FACE MASK DETECTION USING CONVOLUTION NEURAL NETWORK

A PROJECT REPORT

Submitted by

Aryan Kashyap (20BAI10072)

Shrey Khanduja (20BAI10194)

Sagar Maheshwari (20BAI10339)

Alok Khansali (20BAI10347)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING.

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE

MADHYA PRADESH – 466114

APRIL 2022

BONAFIDE CERTIFICATE

Certified that this project report titled **"FACE MASK DETECTION USING CONVOLUTION NEURAL NETWORK"** is the bonafide work of "**ARYAN KASHYAP (Register No :20BAI10072), SHREY KHANDUJA (Register No :20BAI10194), SAGAR MAHESHWARI (Register No :20BAI10339), ALOK KHANSALI (Register No :20BAI10347)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. S SOUNTHARRAJAN

Program Chair

School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY

Pin: - 466001

Dr. J. MANIKANDAN

Project Guide, Assistant Professor

School of Computer Science and Engineering

VIT BHOPAL UNIVERSITY

Pin: - 466001

② ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to **Dr S. SOUNTHARRAJAN**, Head of the Department, School of Computer Science and Engineering for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide **Dr J.MANIKANDAN**, for continually guiding and actively participating in our project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of School of Computer Science and Engineering, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

Abbreviations	Meaning
etc	Et cetera
sklearn	Sci-kit learn
IOT	Internet of things
avg	Average
ROI	Result of information

LIST OF FIGURES AND GRAPHS

LIST OF TABLES

ABSTRACT

The latest epidemic to trigger an international health emergency is Coronavirus illness, according to this report. It is primarily spread through airborne transmission from person to person. The global number of cases has increased due to community transmission.

The individual analyses of two models, the training dataset, and the identification of face masks. TensorFlow is an open-source end-to-end framework for constructing machine learning applications, and it is used to train the datasets. In addition, keras, a submodule that uses tensor flow as its backend, is employed. The loading of data, pre-processing of data, and saving of the model are the primary phases.

In addition, for ML applications in Python, the Scikit-learn package is extremely beneficial. It chooses specialised tools for machine learning and statistical modelling, such as classification and clustering. Other libraries that aid in machine learning applications include Imutils and OpenCV.

Object detection and image classifications are popular research topics in the rapidly growing field of advanced technology, which uses webcams, surveillance cameras, and opensource platforms to detect and identify real-time problems in governmental core-areas such as business shops, airlines, universities, and army bases.

All of this aids in the training of the datasets that are fed into the model, as well as the training of the model that aids in the detection of faces using image recognition. We also come across OpenCV, which serves as the foundation for the subsequent model and aids in detecting picture recognition of a face with and without a mask.

TABLE OF CONTENTS

CHAPTER-1

PROJECT DESCRIPTION AND OUTLINE

1.1 INTRODUCTION

Coronavirus illness 2019 (COVID-19) has infected about 20 million individuals worldwide, according to the World Health Organization's (WHO) annual Situation Report – 205 [5]. To prevent the infection of COVID-19, wearing a mask has become necessary for safety.

In the case of a virus transmitted by sputtering, it appears that wearing a face mask is necessary to protect people and restrict illness spread. The coronavirus pandemic of 2019-20 is presently underway. COVID-19 (coronavirus disease 2019) is an infectious disease with flu-like symptoms at first. COVID-19 originally appeared in China, then soon spread over the rest of the world. When compared to the flu, COVID-19 is considered to be very contagious. We offer a face mask recognition model in this research that detects whether or not a person is wearing a face mask in real time. Such an application can be very valuable for security concerns, such as ensuring that disease transmission is kept to a minimum, especially among children and the elderly.

Face mask detection involves in detecting the location of the face and then determining whether it has a mask on it or not.

1.2 MOTIVATION FOR THE WORK

While many individuals are already convinced of the value of wearing a facial protective mask, as indicated by the World Health Organization (WHO, 2020) and scientific studies. Many people do not wear masks to protect themselves from the illness. As a result of these findings, public health educators and others began to launch mask-wearing prevention initiatives. These efforts focus on informing people about the need of wearing a mask through disseminating preventative posters and sketches. In our situation, we propose to assist these projects and public health sectors by establishing a real-time video-based research system and a linked interactive resource dedicated to assessing face mask wear using a webcam.

1.3 About Introduction to the project including techniques

We will be using TensorFlow and MobilenetV2 to create the model and check its training health, and for scanning the faces we will be using OpenCV, we will be using the model and do the needful. We will also be using Keras which gives fundamental reflections and building units for creation of the model, it also full advantage of TensorFlow, it will help in compiling the overall model.

