GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

EFFICIENT FACE MASK IDENTIFICATION SYSTEM USING DL

A project report submitted in partial fulfilment of the requirement

for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted

by

S.Santhoshi
(17341A05G1)

V.Sai Teja
(17341A05H1)

P.V.Sri Vamsi
(17341A05E1)

V.Sri Vatsav
(17341A05H0)

Under the esteemed guidance of

Mr. K. Phani Babu

Asst.Professor, Dept. of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

GMR Nagar, Rajam – 532127,

Andhra Pradesh, India

August 2021

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the thesis entitled **EFFICIENT FACE MASK IDENTIFICATION SYSTEM USING DL** submitted by **S.Santhoshi, V.Sai Teja, P.V.Sri Vamsi, V.Sri Vatsav (17341A05G1, 17341A05H1, 17341A05E1, 17341A05H0)** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor
Mr.K. Phani Babu
Assistant Professor
Department of CSE
GMRIT, Rajam.

Signature of HOD
Dr. A. V. Ramana
Professor & Head
Department of CSE
GMRIT, Rajam.

The report is submitted for the viva-voce examination held on ……..

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Mr. K. Phani Babu** Assistant Professor, Department of Computer Science and Engineering for his whole hearted and invaluable guidance throughout the project work. Without his sustained and sincere effort, this project work would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr.C.L.V.R.S.V.Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Computer Science and Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

S.Santhoshi **17341A05G1**

V.Sai Teja **17341A05H1**

P.V.Sri Vamsi **17341A05E1**

V.Sri Vatsav **17341A05H0**

ABSTRACT

In recent times, COVID-19 strain is most effective and spreading continuously all over the world. To avoid its effect, people are taking certain measures and masks became a mandatory accessory while coming out. No Mask means No Health. To control and avoid its spread, everyone must wear a mask to make themselves and others ineffective. At crowded places, people without a mask need to be detected and alerted. Observing every person at such places is difficult and face mask detection must be done for the safety of people. A model needs to be trained to detect faces without a mask. It can be implemented using Deep Learning with OpenCV. People without a mask are detected in CCTV and will be warned by the authority.The data set can be trained by using Convolutional Neural Networks(CNN) algorithm. Using this model and data set can produce better results. This can be a solution to avoid the spread of COVID-19 virus.

Keywords: COVID-19, CNN, Deep Learning, OpenCV, CCTV.

INDEX

LIST OF TABLES

LIST OF FIGURES

1.INTRODUCTION

Covid-19 strain is effectively increasing day by day. It has been identified as the most dangerous one among all the other strains and spreads. It is spreading very easily and causing various illnesses like cough,cold etc and even resulting in death. Most of the people have died because of this corona. Everyone needs to take care of themselves and their surroundings. To avoid it, certain measures need to be followed. Maintaining social distance,washing hands,avoiding touching face,eyes etc. One such is wearing a mask while moving out of our home.

The spread can be reduced if people strictly wear a mask and follow other preventive measures. But, most of the people are not taking it seriously and are not wearing the mask. They are allowing the virus into them without intention. People without masks need to be warned and intimated to reduce and avoid the spread. People moving in crowded places must wear a mask. They need to be identified and warned. Everywhere CCTV cameras are arranged for security purposes.

Those CCTV cameras can now be used for mask detection. Face mask detection is a method of detecting a person without a face mask. Observing each person in a crowded place is a huge task. So, an automatic system needs to be built which detects the person without mask and intimates the authorities. That system captures the images of every person and differentiates them. Such a model can be built by using deep learning algorithms.

The Detection of wearing a face mask cannot be determined by large and small appearance variations between different types of images. Object detection method is mainly used for this face mask detection. A deep learning model is trained and is connected to the systems.The learning algorithm Convolution Neural Network (CNN) is used for feature extraction from the images and footage where the features are learned by multiple hidden layers.

All the images in the dataset are converted to binary numpy arrays using LabelBinarizer before being fed into the model. This is a method used to convert the multi class labels into binary labels. It uses an inverse transform method for conversion. Then the dataset is pre-processes using different methods like Image data generator etc.

The Convolutional Neural Network (CNN) is trained by images for classification with ImageNet and some deep learning models like VGG16, VGG19, ResNet50, Inception V3 and Xception. In this work, the model is trained using InceptionV3 and Xception algorithms of

transfer learning. It is a Machine learning method where a model developed for one task can be reused as a starting point for a model of a second task.

Inception V3 module is to act as a "multi-level feature extractor" by computing *1×1*, *3×3*, and *5×5* convolutions within the *same* module of the network.The weights for Inception V3 are smaller than both VGG and ResNet. Xception is an extension of the Inception architecture which replaces the standard Inception modules with depth wise separable convolutions. Xception sports the smallest weight serialization at only 91MB.

1.1 Objective Of The Work:

The motto of this project is to identify the people without masks when moving in crowded places. Those people are identified and warned by informing to authority. This must be done to reduce and stop the spread.

1.2 Organization Of Thesis:

The remainder of the work is organized as follows: the second section consists of related works and similar approaches to object detection and face mask detection. All the methods used in this work include techniques,functions and the dataset are clearly mentioned in the next section. In the next section, evaluation metrics and results, and conclusion are shown and explained. All the experimental results are reviewed and future work is stated.

2.LITERATURE SURVEY

Immediately after the pandemic started spreading rapidly, most of the researchers and students started working on the projects to find the causes and remedies to stop the spread. They have developed and implemented many machine learning and deep learning projects which finds the reason behind the spread of Covid-19. Many projects found out that cough is one of the symptoms and if those patients sneeze in presence of any other person, it would easily spread and other people will be infected.

And one more analysis is that proper wearing of masks can avoid the spread. If any person is without a mask or didn't wear a mask properly, it would lead to the spread and attack of covid-19. So,detection of face masks must be done in crowded places.
The motto of this project is to identify the people without masks and immediately warn them to wear them by informing authorities to avoid the spread.

Mohammad Marufur et al. proposed a framework for detecting people and monitoring them in CCTV cameras.they have used Convolutional Neural Networks(CNN) for image recognition and feature extraction. Those features are then used by dense neural networks for classification. The person without mask is captured along with image and time and that information is sent to authority for alerting the person. Their proposed model achieved 98.7% accuracy.

Abdellah Oumina et al. designed a model for face mask detection by using CNN and the extracted features are processed using different machine learning classifiers like Support Vector Machine(SVM) and K-Nearest Neighbors(KNN). Those classifiers are used and examined to get more accuracy and precision. The designed model got 97.1% as the best classification rate.

Mohammed Loey et al. proposed a system which targets at detecting the faces with and without mask. It consists of two components.For feature extraction, a deep transfer learning model ResNet-50 is used. The extraction model has 16 blocks with different convolution sizes. Based on those features, images are classified using a deep network model called YOLO v2 which is a composition of feature extraction and detection networks.

Jan Zhang et al. proposed and designed a work which targets detailed and proper wearing of masks. In this work, they performed two tasks. One is extracting multiple features from images and giving weights to them using the attention module. And then renowned features are extracted using R-CNN. This proposed model achieved 84.1% mAP.

Jignesh Chowdary et al. designed a model for face mask detection and trained a model using transfer learning technique. Transfer learning has many pre-trained models. One such is InceptionV3 which is used in this. It is trained and tested on Data set and achieved 99.9% accuracy while training and 100% while testing.

Aniruddha Srinivas Joshi et al. proposed a model for face mask detection by cascaded framework and retina face mask using mobileNet in videos. This technique was used in videos for collection of data through web and acquiring data and achieved 86.6% accuracy while cascaded and 89.1% of retina accuracy in mask wearing.

Alok Negi et al. proposed two deep neural techniques CNN model and VGG16 with transfer learning for the face mask detection. CNN model achieved training, validation and testing accuracy 97.42% while another model VGG16 achieved 98.97% accuracy.

Isunuri B Venkateswarlu et al. proposed MobileNet with a global pooling block for face mask detection to perform a flatten of the feature vector. A fully connected dense layer associated with the softmax layer has been utilized for classification. MobileNet with global max pooling records best performance of 99.56% accuracy.

Harish Adusumalli et al. proposed a method which employs TensorFlow and OpenCV to detect face masks. OpenCV is primarily used for Computer Vision Applications..which..involves..Facial..recognition,Object..Detection,Segmentation and recognition.

Bin Xue et al. proposed RETINAFACE algorithm, which effectively realizes the detection of mask wearing, and realizes the function of judging whether the mask is worn correctly. ROC curve 0.6 increases sharply, and the AUC area reaches 0.94, indicating that the classification of the mode.

3.PROBLEM STUDY

3.1 Problem Statement:

When people are moving in crowded places, there a lot of chances for increase of spread of corona virus. If people wear mask and follow all certain preventive measures strictly, then the spread can be decreased and avoided. But, people are not following and become a reason for the spread unintentionally. To avoid it, people need to be alerted and warned for disobeying the rules. It is difficult to check everyone when moving in the crowd. So, a model needs to be trained to identify the persons without mask and inform authorities to warn them.

3.2 Solution:

For the problem, a model which automatically scans the faces and classifies the unmasked is required. Then, that information need to be sent to authorities. A deep learning model can be trained in such a way as mentioned above for mask detection. That model is called as our Efficient face mask identification system.

3.3 Task Analysis:

Face mask Identification System can perform following tasks:

- **Capture the faces:**

  The system captures the faces from the images captured using the camera.

- **Feature Extraction:**

  From the faces captured, features are extracted using the trained algorithm

- **Classification:**

  Based on those features, the faces are classified as masked or unmasked

- **Informing to authorities:**

  All the unmasked people are identified and that information is send to authorities to warn them.

3.4 Process Model:

The proposed method labels the images with or without mask along with the accuracy. The features are extracted and classified using InceptionV3 and Xception of transfer learning.

Each model has its own features and accuracy. The below diagram illustrates the process flow of the proposed system.

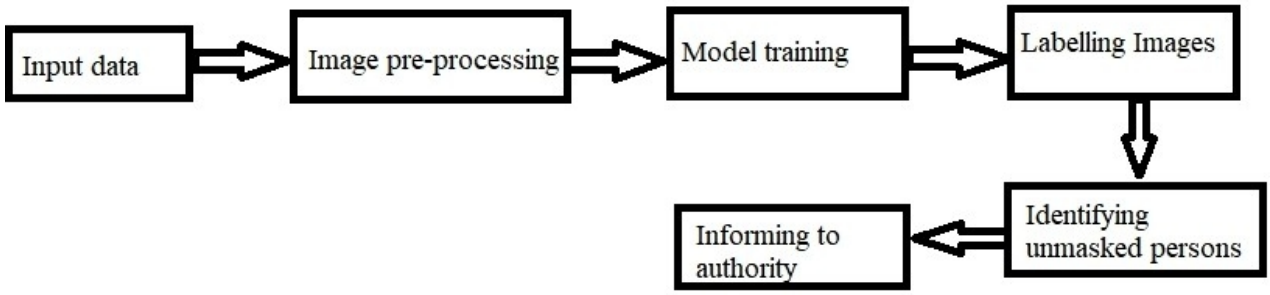

Figure 3.4.1: Process flow of the proposed system

3.5 Requirement Specification:

1. **Software Requirements:**
   - Visual Studio Code
   - Google colab
2. **Functional Requirements :**
   - i. **Dataset:**
     - Must have unbiased dataset
     - Must have over 1500+ images in both classes
     - Dataset must not reuse the same images in training and testing phases
   - ii. **Detector:**
     - Capture the faces
     - Extract the features
     - Image classification
     - Informing to authorities
3. **Non-functional Requirements:**
   - All the data must be secured
   - Time is required to develop the complete project
4. **System Requirements:**
   - Able to load the mask detection and classifier model
   - Able to detect faces in images or video stream
   - Able to extract features from the face image

- End position of the face must be fit inside the webcam frame and must be clear to camera
- Able to detects masks in .jpg, .png , .gif format images
- Result must be viewed by the showing the probability along with label as **Mask** or **No Mask**

3.6 Use case Modelling:

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation. A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

Use case diagrams can be used for requirement analysis and high level design model the context of a system,reverse Engineering,forward Engineering.

