A Mini Project Report

On

**FACE MASK DETECTION**

Submitted in partial fulfillment of the

Requirements for the award of degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

by

**NAMPELLI SRILEKHA**

**(17H61A0533)**

**KOKKONDA SRAVYA**

**(17H61A0542)**

**THANIRU SRINU BABU**

**(17H61A0552)**

**Under the Guidance of**

**Mr. RAJASEKHAR REDDY**

**Associate Professor, Dept. of CSE**



**Department of Computer Science and Engineering**

**ANURAG GROUP OF INSTITUTIONS**

**(Formerly CVSR College of Engineering)**

**(An Autonomous Institution, Approved by AICTE and NBA Accredited)**

**Venkatapur (V), Ghatkesar (M), Medchal(D)., T.S-500088**

**(2016-2020)**

****

**CERTIFICATE**

This is to certify that the project entitled **“FACE MASK DETECTION”**  being submitted by **Nampelli Srilekha** bearing the Hall Ticket number **17H61A0533**, **Kokkonda Sravya** bearing the Hall Ticket number **17H61A0542** and **Thaniru Srinu Babu** bearing the Hall Ticket number **17H61A0552** in partial fulfillment of the requirements for the award of the degree of the **Bachelor of Technology** in **Computer Science and Engineering** to **Anurag Group of Institutions** **(Formerly** **CVSR College of Engineering)** is a record of bonafide work carried out by them under my guidance and supervision from August 2020 to March 2021.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

**Internal Guide Examinar External**

**Mr. Rajasekhar Reddy**

**(Associate Professor, Dept. of CSE)**

**Dr.G.Vishnu Murthy,**

**Professor and Head, Dept. of CSE**

**ACKNOWLEDGEMENT**

It is our privilege and pleasure to express profound sense of respect, gratitude and indebtedness to our guide **Mr. Rajasekhar Reddy, Associate Professor**, Dept. of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his/her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout this dissertation work.

We express our sincere gratitude to **Dr. G.Vishnu Murthy, Professor & Head**, Department of Computer Science and Engineering, Anurag Group of Institutions (Formerly CVSR College of Engineering), for his suggestions, motivations and co-operation for the successful completion of the work.

We extend our sincere thanks to **Dr. V.Vijaya Kumar**, **Dean,** Research and Development, Anurag Group of Institutions, for his encouragement and constant help.

We extend our sincere thanks to **Dr. K.S.Rao, Director**, Anurag Group of Institutions for his encouragement.

Nampelli Srilekha (17H61A0533)

Kokkonda Sravya (17H61A0542)

Thaniru Srinu Babu (17H61A0552)

**DECLARATION**

We hereby declare that the project work entitled “ **FACE MASK DETECTION**” submitted to the **Anurag Group of Institutions(Formerly CVSR College of Engineering)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Mr. Rajasekhar Reddy, Associate Professor** and this project work have not been submitted to any other university for the award of any other degree or diploma.

Nampelli Srilekha (17H61A0533)

Kokkonda Sravya (17H61A0542)

Thaniru Srinu Babu (17H61A0552)

Date:

**ABSTRACT**

The end of 2019 witnessed the outbreak of Coronavirus Disease 2019 (COVID-19), which has continued to be the cause of plight for millions of lives and businesses even in 2020. Studies have proved that wearing a face mask significantly reduces the risk of viral transmission as well as provides a sense of protection. However, it is not feasible to manually track the implementation of this policy. Technology holds the key here. We introduce a Deep Learning based system that can detect instances where face masks are not used properly. Our system consists of Convolutional Neural Network (CNN) architecture capable of detecting masked and unmasked faces and can be integrated with pre-installed CCTV cameras. This will help track safety violations, promote the use of face masks, and ensure a safe working environment.

|  |  |  |
| --- | --- | --- |
|  | **CONTENT** |  |
| **S.NO** | | **PAGE NO** |
|  | 1. Introduction | 1 |
|  | 1.1. Motivation | 1 |
|  | 1.2. Problem Definition | 1 |
|  | 1.3. Objective of the Project | 2 |
|  | 1. Literature Survey | 3 |
|  | 1. Analysis | 5 |
|  | 3.1. Existing System | 5 |
|  | 3.2. Proposed System | 5 |
|  | 3.3. Software Requirement Specification | 6 |
|  | 3.3.1 Purpose | 6 |
|  | 3.3.2 Scope | 7 |
|  | 3.3.3 Overall Description | 7 |
|  | 1. Design | 8 |
|  | 4.1. UML diagrams | 8 |
|  | 1. Implementation | 13 |
|  | 5.1. Modules | 13 |
|  | 5.2. Module description | 13 |
|  | 5.3. Introduction of technologies used | 14 |
|  | 5.4. Sample Code | 16 |
|  | 1. Test cases | 30 |
|  | 1. Screenshots | 31 |
|  | 1. Conclusion | 32 |
|  | 1. Future Enhancement | 33 |
|  | 10.       Bibliography | 34 |

**1. INTRODUCTION**

In this tough Covid-19 times, masks are openly responsible for everyone, and the reason behind this mask is to prevent the spread of coronavirus for the protection of others. Face masks are very effective in preventing infection, after filtering out the majority of cells and creating a strong impression on your nose and mouth, the prevention says. Face Mask Detection system uses two stage architecture to detect people without masks. This architecture is capable of detecting masked and unmasked faces. This model can therefore be used in real-time applications which require face-mask detection for safety purposes due to the outbreak of Covid-19. We can integrate this model with pre-installed CCTV cameras in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

**1.1. MOTIVATION**

In the present scenario due to Covid-19, there are no efficient face mask detection applications which are now in high demand for transportation means, densely populated areas, residential districts, large-scale manufacturers and other enterprises to ensure safety. The main aim of this project is to help track safety violations, promote the use of facemasks, and ensure a safe working environment.

**1.2. PROBLEM DEFINITION**

The World Health Organization (WHO) reports suggest that the two main routes of transmission of the COVID-19 virus are respiratory droplets and physical contact.

Respiratory droplets are generated when an infected person coughs or sneezes. Any person in close contact (within 1 m) with someone who has respiratory symptoms (coughing, sneezing) is at risk of being exposed to potentially infective respiratory droplets.

Droplets may also land on surfaces where the virus could remain viable; thus, the immediate environment of an infected individual can serve as a source of transmission (contact transmission). Wearing a medical mask is one of the prevention measures that can limit the spread of certain respiratory viral diseases, including COVID-19.

