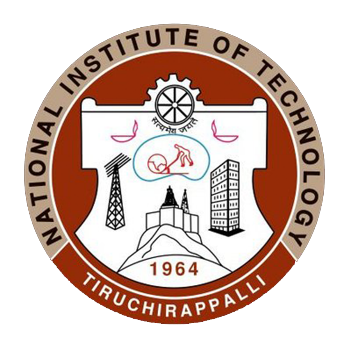
**NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI-620015**



**Department of Computer Applications**

**Project Report Phase-I**

**Python And R Programming**

**CA 727**

**REAL-TIME FACE MASK RECOGISATION SYSTEM**

***Submitted To: Submitted By:***

**Dr. S. SANGEETHA Akshay Jain (205119007)**

**Rakshat Puranik (205119076)**

**Sumit Parmar (205119102)**

**INTRODUCTION**

**Subtopics**

1. [What is Face Detection?](https://www.mygreatlearning.com/blog/real-time-face-detection/#ad1)
2. [Face Detection Methods](https://www.mygreatlearning.com/blog/real-time-face-detection/#ad2)
3. [Face recognition](https://www.mygreatlearning.com/blog/real-time-face-detection/#ad4)

**What** **is Face Detection?**

The goal of face detection is to determine if there are any faces in the image or video. If multiple faces are present, each face is enclosed by a bounding box and thus we know the location of the faces.

## ****Face Detection Methods****

There are two main approaches for Face Detection:

1. Feature Base Approach
2. Image Base Approach

## ****Face Recognition****

## Face recognition is a method of identifying or verifying the identity of an individual using their face. There are various algorithms that can do face recognition but their accuracy might vary.

**BLOCK DIAGRAM**

**Phase #1: Training Face Mask Detector**

Load face mask dataset

Train face mask classifier with Keras/TensorFlow

Serialize face mask classifier to disk

**Phase #2: Apply Face Mask Detector**

Load face mask dataset

Train face mask classifier with Keras/TensorFlow

Serialize face mask classifier to disk

Show results

Apply face mask classifier to each face ROI to determine “mask” or “no mask”

**MODULES**

1. Mask Training
2. Mask Detection Using Image
3. Mask Detection Using Video

A picture containing bag

Description automatically generated

**Code for Mask Training Module**

import numpy as np

import os

import matplotlib.pyplot as plt

from imutils import paths

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 ImageDataGenerator

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

import PIL

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential

dataset=r'C:\Python38\Projects\face-mask-detector\dataset'

imagePaths=list(paths.list\_images(dataset))

data=[]

labels=[]

for i in imagePaths:

label=i.split(os.path.sep)[-2]

labels.append(label)

image=load\_img(i,target\_size=(224,224))

image=img\_to\_array(image)

image=preprocess\_input(image)

data.append(image)

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

labels=np.array(labels)

lb=LabelBinarizer()

labels=lb.fit\_transform(labels)

labels=to\_categorical(labels)

train\_X,test\_X,train\_Y,test\_Y=train\_test\_split(data,labels,test\_size=0.20,stratify=labels,random\_state=10)

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,vertical\_flip=True,fill\_mode='nearest')

baseModel=MobileNetV2(weights='imagenet',include\_top=False,input\_tensor=Input(shape=(224,224,3)))

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)

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

for layer in baseModel.layers:

layer.trainable=False

learning\_rate=0.001

Epochs=20

BS=12

opt=Adam(lr=learning\_rate,decay=learning\_rate/Epochs)

model.compile(loss='binary\_crossentropy',optimizer=opt,metrics=['accuracy'])

H=model.fit(

aug.flow(train\_X,train\_Y,batch\_size=BS),

steps\_per\_epoch=len(train\_X)//BS,

validation\_data=(test\_X,test\_Y),

validation\_steps=len(test\_X)//BS,

epochs=Epochs)

model.save(r'C:\Python38\Projects\face-mask-detector\mobilenet\_v2.model')

predict=model.predict(test\_X,batch\_size=BS)

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

print(classification\_report(test\_Y.argmax(axis=1),predict,target\_names=lb.classes\_))

# 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["acc"], label="train\_acc")

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

plt.title("Training Loss and Accuracy")

plt.xlabel("Epoch #")

plt.ylabel("Loss/Accuracy")

plt.legend(loc="lower left")

plt.savefig(r'C:\Python38\Projects\face-mask-detector\plot\_v2.png')

**SNAPSHOTS**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, table

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

--END--