comparison the two best methodology is CNN and PCA with AAN both recognition rate are approximately same then I am using for the CNN for the face mask detection mode.

### 3.6 Dataset

The dataset which we have used consists of 3835 total images out of which 1916 are of masked faces and 1919 are of unmasked faces. All the images are actual images extracted from Bing Search API, Kaggle datasets and RMFD dataset. From all the three sources, the proportion of the images is equal. The images cover diverse races i.e Asian, Caucasian etc. The proportion of masked to unmasked faces determine that the dataset is balanced.

We need to split our dataset into three parts: training dataset, test dataset and validation dataset. The purpose of splitting data is to avoid overfitting which is paying attention to minor details/noise which is not necessary and only optimizes the training dataset accuracy. We need a model that performs well on a dataset that it has never seen (test data), which is called generalization. The training set is the actual subset of the dataset that we use to train the model. The model observes and learns from this data and then optimizes its parameters. The validation dataset is used to select hyperparameters (learning rate, regularization parameters). When the model is performing well enough on our validation dataset, we can stop learning using a training dataset. The test set is the remaining subset of data used to provide an unbiased evaluation of a final model fit on the training dataset. Data is split as per a split ratio which is highly dependent on the type of model we are building and the dataset itself. If our dataset and model are such that a lot of training is required, then we use a larger chunk of the data just for training which is our case. If the model has a lot of hyperparameters that can be tuned, then we need to take a higher amount of validation dataset. Models with a smaller number of hyperparameters are easy to tune and update, and so we can take a smaller validation dataset.

In our approach, we have dedicated 80% of the dataset as the training data and the remaining 20% as the testing data, which makes the split ratio as 0.8:0.2 of train to test set. Out of the training data, we have used 20% as a validation data set. Overall, 64% of the dataset is used for training, 16% for validation and 20% for testing.

### 3.7 Training

At the training time, for each pixel, we compare the default bounding boxes having different sizes and aspect ratios with ground truth boxes and finally use Intersection over Union (IoU) method to select the best matching box. IoU evaluates how much part of our predicted box match with the ground reality. The values range from 0 to 1 and increasing values of IoU

determine the accuracies in the prediction; the best value being the highest value of IoU. The equation and pictorial description of IoU is given as follow:

$$IoU(B_1, B_2) = \frac{B_1 \cap B_2}{B_1 \cup B_2}$$

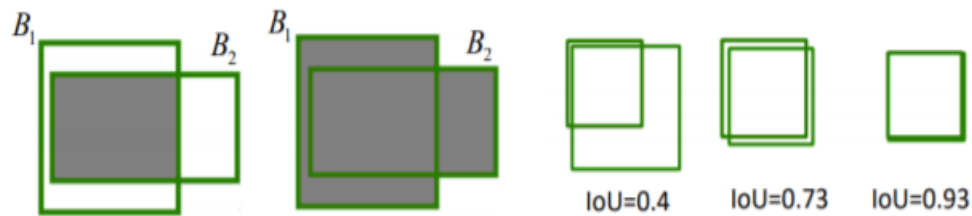


Figure 3.4: Pictorial representation of IoU.

### 3.8 Hyperparameters

A hyperparameter is a parameter or a variable we need to set before applying an algorithm into a dataset. These parameters express the “High Level” properties of the model such as its complexity or how fast it should learn. Hyperparameters are fixed before the actual training process begins. They can be divided into two categories: optimizer hyperparameters and model hyperparameters.

Optimizer parameters help us to tune or optimize our model before the actual training process starts. Some common optimizer hyperparameters are as follows. Learning rate is a hyperparameter that controls how much we are adjusting the weights of our neural network with respect to the gradient. Mini-batch size is a hyperparameter that influences the resource requirements of the training and impacts training speed and number of iterations. Epochs are the hyperparameters that determine the frequency of running the model. One epoch is when an entire dataset is passed forward and backward through the neural network only once. Model hyperparameters are parameters that are more involved in the architecture or structure of the model. They help us to define our model complexity based on the different layers like the input layer, hidden layer, and output layer of a neural network.

Initially, we trained with different values of hyperparameters by changing one and keeping the other constant and noted down the results in each case. We selected the hyperparameters that produced better performance through evaluation metrics. We have chosen the hyperparameters

as follows: initial learning rate is taken as 0.0001, batch size is taken to be 32 and number of epochs as 20. In our case, the target size is also one of the hyperparameters which we kept (224, 224, 3) as it is default input shape of MobileNetV2.

### 3.9 Face detection using OPEN-CV DNN

This model is included in the GitHub repository of OpenCV, starting from the latest version. It is established on the 'Single Shot Multi-box Detector' (SSD) and uses 'ResNet-10' architecture as the base-model. The Single Shot Multi-box Detector is similar to YOLO technique which takes only one shot to detect multiple objects present in an image using Multibox. It is significantly faster in speed and high-accuracy object detection algorithm. The images from which the model is trained have not been disclosed. Two versions of the model are made available by OpenCV:

- i.) CaffeImplementation (Jia et al., 2014) (Floating point 16 version)
- ii.) Original TensorFlow Implementation (8-bit quantized version)

Caffe is a deep learning framework developed as a more faster, powerful and efficient alternative as compared to other object detection methods and is created and managed by Berkeley AI Research (BAIR) and community contributors. We have used the Caffe model for our implementation for SSDMNv2 model to detect faces for the detection of facial masks. For this, the Caffe model and prototxt files were loaded using `cv2.dnn.readNet` ("path/to/prototxtfile", "path/to/caffemodelweights"). After applying the face detection model, we get the number of faces detected, the location of their bounding boxes, and the confidence score in those predictions. These outputs are then used as input for the face mask classifier. Using this approach to detect faces allows for realtime detection without much resource usage. It can also detect faces in different orientations, i.e., left, right, top, and bottom, with good accuracy. It is also able to detect the face mask of different sizes, i.e., big or small.

### 3.10. Classification of images using MobileNetV2

MobileNetV2 is a Deep Neural Network that has been deployed for the classification problem. Pretrained weights of ImageNet were loaded from TensorFlow. Then the base layers are frozen to avoid impairment of already learned features. Then new trainable layers are added, and these layers are trained on the collected dataset so that it can determine the features to classify a face wearing a mask from a face not wearing a mask. Then the model is fine-tuned, and then the weights are saved. Using pre-trained models helps avoid unnecessary computational costs

and helps in taking advantage of already biased weights without losing already learned features. This plan of action is shown in Fig. 3.5.

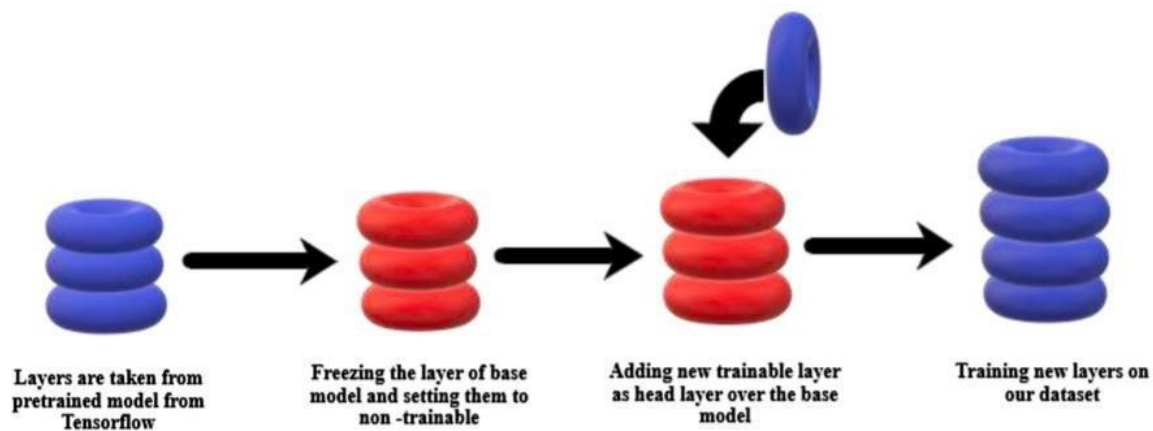


Fig. 3.5. Pipeline of using Pretrained Model.