In this study, medical masks are defined as surgical or procedure masks that are flat or pleated (some are shaped like cups); they are affixed to the head with straps. They are tested for balanced high filtration, adequate breathability and optionally, fluid penetration resistance. The problem here is to analyze a set of video streams/images to identify people who are compliant with the government rule of wearing medical masks. This could help the government to take appropriate action against people who are non-compliant.

**1.3. OBJECTIVES OF THE PROJECT**

The objectives of the project can be described as:

* Check individuals and crowds whether they are wearing masks in public or not.
* Use digital screens to remind visitors to wear masks.
* Alert staff when no masks are detected.
* Works with existing cameras.

**2. LITERATURE SURVEY**

Under the current Covid-19 lockdown time such a system is definitely important to prevent the spread in many use cases. Following are few use cases which will benefit out of this system.

**Airports**: The proposed System can be of great importance at airports to detect travelers without masks. Traveler’s data can be captured as videos in the system at the entrance. Any traveler found to be without a face mask, an alarm alerting the airport authorities is sent so that they could take quick action.

**Hospitals**: The proposed system can be integrated with CCTV cameras and that data may be administered to see if their staff is wearing masks. If some health worker is found without a mask, they can receive a reminder notification to wear a mask.

**Offices**: The proposed system can help in maintaining safety standards to prevent the spread of Covid-19 or any such air borne disease. If some employee is not wearing a mask, they can receive a reminder notification to wear a mask.

The choice of a system must be based on the best performance. Hence the above performance metrics may be considered for coming out with the best system so that it can be implemented at large scale. The system has been worked with the following classifier:

**MobilenetV2**: MobileNetV2 is a state of the art for mobile visual recognition including classification, object detection and semantic segmentation. This classifier uses Depth wise Separable Convolution which is introduced to dramatically reduce the complexity cost and model size of the network, and hence is suitable for Mobile devices, or devices that have low computational power. In MobileNetV2, another best module that is introduced is inverted residual structure. Non-linearity in narrow layers is deleted. Keeping MobileNetV2 as the backbone for feature extraction, best performances are achieved for object detection and semantic segmentation.

For this classifier, the following Adam optimizer is applied to check the performance of the system

**ADAM**: Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, is based on adaptive estimates of lower-order moments. This method is computationally efficient, and can execute with little memory requirements. It is invariant to diagonal rescaling of the gradients, which is well suited for problems which are large in terms of data and/or parameters. The hyper-parameters have intuitive interpretations and they typically require little tuning. Empirical results show that Adam works well in practice and compares favorably to other stochastic optimization methods.

**3. ANALYSIS**

**3.1. EXISTING SYSTEMS**

As the world began implementing precautionary measures against the Coronavirus, numerous implementations of Face Mask Detection systems came forth. (Ejazetal.,) have performed facial recognition on masked and unmasked faces using Principal Component Analysis (PCA). However, the recognition accuracy drops to less than 70% when the recognized face is masked.

**3.1.1. DRAWBACKS OF EXISTING SYSTEMS**

* Accuracy of the model is very less.
* Availability of Internet always for detection.

**3.2. PROPOSED SYSTEM**

We proposed a two-stage architecture for detecting masked and unmasked faces by improvising the accuracy of the model using Deep Learning framework and also extending its functionality to face mask detection in webcam stream. The flow to identify the person in the webcam wearing the face mask or not.

The process is two-fold.

* To identify the faces in the webcam.
* Classify the faces based on the mask.

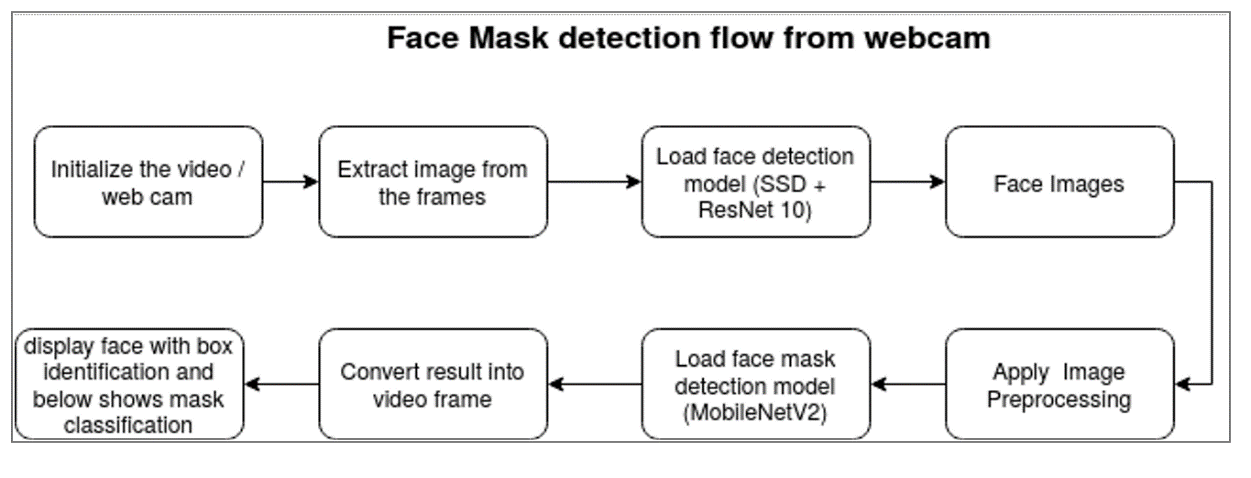


Fig 3.2 Proposed System

We propose a two-stage architecture for detecting masked and unmasked faces and localizing them. The first stage of our architecture includes a Face Detector, which localizes multiple faces in images of varying sizes and detects faces even in overlapping scenarios. The detected faces (regions of interest) extracted from this stage are then batched together and passed to the second stage of our architecture, which is a CNN based Face Mask Classifier. The results from the second stage are decoded and the final output is the image with all the faces in the image correctly detected and classified as either masked or unmasked faces.

* + 1. **ADVANTAGES OF PROPOSED SYSTEM**
* Improvised accuracy of the model.
* No need of Internet always for detection.
* Easy Implementation

**3.3. SOFTWARE REQUIREMENT SPECIFICATION**

**TECHNOLOGY**

* CNN-MobileNetV2
* Python

**TOOLS**

* Tensorflow
* Keras
* Opencv
* Matplotlib
* Numpy
* Scipy
* Anaconda Prompt

**3.3.1. PURPOSE**

Our face mask detector is least complex in structure and gives quick results and hence can be used in CCTV footages to detect whether a person is wearing a mask perfectly so that he does not pose any danger to others. Mass screening is possible and hence can be used in crowded places like railway stations, bus stops, markets, streets, mall entrances, schools, colleges, etc. By monitoring the placement of the face mask on the face, we can make sure that an individual wears it the right way and helps to curb the scope of the virus.