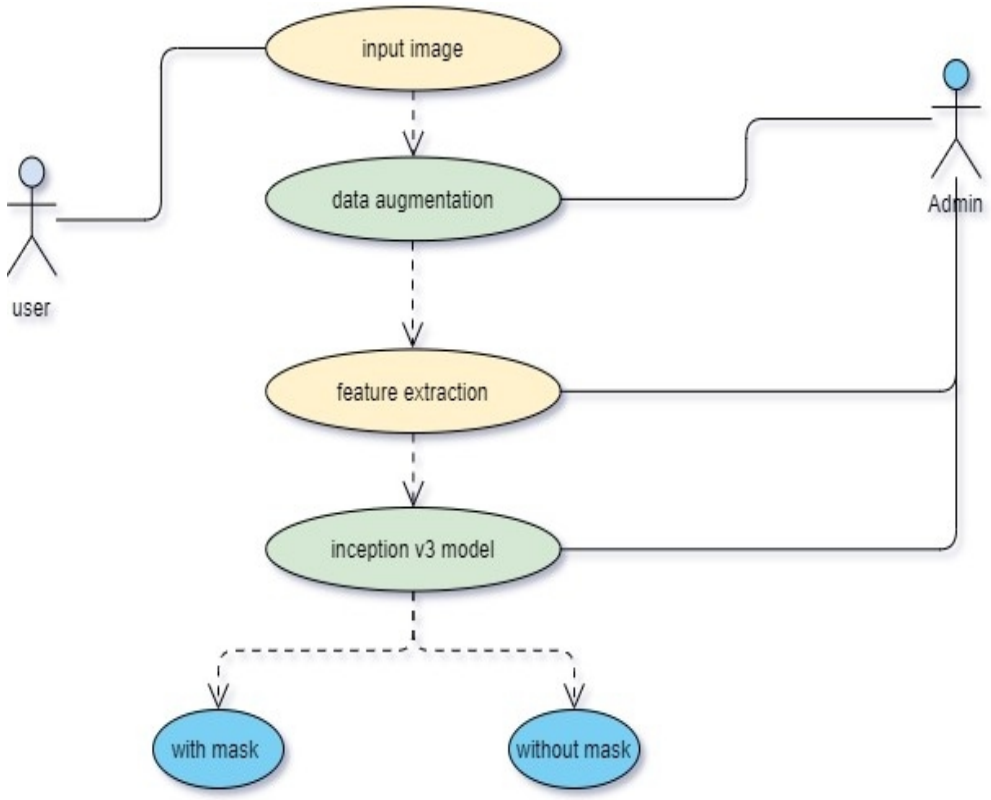

Figure 3.6.1: Use Case Diagram of InceptionV3 Model

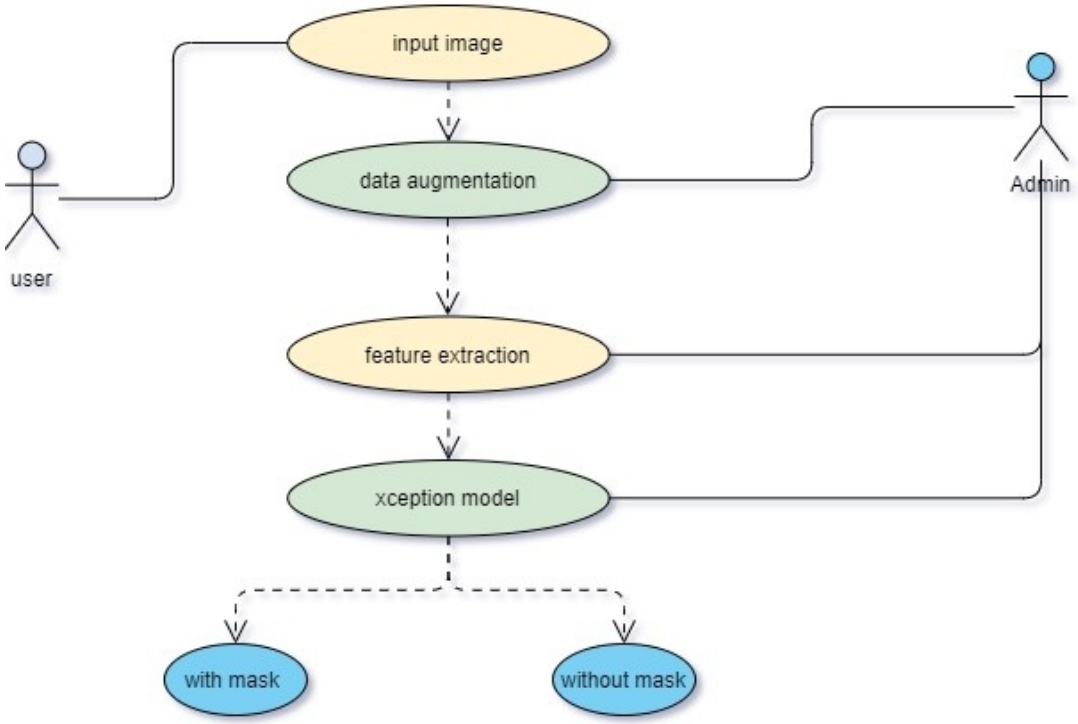

Figure 3.6.2: Use Case Diagram of XceptionV3 Model

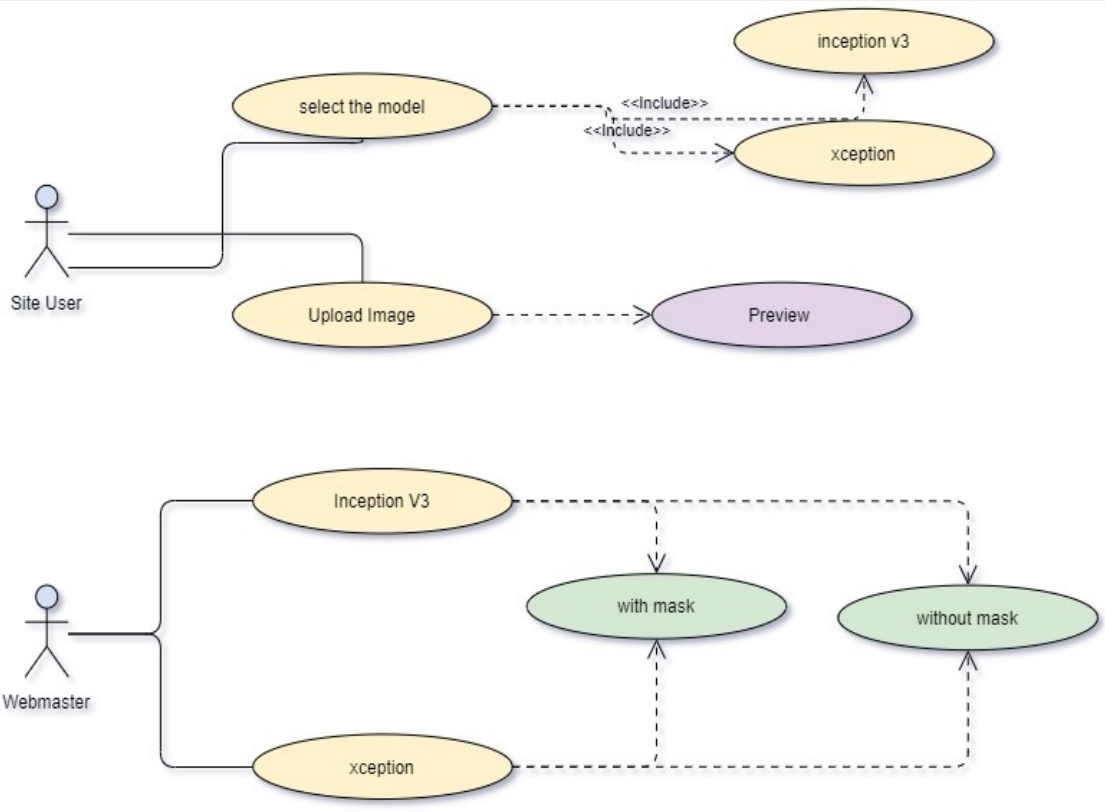

Figure 3.6.3: Use Case Diagram of Web Interfacing

4.METHODOLOGY

4.1 Dataset:

The algorithms are tested on a dataset which consists of various images of faces with mask and without mask. Some of the sample images from the dataset are shown below.

Figure 4.1.1: Images without mask

Figure 4.1.2: Images with mask

As the data is collected from different sources, the data may not be similar in terms of quality,shape,resolution etc. Because of this, it is hard to extract the features and perform classification. So, image pre-processing and augmentation needs to be done to make all the images look similar in terms of different parameters. Data augmentation includes different rotation,flipping,shearing,etc. At last, all the images are converted into 224*224 size.

Class Label	Sample Count
Faces with mask	1900
Faces without mask	2165
Total	4065

Table 4.1: Classes of images and count of their samples

4.2 Algorithms:

There are many deep learning models like VGG16, VGG19, ResNet50, Inception V3 and Xception which can be used for face mask detection. Our proposed system uses InceptionV3 and Xception to build the model. These models are trained on large datasets and extracts features from images and classifies them. In this work, we have compared two algorithms to find the best and optimum model for the problem raised.

InceptionV3:

InceptionV3 is a convolutional neural network which is used for image analysis and object detection. It can load a million size dataset and 48 layers deep. The weights in this model are smaller than other models like VGG and ResNet. It is a multilevel feature extractor.

In this model, each layer performs different operations and forwards the output to the next layer. Each layer performs convolution, avg pooling, max pooling,concatenation. InceptionV3 does 1*1,3*3,5*5 convolution transformation and then a 3*3 max pooling. All that data is filtered and concatenated and passed to the next layer. Every time a filter is added, all the inputs need to convolve to get a single output.

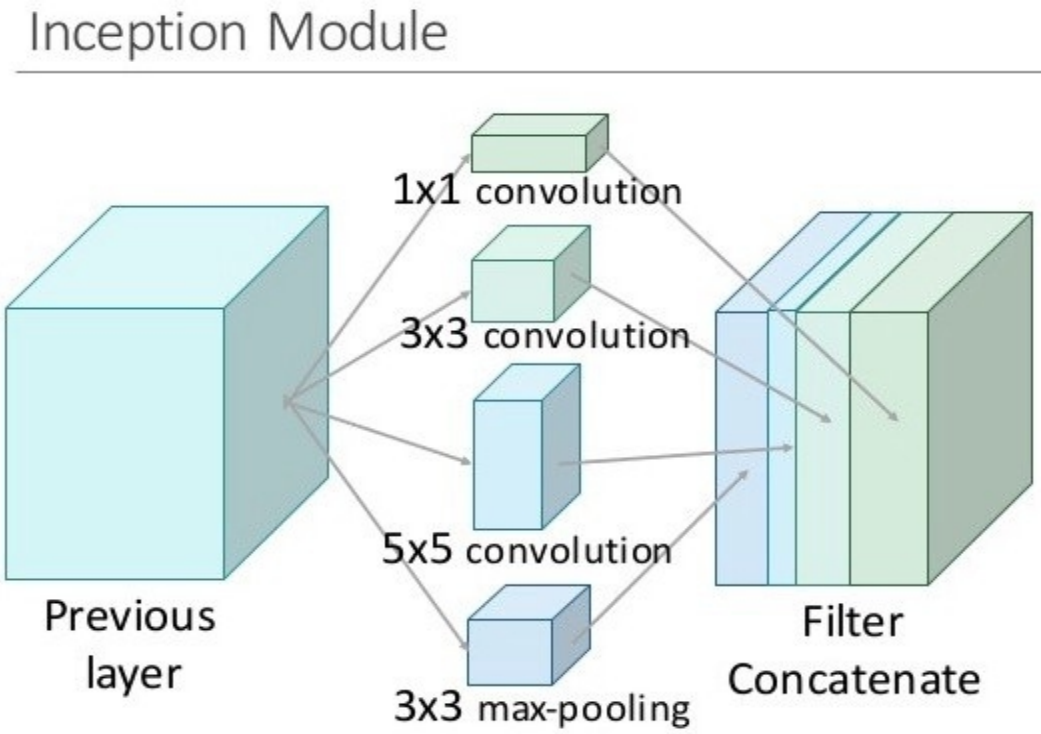

Figure 4.2.1: Architecture of InceptionV3 model

Convolution is a process of applying filters to an input which results in an activation. It is a mathematical combination of 2 functions to produce a third function.A convolution converts all the pixels in its receptive field into a single value. For example, if you would apply a convolution to an image, you will be decreasing the image size as well as bringing all the information in the field together into a single pixel. The final output of the convolution layer is a vector. Convolutional layer, as mentioned above this layer consist of sets of Filters or Kernel. They have a key job of carrying out the convolution operation in the first part of the layer.

The filters take a subset of the input data.The operations performed by this layer are linear multiplications with the objective of extract the high-level features such as edges, from the input image as a convolution operational activity.Since convolutional operation at this layer is a linear operation and the output volume is obtained by stacking the activation maps of all filters along the depth dimension. Linear operation mostly involves the multiplication of weights with the input actually same as in traditional neural network. The mathematical equation is **input+filter/kernel→ feature map.**

Convolution networks are not just limited to only one convolution layer. The first layer is responsible for capturing the low-level features such as colour, edges, gradient orientation etc.

- **Conventional Convolution Layer** – This layer receives a single input which is a feature map and it computes its output by convolving filters across the feature maps from the previous layer.
- **Dynamic Convolution Layer** – This layer receives two input, first one as a feature map from the previous layer and second one a filter.

Image Matrix

1	0	0	1	1
1	0	1	1	1
0	1	1	0	1
0	1	0	1	0
1	1	1	0	1

Filter Matrix

1	0	0
1	0	0
0	1	1

Figure 4.2.2:Example of Convolution layer

Average pooling is done after the convolution layer. Pooling is a process of reducing the spatial resolution without changing the 2D representation. Average pooling returns the average of all the values from the portion of images. This method smooths out the image and is used when the focus is on the lighter pixels. Example of avg pooling is shown below.

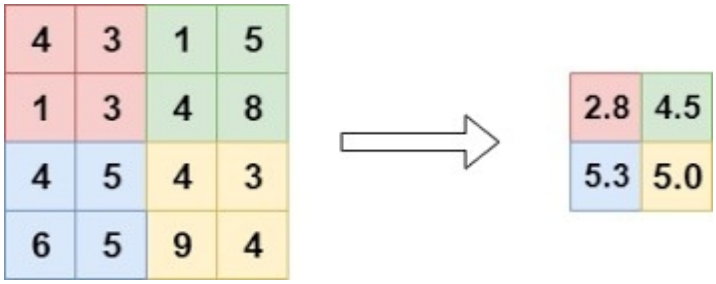

Figure 4.2.3:Example of average pooling

Average Pooling Using Keras:

```
import numpy as np
from keras.models import Sequential
from keras.layers import AveragePooling2D
# define input image
image = np.array([[2, 2, 7, 3],
                  [9, 4, 6, 1],
                  [8, 5, 2, 4],
                  [3, 1, 2, 6]])
image = image.reshape(1, 4, 4, 1)
# define model containing just a single average pooling layer
model = Sequential(
   [AveragePooling2D(pool_size = 2, strides = 2)])
# generate pooled output
output = model.predict(image)
# print output image
output = np.squeeze(output)
print(output)
```

Max Pooling is also a type of pooling which is used when the focus is on the brighter pixels. It is completely opposite to avg pooling. This method returns the maximum value from the portion of images covered by kernel.InceptionV3 uses max pooling. Example is shown in the below figure.