### 3.11 Algorithms explaining the complete pipeline

The proposed SSDMNV2 methodology has been clearly explained using the two algorithms as shown below. First the images were preprocessed and were trained on the whole dataset. Second the model trained in the first part was used to detect the face mask with the appropriate accuracy. In Algorithm 1 shown below, images along with their pixel values were taken as an input, resized and normalized. Data augmentation technique was then applied on the images in order to get more accuracy. Data was then splitted into training and testing batches and MobilenetV2 model was applied on it. Adam optimizer was used to compile the whole model. In Algorithm 2 the model trained in previous part was then deployed on both classification on static images and on real time webcam. If the faces are detected using SSD, a bounding box showing the face of the person wearing a mask is shown in the output.

---

#### ALGORITHM 1: Pre-processing and Training on Dataset

**INPUT:** Images along with their pixels values

**OUTPUT:** Trained Model

**STEP 1:** Load Images and their pixel values.

**STEP 2:** Process the images, i.e., resizing, normalization, and conversion to a 1D array.

**STEP 3:** Load the Filenames and their respective labels.

**STEP 4:** Perform Data augmentation and then split data into training and testing batches.

**STEP 5:** Load MobilenetV2 model from Keras. Train it on training batches and compile it using Adam optimizer.

**STEP 6:** Save the model for future use.

---

---

**ALGORITHM 2: Deployment of Face Mask Detector**

**INPUT:** Choice of deployment and Files(optional).

**OUTPUT:** Images classified into the mask and no mask or Classification in Real-time.

**STEP 1:** Load saved classifier from disk. Also, load face detector from OpenCV.

**STEP 2:** If the choice is classification on image:

    Load Image(s)

**STEP 2.1:** Apply face detection model to Detect faces in an image

**STEP 2.2:** If Faces are detected:

    Crop face to bounding box coordinates from face detection model

    Get predictions from the face classifier model.

    Show predictions and save resultant image.

    Else:

        Show no output

**STEP 3:** If the choice is classification in real-time:

    Load real-time feed from OpenCV

    Read the feed frame by frame.

**STEP 3.1:** Apply face detection model to Detect faces in Frames read in real-time

**STEP 3.2:** If Faces are detected:

    Crop face to bounding box coordinates from face detection model

    Get predictions from the face classifier model.

    Show output in a real-time feed

    Else:

        Show normal feed

**STEP 4:** End stream when q is pressed

---

## CHAPTER 4

### COVID-19: FACE MASK DETECTION USING TENSORFLOW AND OPENCV

In these tough *COVID-19* times, wouldn't it be satisfying to do something related to it? I decided to build a very simple and basic Convolutional Neural Network (CNN) model using *TensorFlow* with *Keras* library and *OpenCV* to detect if you are wearing a face mask to protect yourself. Interesting! Isn't it?



Photo by [Macau Photo Agency](#) on [Unsplash](#)

For building this model, I will be using the face mask dataset provided by [Prajna Bhandary](#). It consists of about **1,376** images with **690** images containing people with face masks and **686** images containing people without face masks.

I am going to use these images to build a *CNN* model using TensorFlow to detect if you are wearing a face mask by using the *webcam* of your *PC*. Additionally, you can also use your *phone's camera* to do the same!

## Step 1: Data Visualization

In the first step, let us visualize the total number of images in our dataset in both categories. We can see that there are **690** images in the 'yes' class and **686** images in the 'no' class.

|                                                      |     |
|------------------------------------------------------|-----|
| The number of images with facemask labelled 'yes':   | 690 |
| The number of images without facemask labelled 'no': | 686 |

## Step 2: Data Augmentation

In the next step, we **augment** our dataset to include more number of images for our training. In this step of *data augmentation*, we *rotate* and *flip* each of the images in our dataset. We see that, after data augmentation, we have a total of **2751** images with **1380** images in the 'yes' class and **1371** images in the 'no' class.

|                                  |                     |                         |      |
|----------------------------------|---------------------|-------------------------|------|
| Percentage of positive examples: | 50.163576881134134% | number of pos examples: | 1380 |
| Percentage of negative examples: | 49.836423118865866% | number of neg examples: | 1371 |

## Step 3: Splitting the data

In this step, we **split** our data into the **training set** which will contain the images on which the CNN model will be trained and the **test set** with the images on which our model will be tested.

In this, we take **split\_size=0.8**, which means that **80%** of the total images will go to the *training set* and the remaining **20%** of the images will go to the *test set*.

|                                                                           |      |
|---------------------------------------------------------------------------|------|
| The number of images with facemask in the training set labelled 'yes':    | 1104 |
| The number of images without facemask in the training set labelled 'yes': | 276  |
| The number of images with facemask in the test set labelled 'no':         | 1096 |
| The number of images without facemask in the test set labelled 'no':      | 1096 |

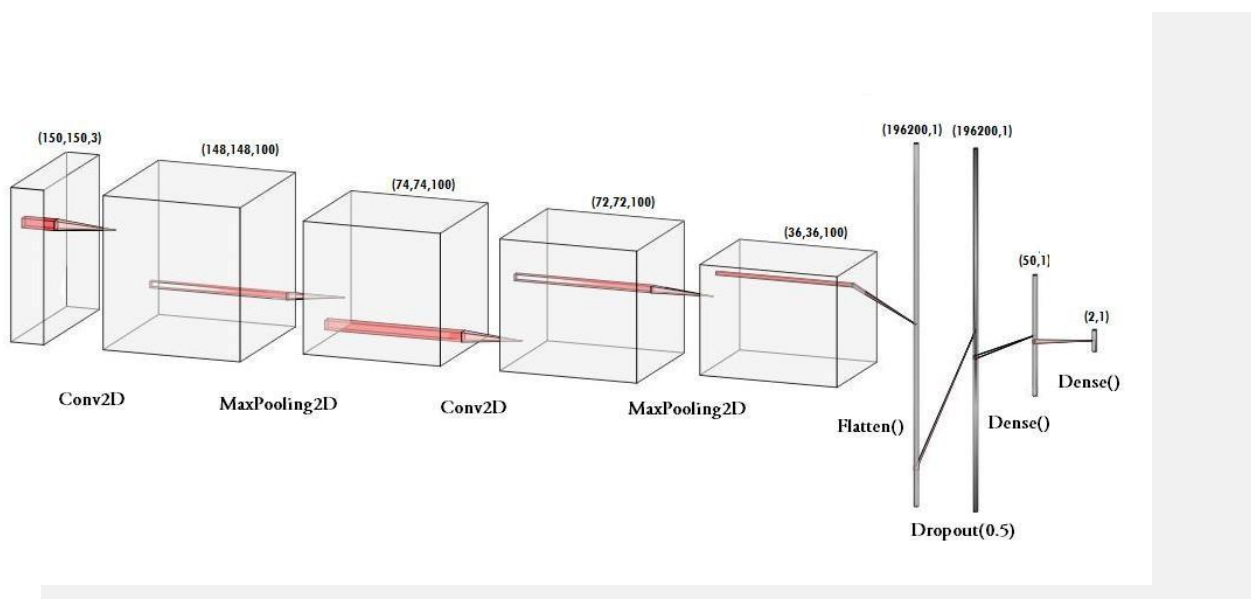
After splitting, we see that the desired percentage of images have been distributed to both the training set and the test set as mentioned above.

## Step 4: Building the Model

In the next step, we build our **Sequential CNN model** with various layers such as *Conv2D*, *MaxPooling2D*, *Flatten*, *Dropout* and *Dense*. In the last Dense layer, we use the ‘*softmax*’ function to output a vector that gives the *probability* of each of the two classes.