1.4 Problem Statement

It is observed that many people don't wear masks while going outside and are sometimes not noticed by the security personnel. An image with various properties is used to detect a face. Face detection necessitates the recognition of facial expressions, face tracking, and position estimation, which is a tedious task. We will be trying to solve the issue by creating a face mask detector model and identifying the person on live image/video stream wearing face mask using image recognition in Machine Learning and Healthcare.

1.5 Objective of work

The objective of the project is to determine whether the person in front of the camera is wearing a mask or not through live streaming. The plan is use AI and ML tools to create a model by training a dataset and then using the model for face detection.

1.6 Organization of the project

Our group after identifying this issue have organized this project by using machine learning and artificial intelligence which will help in the conduction of image recognition. This project is on a small scale which can be developed in the future.

1.7 Literature Review

Table 1.1 Literature review

Sl. No.	Title of the Paper	Journal Name, Publisher Name, Year of Publication and Volume & Issue Number (Only SCI)	Author Name	Problem Addressed / Problem Statement	Methods/ Technologies Used	Author Contribution	Shortcomings/ Deficiency / Assumption Made (Research Gap)
1.	Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV	17th INDICON 2020,IEEE, 2020,978-1-7281-6916-3	Arjya Das; Mohammad Wasif Ansari; Rohini Basak	To resolve the issue on face mask for aversion of covid 19.	ML packages, TensorFlow, OpenCV.	Used basic ML tools and simplified techniques the method has achieved reasonably high accuracy.	Dataset contains huge number of images to get high level accuracy.
2.	Face Mask Detection Using OpenCV	ICICV 3rd conference 2021, IEEE, 978-1-6654-1960-4	Harish Adusumalli; D. Kalyani; R.Krishna Sri; M. Pratapteja; P V R D Prasada Rao	A system to replace humans to check masks on the faces of people for face detection.	Dataset Collection, Training a model to detect face masks, Detecting the person not wearing a mask	Author has used data sets,models to create a system of face mask detection	To achieve high levels of accuracy, the dataset contains a large number of photographs. Using pre-trained models, which may or may not provide good accuracy.
3.	Face mask detection using MobileNet and Global Pooling Block	CICT 2020 IEEE 4th Conference on Information & Communication Technology, IEEE, 978-0-7381-2447-6	Isunuri B Venkateswarlu; Jagadeesh Kakarla; Shree Prakash	Using MobileNet with a Global pooling block model for face mask detection to automate face mask detection and replace human observation: problem statement	Mobilenet and global pooling block	Using mobile net and pooling to achieve high accuracy.	using pre trained models which might not give good accuracy sometimes
4.	Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic	2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) ,IEEE, 978-1-7281-9675-6	Samuel Ady Sanjaya; Suryo Adi Rakhmawan	use a regionally automated face mask recognition in order to validly control the implementation of the government law	MobileNetV2	Used MobileNetV2 in order to train the data	Only MobileNetV2 will not suffice the whole training of the model.

CHAPTER-2:

RELATED WORK INVESTIGATION

2.1 INTRODUCTION

The coronavirus pandemic has resulted in unprecedented levels of international scientific cooperation. In many ways, artificial intelligence (AI) based on machine learning and deep learning can aid in the fight against Covid-19. Researchers and physicians can use machine learning to analyse large amounts of data to estimate the spread of COVID-19, serve as an early warning system for possible pandemics, and classify vulnerable people. To combat and anticipate new diseases, investment for emerging technology such as artificial intelligence, IoT, big data, and machine learning is required.

There are many attempts towards face mask detection which use datasets available in the present. But the size of datasets are too varied. Some of the datasets use repeated images which aids in false accuracy. The softwares or systems work enough for face detection but flaws tend to happen. The main problem to be resolved is to check whether the person is wearing a mask correctly or not.

2.2 Core area of project

The deep learning techniques are implemented using OpenCV to detect live stream objects, TensorFlow, and Keras to implement the deep learning methods in this study. The most important aspects in advanced technology development to detect images utilising a surveillance camera, drone camera, and other web-based technologies are object detection and image processing. Data pre-processing is essential for delivering clean data, identifying key features from a given dataset, and boosting model performance in text classification.

Similarly, picture pre-processing and image classification are the most essential object identification and image classification methods for improving image data and enhancing image features for further processing tasks. If we have less image datasets using augmentation approaches, picture pre-processing is used to adjust the position of the data, boost the brightness of the image, and mostly create new artificial data.

2.3 Existing Approaches/Methods

There are few existing approaches by some organizations like

2.3.1 Approaches/Methods-1

[1] Arduino-based autonomous face detection system was proposed to detect the live streaming video to ensuring the intelligence security system.

2.3.2 Approaches/Methods-2

Leeway Hertz algorithm is an algorithm which is used for facial recognition.

2.3.3 Approaches/Methods-3

Similarly, a company named Tryolabs which is involved in face mask detection.