**3.3.2. SCOPE**

The model is developed using deep learning technology with the concept of a convolution neural network for detecting and classifying whether people are wearing masks or not. It works similarly to how our brain works. In a similar way, the computer is able to perform image classification by looking for low-level features such as edges and curves and then building up to more abstract concepts through a series of convolutional layers. Hence, the face image of a person is divided into 2 classes with Mask and No Mask. This is a faster, cost-effective model.

**3.3.3. OVERALL DESCRIPTION**

Face mask detection has been accomplished by adopting Deep Learning techniques. We have designed our project into two phases: training face mask detector and implementing face mask detector. The dataset is loaded for the model to be trained and the model is serialized in the training phase. Further, the trained model is loaded, the faces are detected in images and video streams and then the region of interest (ROI) is extracted. Finally, the face mask detector is applied and the images or faces in the video streams are classified as Mask or No Mask.

**4. DESIGN**

**4.1. UML DIAGRAMS**

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

**What is UML?**

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it’s one of the most popular [business process modeling techniques](http://tallyfy.com/business-process-modeling-techniques).

It is based on diagrammatic representations of software components. As the old proverb says: “a picture is worth a thousand words”. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

**What is the use of UML?**

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several [business processes](http://tallyfy.com/business-process) or [workflows](https://tallyfy.com/what-is-a-workflow/). For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

**Types of UML Diagrams**

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

**4.1.1. CLASS DIAGRAM**

Class UML diagram is the most common diagram type for software documentation. Since most software being created nowadays is still based on the [Object-Oriented Programming paradigm](https://en.wikipedia.org/wiki/Object-oriented_programming), using class diagrams to document the software turns out to be a common-sense solution. This happens because OOP is based on classes and the relations between them.

In a nutshell, class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviors (also referred to as member functions). More specifically, each class has 3 fields: the class name at the top, the class attributes right below the name, the class operations/behaviors at the bottom. The relation between different classes (represented by a connecting line), makes up a class diagram.

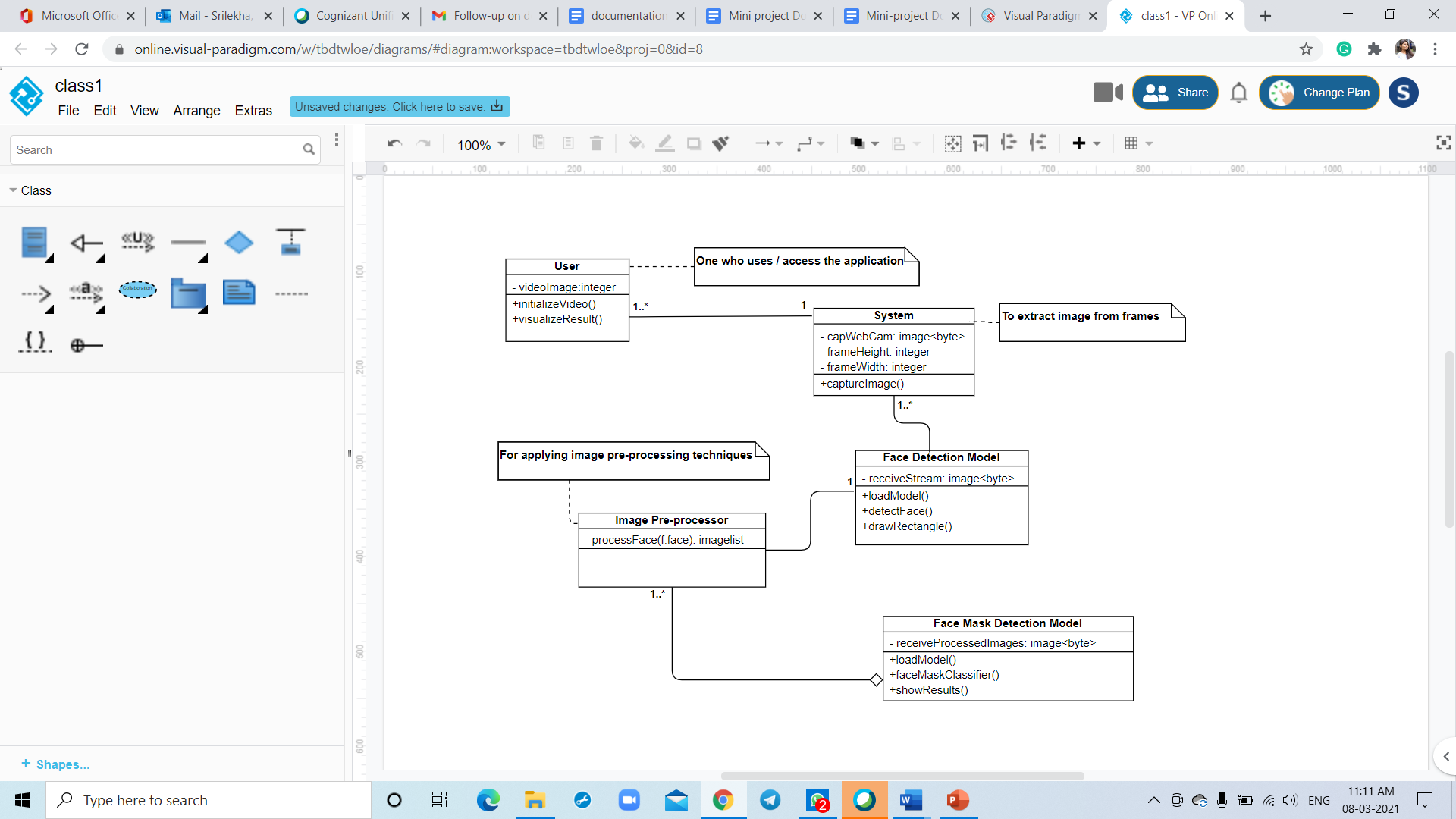
****

Fig 4.1.1 Class Diagram

**4.1.2. USECASE DIAGRAM**

A cornerstone part of the system is the [functional requirements](https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/) that the system fulfills. Use Case diagrams are used to analyze the system’s [high-level requirements](http://www.testablerequirements.com/testablerequirements/ident_hlrs.htm). These requirements are expressed through different use cases. We notice three main components of this UML diagram:

* Functional requirements – represented as use cases; a verb describing an action
* Actors – they interact with the system; an actor can be a human being, an organization or an internal or external application
* Relationships between actors and use cases – represented using straight arrows

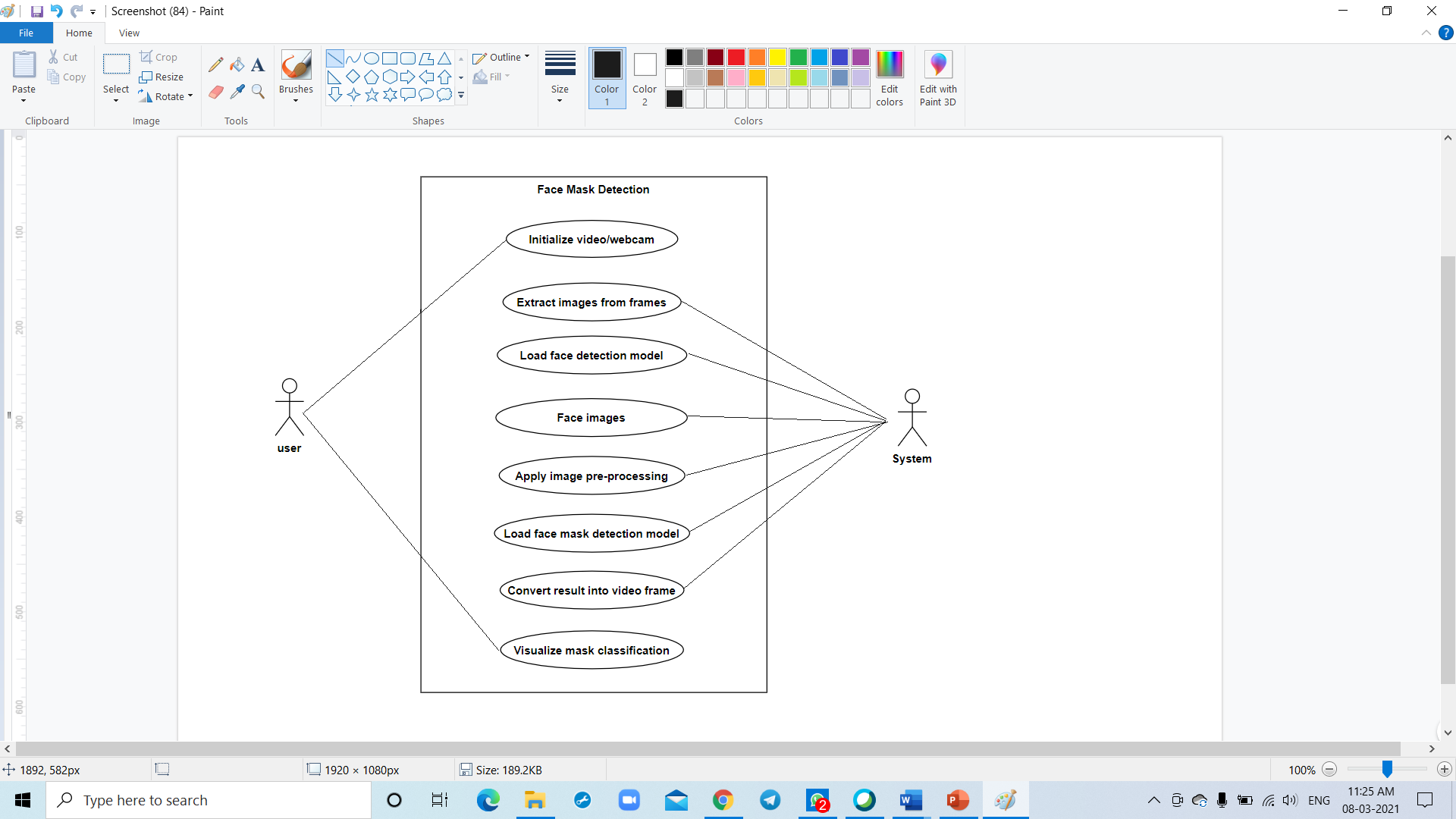
****

Fig 4.1.2 Use Case Diagram

**4.1.3. ACTIVITY DIAGRAM**

Activity diagrams are probably the most important UML diagrams for doing [business process modeling](https://tallyfy.com/business-process-modeling/). In software development, it is generally used to describe the flow of different activities and actions. These can be both sequential and in parallel. They describe the objects used, consumed or produced by an activity and the relationship between the different activities. All the above are essential in business process modeling.