### Sequential Model for Face Mask detection

Here, we use the ‘*adam*’ optimizer and ‘*binary\_crossentropy*’ as our loss function as there are only two classes. Additionally, you can even use the *MobileNetV2* for better accuracy.



### CNN Model for Face Mask (Source — Self)

## Step 5: Pre-Training the CNN model

After building our model, let us create the ‘*train\_generator*’ and ‘*validation\_generator*’ to fit them to our model in the next step. We see that there are a total of **2200** images in the **training set** and **551** images in the **test set**.

|                                           |                                          |
|-------------------------------------------|------------------------------------------|
| Found 2200 images belonging to 2 classes. | Found 551 images belonging to 2 classes. |
|-------------------------------------------|------------------------------------------|

## Step 6: Training the CNN model

This step is the main step where we fit our images in the training set and the test set to our Sequential model we built using *keras* library. I have trained the model

for **30 epochs** (iterations). However, we can train for more number of epochs to attain higher accuracy lest there occurs *over-fitting*.

```
history = model.fit_generator(train_generator,
                             epochs=30,
                             validation_data=(validation_generator,
                                              validation_generator),
                             callbacks=[checkpoint])>>Epoch 30/30
220/220 [=====] - 231s 1s/step - loss: 0.0368 - acc: 0.9886 -
val_loss: 0.1072 - val_acc: 0.9619
```

We see that after the 30th epoch, our model has an accuracy of **98.86%** with the training set and an accuracy of **96.19%** with the test set. This implies that it is well trained without any over-fitting.

#### Step 7: Labeling the Information

After building the model, we label two probabilities for our results. [**'0'** as **'without\_mask'** and **'1'** as **'with\_mask'**]. I am also setting the boundary rectangle color using the RGB values. [**'RED'** for **'without\_mask'** and **'GREEN'** for **'with\_mask'**]

```
labels_dict={0:'without_mask',1:'with_mask'}
color_dict={0:(0,0,255),1:(0,255,0)}
```