2.4 Pros and cons of the stated Approaches/Methods

These foreign companies' new technology could be prone to some of the same pitfalls as facial recognition. Many of the training datasets used for facial recognition are dominated by light-skinned individuals. Researchers found that the algorithm was less than 70 percent accurate in identifying new faces. One more aspect of machine learning to take into account it's possible that the machine learning models could pick up on other "background" features, which leads to mistakes hence being a limitation to the technique .

These are capable enough detecting face from farther distance through security cameras detecting multiple faces at a time.

2.5 Issues/Observations from investigation

We can make following observations:

- Even though these algorithms and AI softwares are being used in public for face detection they still lag in certain areas.
- Due to their accuracy being around 70%, many detections go wrong hence not solving the objective completely.
- Although at a small scale, we have attempted to increase the accuracy to 95% which definitely gives better results.

2.6 Summary

It can be concluded that AI and ML has an important role in this pandemic situation as it upholds the norms which are to be followed. They are essential towards healthcare but still always have room for improvement.

CHAPTER-3:

REQUIREMENT ARTIFACTS

3.1 INTRODUCTION

Due to miniature version of our project, the objective can be satisfied with minimal requirements. Generally, for an AI and ML project computer system with sufficient RAM and graphic card is enough for project to work along with storage which is enough to store data.

For the project, we need a computer system with a GPU to train and create the model. The project being based on face detection also uses a webcam be it inbuilt or an external. A dataset with images of no mask and with mask is essential. Code editors such as VS Code or Anaconda which can use TensorFlow and its modules is required for implementation.

Face Mask Detector functional requirements are that the system must be able to accurately load the face mask classifier model and detect faces in pictures or video streams.

3.2 Hardware and Software requirements

Requirements are as follows: -

1. Computer system with a working webcam.
2. A GPU setup is recommended for quicker and better results.
3. Minimum 4Gb RAM with basic storage available on System.
4. A Quad core CPU.

3.3 Specific Project Requirements

3.3.1 A Data set containing images with mask and without masks with different complexities

3.3.2 Special modules and libraries are required to be preinstalled into the system, for e.g., TensorFlow, Imutils, OpenCV etc.

3.3.3 Dataset images have been scanned for the protection before its utilization. (Sample images)

Figure 3.1 Sample dataset

Figure 3.1 shows sample dataset of images with mask and without mask.

3.4 Summary

As per stated above our project is a small scaled one hence requiring minimal specifications however greater specifications are recommended for quicker results and also for large scale implementations.

CHAPTER-4:

DESIGN METHODOLOGY AND ITS NOVELTY

4.1 Methodology and goal

Our proposed ML model is a solution to healthcare problem of people not abiding by rules on wearing a face mask for safety. Our face detector model will be able to detect the object i.e., the face of the person on our webcam and classify it into with mask/without_mask categories. After reading the pictures, the model will deal with issues regarding image height, width, colour, face prediction accuracy etc.

To identify the person on live image/ video stream wearing face mask using image recognition in Machine Learning for healthcare.

Method:

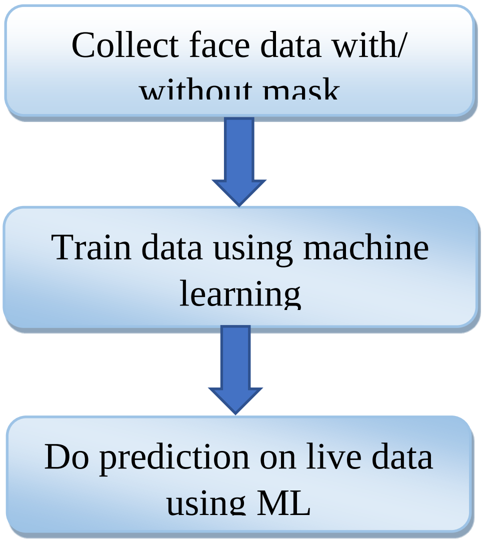


Figure 4.1Methodology

Figure 4.1 shows the design and methodology of the project.

4.2 Functional modules design and analysis

TensorFlow is an open-source end-to-end platform for creating Machine Learning applications. It is a symbolic math library that uses dataflow and differentiable programming to perform various tasks focused on training and inference of deep neural networks. It allows developers to create machine learning applications using various tools, libraries, and community resources. TensorFlow and Keras work hand-in-hand. Keras is a python library which runs with TensorFlow at its backend. It is also focused on neural networks maintaining the mathematical details. In general, it follows the following basic steps:

- Load the data
- Pre-process the data
- Define and compile the model
- Save the model.

There are some modules present in keras library:

1. Pre-processing – Converts raw data on the disk into a trained dataset ready for utilization.
2. Layers – These are the building blocks of keras library. It eases in creating the neural networks/ learning model, it is mainly used to compute mathematical details of the data.
3. Optimizer – It is used for compilation of the model
4. [4] MobileNetV2 – It helps in creating the image classification model.

