**Face-Mask Recognition using Deep Learning**

This project uses a Deep Neural Network, more specifically a Convolutional Neural Network, to differentiate between images of people with and without masks. The CNN manages to get an accuracy of 98.2% on the training set and 97.3% on the test set. Then the stored weights of this CNN are used to classify as mask or no mask, in real time, using OpenCV. With the webcam capturing the video, the frames are preprocessed and fed to the model to accomplish this task. The model works efficiently with no apparent lag time between wearing and removing mask and display of prediction.

The model is also capable of predicting multiple faces with or without masks at the same time.

The main objective of this project is to determine whether a person is wearing a mask or not in real-time. This analysis helps us prevent the transmission of deadly viruses such as COVID-19, etc. It also is an effective way of mask detection for a large gathering of people, such as in a function, crowds or anything which involves gathering of people. This project is used for effective detection and is also simple to implement in real life applications such as cameras, CCTVS etc.

**Code:**

import numpy as np

import keras

import keras.backend as k

from keras.layers import Conv2D,MaxPooling2D,SpatialDropout2D,Flatten,Dropout,Dense

from keras.models import Sequential,load\_model

from keras.optimizers import Adam

from keras.preprocessing import image

import cv2

import datetime

# BUILDING MODEL TO CLASSIFY BETWEEN MASK AND NO MASK

#model=Sequential()

#model.add(Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))

#model.add(MaxPooling2D() )

#model.add(Conv2D(32,(3,3),activation='relu'))

#model.add(MaxPooling2D() )

#model.add(Conv2D(32,(3,3),activation='relu'))

#model.add(MaxPooling2D() )

#model.add(Flatten())

#model.add(Dense(100,activation='relu'))

#model.add(Dense(1,activation='sigmoid'))

#

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

#

#from keras.preprocessing.image import ImageDataGenerator

#train\_datagen = ImageDataGenerator(

# rescale=1./255,

# shear\_range=0.2,

# zoom\_range=0.2,

# horizontal\_flip=True)

#

#test\_datagen = ImageDataGenerator(rescale=1./255)

#

#training\_set = train\_datagen.flow\_from\_directory(

# 'train',

# target\_size=(150,150),

# batch\_size=16 ,

# class\_mode='binary')

#

#test\_set = test\_datagen.flow\_from\_directory(

# 'test',

# target\_size=(150,150),

# batch\_size=16,

# class\_mode='binary')

#

#model\_saved=model.fit\_generator(

# training\_set,

# epochs=10,

# validation\_data=test\_set,

#

# )

#

#model.save('mymodel.h5',model\_saved)

#To test for individual images

mymodel=load\_model('mymodel.h5')

test\_image=image.load\_img(r'D:/winsem/ai/Project2/FaceMaskDetector/test/with\_mask/1-with-mask.jpg',

target\_size=(150,150,3))

test\_image

test\_image=image.img\_to\_array(test\_image)

test\_image=np.expand\_dims(test\_image,axis=0)

mymodel.predict(test\_image)[0][0]

# IMPLEMENTING LIVE DETECTION OF FACE MASK

mymodel=load\_model('mymodel.h5')

cap=cv2.VideoCapture(0,cv2.CAP\_DSHOW)

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

while cap.isOpened():

\_,img=cap.read()

face=face\_cascade.detectMultiScale(img,scaleFactor=1.1,minNeighbors=4)

for(x,y,w,h) in face:

face\_img = img[y:y+h, x:x+w]

cv2.imwrite('temp.jpg',face\_img)

test\_image=image.load\_img('temp.jpg',target\_size=(150,150,3))

test\_image=image.img\_to\_array(test\_image)

test\_image=np.expand\_dims(test\_image,axis=0)

pred=mymodel.predict(test\_image)[0][0]

if pred==1:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)

cv2.putText(img,'NO MASK',((x+w)//2,y+h+20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,0,255),3)

else:

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)

cv2.putText(img,'MASK',((x+w)//2,y+h+20),cv2.FONT\_HERSHEY\_SIMPLEX,1,(0,255,0),3)

datet=str(datetime.datetime.now())

cv2.putText(img,datet,(400,450),cv2.FONT\_HERSHEY\_SIMPLEX,0.5,(255,255,255),1)

cv2.imshow('img',img)

if cv2.waitKey(1)==ord('q'):

break

cap.release()

cv2.destroyAllWindows()