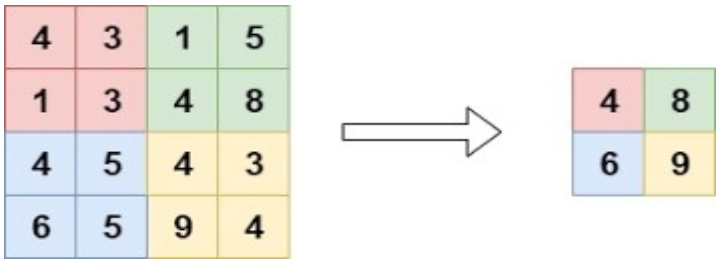

Figure 4.2.4:Example of max pooling

Max Pooling Using Keras:

```
import numpy as np
from keras.models import Sequential
from keras.layers import MaxPooling2D
# define input image
image = np.array([[2, 2, 7, 3],
                  [9, 4, 6, 1],
                  [8, 5, 2, 4],
                  [3, 1, 2, 6]])
image = image.reshape(1, 4, 4, 1)
# define model containing just a single max pooling layer
model = Sequential(
   [MaxPooling2D(pool_size = 2, strides = 2)])
# generate pooled output
output = model.predict(image)
# print output image
output = np.squeeze(output)
print(output)
```

Xception:

It is a convolution neural network which is defined as extreme inception.It is 71 layers deep.Classify is used in this model to classify the images. It has similar features like InceptionV3 but the inception modules are replaced by depthwise separable convolutions. Xception deals with the depth dimensions rather than spatial dimensions. These convolutions are more efficient than the classical convolutions in terms of computation time. InceptionV3 and Xception share the same number of parameters.

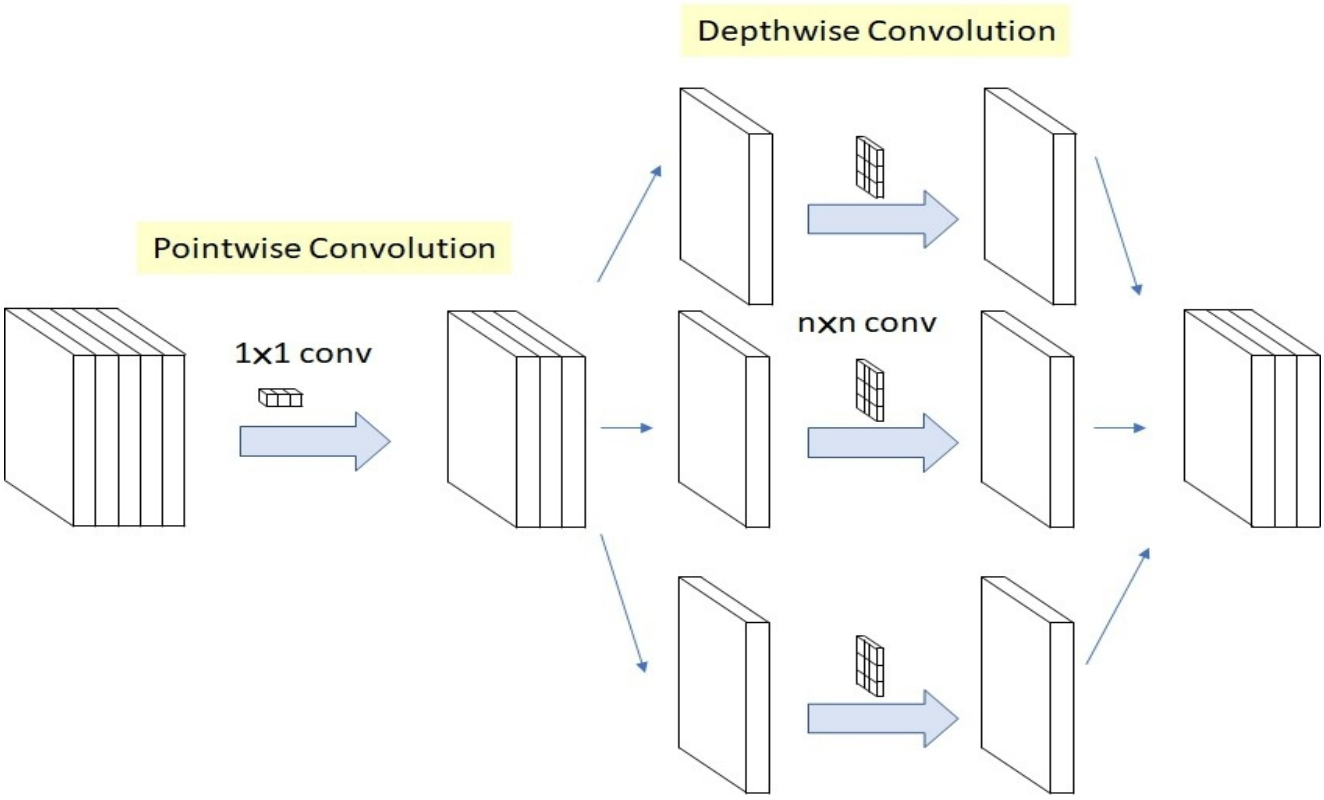

Figure 4.2.5:Architecture of Xception model

Depthwise separable convolution is a conjunction of pointwise and depthwise convolutions. Xception network comprises 3 flows. The data first passes through entry flow and then through middle flow which is repeated 8 times and then through the exit flow. Each flow performs different operations like convolution, max pooling, separable convolution and an activation function.Based on the number of labels used in the model, the activation function varies. Activation function is a node that is placed in between or at the end of the neural network to decide whether the neuron should be eliminated or not.

4.3 Technologies Used:

- **OpenCV:**

To detect the faces, images need to be captured from the video. The proposed system need to be attached with the CCTV cameras for capturing. In such case, a real-time computer vision is necessary. OpenCV is the one which is suitable to perform that function. It is a real-time computer vision which is open source and a machine learning software library. It was built to provide a common infrastructure for computer vision applications. It is easy to use and to modify the code. It is a package of many useful and optimized algorithms used to

detect, recognize faces and objects, track movements etc. It can find similar images from database, can put images together to produce high resolution image.

- **TensorFlow:**

It is an open source platform for managing all aspects of machine learning system and mainly focuses on training machine learning models. The APIs of tensorflow are hierarchically arranged with the high level APIs built on low level APIs. To define and train the models tf.keras is used which a high level API and it is also useful to make predictions. TensorFlow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster. Tensor flow can be used for

- ✓ Easy model building
- ✓ Robust ML production anywhere
- ✓ Powerful experimentation for research

- **Streamlit:**

It is an open source app framework for machine learning and data science**.** Streamlit lets you turn data scripts into shareable web apps in minutes. . We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app. Use the this command to install **pip install streamlit.** After installation, run this command to run the app **streamlit run <yourscript.py> .**

4.4 Behavioural Aspects of the System:

Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. It is used by developers to understand the flow of programs on a high level. It also enables them to figure out constraints and conditions that cause particular events. A flow chart converges into being an activity diagram if complex decisions are being made.

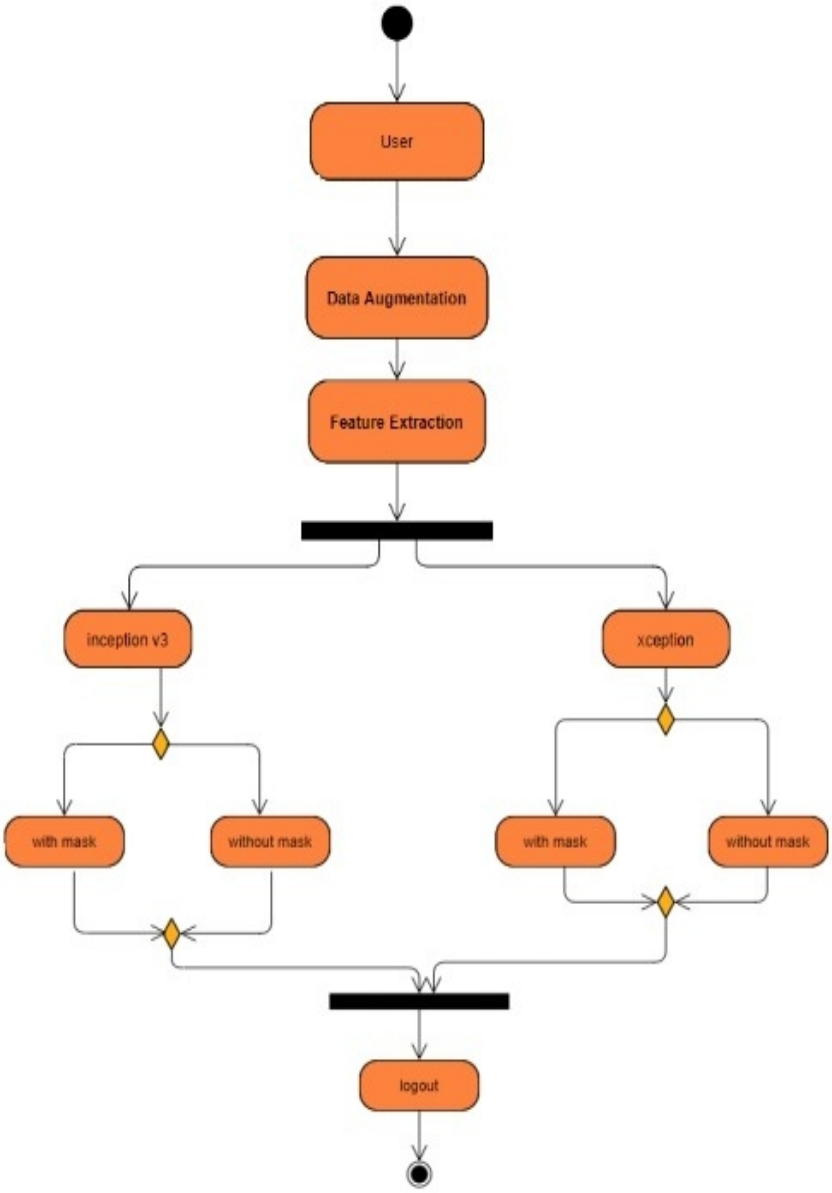

Figure 4.4.1: Activity Diagram

4.5 Flow Chart:

The below figure depicts the flow chart of the system. All the steps done in for the system are clearly mentioned. Those steps are explained in the next chapter. It is started from importing the required and necessary packages and ended with connecting the web which gives the results. As model is trained using two different algorithms, graphs are also plotted.

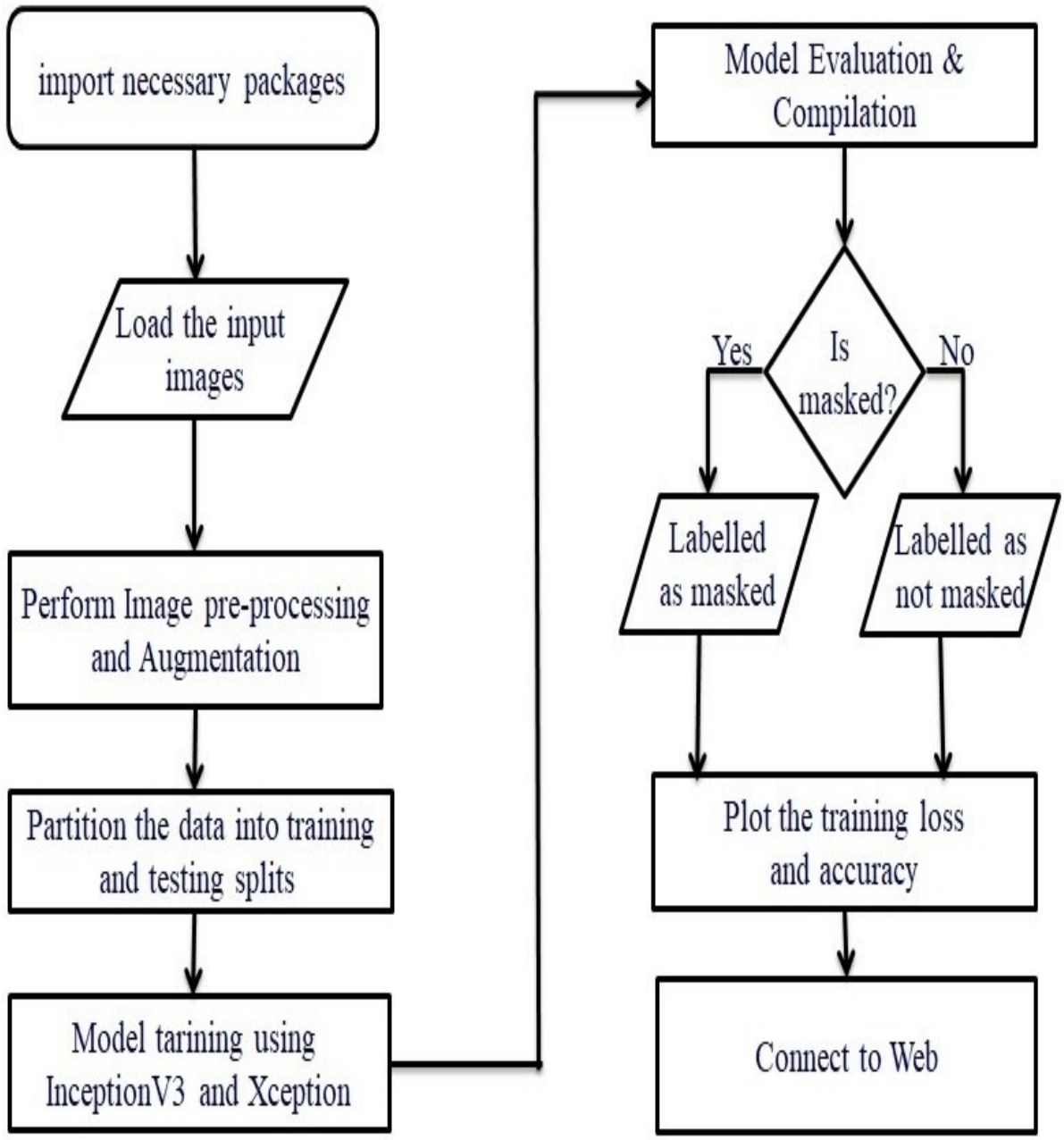

Figure 4.5.1: Flowchart of the proposed system

4.6 Description of steps:

Step-1:Import the necessary packages

from tensorflow.keras.optimizers import Adam:

It is an optimizer that implements the adam algorithm. It is used for designing deep neural networks and is based on adaptive estimation of first and second order moments.

from tensorflow.keras.utils import to_categorical:

To-categorical is a converter which is used to convert a class vector to a binary matrix.

from sklearn.preprocessing import LabelBinarizer:

Label binarizer is used to convert multiclass labels to binary labels by using inverse transform method.

from sklearn.model_selection import train_test_split:

train_test_split is used to divide the dataset into two subsets. It is a technique to calculate the performance of the algorithm.

from sklearn.metrics import classification_report:

Classification_report is used to generate the training report.

from imutils import paths:

Imutils is a series of functions to perform on an image like translation,rotation,resizing etc. It is also useful to display the matplotlib images with openCV and python. This package is used to change the paths of images into one list under one path.

import matplotlib.pyplot as plt:

Pyplot is a collection of functions in the matplotlib package. It is used to create a figure,create a plotting area,lines etc.

import numpy as np:

Numpy is a python library which is used when working with arrays concept.Creating a namespace for numpy as np.

import tensorflow_hub as hub:

Tensorflow hub is a storage of trained ML models which can be used and deployed anywhere. Creating a namespace for tensorflow_hub as hub.

import tensorflow as tf:

Tensorflow is used to create dataflow graphs which is used to understand how the data is moving through it. It is used for classification,prediction,understanding,creation etc.Creating a namespace for tensorflow as tf.

from tensorflow.keras.preprocessing.image import img_to_array:

It is one of the pre-processing techniques applied on dataset to convert the images into array form.

from tensorflow.keras.preprocessing.image import ImageDataGenerator:

It is also one of the pre-processing techniques applied on dataset to perform transformations and normalization techniques during training the data.

from tensorflow.keras.preprocessing.image import load_img:

This function is used to load the image along with some arguments such as grayscale,target_size,color_mode etc.if grayscale is opted as true then image is loaded as grayscale.if target_size is none, then it defaults to original size.

from tensorflow.keras import layers

from tensorflow.keras.layers import Input

import argparse:

Argparse is a parser which is the most common and friendly command line interface for command line options and arguments. It gives errors and help messages when invalid arguments are entered.

from tensorflow.keras.applications import InceptionV3:

This function is used to import the InceptionV3 classes and all the functions in it.

from tensorflow.keras.applications import Xception:

This function is used to import the Xception classes and all the functions in it.