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Imutils, a series of convenience functions to make basic image processing operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.

[2] OpenCV, is a library of Python bindings designed to solve computer vision problems. OpenCV makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB- style syntax. All the OpenCV array structures are converted to and from NumPy arrays.

4.3 Software Architecture designs

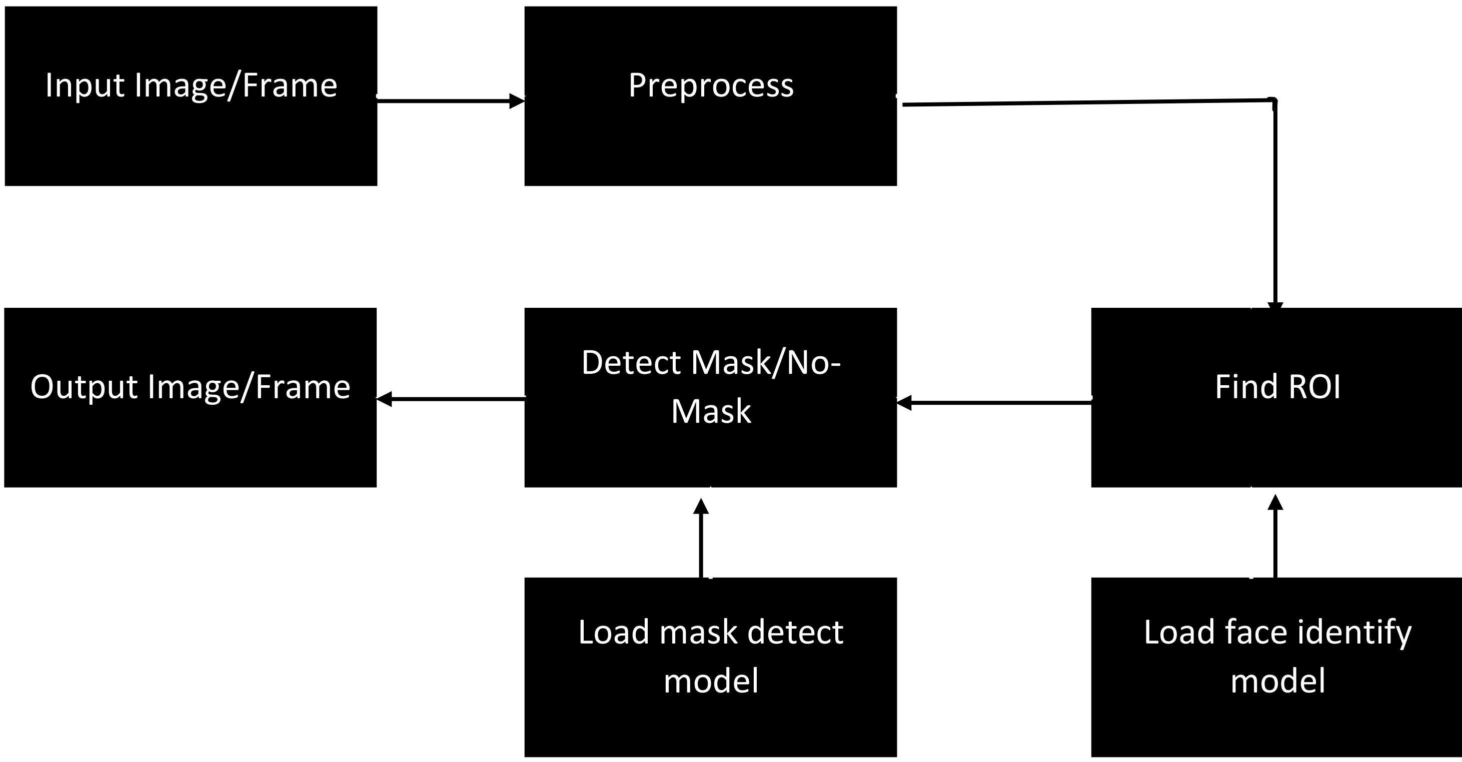


Figure 4.2 System Architecture

Figure 4.1 shows the system architecture of the project

4.4 Subsystem services

Visual Studio is a lightweight environment and a powerful code editor which has been used for the coding. It uses the resources and modules present in the computer system for executing the project. VS Code is able to execute C++, python, JAVA and many more languages, thus providing with a variety of features.

It also highlights the functions, strings, modules distinctively which helps in easy interpretation. The program for the project has been done in python language which is a very efficient language supported by VS Code. The extensions in VS Code also improves coding (e.g., better syntax).

4.5 User interface designs

We'd designed friendly interface for people, in this people will be able to see their live image on screen that would be labelled with two categories i.e., with_mask and without_mask.