#### Step 8: Importing the Face detection Program

After this, we intend to use it to detect if we are wearing a face mask using our PC's webcam. For this, first, we need to implement face detection. In this, I am using the *Haar Feature-based Cascade Classifiers* for detecting the features of the face.

```
face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

This cascade classifier is designed by *OpenCV* to detect the *frontal face* by training thousands of images. The .xml file for the same needs to be downloaded and used in detecting the face. I have uploaded the file in my GitHub repository.

## Step 9: Detecting the Faces with and without Masks

In the last step, we use the OpenCV library to run an infinite loop to use our web camera in which we detect the face using the *Cascade Classifier*. The code `webcam = cv2.VideoCapture(0)` denotes the usage of webcam.

The model will predict the possibility of each of the two classes ([without\_mask, with\_mask]). Based on which probability is higher, the label will be chosen and displayed around our faces.

## 4.1 CORPORATED PACKAGES

### A. TensorFlow

TensorFlow, an interface for expressing machine learning algorithms, is utilized for implementing ML systems into fabrication over a bunch of areas of computer science, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research [18]. In the proposed model, the whole Sequential CNN architecture (consists of several layers) uses TensorFlow at backend. It is also used to reshape the data (image) in the data processing.

### B. Keras

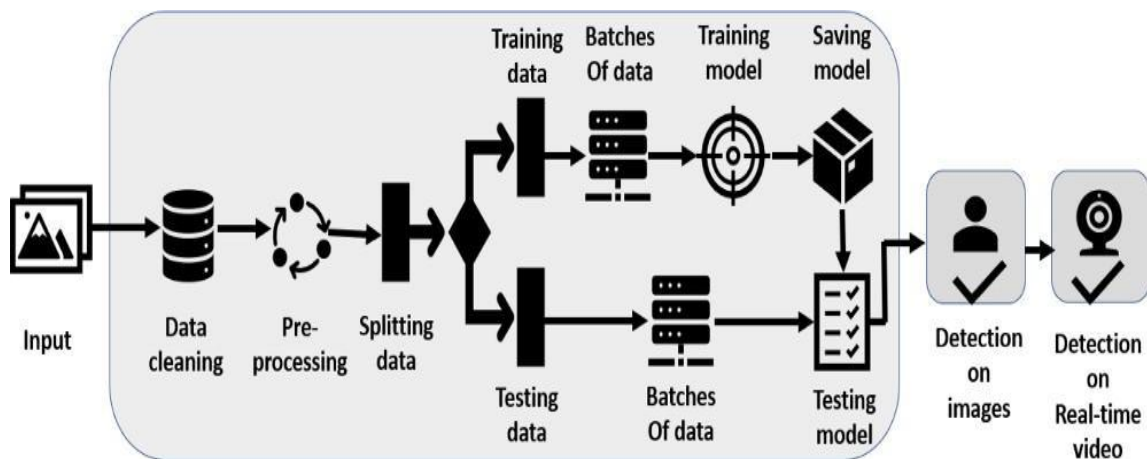
Keras gives fundamental reflections and building units for creation and transportation of ML arrangements with high iteration velocity. It takes full advantage of the scalability and cross-platform capabilities of TensorFlow. The core data structures of Keras are layers and models [19]. All the layers used in the CNN model are implemented using Keras. Along with the conversion of the class vector to the binary class matrix in data processing, it helps to compile the overall model.

### C. OpenCV

OpenCV (Open Source Computer Vision Library), an open-source computer vision and ML software library, is utilized to differentiate and recognize faces, recognize objects, group movements in recordings, trace progressive modules, follow eye gesture, track camera actions, expel red eyes from pictures taken utilizing flash, find comparative pictures from an image database, perceive landscape and set up markers to overlay it with increased reality and so forth [20]. The proposed method makes use of these features of OpenCV in resizing and color conversion of data image

## 4.2 Pre-processing

The Dataset from Masked face recognition and application contained a lot of noise, and a lot of repetitions were present in the images of this dataset. Since a good dataset dictates the accuracy of the model trained on it, so the data from the above-specified datasets were taken. They were then processed, and also all the repetitions were removed manually. The data cleaning was done manually to remove the corrupt images which were found in the said dataset. Finding these images was a vital part. As it is well known, the corrupt image was a tricky task, but due to valid efforts, we divided the work and cleaned the data set with mask images and without mask images. Cleaning, identifying, and correcting errors in a dataset removes adverse effects from any predictive model. This part explains the procedure of pre-processing the data and then training on data. First, we define a function name sorted alphanumerically to sort the list in lexicographical order. A function pre-processing is defined, which takes the folder to the dataset as input, then loads all the files from the folder and resizes the images according to the SSDMNv2 model. Then the list is sorted using sorted alphanumerically, and then the images are converted into tensors. Then the list is converted to NumPy array for faster calculation. After this, the process of data augmentation is applied to increase the accuracy after training the model.



**Fig. 4.1.Flow Diagram of the SSDMNv2 model.**

### 4.3 FLOW DIAGRAM:



Figure 4.2: Flow diagram for face mask detection