Step-2:Load the input images

Figure 4.6.1:Loading of dataset into the model

imagePaths = list(paths.list_images("/content/drive/MyDrive/DataSet"))

print(imagePaths)

The data is loaded and is stored in imagePaths. Then it is printed to get the data. Data is divided into channels and size of the data is as mentioned. Number of labels for the data is two. For data and labels, two arrays have been created as follows.

```
data  =  []
labels = []
IMG_SIZE = 224
CHANNELS = 3
N_LABELS=2
```

Step-3:Perform Image pre-processing and augmentation

```
# load the input image (224x224) and preprocess it
  image = load_img(imagePath, target_size=(IMG_SIZE, IMG_SIZE))
  image = img_to_array(image)
  image = image/255
#image = preprocess_input(image)
  # update the data and labels lists, respectively
  data.append(image)
  labels.append(label)
data = np.array(data, dtype="float32")
labels = np.array(labels)
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
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")
```

Step-4:Partition the data into training and test splits

This can be done by using the train_test_split function as follows:

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

trainX and testX are used to make up the image itself while trainY and testY are used to make up the labels in the above line of code.

Next phase is training the model using 2 algorithms. Different methods are used to train the model like flatten,dense,dropout for feature extraction. Activation functions like ReLu, sigmoid are also used.

Step-5:Model training using InceptionV3 and Xception

InceptionV3:

```
feature_extractor_layer=InceptionV3(weights="imagenet",
include_top=False,input_tensor=Input(shape=(IMG_SIZE,IMG_SIZE,CHANNELS)))
feature_extractor_layer.trainable = False
model =
   tf.keras.Sequential([ feature_e
   xtractor_layer,
   layers.Flatten(name="flatten"),
   layers.Dense(1024, activation='relu', name='hidden_layer'),
   layers.Dropout(0.5),
   layers.Dense(N_LABELS, activation='sigmoid', name='output')    ])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Functional)    (None, 5, 5, 2048)        21802784
_________________________________________________________________
flatten (Flatten)            (None, 51200)             0
_________________________________________________________________
hidden_layer (Dense)         (None, 1024)              52429824
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
output (Dense)               (None, 2)                 2050
=================================================================
Total params: 74,234,658
Trainable params: 52,431,874
Non-trainable params: 21,802,784
_________________________________________________________________
```

Figure 4.6.2:Trained InceptionV3 model

Xception:

```
feature_extractor_layer = Xception(weights="imagenet", include_top=False,
input_tensor=Input(shape=(IMG_SIZE,IMG_SIZE,CHANNELS)))
feature_extractor_layer.trainable = False
model =
   tf.keras.Sequential([ feature_e
   xtractor_layer,
   layers.Flatten(name="flatten"),
   layers.Dense(1024, activation='relu', name='hidden_layer'),
   layers.Dropout(0.5),
   layers.Dense(N_LABELS, activation='sigmoid', name='output')
])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
xception (Functional)        (None, 7, 7, 2048)        20861480
_________________________________________________________________
flatten (Flatten)            (None, 100352)            0
_________________________________________________________________
hidden_layer (Dense)         (None, 1024)              102761472
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
output (Dense)               (None, 2)                 2050
=================================================================
Total params: 123,625,002
Trainable params: 102,763,522
Non-trainable params: 20,861,480
_________________________________________________________________
```

Figure 4.6.3:Trained Xception model

In the above code features are extracted from the images separately for each model. And then performing operations on layers like flatten,dense,dropout.

Dropout is used to prevent overfitting. It is only applied on layers when training is set to true to avoid the dropout of values during inference.

Flatten is used to flatten any argument given. It is used to reshape in a way that is equal to the number of elements present in the tensor.

Dense is a deeply connected neural network layer which receives an input and performs the following operation to give the output.

output=activation function(dot(input,kernel)+bias)

In this, two activation functions are used. They are ReLu and Sigmoid. Activation function is a node placed at the end or in between the network to determine the output of a neural network model. It determines whether a neuron should be terminated or not. It is useful to normalize the output if the input is in the range of 0 to 1.

ReLu is a rectified linear activation function which outputs the input directly if the input is positive, otherwise outputs zero. It allows the models to learn faster and perform better.The advantage is it does not activate all the neurons at a time.It overcomes the vanishing gradient problem.

Sigmoid is the most suitable activation function when there are only 2 labels either 0 or 1 in the model. It is mainly used for those models where we have to predict the probability as the output.

Next phase in the process is model evaluation and compilation. In this phase, the trained model is executed and evaluated against the evaluation metrics. For each model,learning rate,epochs,batch size varies.The evaluation metrics are precision,recall,f1-score and support.

Step-6:Model Evaluation and Compilation

InceptionV3:

```
LR = 1e-5 # Keep it small when transfer learning
EPOCHS = 20
BS = 256
#for sigmod , we use the binary classification is binary_crossentropy
model.compile( optimizer=tf.keras.optimizers.Adam(learning_rate=
LR), loss="binary_crossentropy",
metrics=["accuracy"])
import time
start = time.time()
history = model.fit(aug.flow(trainX, trainY, batch_size=BS),
steps_per_epoch=len(trainX) // BS,
```

validation_data=(testX, testY),

epochs=EPOCHS)

print('\nTraining took {}'.format((time.time()-start)))

Learning Rate: It is the parameter which controls the model to change it as a response to error time when the model weights are updated. A good learning rate is either 0.1 or 0.01.

Epochs: It is used to train the neural network. It indicates the number of times the algorithm will work through the training dataset.

If batch size is equal to complete training dataset, then epochs are equal to number of iterations.

```
Epoch 1/20
12/12 [==============================] - 425s 35s/step - loss: 0.3115 - accuracy: 0.8896 - val_loss: 0.0679 - val_accuracy: 0.9841
Epoch 2/20
12/12 [==============================] - 418s 35s/step - loss: 0.0873 - accuracy: 0.9685 - val_loss: 0.0634 - val_accuracy: 0.9890
Epoch 3/20
12/12 [==============================] - 417s 35s/step - loss: 0.0687 - accuracy: 0.9768 - val_loss: 0.0608 - val_accuracy: 0.9927
Epoch 4/20
12/12 [==============================] - 417s 35s/step - loss: 0.0522 - accuracy: 0.9851 - val_loss: 0.0599 - val_accuracy: 0.9878
Epoch 5/20
12/12 [==============================] - 418s 35s/step - loss: 0.0499 - accuracy: 0.9857 - val_loss: 0.0548 - val_accuracy: 0.9927
Epoch 6/20
12/12 [==============================] - 418s 35s/step - loss: 0.0506 - accuracy: 0.9834 - val_loss: 0.0510 - val_accuracy: 0.9927
Epoch 7/20
12/12 [==============================] - 418s 35s/step - loss: 0.0434 - accuracy: 0.9874 - val_loss: 0.0471 - val_accuracy: 0.9927
Epoch 8/20
12/12 [==============================] - 418s 35s/step - loss: 0.0414 - accuracy: 0.9871 - val_loss: 0.0451 - val_accuracy: 0.9915
Epoch 9/20
12/12 [==============================] - 418s 35s/step - loss: 0.0341 - accuracy: 0.9897 - val_loss: 0.0434 - val_accuracy: 0.9915
Epoch 10/20
12/12 [==============================] - 418s 35s/step - loss: 0.0298 - accuracy: 0.9934 - val_loss: 0.0412 - val_accuracy: 0.9939
Epoch 11/20
12/12 [==============================] - 417s 35s/step - loss: 0.0449 - accuracy: 0.9861 - val_loss: 0.0401 - val_accuracy: 0.9927
Epoch 12/20
12/12 [==============================] - 417s 35s/step - loss: 0.0291 - accuracy: 0.9907 - val_loss: 0.0399 - val_accuracy: 0.9902
Epoch 13/20
12/12 [==============================] - 417s 35s/step - loss: 0.0287 - accuracy: 0.9897 - val_loss: 0.0398 - val_accuracy: 0.9939
Epoch 14/20
12/12 [==============================] - 418s 35s/step - loss: 0.0231 - accuracy: 0.9940 - val_loss: 0.0384 - val_accuracy: 0.9927
Epoch 15/20
12/12 [==============================] - 419s 35s/step - loss: 0.0318 - accuracy: 0.9894 - val_loss: 0.0394 - val_accuracy: 0.9878
Epoch 16/20
12/12 [==============================] - 418s 35s/step - loss: 0.0298 - accuracy: 0.9901 - val_loss: 0.0387 - val_accuracy: 0.9939
Epoch 17/20
12/12 [==============================] - 418s 35s/step - loss: 0.0248 - accuracy: 0.9920 - val_loss: 0.0390 - val_accuracy: 0.9927
Epoch 18/20
12/12 [==============================] - 417s 35s/step - loss: 0.0299 - accuracy: 0.9917 - val_loss: 0.0429 - val_accuracy: 0.9878
Epoch 19/20
12/12 [==============================] - 417s 35s/step - loss: 0.0208 - accuracy: 0.9937 - val_loss: 0.0420 - val_accuracy: 0.9915
Epoch 20/20
12/12 [==============================] - 417s 35s/step - loss: 0.0234 - accuracy: 0.9920 - val_loss: 0.0432 - val_accuracy: 0.9915

Training took 8676.431966304779
```

Figure 4.6.4:Evaluated and compiled InceptionV3 model

Xception:

LR = 1e-5 # Keep it small when transfer learning

EPOCHS = 20

BS = 256

#for sigmoid , we use the binary classification is binary_crossentropy

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=

LR),

```
loss="binary_crossentropy",
metrics=["accuracy"])
import time
start = time.time()
history = model.fit(aug.flow(trainX, trainY, batch_size=BS),
steps_per_epoch=len(trainX) // BS,
validation_data=(testX, testY),
epochs=EPOCHS)
print('\nTraining took {}'.format((time.time()-start)))
```

```
Epoch 1/20
12/12 [==============================] - 522s 45s/step - loss: 0.2372 - accuracy: 0.9039 - val_loss: 0.0782 - val_accuracy: 0.9841
Epoch 2/20
12/12 [==============================] - 524s 44s/step - loss: 0.0595 - accuracy: 0.9824 - val_loss: 0.0747 - val_accuracy: 0.9829
Epoch 3/20
12/12 [==============================] - 527s 44s/step - loss: 0.0458 - accuracy: 0.9857 - val_loss: 0.0712 - val_accuracy: 0.9841
Epoch 4/20
12/12 [==============================] - 535s 45s/step - loss: 0.0407 - accuracy: 0.9857 - val_loss: 0.0643 - val_accuracy: 0.9866
Epoch 5/20
12/12 [==============================] - 526s 45s/step - loss: 0.0320 - accuracy: 0.9894 - val_loss: 0.0539 - val_accuracy: 0.9866
Epoch 6/20
12/12 [==============================] - 520s 44s/step - loss: 0.0345 - accuracy: 0.9891 - val_loss: 0.0480 - val_accuracy: 0.9866
Epoch 7/20
12/12 [==============================] - 527s 45s/step - loss: 0.0279 - accuracy: 0.9899 - val_loss: 0.0454 - val_accuracy: 0.9853
Epoch 8/20
12/12 [==============================] - 527s 45s/step - loss: 0.0216 - accuracy: 0.9940 - val_loss: 0.0455 - val_accuracy: 0.9853
Epoch 9/20
12/12 [==============================] - 525s 44s/step - loss: 0.0270 - accuracy: 0.9917 - val_loss: 0.0455 - val_accuracy: 0.9853
Epoch 10/20
12/12 [==============================] - 530s 46s/step - loss: 0.0207 - accuracy: 0.9947 - val_loss: 0.0421 - val_accuracy: 0.9853
Epoch 11/20
12/12 [==============================] - 536s 45s/step - loss: 0.0231 - accuracy: 0.9937 - val_loss: 0.0387 - val_accuracy: 0.9853
Epoch 12/20
12/12 [==============================] - 523s 44s/step - loss: 0.0182 - accuracy: 0.9954 - val_loss: 0.0387 - val_accuracy: 0.9866
Epoch 13/20
12/12 [==============================] - 530s 45s/step - loss: 0.0192 - accuracy: 0.9950 - val_loss: 0.0358 - val_accuracy: 0.9878
Epoch 14/20
12/12 [==============================] - 528s 45s/step - loss: 0.0125 - accuracy: 0.9957 - val_loss: 0.0375 - val_accuracy: 0.9866
Epoch 15/20
12/12 [==============================] - 526s 44s/step - loss: 0.0162 - accuracy: 0.9964 - val_loss: 0.0354 - val_accuracy: 0.9878
Epoch 16/20
12/12 [==============================] - 537s 45s/step - loss: 0.0151 - accuracy: 0.9958 - val_loss: 0.0338 - val_accuracy: 0.9890
Epoch 17/20
12/12 [==============================] - 518s 44s/step - loss: 0.0164 - accuracy: 0.9937 - val_loss: 0.0331 - val_accuracy: 0.9866
Epoch 18/20
12/12 [==============================] - 532s 45s/step - loss: 0.0154 - accuracy: 0.9964 - val_loss: 0.0332 - val_accuracy: 0.9878
Epoch 19/20
12/12 [==============================] - 527s 45s/step - loss: 0.0135 - accuracy: 0.9967 - val_loss: 0.0330 - val_accuracy: 0.9878
Epoch 20/20
12/12 [==============================] - 535s 45s/step - loss: 0.0144 - accuracy: 0.9960 - val_loss: 0.0330 - val_accuracy: 0.9927

Training took 10885.230921983719
```

Figure 4.6.5:Evaluated and compiled Xception model

Confusion Matrix:

InceptionV3:

```
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
print(classification_report(testY.argmax(axis=1), predIdxs,target_names=lb.classes_))
```

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

   with_mask       0.99      0.99      0.99       433
without_mask       0.99      0.99      0.99       386

    accuracy                           0.99       819
   macro avg       0.99      0.99      0.99       819
weighted avg       0.99      0.99      0.99       819
```

Figure 4.6.6:Evaluation metrics of InceptionV3 model

Xception:

```
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
predIdxs = np.argmax(predIdxs, axis=1)
print(classification_report(testY.argmax(axis=1), predIdxs,target_names=lb.classes_))
```

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

   with_mask       1.00      0.99      0.99       433
without_mask       0.99      1.00      0.99       386

    accuracy                           0.99       819
   macro avg       0.99      0.99      0.99       819
weighted avg       0.99      0.99      0.99       819
```

Figure 4.6.7:Evaluation metrics of Xception model

Evaluation Metrics:

Precision: The ability of a model to identify only the relevant data points.