[3] Face Mask Detection Platform uses Artificial Network to recognize is a user is not wearing a mask. The app can be connected to any existing or new IP cameras to detect people without a mask. App user can also add faces and phone numbers to send them an alert in case they are not wearing a mask. If the camera capture an unrecognized face.

4.6 Summary

Overall, our project has a simple design and it is completely based on Machine learning. We've used the appropriate modules of python which have good hands on Neural networks. Our project has a straightforward architecture and is entirely based on machine learning. We've chosen Python libraries that are well-versed in neural networks.

CHAPTER-5

TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 OUTLINE

The code has been divided into two parts, the first part targets towards the creation of the model and training the dataset. It will check the accuracy of the model by training its health and giving the resulted graph. The second focuses on face detection implementation by connection the webcam to the code and then identifying face masks in livestream/ video stream.

5.2 Technical coding and code solutions

- **<u>Training Dataset</u>: -**

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import os

# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-4      # initial learning rate - learning rate is less then loss will be calculated correctly(0.00001)
EPOCHS = 20
BS = 32

# Directory shows where the dataset is present
DIRECTORY = r"E:\Face-Mask-Detection-master\dataset"
# categories of data set
CATEGORIES = ["with_mask", "without_mask"]

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

```

```python
# appending all image arrays in datalist
data = []
# append all the data related to dataset which is with_mast and without_mask
labels = []


# looping through the dataset("CATEGORIES")
for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):                                # listdir - list down all the images in particular directory
        img_path = os.path.join(path, img)                      # joining the path to corresponding image
        image = load_img(img_path, target_size=(224, 224))      # load_img - loads the image path, target size is height and width of the image
        image = img_to_array(image)                             # saving the image to a varable 'image'
        image = preprocess_input(image)                         # preprocess input is coming from mobileNet

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

# data is in numerical value but labels are still alphabetical values - so converting them
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)                               # converting dataset to numerical values like 0 and 1
labels = to_categorical(labels)


# converting the data to numpy arrays
data = np.array(data, dtype="float32")
labels = np.array(labels)

# train test split - 20% to testing and 80% for training the model
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)

```

```python

# construct the training image generator for data augmentation using ImageDataGenerator - used for creating many images using single images but with different properties
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

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

# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)            # relu is goto activation function for non linear use Cases
headModel = Dropout(0.5)(headModel)                             # dropout is used for avoiding the overfitting of the model
headModel = Dense(2, activation="softmax")(headModel)           # softmax function is probability based function (2 describes with mask and without mask)


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


```

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


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


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

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

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

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


```

```python
# serialize the model to disk
print("[INFO] saving mask detector model...")

# An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.
model.save("mask_detector.model", save_format="h5")


# plot the training loss and accuracy using matplotlib
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="center right")
plt.savefig("plot.png")                    # saving the image using matplotlib      You, 4 weeks ago • Added dat
```

Output: -

```
Epoch 1/20
95/95 [==============================] - 75s 790ms/step - loss: 0.3265 - accuracy: 0.8553 - val_loss: 0.0890 - val_accuracy: 0.9844
Epoch 2/20
95/95 [==============================] - 71s 752ms/step - loss: 0.1173 - accuracy: 0.9572 - val_loss: 0.0585 - val_accuracy: 0.9883
Epoch 3/20
95/95 [==============================] - 70s 737ms/step - loss: 0.0842 - accuracy: 0.9687 - val_loss: 0.0441 - val_accuracy: 0.9883
Epoch 4/20
95/95 [==============================] - 68s 715ms/step - loss: 0.0615 - accuracy: 0.9792 - val_loss: 0.0413 - val_accuracy: 0.9883
Epoch 5/20
95/95 [==============================] - 68s 713ms/step - loss: 0.0600 - accuracy: 0.9789 - val_loss: 0.0378 - val_accuracy: 0.9909
Epoch 6/20
95/95 [==============================] - 70s 735ms/step - loss: 0.0458 - accuracy: 0.9848 - val_loss: 0.0360 - val_accuracy: 0.9922
Epoch 7/20
95/95 [==============================] - 67s 710ms/step - loss: 0.0536 - accuracy: 0.9842 - val_loss: 0.0343 - val_accuracy: 0.9909
Epoch 8/20
95/95 [==============================] - 73s 764ms/step - loss: 0.0311 - accuracy: 0.9885 - val_loss: 0.0299 - val_accuracy: 0.9935
Epoch 12/20
95/95 [==============================] - 71s 748ms/step - loss: 0.0374 - accuracy: 0.9871 - val_loss: 0.0283 - val_accuracy: 0.9935
Epoch 13/20
95/95 [==============================] - 76s 805ms/step - loss: 0.0304 - accuracy: 0.9901 - val_loss: 0.0271 - val_accuracy: 0.9909
Epoch 14/20
95/95 [==============================] - 69s 723ms/step - loss: 0.0291 - accuracy: 0.9914 - val_loss: 0.0256 - val_accuracy: 0.9922
Epoch 15/20
95/95 [==============================] - 73s 771ms/step - loss: 0.0263 - accuracy: 0.9927 - val_loss: 0.0262 - val_accuracy: 0.9948
Epoch 16/20
95/95 [==============================] - 72s 757ms/step - loss: 0.0315 - accuracy: 0.9888 - val_loss: 0.0302 - val_accuracy: 0.9922
Epoch 17/20
95/95 [==============================] - 72s 756ms/step - loss: 0.0270 - accuracy: 0.9898 - val_loss: 0.0263 - val_accuracy: 0.9935
Epoch 18/20
95/95 [==============================] - 74s 776ms/step - loss: 0.0235 - accuracy: 0.9934 - val_loss: 0.0242 - val_accuracy: 0.9922
Epoch 19/20
95/95 [==============================] - 70s 741ms/step - loss: 0.0286 - accuracy: 0.9901 - val_loss: 0.0247 - val_accuracy: 0.9935
Epoch 20/20
95/95 [==============================] - 55s 577ms/step - loss: 0.0308 - accuracy: 0.9885 - val_loss: 0.0227 - val_accuracy: 0.9935
```

```
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767

[INFO] saving mask detector model...
```