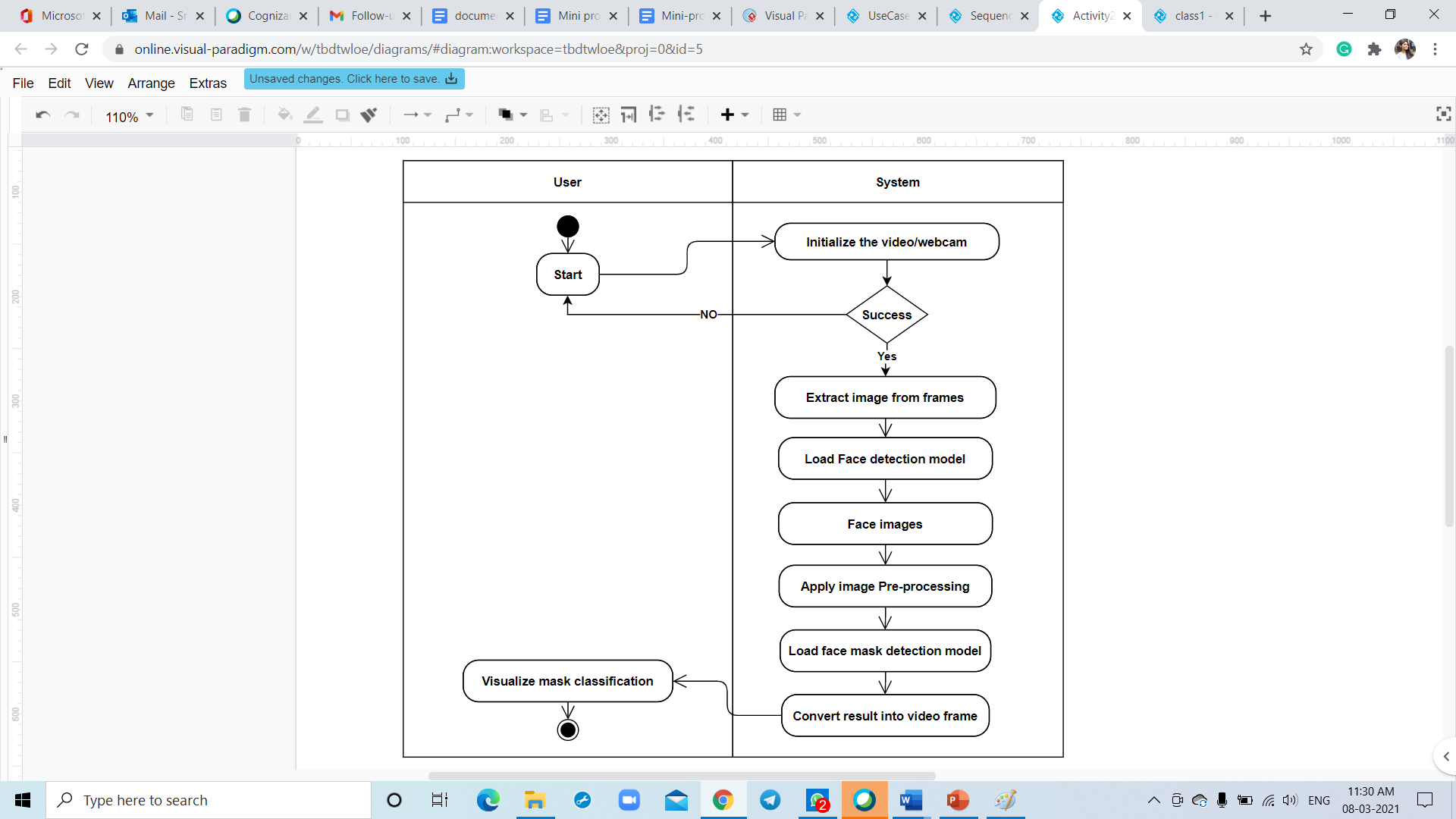
****

Fig 4.1.3 Activity Diagram

**4.1.4. SEQUENCE DIAGRAM**

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature.

As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner. To get a better idea, check the example of a UML sequence diagram below.

As the name suggests, structural diagrams are used to depict the structure of a system. More specifically, it is used in software development to represent the architecture of the system and how the different components are interconnected (not how they behave or communicate, simply where they stand).

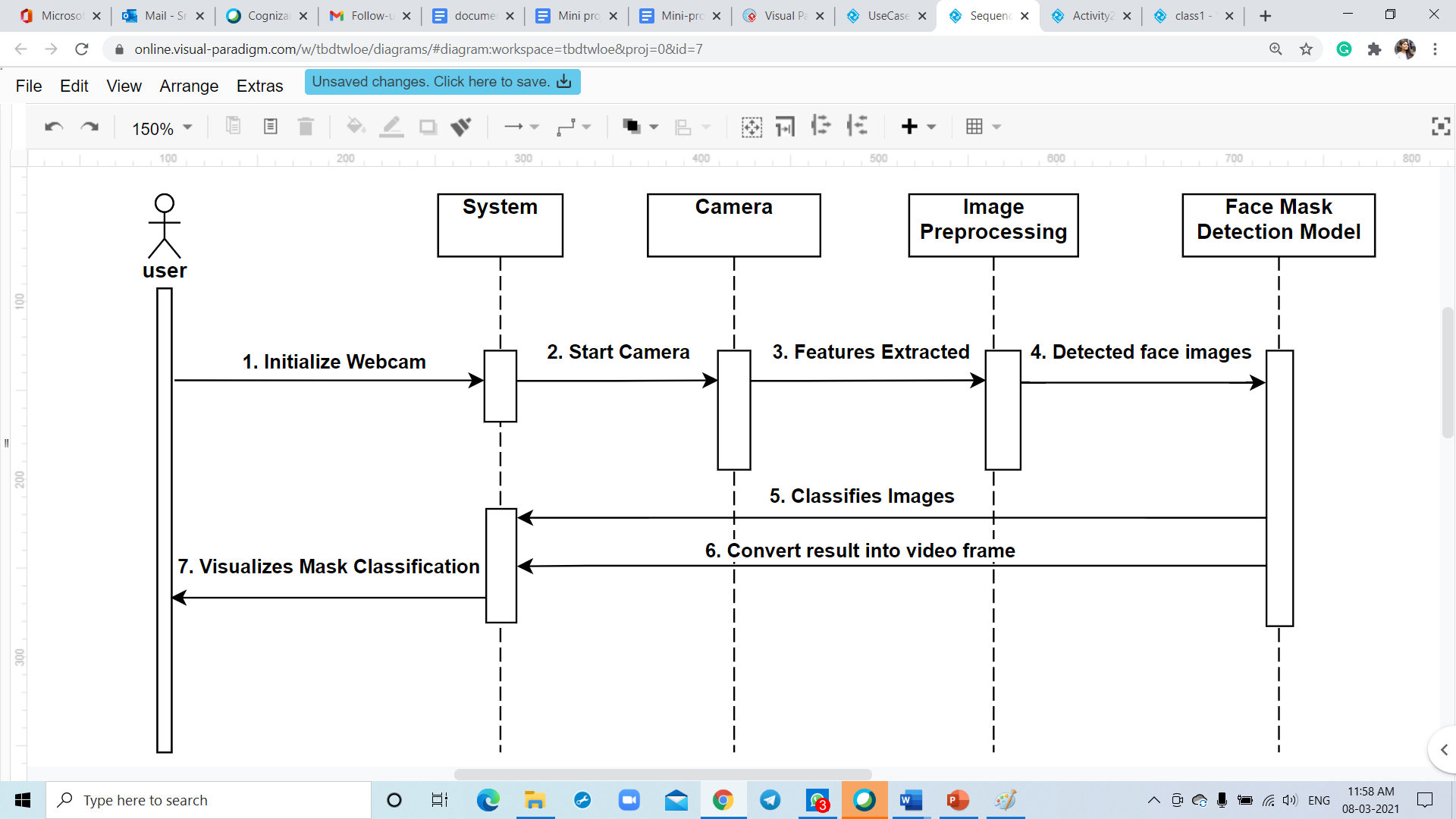
****

Fig 4.1.4 Sequence Diagram

**5.** **IMPLEMENTATION**

**5.1. MODULES**

* User
* Tensorflow
* Keras
* Opencv
* Matplotlib
* Numpy
* Anaconda Prompt

**5.2. MODULE DESCRIPTION**

**User**

In this project, the first module is for users. They are made to initialize a webcam / video stream and access the results. They are shown the result out of the two classes taken are Mask and No Mask.