### 4.4 Data augmentation

For the training of SSDMNv2 model, an enormous quantity of data is necessary to perform training effectively since due to the nonavailability of an adequate amount of data for training the proposed model. The method of data augmentation is used to solve this issue. In this technique, methods like rotation, zooming, shifting, shearing, and flipping the picture are used for generating numerous versions of a similar picture. In the proposed model, image augmentation is used for the data augmentation process. A function image data generation is created for image augmentation, which returns test and train batches of data.

## CHAPTER 5

### EXPERIMENTAL RESULTS ANALYSIS

All the experimental trials have been conducted on a laptop equipped by an Intel i7-8750H processor (4.1 GHz), 16 GB of RAM with 1050ti max-Q with 4 GB of VRAM. The Jupyter Notebook software equipped with Python 3.8 kernel was selected in this research for the development and implementation of the different experimental trails. The metrics selected for evaluation of SSDMNV2 model is explained below.

$$Accuracy = \frac{Tp + Tn}{(Tp + Fp + Fn + Tn)}$$

$$Precision = \frac{Tp}{(Tp + Fp)}$$

$$Recall = \frac{Tp}{(Tp + Fn)}$$

$$f1 \text{ score} = 2 * \frac{Recall * Precision}{(Recall + Precision)}$$

Where Tp = True positive,

Tn = True negative,

Fp = False positive,

Fn = False negative

In above formulas, True positive values refer to images which were labelled true and after prediction by model gave true result. Likewise, for True negative refers to images which were labelled true but after prediction resulted in false result. False positive refers to images which were labelled false and after prediction resulted in false hence false positives. False negative refers to images which were labelled false and

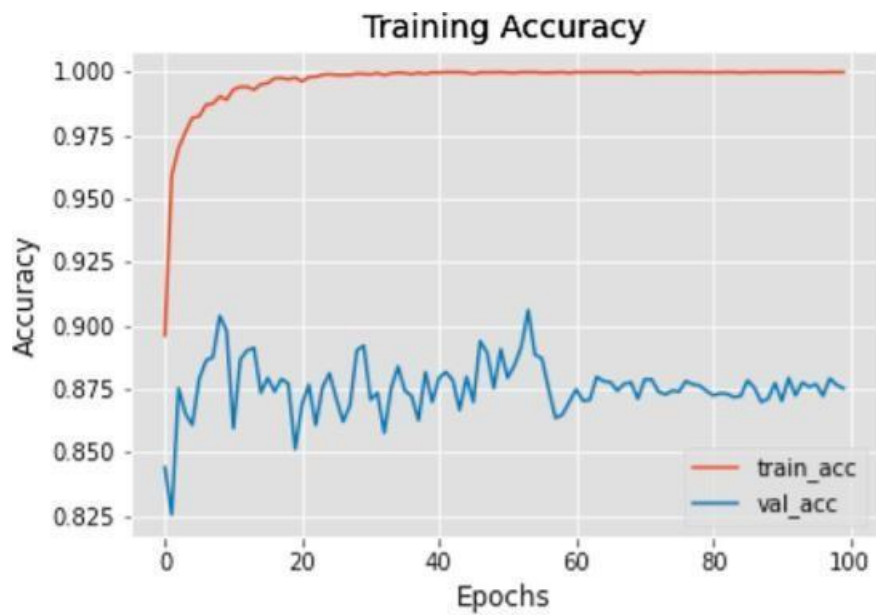


Fig. 5.1. Training accuracy curve (Without data augmentation).

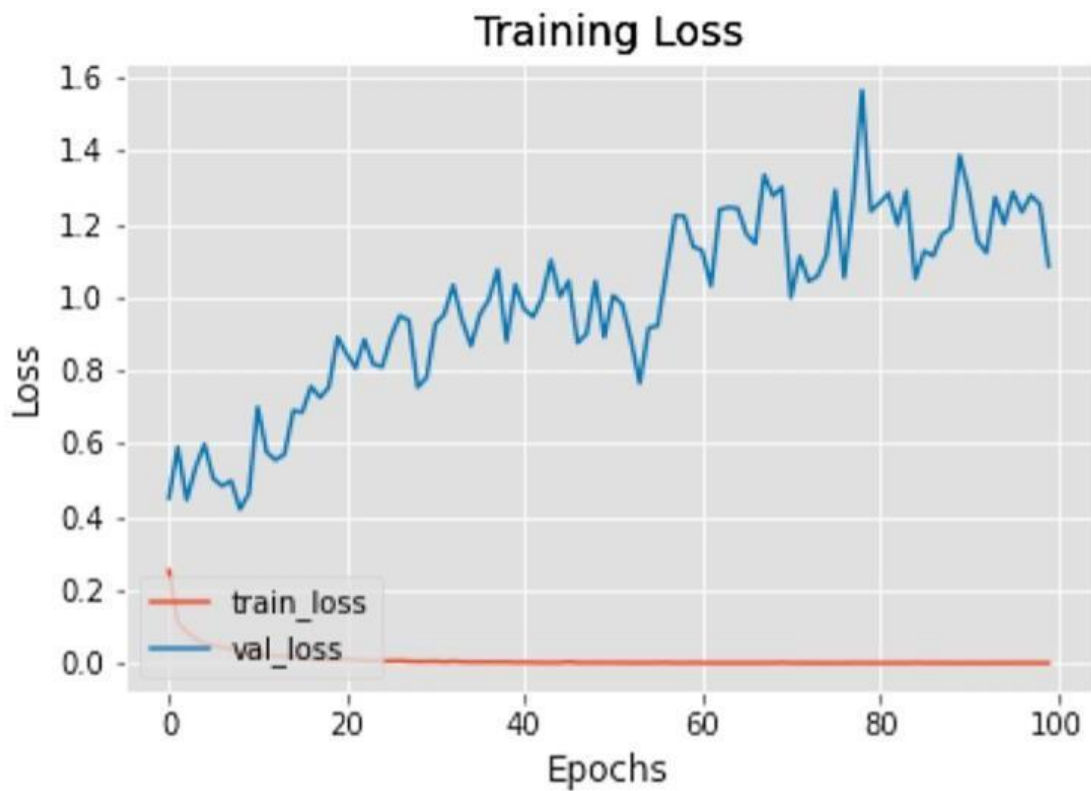


Fig. 5.2. Training loss curve (Without data augmentation)

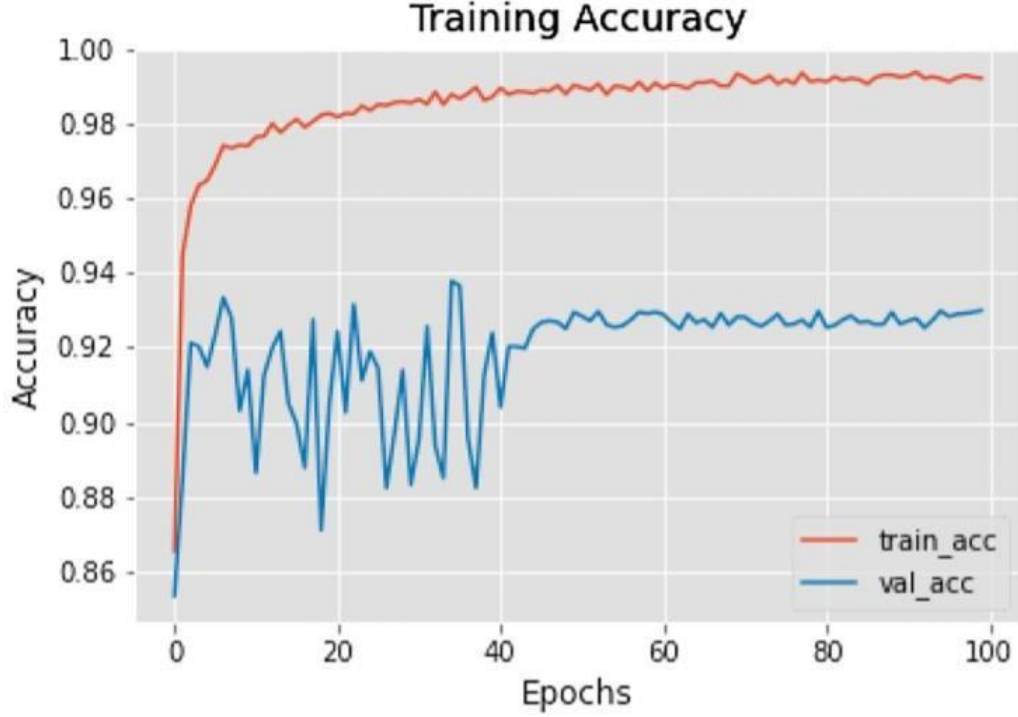


Fig. 5.3. Training accuracy curve.

The performance of RetinaFaceMask is compared with a public baseline result published by the creator of the dataset [8]. Due to the limited methods proposed for this dataset, we also used ResNet and MobileNet as different backbones for comparison. The accuracy is given in Table

5.1. Table 5.1 showed that RetinaFaceMask with ResNet-backbone achieved higher accuracy than with MobileNet-backbone by around 10% in precision and recall respectively. In addition, RetinaFaceMask with ResNet-backbone achieves the state-of-the-art result compared to the baseline model. Particularly, RetinaFaceMask is 2.3% and 1.5% higher than the baseline result in face and mask detection precision respectively, and 11.0% and 5.9% higher than baseline for recall.

Table 5.1: Detection Accuracy of RetinaFaceMask

| Model                    | Face      |        | Mask      |        |
|--------------------------|-----------|--------|-----------|--------|
|                          | Precision | Recall | Precision | Recall |
| baseline [8]             | 89.6%     | 85.3%  | 91.9%     | 88.6%  |
| RetinaFaceMask+MobileNet | 83.0%     | 95.6%  | 82.3%     | 89.1%  |
| RetinaFaceMask+ResNet    | 91.9%     | 96.3%  | 93.4%     | 94.5%  |

Some Experiment results are shown in Fig. 5.4, where the red and green boxes refer to the face and mask predictions, respectively. Figure 5.4 indicates that RetinaFaceMask can recognize the confusing face without mask.



Figure 5.4: Examples of Detection Results.



Figure 5.5: Loss graph for ResNet50.

After observing these results, we can conclude that the lowest performance amongst these was of ResNet50. Xception and MobileNetV2 both performed quite similar and to select the best from them was the 100% precision of unmasked faces by MobileNetV2. Since our detector can take no risk by falsely concluding that an unmasked face is a masked face, precision is the key metric to decide between these models. Since, the best performance was given by MobileNetV2, therefore, we decided to use MobileNetV2 as a base model for our face mask detector algorithm.

## Inference

We implemented our model on images containing one and more faces. We also implemented it on videos and live video streams by removing and wearing masks one by one. Some screenshots of the results are shown below:

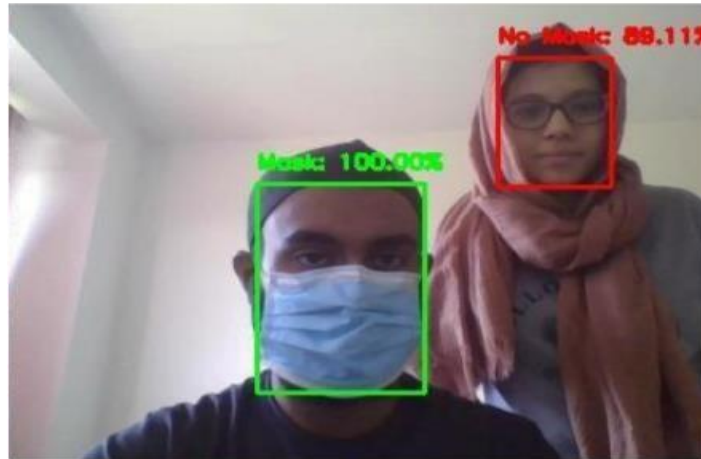


Figure 5.6: One person with mask and one person without mask.

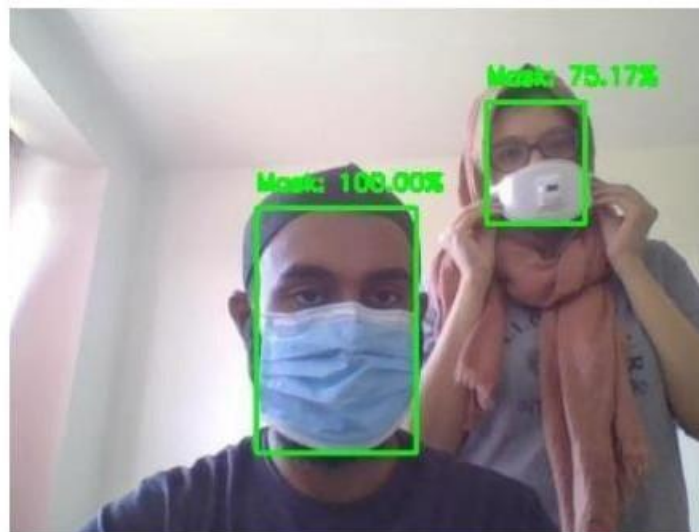


Figure 5.7: Both persons with masks.



Figure 5.8: Both persons without masks.

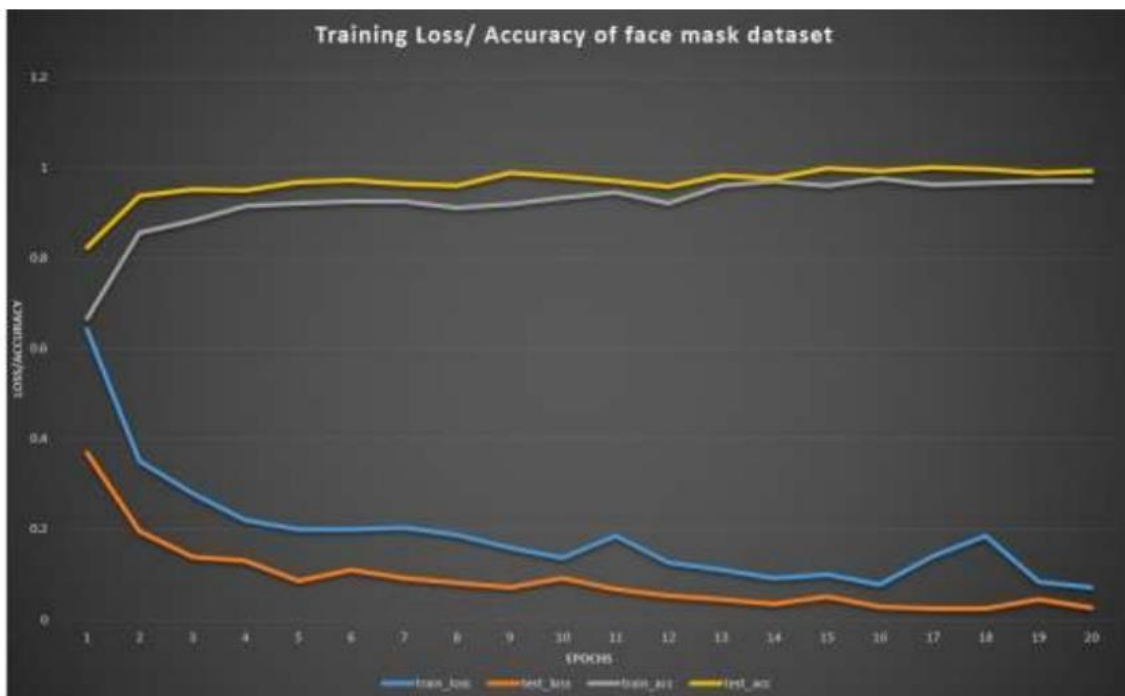


Figure 5.9: Training Loss/Accuracy curves of face mask detection dataset.

### Face Mask Classifier Performance Metrics