Figure 5.1Output of first half of the code

Figure 5.1 shows the creation of the model and displays the formulated results.

- **Face Mask Detection**: -

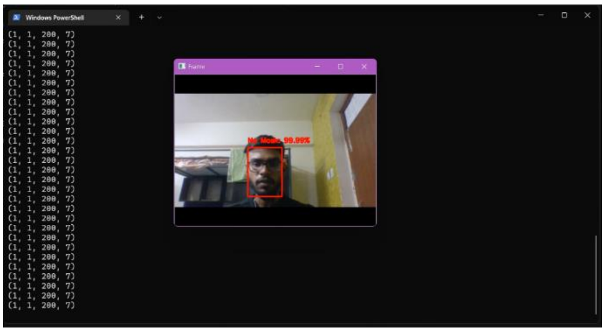
```python
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os


# facenet - detecting face
def detect_and_predict_mask(frame, faceNet, maskNet):
    # grab the dimensions of the frame and then construct a blob
    # from it
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))
      You, 7 days ago • Added face detector model and detecting mask file…
    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

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

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

```

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

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

            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)

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

    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
```

```python
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)  # location is x and y direction of the rectangle and prediction is the accuracy of mask.

# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

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

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

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

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

    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
```

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

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

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

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

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

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

Output: -



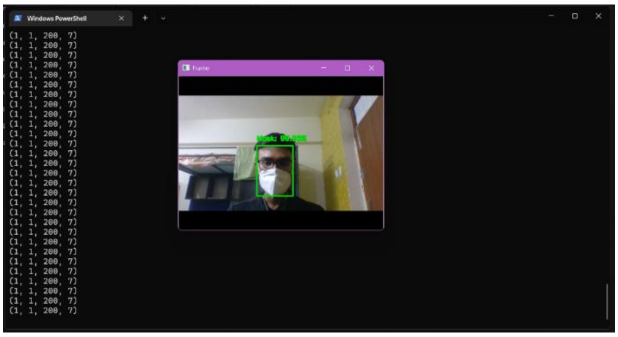


Figure 5.2 Output of second half of code

Figure 5.2 shows the output of detecting the face with mask and without mask.

5.3 Working Layout of Forms

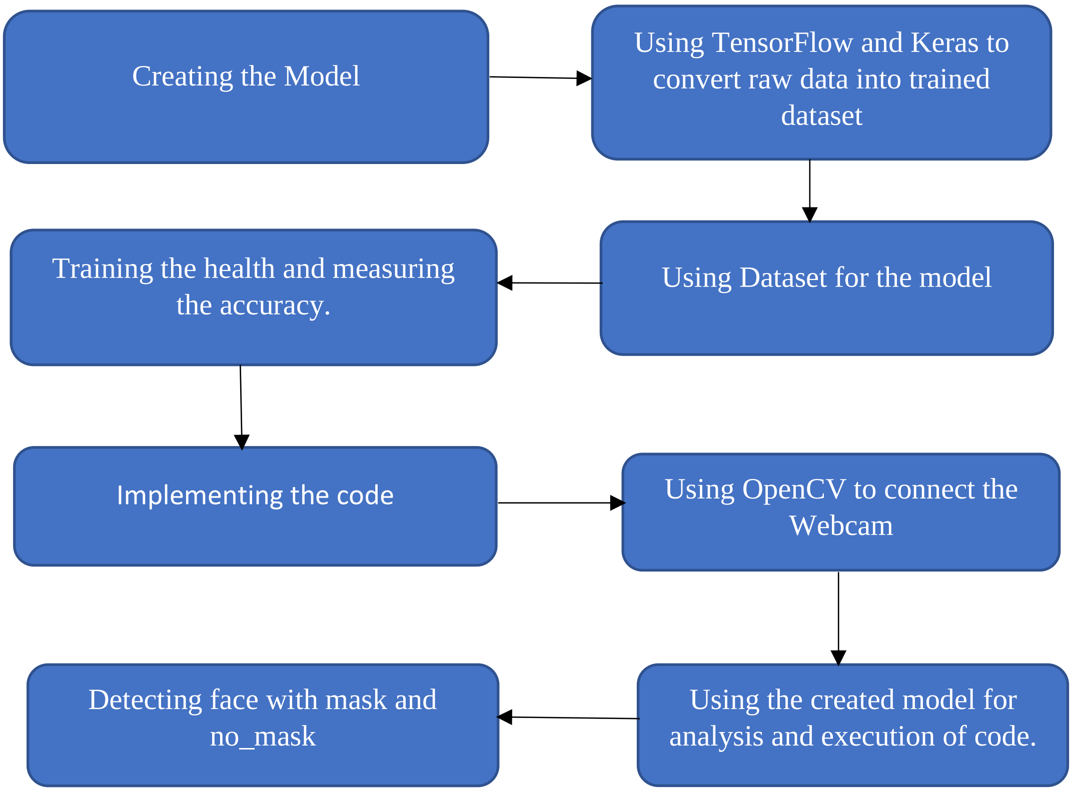


Figure 5.3 Working layout

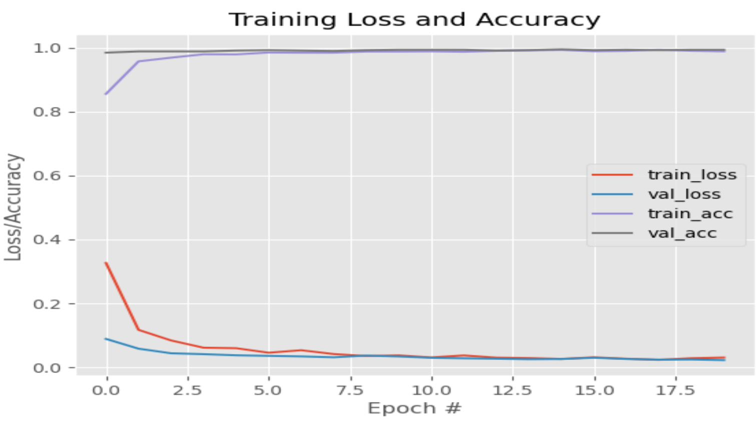
Figure 5.3 shows the whole working layout, how's the work is being carried out.

5.4 Test and validation

```
Epoch 12/20
95/95 [==============================] - 71s 748ms/step - loss: 0.0374 - accuracy: 0.9871 - val_loss: 0.0283 - val_accuracy: 0.9935
Epoch 13/20
95/95 [==============================] - 76s 805ms/step - loss: 0.0304 - accuracy: 0.9901 - val_loss: 0.0271 - val_accuracy: 0.9909
Epoch 14/20
95/95 [==============================] - 69s 723ms/step - loss: 0.0291 - accuracy: 0.9914 - val_loss: 0.0256 - val_accuracy: 0.9922
Epoch 15/20
95/95 [==============================] - 73s 771ms/step - loss: 0.0263 - accuracy: 0.9927 - val_loss: 0.0262 - val_accuracy: 0.9948
Epoch 16/20
95/95 [==============================] - 72s 757ms/step - loss: 0.0315 - accuracy: 0.9888 - val_loss: 0.0302 - val_accuracy: 0.9922
Epoch 17/20
95/95 [==============================] - 72s 756ms/step - loss: 0.0270 - accuracy: 0.9898 - val_loss: 0.0263 - val_accuracy: 0.9935
Epoch 18/20
95/95 [==============================] - 74s 776ms/step - loss: 0.0235 - accuracy: 0.9934 - val_loss: 0.0242 - val_accuracy: 0.9922
Epoch 19/20
95/95 [==============================] - 70s 741ms/step - loss: 0.0286 - accuracy: 0.9901 - val_loss: 0.0247 - val_accuracy: 0.9935
Epoch 20/20
95/95 [==============================] - 55s 577ms/step - loss: 0.0308 - accuracy: 0.9885 - val_loss: 0.0227 - val_accuracy: 0.9935
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767