Precision=TP/(TP+FP)

Recall: The ability of a model to find all the relevant cases within a data set.

Recall=TP/(TP+FN)

F1-Score: It is mostly used to compare the performance of 2 classifiers. It is the harmonic mean of precision and recall. A good f1-score means that it should have a low false positives and false negatives.

F1-Score=2*(Precision*Recall)/(Precision+Recall)

Support: It is mainly used to evaluate the quality of output of a classifier on a dataset. Number of actual occurrences of the class in a specified dataset.

Support=No. Of occurrences/Total Occurrences

5.IMPLEMENTATION IN LOCAL SYSTEM

5.1 Input and Output Of InceptionV3 Model:

Input Image Without Mask:

Figure 5.1.1:Image of a person without mask

Output Image Without Mask:

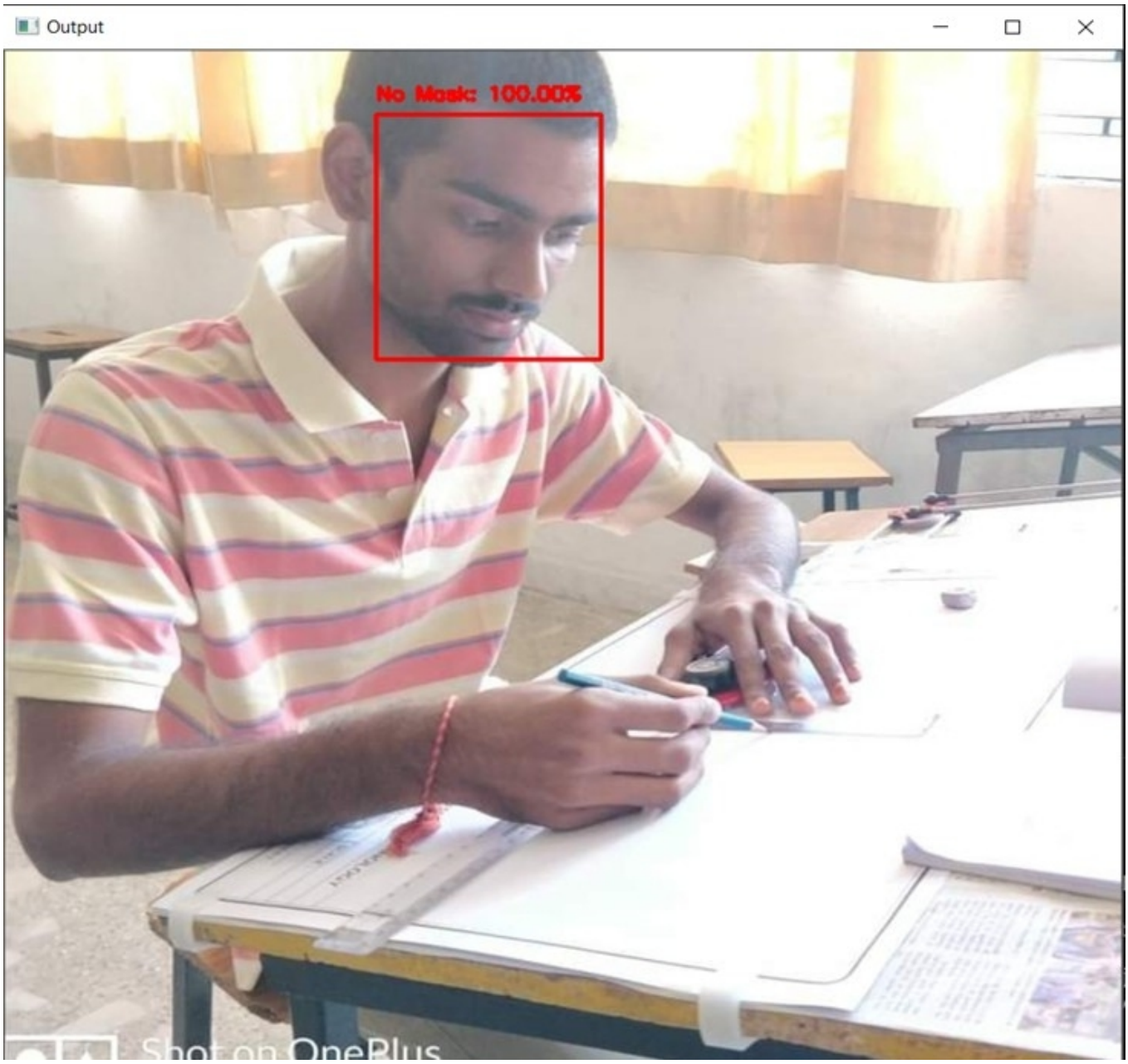

Figure 5.1.2: Output image with label that denotes no mask using InceptionV3 model

Input Image With Mask:

Figure 5.1.3:Image of people with masks

Output Image With Mask:

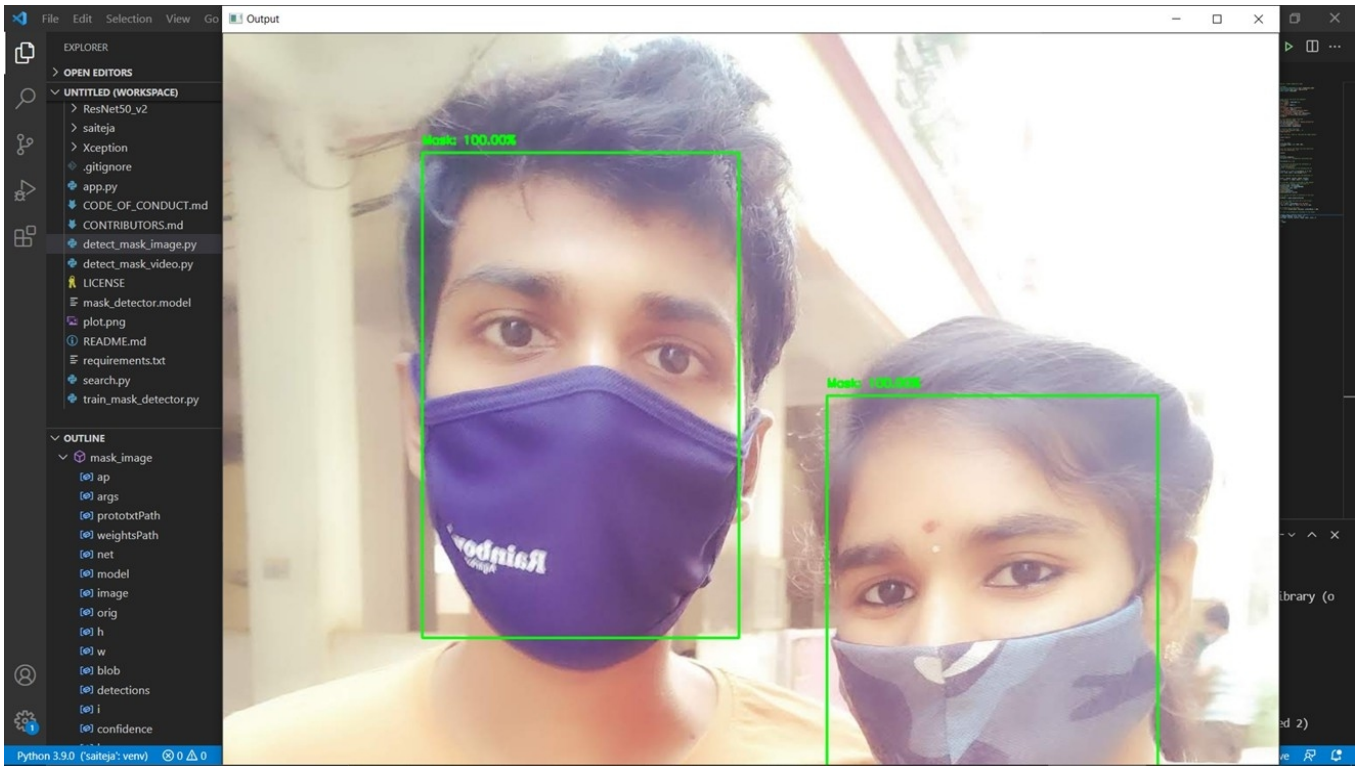

Figure 5.1.4:Output image with label that denotes mask usingInceptionV3 model

5.2 Input and Output of Xception Model:

Input Image Without Mask:

Figure 5.2.1:Image of person without mask

Output Image Without Mask:

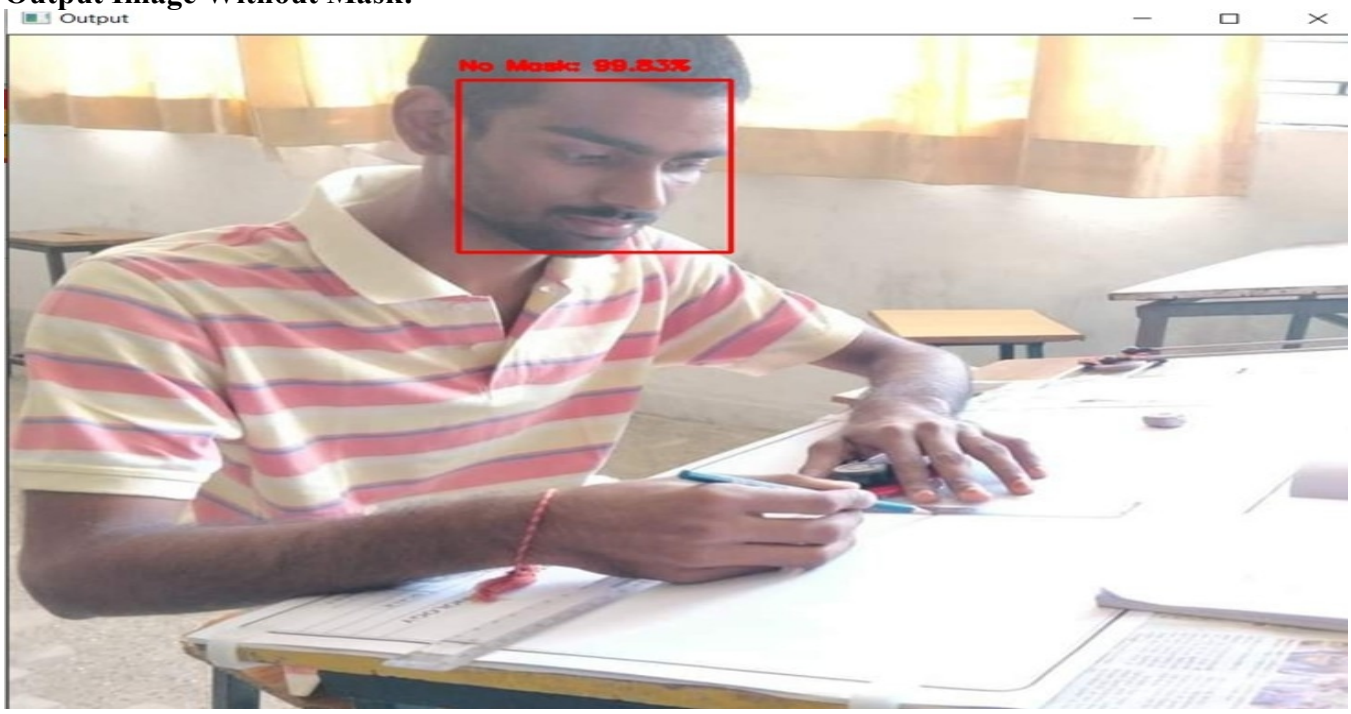

Figure 5.2.2:Output image with label denoting that person is without mask using Xception model

Input Image With Mask:

Figure 5.2.3:Image of person with mask

Output Image With Mask:

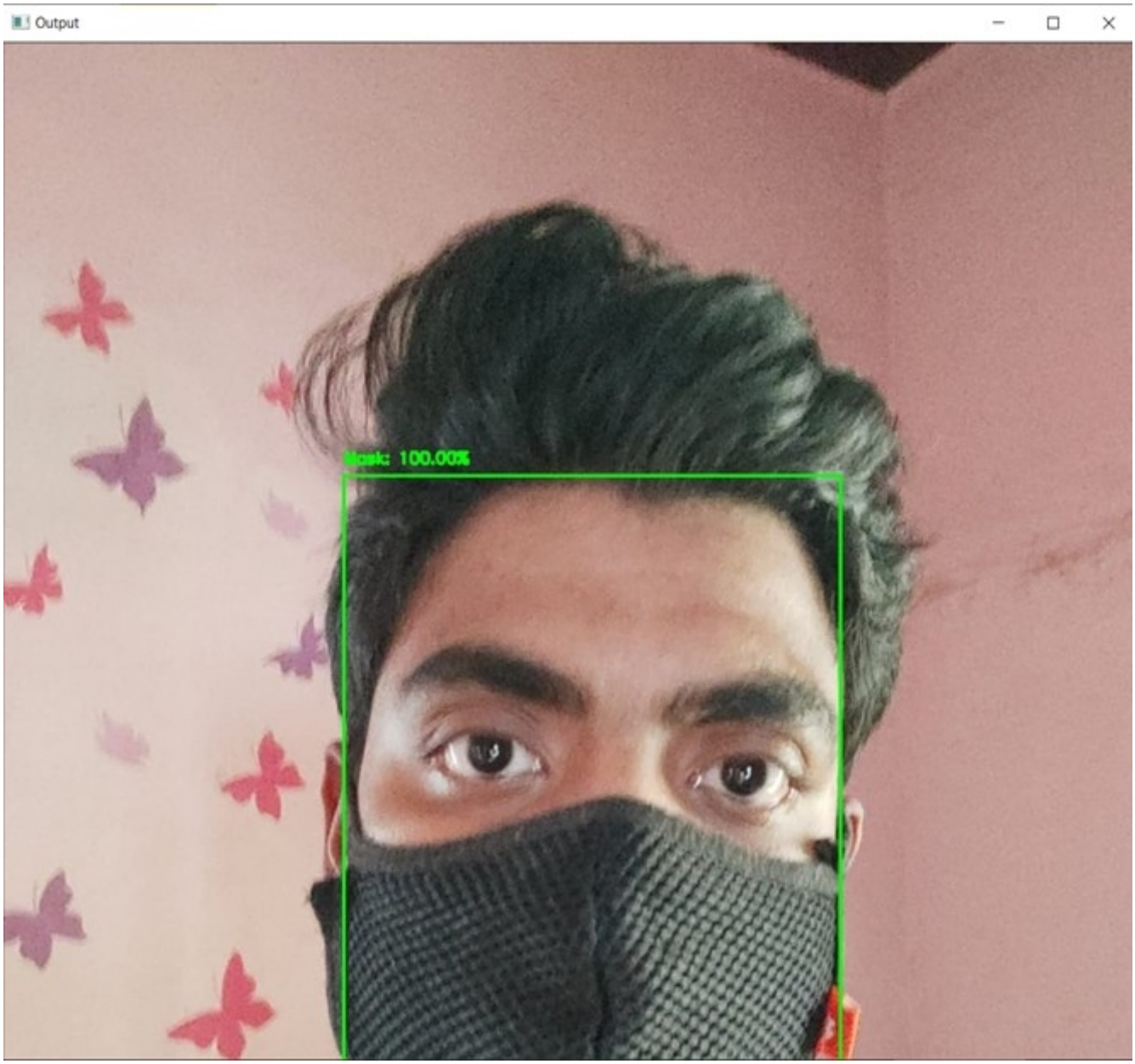

Figure 5.2.4:Output image with label denoting that person is with mask using Xception model

5.3 Detection of face mask in real-time monitoring:

Using Inception Model:

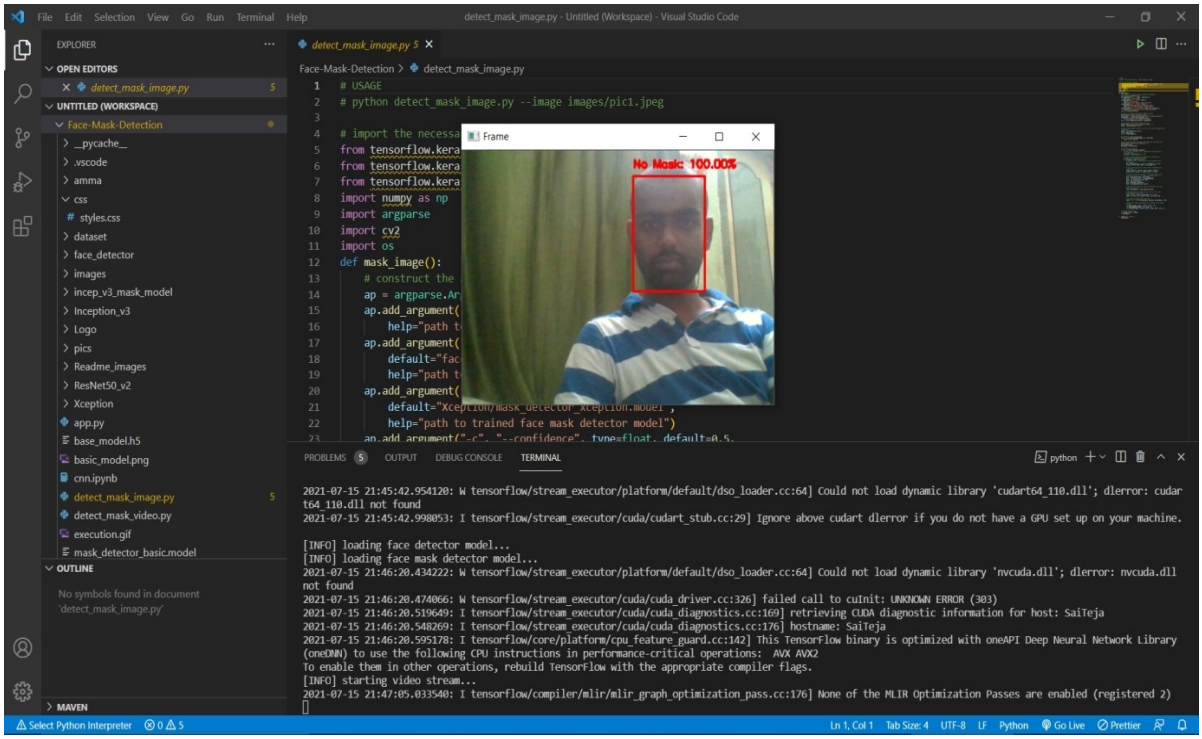

Figure 5.3.1: Classifying the face as no mask using InceptionV3 model

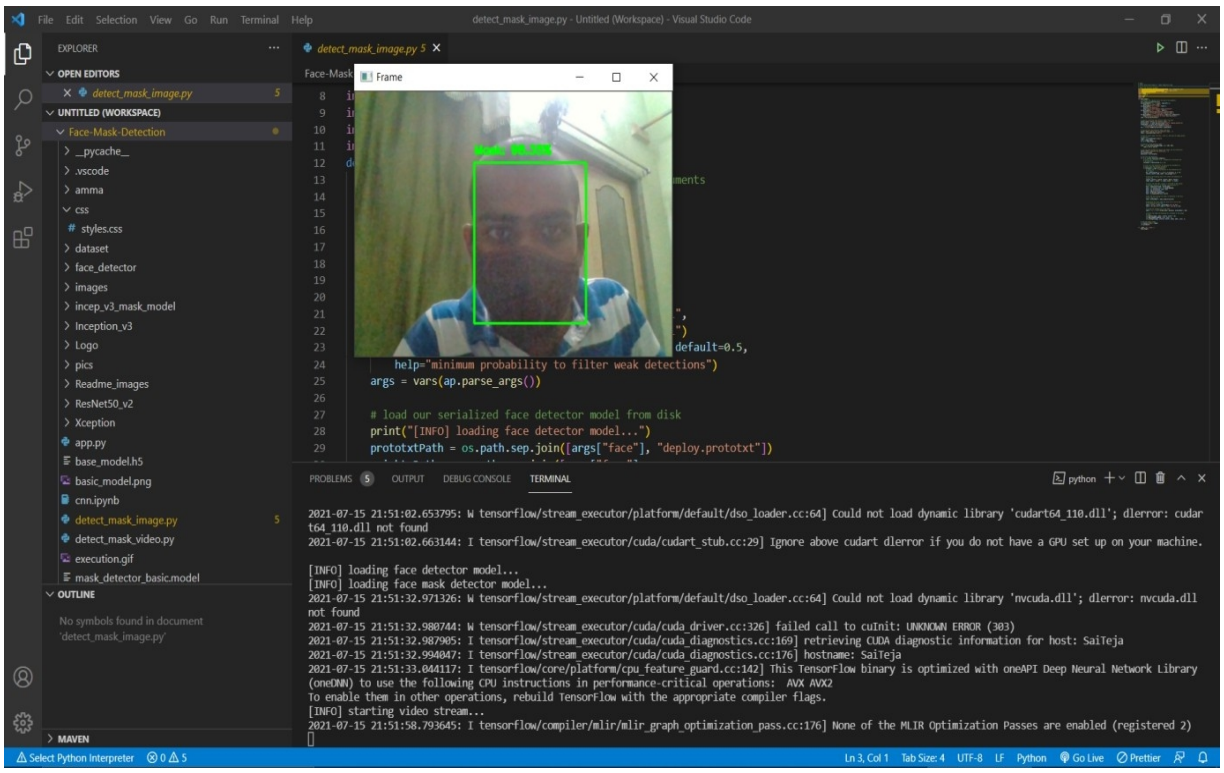

Figure 5.3.2: Classifying the face as masked using InceptionV3 model

Using Xception Model:

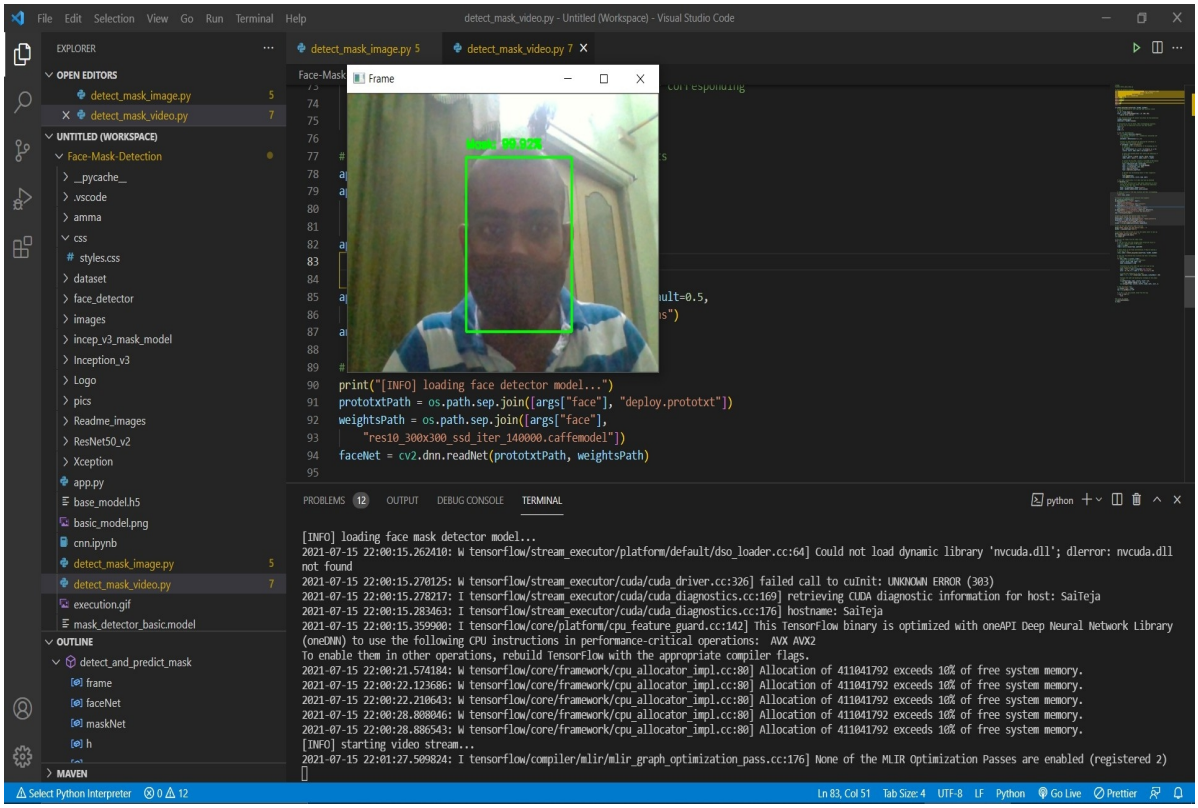

Figure 5.3.3: Classifying the face as masked using Xception model

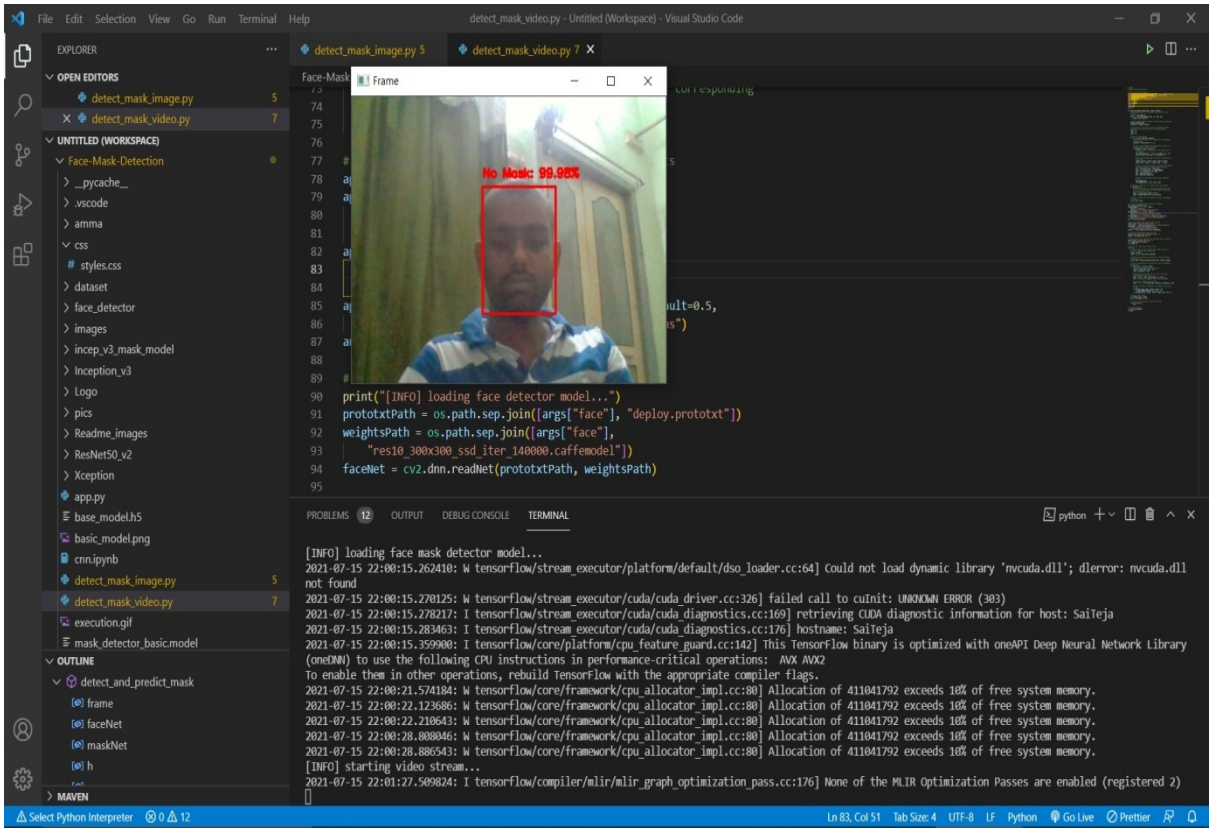

Figure 5.3.4: Classifying the face as no mask using Xception model

6.RESULT ANALYSIS

Training loss and accuracy:

InceptionV3:

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

Figure 6.1:Graph depicting the loss and accuracy of inceptionV3 model

Xception:

```
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), history.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), history.history["val_loss"], label="val_loss")
```

```
plt.plot(np.arange(0, N), history.history["accuracy"], label="accuracy")
plt.plot(np.arange(0, N), history.history["val_accuracy"], label="val_accuracy")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

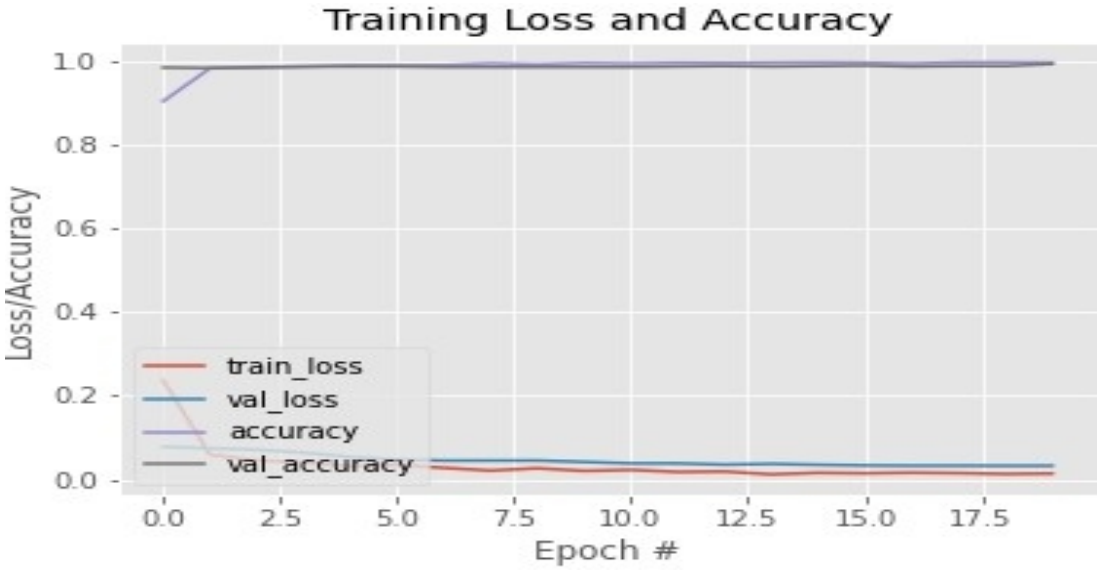

Figure 6.2:Graph depicting the loss and accuracy of Xception model

Trained Model	Accuracy
InceptionV3	99.15%
Xception	99.27%

Table.6.1: Comparison of trained models with their accuracy

7.CONNECTING TO WEB

7.1 Components:

1. **Buttons:**

- **Upload or Browse Files:** This button is used to upload the files in the model trained for classification
- **Process Button:** After the image is uploaded, click on process button to process the model and give output.

2. **Message:**

- After uploading the image using browse files, a message as **image uploaded successfully** is flashed on the screen

7.2 Input and Output of Each Model:

Input Image:

Figure 7.2.1: Input image to check the output when connected with web

Uploading of Image into InceptionV3 Model:

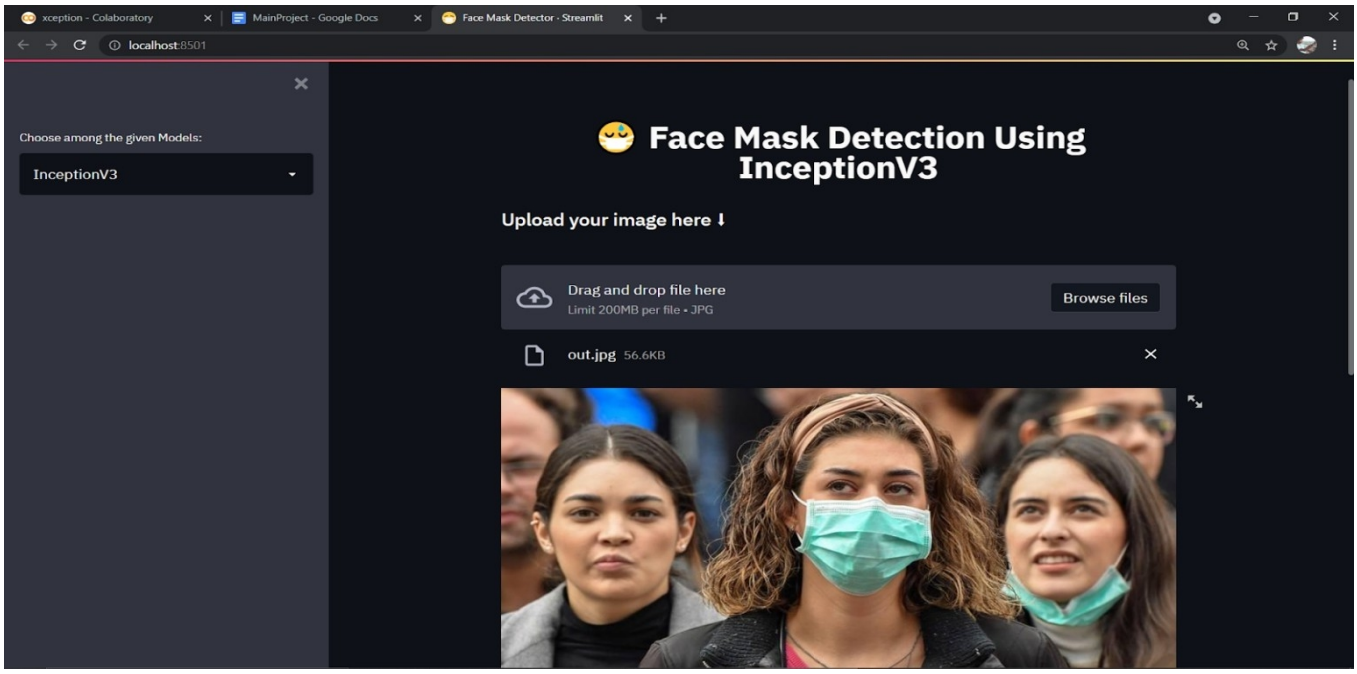

Figure 7.2.2: Uploading of image to check InceptionV3 model

Output of InceptionV3 Model:

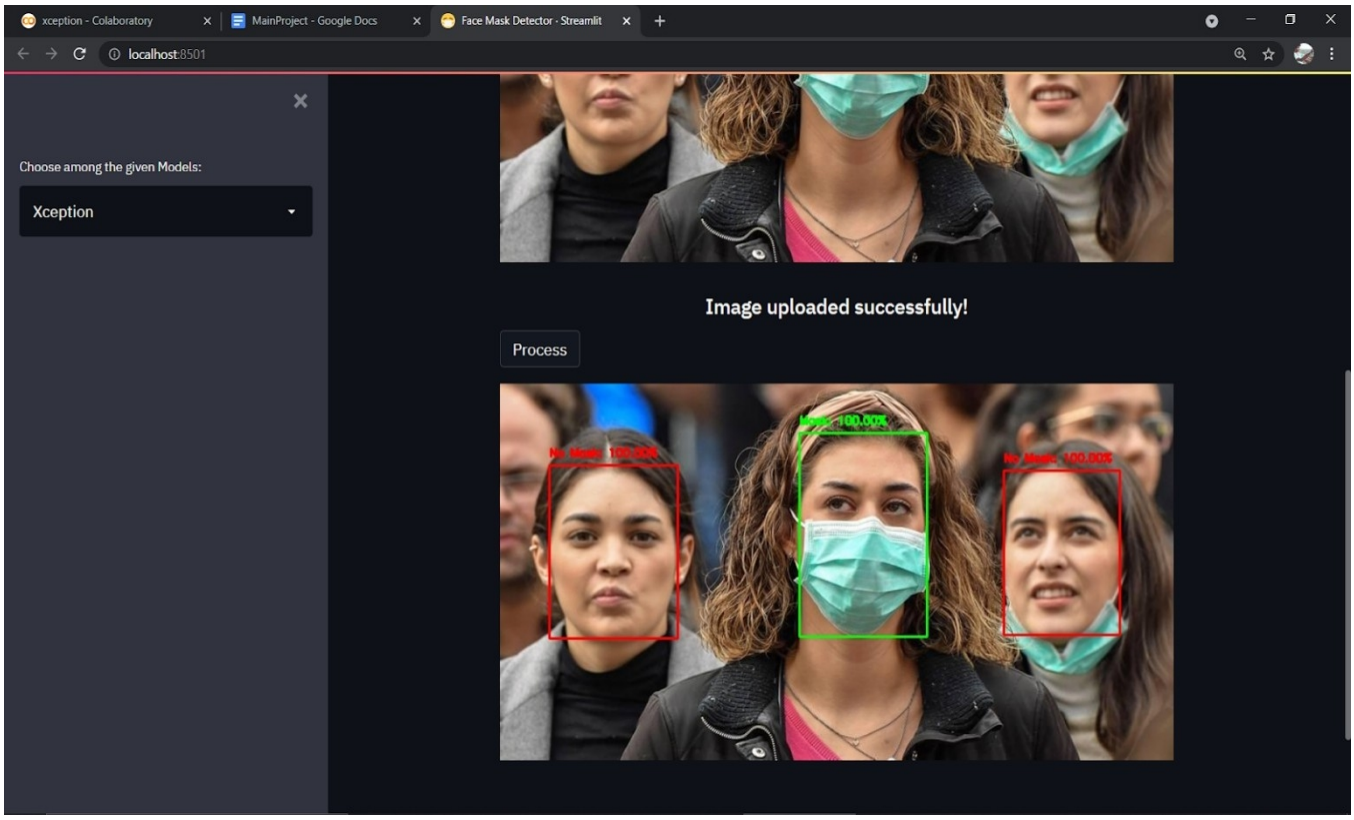

Figure 7.2.3: Classifies the persons with labels when connected with web using InceptionV3 model

Uploading of Image into Xception Model:

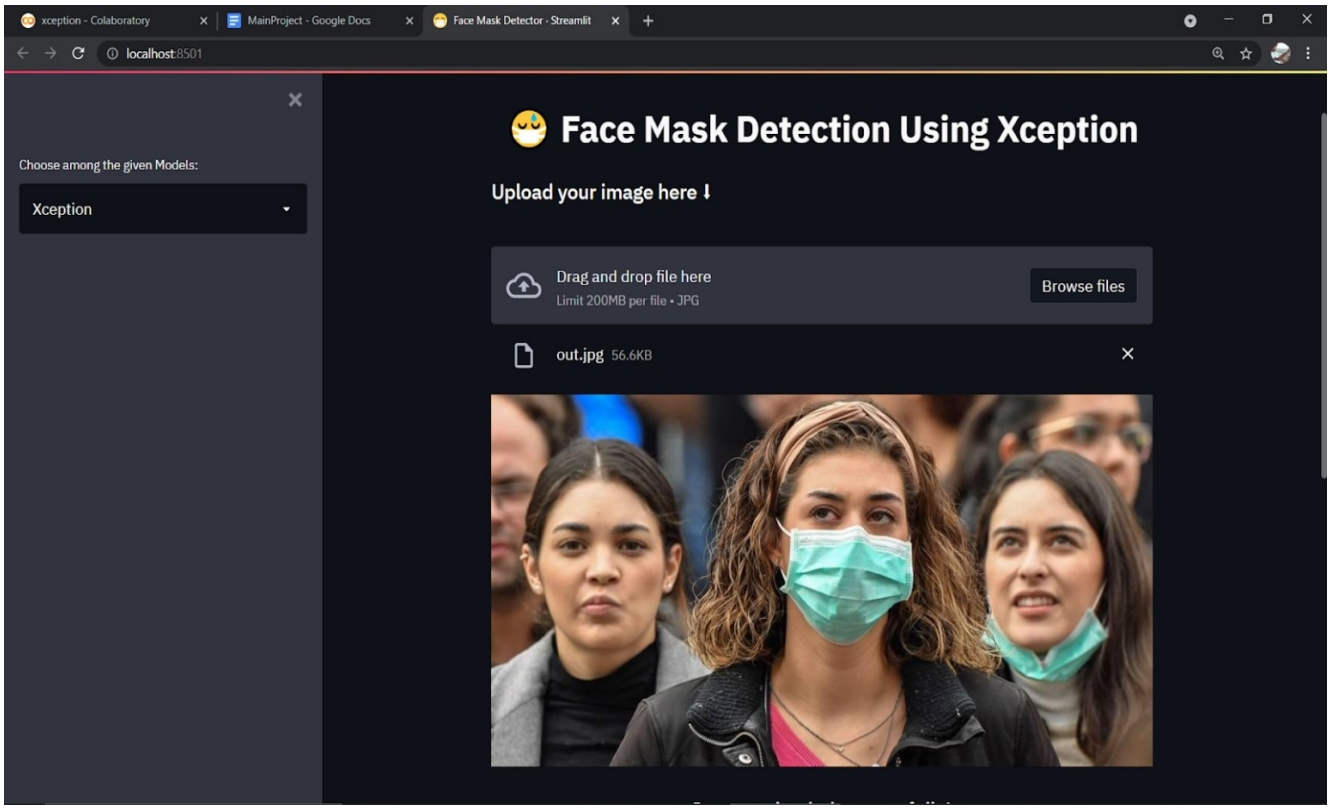

Figure 7.2.4: Uploading of image to check Xception model

Output of XceptionV3 Model:

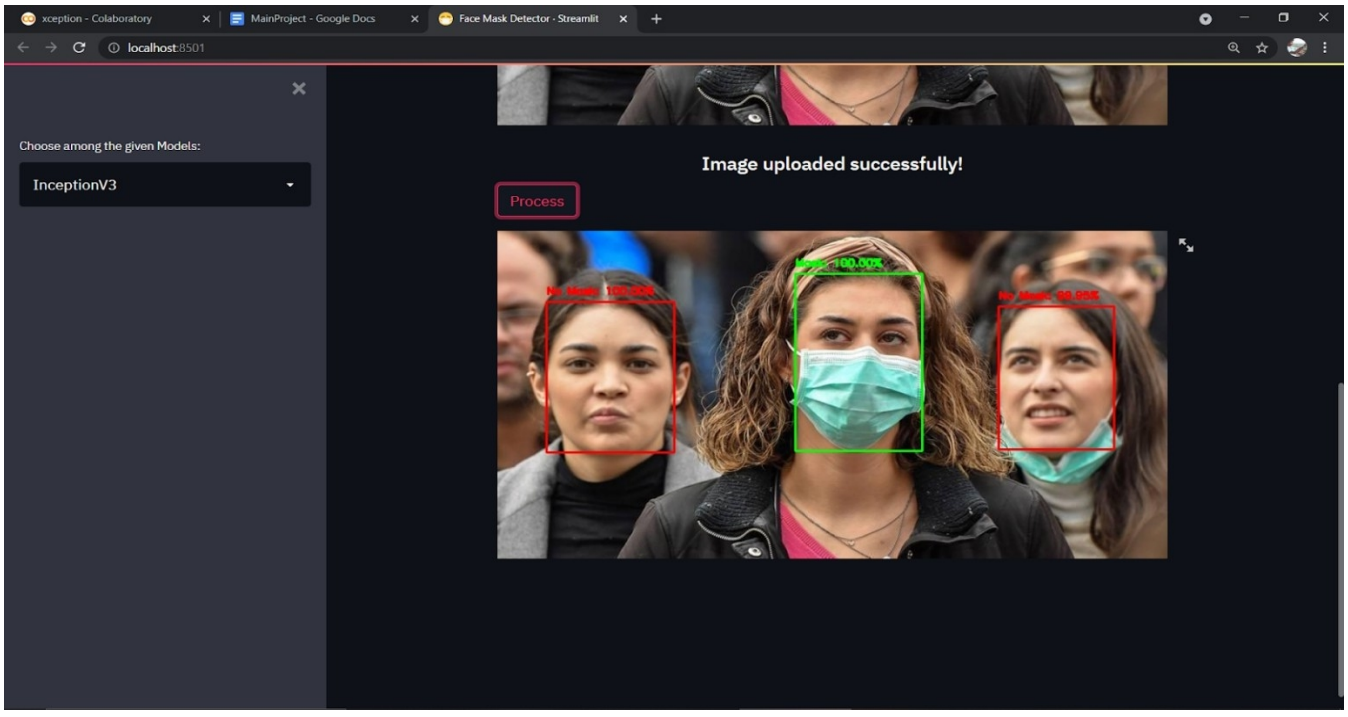

Figure 7.2.5: Classifies the persons with labels when connected with web using Xception model

8.CONCLUSION

The main objective of the proposed system is to identify the unmasked people when moving in crowded places. The corona virus is spreading rapidly as the people are not strictly following the rules of the pandemic. Face mask is a mandatory accessory and the most important preventive measure to be followed by everyone. The proposed system is made by using two different deep learning algorithms which are efficient and useful for object detection. Those algorithms are used to train the model for detecting the mask on a face. An image dataset is used which consists of images of faces with mask and without mask. The algorithms extract the features and label the images as masked and unmasked as per the training. The unmasked people are identified and informed to authorities to alert and warn them. Each algorithm detects with an accuracy of 99.15% and 99.27%. This system is very helpful and makes people strictly wear the mask for the safety of everyone.

9.FUTURE WORK

The proposed system detects the unmasked people and informs authorities.They receive the information and reach them to give warning. It takes more time as it requires manpower. If there is any network issue, then also it is impossible to reach the information on time. So,IoT devices like sensors need to be placed in the surroundings to alert them immediately after detection. Alarms can also be placed and need to be raised at the place where the unmasked people are identified. By the rise of alarm, people can be alerted.

APPENDICES

A.1 InceptionV3 Model:

```
# USAGE
# python detect_mask_image.py --image images/pic1.jpeg
# 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
import numpy as np
import argparse
import cv2
import os
def mask_image():
# construct the argument parser and parse the arguments
	ap = argparse.ArgumentParser()
	ap.add_argument("-i", "--image", required=True,help="path to input image")
	ap.add_argument("-f","--face", type=str,default="face_detector",
	help="path to face detector model directory")
ap.add_argument("-m","--
model",type=str,default="InceptionV3/mask_detector_inceptionv3.model",help="path to
trained face mask detector model")
ap.add_argument("-c","--confidence",type=float,default=0.5,help="minimum probability to
filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk
	print("[INFO] loading face detector model...")
	prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
		weightsPath=os.path.sep.join([args["face"],"res10_300x300_ssd_iter_140000.
	caffemodel"])
	net = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model from disk
	print("[INFO] loading face mask detector model...")
```

```
        model = load_model(args["model"])

# load the input image from disk, clone it, and grab the image spatial
# dimensions
        image = cv2.imread(args["image"])
        orig = image.copy()
        (h, w) = image.shape[:2]
# construct a blob from the image
        blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
        print("[INFO] computing face detections...")
        net.setInput(blob)
        detections = net.forward()

# 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 = image[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)
face = np.expand_dims(face, axis=0)

# pass the face through the model to determine if the face
# has a mask or not
(mask, withoutMask) = model.predict(face)[0]
# 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(image, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)

# show the output image
cv2.imshow("Output", image)
cv2.waitKey(0)
if__name__== "_main_":
mask_image()
```

A.2 Xception Model:

```
# USAGE
# python detect_mask_image.py --image images/pic1.jpeg
# 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
import numpy as np
import argparse
import cv2
import os
def mask_image():
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
help="path to input image")
ap.add_argument("-f", "--face", type=str,
default="face_detector",
help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
default="Xception/mask_detector_xception.model",
help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector model from disk
print("[INFO] loading face detector model...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
"res10_300x300_ssd_iter_140000.caffemodel"])
net = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model from disk
print("[INFO] loading face mask detector model...")
```

```
model = load_model(args["model"])
# load the input image from disk, clone it, and grab the image spatial
# dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(h, w) = image.shape[:2]
# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
(104.0, 177.0, 123.0))
# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()

# 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 = image[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)
face = np.expand_dims(face, axis=0)
# pass the face through the model to determine if the face
# has a mask or not
(mask, withoutMask) = model.predict(face)[0]
# 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(image, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
# show the output image
cv2.imshow("Output", image)
cv2.waitKey(0)
if__name___== "_main_":
mask_image()
```

A.3 Interfacing With Web:

```
import streamlit as st
from PIL import Image, ImageEnhance
import numpy as np
import cv2
import os
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
import detect_mask_image

# Setting custom Page Title and Icon with changed layout and sidebar state
st.set_page_config(page_title='Face Mask Detector', page_icon='
def local_css(file_name):
    """ Method for reading styles.css and applying necessary changes to HTML"""
    with open(file_name) as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)

def mask_image_Inceptionv3_mask_detector():
    global RGB_img
    # load our serialized face detector model from disk
    print("[INFO] loading face detector model...")
    prototxtPath = os.path.sep.join(["face_detector", "deploy.prototxt"])
    weightsPath = os.path.sep.join(["face_detector",
                                    "res10_300x300_ssd_iter_140000.caffemodel"])
    net = cv2.dnn.readNet(prototxtPath, weightsPath)

    # load the face mask detector model from disk
    print("[INFO] loading face mask detector model...")
    model = load_model("./Inception_v3/Inceptionv3_mask_detector.model")
    # load the input image from disk and grab the image spatial
    # dimensions
    image = cv2.imread("./images/out.jpg")
```

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

 # construct a blob from the image
 blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
                              (104.0, 177.0, 123.0))

 # pass the blob through the network and obtain the face detections
 print("[INFO] computing face detections...")
 net.setInput(blob)
 detections = net.forward()

# 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 = image[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)
            face = np.expand_dims(face, axis=0)

    # pass the face through the model to determine if the face
    # has a mask or not
            (mask, withoutMask) = model.predict(face)[0]

     # 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(image, label, (startX, startY - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
mask_image_Inceptionv3_mask_detector()

def mask_image_Xception_mask_detector():
    global RGB_img
    # load our serialized face detector model from disk
    print("[INFO] loading face detector model...")
    prototxtPath = os.path.sep.join(["face_detector", "deploy.prototxt"])
    weightsPath = os.path.sep.join(["face_detector",
                                    "res10_300x300_ssd_iter_140000.caffemodel"])
    net = cv2.dnn.readNet(prototxtPath, weightsPath)

    # load the face mask detector model from disk
```

```
print("[INFO] loading face mask detector model...")
model = load_model("Xception/Xception_mask_detector.model")
# load the input image from disk and grab the image spatial
# dimensions
image = cv2.imread("./images/out.jpg")
(h, w) = image.shape[:2]

# construct a blob from the image
blob = cv2.dnn.blobFromImage(image, 1.0, (300, 300),
                             (104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections
print("[INFO] computing face detections...")
net.setInput(blob)
detections = net.forward()
# 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 = image[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)
        face = np.expand_dims(face, axis=0)

        # pass the face through the model to determine if the face
        # has a mask or not
        (mask, withoutMask) = model.predict(face)[0]

        # 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(image, label, (startX, startY - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(image, (startX, startY), (endX, endY), color, 2)
        RGB_img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
mask_image_Inceptionv3_mask_detector()

def mask_detection():
    local_css("css/styles.css")
    activities = ["Xception", "InceptionV3"]
    st.set_option('deprecation.showfileUploaderEncoding', False)
    choice = st.sidebar.selectbox("Choose among the given Models:", activities)
```

```
    if choice == ‘Xception’:
        st.markdown('<h1 align="center">
        st.markdown("### Upload your image here ⬇  ")
        image_file = st.file_uploader("", type=['jpg']) # upload image
        if image_file is not None:
            our_image = Image.open(image_file) # making compatible to PIL
            im = our_image.save('./images/out.jpg')
            saved_image = st.image(image_file, caption='', use_column_width=True)
                        st.markdown('<h3 align="center">Image uploaded successfully!</h3>',
unsafe_allow_html=True)
            if st.button('Process'):
                mask_image_ResNet50_mask_detector()
                st.image(RGB_img, use_column_width=True)

    if choice == 'InceptionV3':
        st.markdown('<h1 align="center">
        st.markdown("### Upload your image here ⬇  ")
        image_file = st.file_uploader("", type=['jpg']) # upload image
        if image_file is not None:
     our_image = Image.open(image_file) # making compatible to PIL
            im = our_image.save('./images/out.jpg')
            saved_image = st.image(image_file, caption='', use_column_width=True)
                        st.markdown('<h3 align="center">Image uploaded successfully!</h3>',
unsafe_allow_html=True)
            if st.button('Process'):
                mask_image_Inceptionv3_mask_detector()
                st.image(RGB_img, use_column_width=True)

mask_detection()
```

REFERENCES

[1] Mohammad Marufur Rahman,Saifuddin Mahmud et al.,,"*An Automated System To Limit COVID-19 Using Facial Mask Detection In Smart City Network" ,* IEEE 2020.

[2] Abdellah Oumina,Mustapha Hamdi et al.,, "*Control The COVID-19 Pandemic: Face Mask Detection Using Transfer Learning*", IEEE 2021.

[3] Mohamed Loey,Gunasekaran Manogaran et al., "*Fighting Against COVID-19: A Novel Deep Learning Model Based On YOLO-v2 with ResNet-50 For Medical Face Mask Detection"*, November 2020.

[4] Jan Zhang,Feiteng Han et al., "*A Novel Detection Framework About Conditions Of Wearing Face Mask For Helping Control The Spread Of COVID-19"*, IEEE 2021.

[5] G. Jignesh Chowdary, Narinder Singh Punn, et al., " *Face Mask Detection using Transfer Learning of InceptionV3",* October 2020.

[6] Aniruddha Srinivas Joshi, Shreyas Srinivas Joshiy, et al., "*Deep Learning Framework to Detect Face Masks from Video Footage*", 2021.

[7] Alok Negi, Krishan Kumar, et al., " *Deep Neural Architecture for Face Mask Detection on Simulated Masked Face Dataset against Covid-19 Pandemic",* 2021.

[8] Isunuri B Venkateswarlu, Jagadeesh Kakarla, et al.,"*Face Mask Detection Using MobileNet And Global Pooling Block*" , IEEE 2020.

[9] Harish Adusumalli, D.Kalyani et al., "*Face Mask Detection Using OpenCV* " , IEEE 2021.

[10] Bin Xue, Jianpeng Hu et al., "*Intelligent Detection And Recognition System For Mask Wearing Based On Improved RetinalFace Algorithm* " , IEEE 2021.