**Tensorflow**

TensorFlow is a framework created by Google for creating Deep Learning models. Deep Learning is a category of machine learning models (=algorithms) that use multi-layer neural networks.



**Keras**

Keras is a powerful and easy-to-use free open-source Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in just a few lines of code. There is a requirement of TensorFlow in our project, that’s the main reason why we have incorporated the Keras module into the project.

**Opencv**

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human.

**Matplotlib**

The main purpose of Matplotlib is toascertain the results in the required graphical representations. There are various functions available in this module that allows us to plot different kinds of graphs. In this project, we have used this library for the sake of knowing the accuracy and losses across the iterations.

**Numpy**

NumPy or Numerical Python is an open-source Python library that makes it easy to complex numerical operations. Working with machine learning and deep learning applications involve complex numerical operations with large datasets. NumPy makes implementing these operations relatively simple and effective when compared to their pure Python implementation.

**Anaconda Prompt**

Anaconda is the Python distribution and the Anaconda Prompt is a command line shell (a program where you type in commands instead of using a mouse). The black screen and text that makes up the Anaconda Prompt doesn't look like much, but it is really helpful for problem solvers using Python.

**5.3. INTRODUCTION TO TECHNOLOGY USED**

**Python**

Python is certainly one of the best languages when working with Machine Learning and AI models as it has many built-in libraries which can be used directly without much implementation and code.

**Python Programming Language**

Python is a high-level, interpreted, interactive, and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

**Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive** − You can sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** − Python supports an Object-Oriented style or technique of programming that encapsulates code within objects.

**Python is a Beginner's Language** − Python is a great language for beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Convolutional Neural Networks**

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and be able to differentiate one from the other. CNN (Convolutional Neural Network) has many versions of pre-trained and well-architected networks for example AlexNet, ResNet, Inception, LeNet, MobileNet and so on. In this project we have chosen the MobileNetV2 due to its lightweight and very efficient mobile-oriented model.

**MobileNetV2**

MobileNetV2 builds upon the ideas from MobileNetV1, using depthwise separable convolution as efficient building blocks. However, V2 introduces two new features to the architecture: 1) linear bottlenecks between the layers, and 2) shortcut connections between the bottlenecks1. The basic structure is shown below.

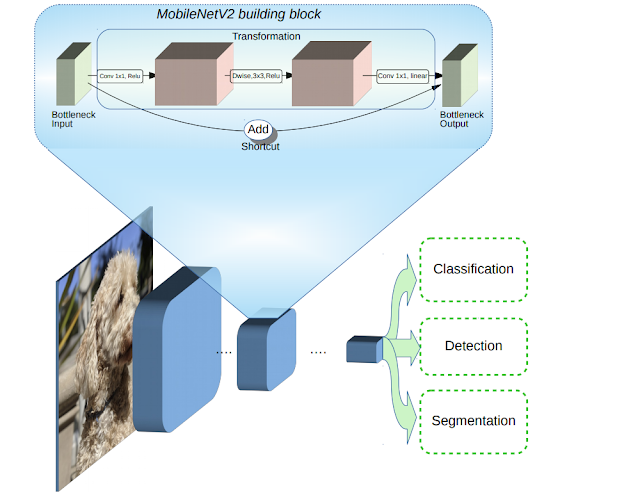


Fig 5.3 MobileNetV2 Architecture

**5.4.** **SAMPLE CODE**

train\_mask\_detector.py

# import the necessary packages

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.layers import AveragePooling2D

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Input

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing.image import img\_to\_array

from tensorflow.keras.preprocessing.image import load\_img

from tensorflow.keras.utils import to\_categorical

from sklearn.preprocessing import LabelBinarizer

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

from imutils import paths

import matplotlib.pyplot as plt

import numpy as np

import os

# initialize the initial learning rate, number of epochs to train for,

# and batch size

INIT\_LR = 1e-4

EPOCHS = 20

BS = 32

DIRECTORY = r"C:\Mask Detection\CODE\Face-Mask-Detection-master\dataset"

CATEGORIES = ["with\_mask", "without\_mask"]

# grab the list of images in our dataset directory, then initialize

# the list of data (i.e., images) and class images

print("[INFO] loading images...")

data = []

labels = []

for category in CATEGORIES:

path = os.path.join(DIRECTORY, category)

for img in os.listdir(path):

img\_path = os.path.join(path, img)

image = load\_img(img\_path, target\_size=(224, 224))

image = img\_to\_array(image)

image = preprocess\_input(image)

data.append(image)

labels.append(category)

# perform one-hot encoding on the labels

lb = LabelBinarizer()

labels = lb.fit\_transform(labels)

labels = to\_categorical(labels)

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

labels = np.array(labels)

(trainX, testX, trainY, testY) = train\_test\_split(data, labels,

test\_size=0.20, stratify=labels, random\_state=42)

# construct the training image generator for data augmentation

aug = ImageDataGenerator(

rotation\_range=20,

zoom\_range=0.15,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.15,

horizontal\_flip=True,

fill\_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are

# left off

baseModel = MobileNetV2(weights="imagenet", include\_top=False,

input\_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the

# the base model

headModel = baseModel.output

headModel = AveragePooling2D(pool\_size=(7, 7))(headModel)

headModel = Flatten(name="flatten")(headModel)

headModel = Dense(128, activation="relu")(headModel)

headModel = Dropout(0.5)(headModel)

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

# place the head FC model on top of the base model (this will become

# the actual model we will train)

model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will

# \*not\* be updated during the first training process

for layer in baseModel.layers:

layer.trainable = False

# compile our model

print("[INFO] compiling model...")

opt = Adam(lr=INIT\_LR, decay=INIT\_LR / EPOCHS)

model.compile(loss="binary\_crossentropy", optimizer=opt,

metrics=["accuracy"])

# train the head of the network

print("[INFO] training head...")