[INFO] saving mask detector model...
```

Figure 5.4 Creation of model

Figure 5.4 shows the model being created and its health being checked and it also displays the formulated results



5.5 Performances Analysis (Graphs/Charts)

Figure 5.5 Performance Analysis

Figure 5.5 shows the training health of the created model in the form of Graph.

Table 5.1 Accuracy table

	Precision	Recall	F1-score	Support
With_mask	0.99	0.99	0.99	383
Without_mask	0.99	0.99	0.99	384
Accuracy			0.99	767
Macro avg	0.99	0.99	0.99	767
Weighted avg	0.99	0.99	0.99	767

5.6 Summary

After executing the code and analysing the results, we observe that our model has a precision and accuracy of 95% i.e., 0.95(given in the table). Hence our objective has been accomplished. Therefore, if you run the code and analyse the results, you will find that the accuracy and accuracy of the model is 95%. In this way, our goal was achieved.

CHAPTER-6

PROJECT OUTCOME AND APPLICABILITY

6.1 OUTLINE

As our project deals with face mask detection. It has applications in many areas which includes medical, securities etc. because our project deals with face mask detection, it is applied in many fields such as medical care and airports, offices etc. Our project will help reduce the spread of infection in the future.

6.2 Key implementations outlines of the system

Our project being at minimal scale doesn't require higher specifications. Hence it can be implemented on any pc or laptop with a Webcam. Because our dataset contains a variety of images. The accuracy is at par with other algorithms even at less amount of images available. Hence our accuracy is better than our algorithms.

6.3 Project applicability on Real-world applications

This project, if developed at higher scale can be used in the real world; this code can be installed in security cameras in Malls', offices and many public places for maintaining covid norms. This code can be installed at small shops where affording security person is difficult, it can also be installed at airports to detect travellers without masks.

It can also be installed at Hospitals to detect staffs not wearing masks during their shifts, if any health worker is found without a mask, they'll receive a notification with a reminder to wear a mask.

6.4 Inference

Our project is future ready project. As we see covid pandemic situation had created big issues in the world, similarly it is possible another situation may arise. So, in that situation also our project will be able to overcome the hazard. Hence our project has the potential to aid the situation.

CHAPTER-7
CONCLUSIONS AND RECOMMENDATION

7.1 OUTLINE

The conclusion of our project is, AI and ML has an important role in healthcare. This work is based on a deep learning approach to detecting facial masks in public places to limit the spread of the coronavirus in the community. We have illustrated the creation of the model. We used basic ML tools to create it and it resulted in high accuracy and precision. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not.

The proposed technique effectively handles blockages in crowded situations by using an ensemble of 1- and 2-stage detectors at the pre-processing level. The face detection project used in many parts of the world for security matters. Better the accuracy, better the results.

7.2 Limitations/Constraints of the system

The limitations we faced is the high grade equipments used for implementing the code at a commercial level. Our system is sufficient enough to run the whole project keeping all the necessary requirements at the minimum scale. Moreover, we’ve not used the pure CNN which gives the best specimen because it takes more time for computation. We have used mobilenetV2 which has a faster approach but lesser precision as compared to pure CNN.

7.3 Future Enhancements

Our code can develop to be used in more than just our systems. Starting from houses till public places. Datasets can be updated for better accuracy. No code is perfect, it can always be improved.

7.4 Inference

Our project is prepared for the future. As we can see, the covid pandemic event caused major problems over the world; similarly, another catastrophe could emerge. As a result, our project has the potential to help.

Appendix A

This Appendix consists the individual analysis of 2 models, training dataset and face mask detection. The datasets are being trained using the module named TensorFlow, it is an open-source end-to-end platform for creating machine leaning applications. Also, a sub module named keras is used which runs with tensor flow at its backend. The main steps followed are loading the data, pre-processing the data, saving the model etc.

Also, a module Scikit-learn is most useful for ML applications in python. It selects special tools for ML and Statistical modelling including classification, clustering etc. There are other libraires also which helps in ML applications like Imutils, OpenCV.

All these help in training the datasets which are driven into the model, also it helps to train the model, which helps in detecting the face using image recognition. We also encounter OpenCV which is the plinth for the following model, it helps in detecting the face with mask/ without mask using image recognition.

REFERENCES

1. Das, M. Wasif Ansari and R. Basak, "Covid-19 face mask detection using tensorflow keras and opencv", 2020 IEEE 17th India Council International Conference (INDICON), pp. 1-5, 2020
2. Khan, S. Chakraborty, R. Astya and S Khepra, "Face Detection and Recognition Using OpenCV", Proceedings - 2019 International Conference on Computing Communication and Intelligent Systems ICCCIS 2019 2019-Janua, pp. 116-119, 2019
3. I. B. Venkateswarlu, J. Kakarla and S. Prakash, "Face Mask Detection using MobileNet and Global Pooling Block", 2020 IEEE 4th Conference on Information Communication Technology (CICT), pp. 1-5, 2020
4. S. A. Sanjaya and S. A. Rakhmawan, "Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic", 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), pp. 1-5, October, 2020
5. W.H.O., “Coronavirus disease 2019 (covid-19): situation report, 205 2020”.