```

]: predict=model.predict(test_X,batch_size=85)
predict=np.argmax(predict,axis=1)
print(classification_report(test_Y.argmax(axis=1),predict,target_names=lb.classes_))

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| with_mask    | 0.96      | 0.99   | 0.98     | 138     |
| without_mask | 0.99      | 0.96   | 0.97     | 138     |
| accuracy     |           |        | 0.97     | 276     |
| macro avg    | 0.98      | 0.97   | 0.97     | 276     |
| weighted avg | 0.98      | 0.97   | 0.97     | 276     |

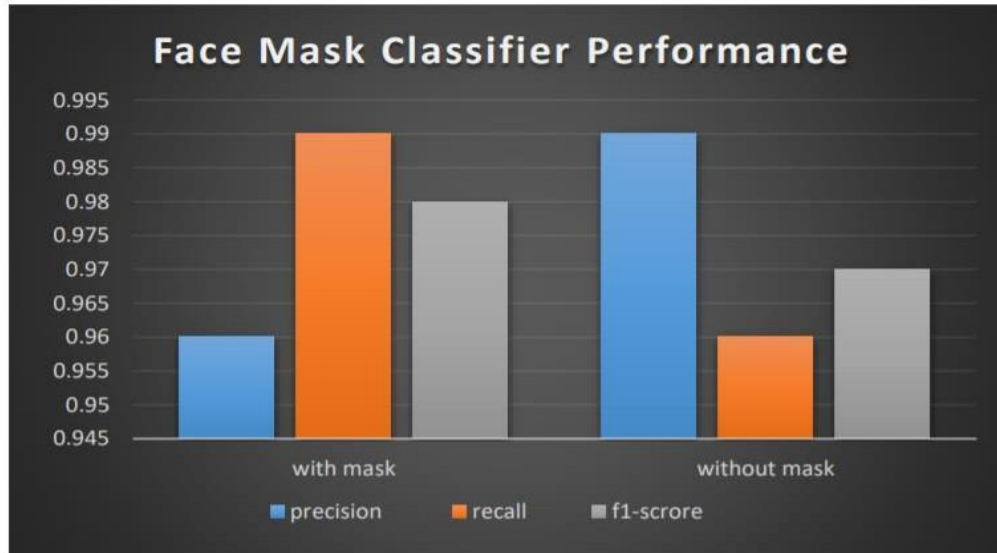


Fig 5.10: Performance Metrics Histogram graph.

## Dataset Statistics

After the annotation, we obtain 35, 806 masked faces with a minimum size of  $32 \times 32$ . The distribution of face sizes is shown in Fig. 4(a), and we find that a face in MAFA covers  $143 \times 143$  pixels on average and spreads across a large variety of sizes. Figure 4(b)-(d) illustrate the statistics of face attributes, including face poses, occlusion degrees and mask types. We can see that most faces in MAFA are front faces, while few faces are purely left or right. In this manner, face detectors that rely on the accurate localization of eyes can be tested as well, while the challenging cases such as left-front and right-front faces can be used to further assess their robustness.

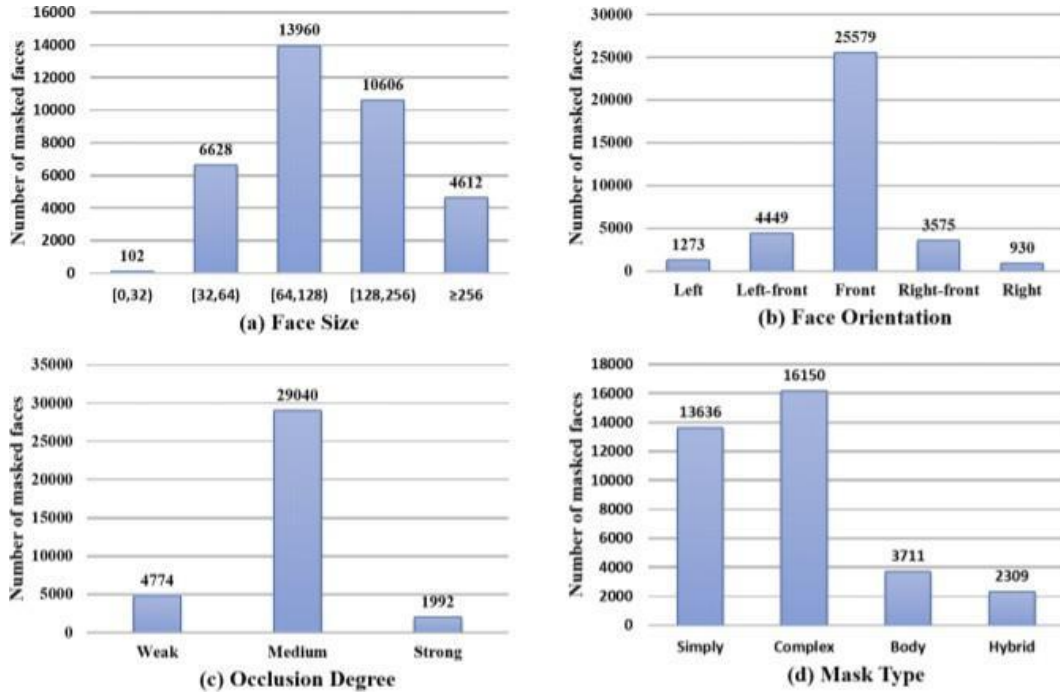


Figure 5.11. Statistics of the masked faces in MAFA

In addition, most faces have medium occlusion, which are also the most common cases we may encounter in cold weather and poor air condition. Furthermore, MAFA contains multiple types of masks (e.g., simple, complex and human body). These masks, which may bring in diversified ‘noise’ when constructing face descriptors, may provide us an opportunity to test the robustness of face detectors. To sum up, all these characteristics discussed above indicate that MAFA is a challenging dataset for face detection, and such a dataset can provide fair comparisons in benchmarking different face detectors.

### **Floating-point 16 version of the original Caffe implementation.**

2.8 bit quantized version using Tensor flow The Caffe model in this face mask detector. There has been a lot of discussion around deep learning based approaches for person detection. This encouraged us to come up with our own algorithm to solve this problem. Our work on facemask detection comprises of data collection to tackle the variance in kinds of face masks worn by the workers. The face mask detection model is a combination of face detection model to identify the existing faces from camera feeds and then running those faces through a mask detection model.



Fig – 5.12: Detection of “Mask” and “No Mask”.

To further investigate the performance of these models, we refer to the representative results shown in Fig. 6 and focus on the failures produced by the seven models. From these results, we find that the feature robustness might be a major cause of the impressive performance of our approach. By refining the noisy face descriptors extracted via VGGFace within the embedding module, the inherent characteristics of faces, even occluded by masks, can be captured by referring to the most representative normal faces (and nonfaces). In this manner, the reference faces (and non-faces) provide us valuable cues in recognizing a real face and refining its location and scale. By contrast, the boosting-based approaches often rely on the heuristic local features directly extracted from the masked face regions, which are less robust when heavy occlusion occurs. As a result, the performances of the two boosting-based face detectors are very low since the local features extracted from masks other than the facial regions are used for face detection.

Actually, feature robustness is with the highest importance in detecting masked faces. For example, NPD [20] uses a scale invariant features called normalized pixel difference and achieves an average precision of 19.6% since such simple features are often not reliable in masked facial regions. On the contrary, all the three DPM-based approaches, including ZR, HH and HPM, adopt the local HoG (Histogram of Oriented Gradients) descriptor [2]. As their DPM-based frameworks can well capture the robust topological information of facial regions from a global perspective, their performances are much better than NPD even in heavy occlusions and complex masks (see Fig. 5.13).