H = model.fit(

aug.flow(trainX, trainY, batch\_size=BS),

steps\_per\_epoch=len(trainX) // BS,

validation\_data=(testX, testY),

validation\_steps=len(testX) // BS,

epochs=EPOCHS)

# make predictions on the testing set

print("[INFO] evaluating network...")

predIdxs = model.predict(testX, batch\_size=BS)

# for each image in the testing set we need to find the index of the

# label with corresponding largest predicted probability

predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report

print(classification\_report(testY.argmax(axis=1), predIdxs,

target\_names=lb.classes\_))

# serialize the model to disk

print("[INFO] saving mask detector model...")

model.save("mask\_detector.model", save\_format="h5")

# plot the training loss and accuracy

N = EPOCHS

plt.style.use("ggplot")

plt.figure()

plt.plot(np.arange(0, N), H.history["loss"], label="train\_loss")

plt.plot(np.arange(0, N), H.history["val\_loss"], label="val\_loss")

plt.plot(np.arange(0, N), H.history["accuracy"], label="train\_acc")

plt.plot(np.arange(0, N), H.history["val\_accuracy"], label="val\_acc")

plt.title("Training Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="lower left")

plt.savefig("plot.png")

detect\_mask\_video.py

# import the necessary packages

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing.image import img\_to\_array

from tensorflow.keras.models import load\_model

from imutils.video import VideoStream

import numpy as np

import imutils

import time

import cv2

import os

def detect\_and\_predict\_mask(frame, faceNet, maskNet):

# grab the dimensions of the frame and then construct a blob

# from it

(h, w) = frame.shape[:2]

blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),

(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections

faceNet.setInput(blob)

detections = faceNet.forward()

print(detections.shape)

# initialize our list of faces, their corresponding locations,

# and the list of predictions from our face mask network

faces = []

locs = []

preds = []

# loop over the detections

for i in range(0, detections.shape[2]):

# extract the confidence (i.e., probability) associated with

# the detection

confidence = detections[0, 0, i, 2]

# filter out weak detections by ensuring the confidence is

# greater than the minimum confidence

if confidence > 0.5:

# compute the (x, y)-coordinates of the bounding box for

# the object

box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])

(startX, startY, endX, endY) = box.astype("int")

# ensure the bounding boxes fall within the dimensions of

# the frame

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

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

# extract the face ROI, convert it from BGR to RGB channel

# ordering, resize it to 224x224, and preprocess it

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

face = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)

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

face = img\_to\_array(face)

face = preprocess\_input(face)

# add the face and bounding boxes to their respective

# lists

faces.append(face)

locs.append((startX, startY, endX, endY))

# only make a predictions if at least one face was detected

if len(faces) > 0:

# for faster inference we'll make batch predictions on \*all\*

# faces at the same time rather than one-by-one predictions

# in the above `for` loop

faces = np.array(faces, dtype="float32")

preds = maskNet.predict(faces, batch\_size=32)

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

# locations

return (locs, preds)

# load our serialized face detector model from disk

prototxtPath = r"face\_detector\deploy.prototxt"

weightsPath = r"face\_detector\res10\_300x300\_ssd\_iter\_140000.caffemodel"

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk

maskNet = load\_model("mask\_detector.model")

# initialize the video stream

print("[INFO] starting video stream...")

vs = VideoStream(src=0).start()

# loop over the frames from the video stream

while True:

# grab the frame from the threaded video stream and resize it

# to have a maximum width of 400 pixels

frame = vs.read()

frame = imutils.resize(frame, width=400)

# detect faces in the frame and determine if they are wearing a

# face mask or not

(locs, preds) = detect\_and\_predict\_mask(frame, faceNet, maskNet)

# loop over the detected face locations and their corresponding

# locations

for (box, pred) in zip(locs, preds):

# unpack the bounding box and predictions

(startX, startY, endX, endY) = box

(mask, withoutMask) = pred

# determine the class label and color we'll use to draw

# the bounding box and text

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

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

# include the probability in the label

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

# display the label and bounding box rectangle on the output

# frame

cv2.putText(frame, label, (startX, startY - 10),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, color, 2)

cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

# show the output frame

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

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

if key == ord("q"):

break

# do a bit of cleanup

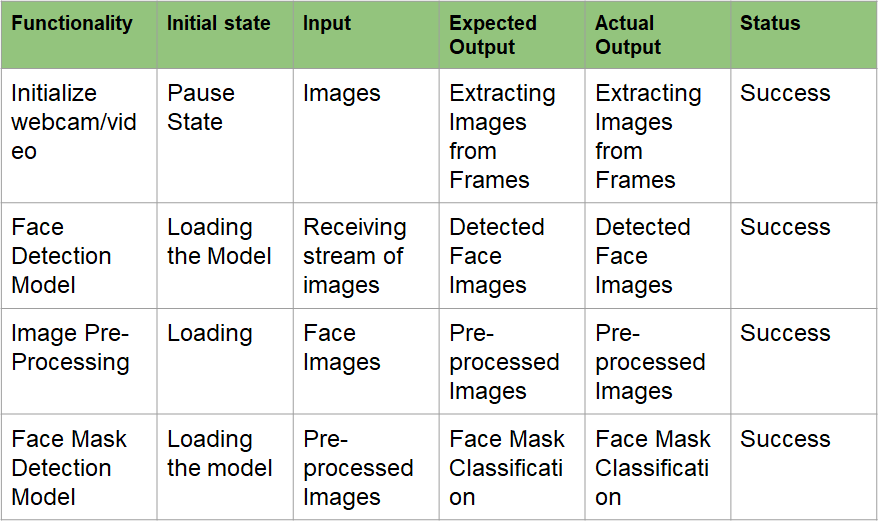
cv2.destroyAllWindows()

vs.stop()

**6.** **TESTING**

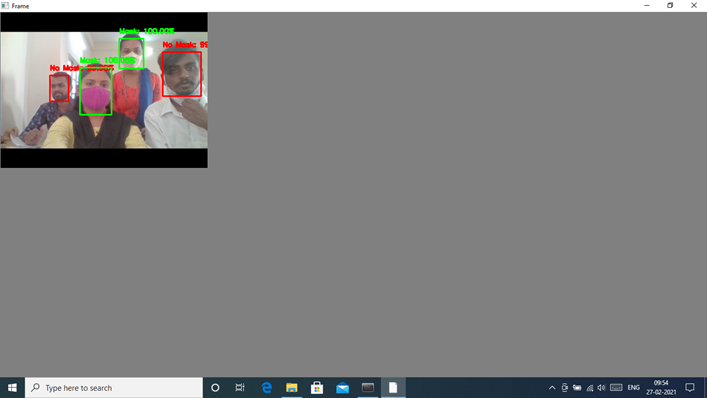
Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results i.e. testing is executing a system to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