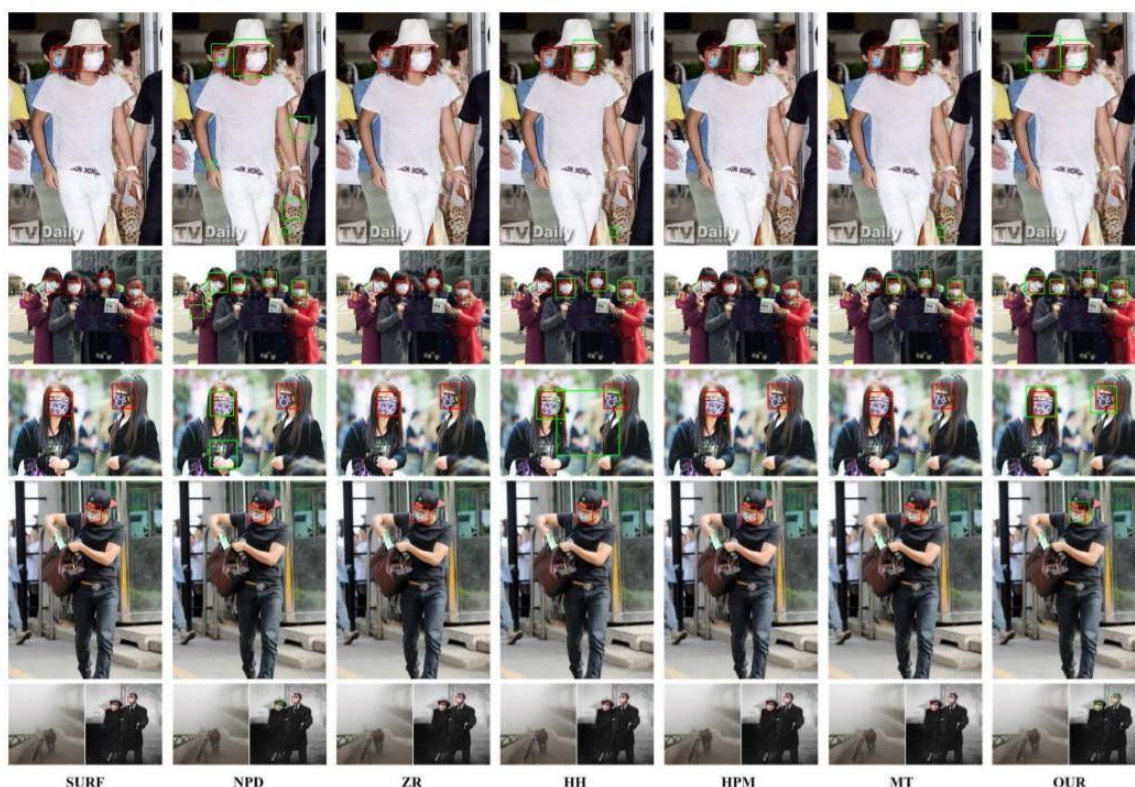


Figure 5.13. Representative detection results of our approach and 6 state-of-the-art face detectors on MAFA (Red: ground-truth, Green: detection results). We see that most face detectors have difficulties to accurately locate the side faces that are heavily occluded by masks.

Last but not least, the features in MT are learned from data. Even though such features are especially designed for masked faces, the overall performance of MT is still very impressive compared with the boosting-based and DPM-based approaches. To sum up, robust features play an important role in detecting masked faces, especially when most facial regions are occluded by masks. Compared with the heuristic features that are sensitive to masks and occlusions, robust topological feature or features learned from data are more robust in detecting faces. These results may point out a future direction of developing robust and effective face descriptors with the assistance of the MAFA dataset.

## Chapter 6

### CONCLUSION

To mitigate the spread of COVID-19 pandemic, measures must be taken. We have modeled a face mask detector using SSD architecture and transfer learning methods in neural networks. To train, validate and test the model, we used the dataset that consisted of 1916 masked faces images and 1919 unmasked faces images. These images were taken from various resources like Kaggle and RMFD datasets. The model was inferred on images and live video streams. To select a base model, we evaluated the metrics like accuracy, precision and recall and selected MobileNetV2 architecture with the best performance having 100% precision and 99% recall. It is also computationally efficient using MobileNetV2 which makes it easier to install the model to embedded systems. This face mask detector can be deployed in many areas like shopping malls, airports and other heavy traffic places to monitor the public and to avoid the spread of the disease by checking who is following basic rules and who is not.

The problem of face detection in the wild have been explored in many existing researches, and the corresponding face detectors have been tested on datasets of normal faces. On these datasets, some face detectors have achieved extremely high performances and it seems to be somehow difficult to further improve them. However, the ‘real wild’ scenarios are much more challenging than expected for containing faces captured at unexpected resolution, illumination and occlusion. In particular, the detection of masked faces is an important task that needs to be addressed so as to facilitate applications such as video surveillance.

As the technology are blooming with emerging trends the availability so we have novel face mask detector which can possibly contribute to public healthcare. The architecture consists of MobileNet as the backbone 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 , Pytorch 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 hyper parameters. 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.

The architecture of RetinaFaceMask consists of ResNet or MobileNet as the backbone, FPN as the neck, and context attention modules as the heads. The strong backbone ResNet and light backbone MobileNet can be used for high and low computation scenarios, respectively. 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. Furthermore, we have proposed a novel context attention head module to focus on the face and mask features, and a novel algorithm object removal cross class, i.e. ORCC, to remove objects with lower confidence and higher IoU. The proposed method achieves state-of-the-art results on a public face mask dataset, where we are 2.3% and 1.5% higher than the baseline result in the face and mask detection precision respectively, and 11.0% and 5.9% higher than baseline for recall.

## 6.1 Sample code

```
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
from playsound import playsound
import numpy as np
import argparse
import imutils
import time
import cv2
import os
from plyer import notification

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, (300, 300),
                                  (104.0, 177.0, 123.0))

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

    # 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 > args["confidence"]:
            # 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))

```

## 6.2 Screenshots

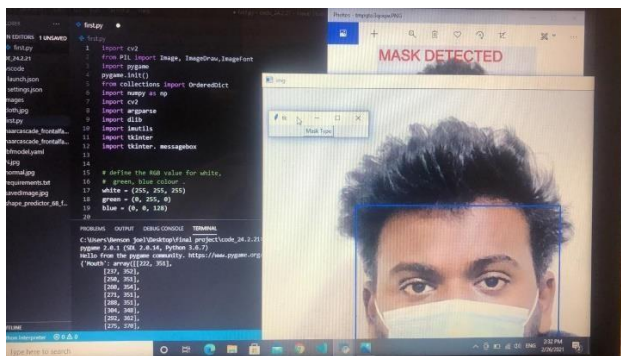


Fig-6.1

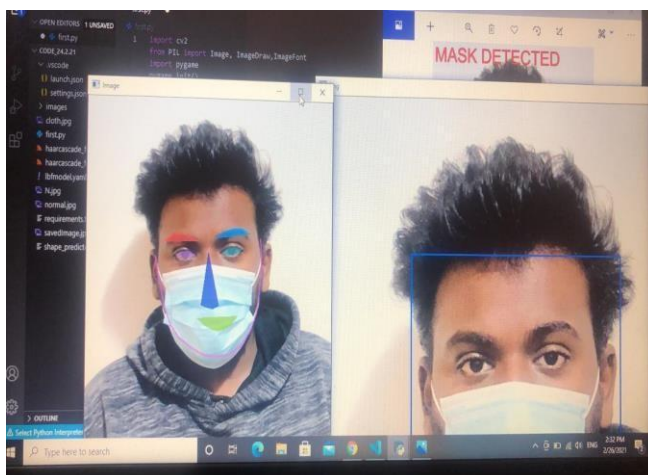
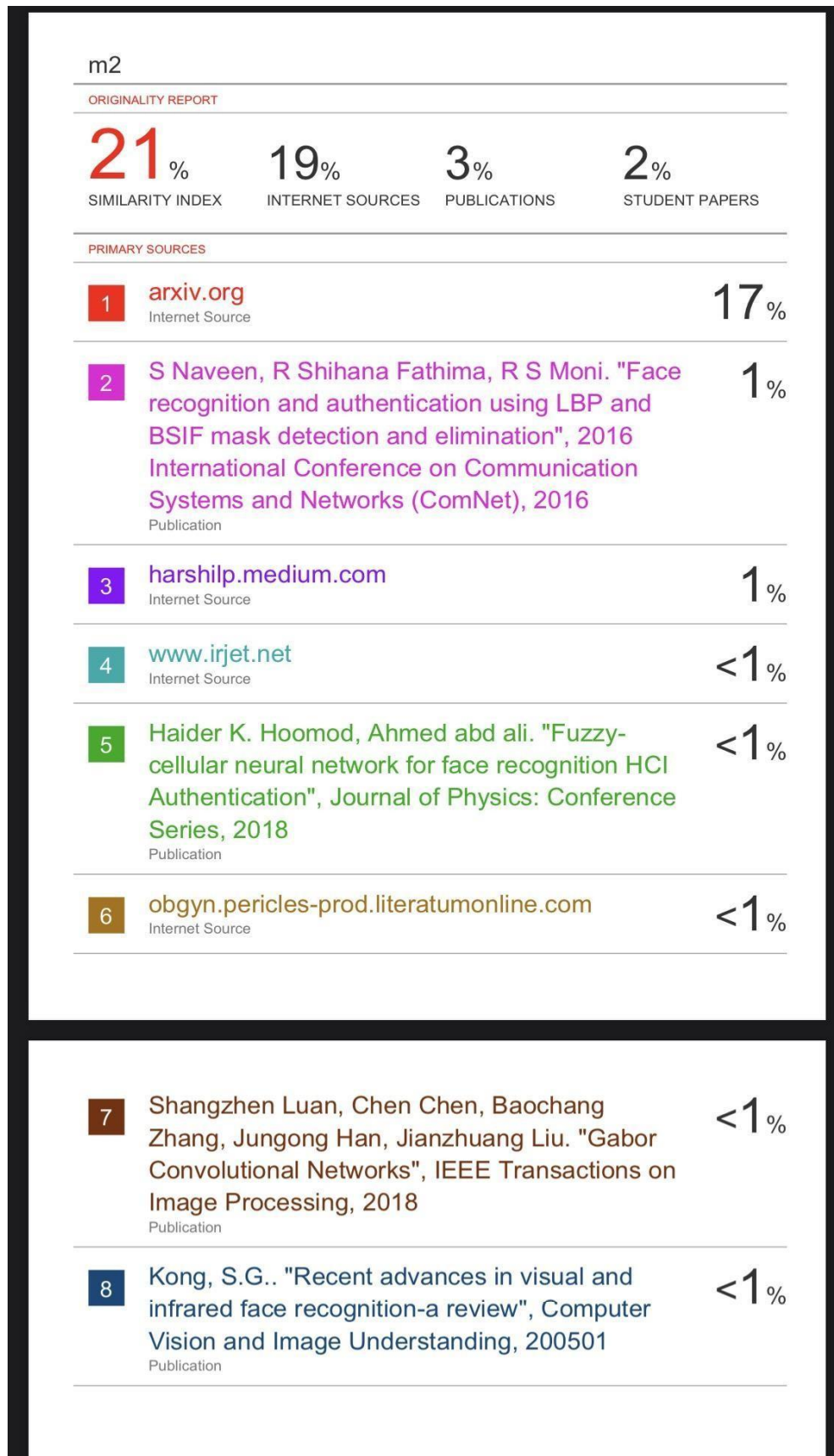