**6.1.** **TEST CASES**

****

**7.** **SCREENSHOTS**

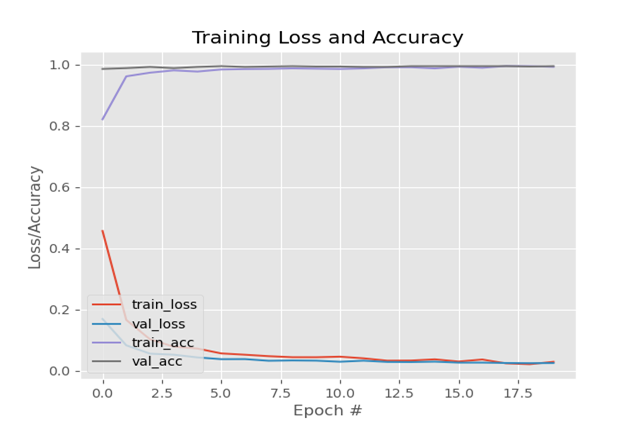
**7.1. SCREENSHOT OF OUTPUT**

****

**7.2. ACCURACY OVERVIEW**

The dataset has been divided into 2 sets, likely a training and validation set.

The accuracy of image classifiers over the training set vs validation.

****

**8. CONCLUSION**

Face Mask Detection model detects whether a person in real-time if he/she is wearing a mask or not with good accuracy. The model is tested with images and real-time video streams. Further, this project can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

**9. FUTURE ENHANCEMENT**

The system may be implemented along with interfacing with alarm and alerting systems in the near future. This system may be integrated with a system which can integrate with a system implementing social distancing which can make it a wholesome system which can bring dramatic impact on the spread.

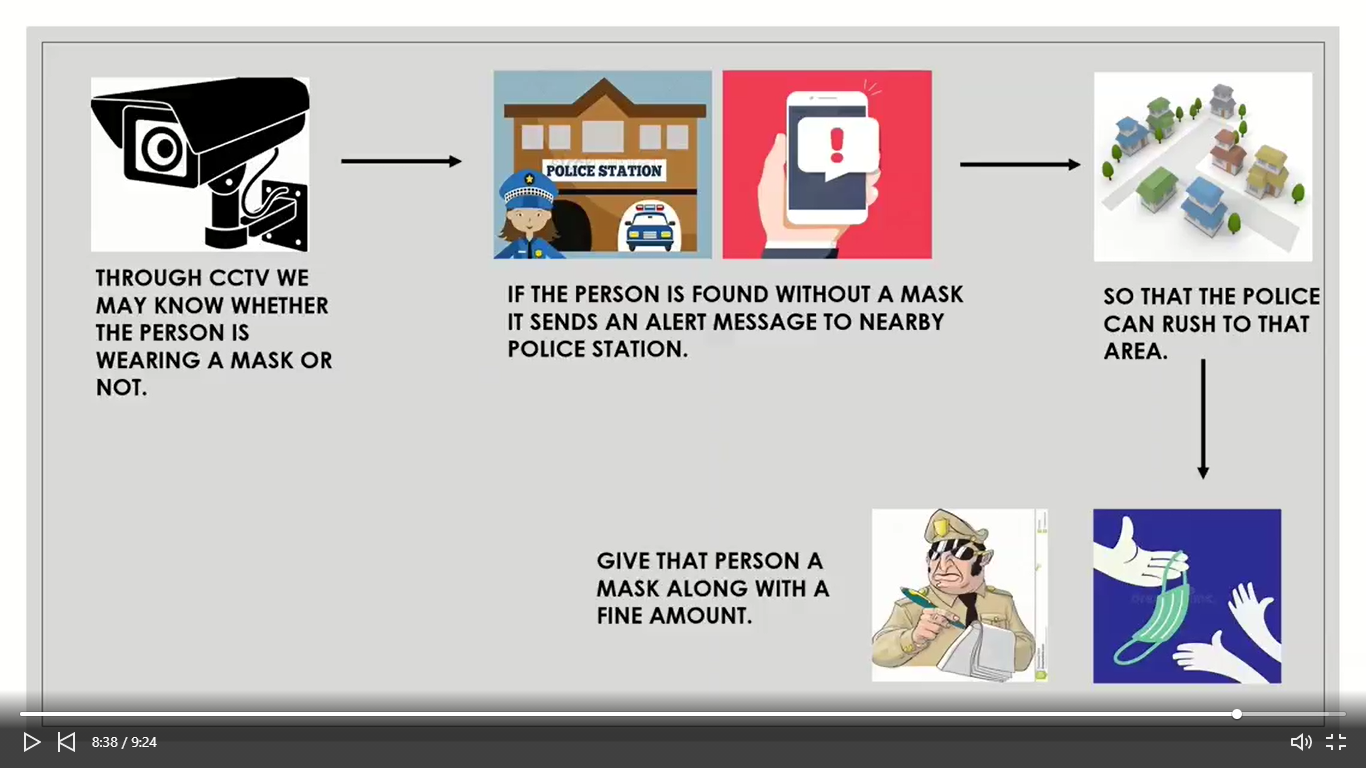


Fig 9 Future Scope

The system can be used in the following places to identify people with or without masks.

* Offices – Manufacturers, retail, other SMEs and corporate giants
* Hospitals/healthcare organizations
* Airports and railway stations
* Sports venues
* Entertainment and hospitality industry
* Densely populated areas

**10.** **BIBLIOGRAPHY**

1. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 4510- 4520, doi: 10.1109/CVPR.2018.00474.
2. Xin, M., Wang, Y. Research on image classification model based on deep convolutional neural networks. J Image Video Proc. 2019, 40 (2019).
3. Sultana, F., A. Sufian, and P. Dutta. "A review of object detection models based on convolutional neural networks." arXiv preprint arXiv:1905.01614 (2019).
4. X. Jia, "Image recognition method based on deep learning," 2017 29th Chinese Control And Decision Conference (CCDC), Chongqing, 2017, pp. 4730-4735, doi: 10.1109/CCDC.2017.7979332.
5. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications - Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam.
6. Adam: A Method for Stochastic Optimization - Diederik P. Kingma, Jimmy Ba
7. B. QIN, D. LI, Identifing facemask-wearing condition using image super-resolution with classification network to prevent covid-19 (2020)
8. “The importance of wearing masks in curtailing the COVID-19 pandemic” - Jose Humphreys.
9. Deep Residual Learning for Image Recognition, He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. arXiv:1512.03385,2015
10. Image Classification Using Convolutional Neural Networks Deepika Jaswal, Sowmya.V, K.P.Soman