Fig-6.2

## 6.3 Plagiarism report



#### 6.4 REFERENCES:

- [ 1] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, July 2002.
- [ 2] T.-H. Kim, D.-C. Park, D.-M. Woo, T. Jeong, and S.-Y. Min, "Multi-class classifier based adaboost algorithm," in *Proceedings of the Second Sinoforeign-interchange Conference on Intelligent Science and Intelligent Data Engineering*, ser. *IScIDE'11*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 122–127.
- [ 3] P. Viola and M. J. Jones, "Robust real-time face detection," *Int. J. Comput. Vision*, vol. 57, no. 2, pp. 137–154, May 2004.
- [ 4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, Dec 2001, pp. I–I.
- [ 5] J. Li, J. Zhao, Y. Wei, C. Lang, Y. Li, and J. Feng, "Towards real world human parsing: Multiple-human parsing in the wild," *CoRR*, vol. abs/1705.07206. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
- [ 6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
- [ 7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015.
- [ 8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- [ 9] K. Li, G. Ding, and H. Wang, "L-fcn: A lightweight fully convolutional network for biomedical semantic segmentation," in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec 2018, pp. 2363–2367.
- [ 10] X. Fu and H. Qu, "Research on semantic segmentation of high-resolution remote sensing image based on full convolutional neural network," in *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*, Dec 2018, pp. 1–4.

- [ 11] S. Kumar, A. Negi, J. N. Singh, and H. Verma, "A deep learning for brain tumor MRI images semantic segmentation using fcn," in 2018 4th International Conference on Computing Communication and Automation (ICCCA), Dec 2018, pp. 1–4.
- [ 12] Tumuluru, Lakshmi Ramani, B., Saibaba, C.M.H., Venkateswarlu, B., Ravinder, N. OpenCV algorithms for facial recognition, International Journal of Innovative Technology and Exploring Engineering, Volume 8, Issue 8, 1 June 2019, Pages 927-933
- [ 13] Malathi, J., T. Satya Nagamani, and K. N. V. S. K. Vijaya Lakshmi. "Survey: Image forgery and its detection techniques." Journal of Physics: Conference Series. Vol. 1228. No. 1. IOP Publishing, 2019.
- [ 14] Krishnaveni, G., B. Lalitha Bhavani, and N. V. S. K. Vijaya Lakshmi. "An enhanced approach for object detection using wavelet based neural network." Journal of Physics: Conference Series. Vol. 1228. No. 1. IOP Publishing, 2019.
- [ 15] Satapathy, Sandeep Kumar, et al. "Deep learning based image recognition for vehicle number information." International Journal of Innovative Technology and Exploring Engineering 8 (2019): 52-55.
- [ 16] Pathak, Mrunal, Vinayak Bairagi, and N. Srinivasu. "Multimodal Eye Biometric System Based on Contour Based E-CNN and Multi Algorithmic Feature Extraction Using SVBF Matching." International Journal of Innovative Technology and Exploring Engineering
- [ 17] Volume-8 Issue-9, July 2019.
- [ 18] .Dudi, B., and Dr V. Rajesh. "Medicinal Plant Recognition Based On CNN And Machine Learning." International Journal Of Advanced Trends In Computer Science And Engineering, ISSN (2019): 2278-3091.
- [ 19] Kishore, P. V. V., et al. "Indian classical dance action identification and classification with convolutional neural networks." Advances in Multimedia 2018 (2018).
- [ 20] Ravi, Sunitha, et al. "Multi modal spatio temporal co-trained CNNs with single modal testing on RGB–D based sign language gesture recognition." Journal of Computer Languages 52 (2019): 88-102.
- [ 21] Kumar, E. Kiran, et al. "Training CNNs for 3-D sign language recognition with color texture coded joint angular displacement maps." IEEE Signal Processing Letters 25.5 (2018): 645-649.

- [ 22] Patel, Ashok Kumar, Snehamoy Chatterjee, and Amit Kumar Gorai. "Development of a machine vision system using the support vector machine regression (SVR) algorithm for the online prediction of iron ore grades." *Earth Science Informatics* 12.2 (2019): 197-210.
- [23] P. A. Rota, M. S. Oberste, S. S. Monroe, W. A. Nix, R. Campagnoli, J. P. Icenogle, S. Penaranda, B. Bankamp, K. Maher, M.-h. Chenet al., "Characterization of a novel coronavirus associated with severe acute respiratory syndrome," *science*, vol. 300, no. 5624, pp. 1394–1399, 2003.
- [24] Z. A. Memish, A. I. Zumla, R. F. Al-Hakeem, A. A. AlRabeeah, and G. M. Stephens, "Family cluster of middle east respiratory syndrome coronavirus infections," *New England Journal of Medicine*, vol. 368, no. 26, pp. 2487–2494, 2013.
- [25] Y. Liu, A. A. Gayle, A. Wilder-Smith, and J. Rocklöv, "The reproductive number of covid-19 is higher compared to sars coronavirus," *Journal of travel medicine*, 2020.
- [26] Y. Fang, Y. Nie, and M. Penny, "Transmission dynamics of the covid-19 outbreak and effectiveness of government interventions: A data driven analysis," *Journal of medical virology*, vol. 92, no. 6, pp. 645–659, 2020.
- [27] N. H. Leung, D. K. Chu, E. Y. Shiu, K.-H. Chan, J. J. McDevitt, B. J. Hau, H.-L. Yen, Y. Li, D. K. M., J. Ip et al., "Respiratory virus shedding in exhaled breath and efficacy of face masks."
- [28] S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, "Rational use of face masks in the covid19 pandemic," *The Lancet Respiratory Medicine*, 2020.