**PROJECT: Face Mask Detection Using Camera**

**Introduction**

Technological advancements has led us to a more active area of research onto which various current and future upcoming tech-innovations are highly focusing onto. This is *Computer Vision.* This refers to a technical field that involves using well designed algorithms to give computer machines the capability to *detect, interpret* and *understand* multimedia objects e,g images and videos.

With promisingly advancing technology, the concept areas of application of artificial intelligence (AI) and machine learning (ML) seems to have no boundaries or limitations. For instance, self-driving cars, image and facial recognition, robotic systems among other more different areas make use of this technology.

Facial detection process starts with *image recognition* where the algorithm handler module in this case raspberry pi, a micro-computer device with high computing capabilities and processing power capable of executing advanced image processing algorithms. For images are used as the input data, they are first processed to give them a suitable format for that purpose through a task termed as *image processing.* This involves converting the image files from their actual formats e.g JPEG, JPG or PNG to more usable data consumable by the support algorithms.

We will be using *Python* as the primary support language for the project development due its various significance which include:

* Simple to learn for beginners
* Simple syntax for easier code understanding
* Large support community hence easier to learn and fix problems.
* Wide range of support libraries developed in Python hence easier

import in to the target project code algorithm.

**TensorFlow**

This forms one of the favorable and popularly known and used python libraries in areas of advanced computations and machine vision. The library provides a variety of utility functions for image data processing which is not limited to image data extraction, resizing and transformation where large set of images are to be ferried across different endpoints.

In application, the library function ***tensorflow.io.read\_file*** which requires and argument input for this case the file name of the image to be processed and the resultant output is a tensor of type ***tensorflow.string.*** For the input to the function is an image, the output is a raw byte data of the image file for instance;

import tensorflow #imports tensorflow

values = tf.io.read\_file('soccer\_ball.jpg')

#creates image object

As a machine learning tensorflow supports other deep learning APIs written in Python language such as keras.

**Keras**

This is a Python language oriented deep learning API that runs on top of tensorflow machine learning platform. Its a quite light-weight tool that has the following qualities:

* Its simple

- It reduces developers cognitive load

* Its flexible

- This tool adopts the principle of disclosure of involved

algorithm complexity.

* Its powerful

- It provides performance strength and scalability

Keras has its core data structures as **layers** and **models.**

Its brought in to python language application using the syntax

import tensorflow #imports tensorflow

from tensorflow import keras

#imports keras module

Keras will form one of our prime tools in actualizing the real-time face mask detection project

**OpenCV**

This forms our second essential feature in the realization of our project. It refers to Open Source Computer Vision(OpenCV) library. It is an open-source software library highly applicable in areas of computer vision and machine learning.

The OpenCV library has a number of optimized algorithms that finds applicability in multiple areas of technological advancement and improvements across various industries.

To be used in a python supported code we invokeit using the command;

import cv2

**Training Dataset**

This is a fundamental aspect of this project. Dataset allows training the project handler algorithm so as to improve its prediction efficiency and performance. For this case we will be using a dataset labeled *face-mask-dataset* and has te following properties:

* Number of images: 1376
* Number of images of people wearing face masks: 690
* Number of images of people not wearing face masks: 686

The project operation is achieved through *test* and *train* python code algorithms shown

*test.py*

# ---------------------------------------------------------

# Prediction:

# - Face Mask Detection

# -------------------

# - Test model algorithm

# - - - - - - - - - -

# December /021

# - - - - - - - - - -

# ---------------------------------------------------------

import cv2

import numpy as np

from keras.models import load\_model

model=load\_model("./model-010.h5")

# imports essential support libraries

labels\_dict={0:'without mask',1:'mask'}

color\_dict={0:(0,0,255),1:(0,255,0)}

# project support dictionaries

size = 4

webcam = cv2.VideoCapture(0)

# Camera object dfn

cascade = "haarcascade\_frontalface\_default.xml"

# use this xml file

# We load the xml file

classifier = cv2.CascadeClassifier(cascade)

while True:

(rval, im) = webcam.read()

im=cv2.flip(im,1,1)

#Flip to act as a mirror

# Resize the image to speed up detection

mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))

# detect MultiScale / faces

faces = classifier.detectMultiScale(mini)

# Draw rectangles around each face

for f in faces:

(x, y, w, h) = [v \* size for v in f]

#Scale the shapesize backup

#Save just the rectangle faces

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

resized=cv2.resize(face\_img,(150,150))

normalized=resized/255.0

reshaped=np.reshape(normalized,(1,150,150,3))

reshaped = np.vstack([reshaped])

result=model.predict(reshaped)

#print(result)

label=np.argmax(result,axis=1)[0]

cv2.rectangle(im,(x,y),(x+w,y+h),color\_dict[label],2)

cv2.rectangle(im,(x,y-40),(x+w,y),color\_dict[label],-1)

cv2.putText(im, labels\_dict[label], (x, y-10),cv2.FONT\_HERSHEY\_SIMPLEX,0.8,(255,255,255),2)

# Show the image

cv2.imshow('LIVE', im)

key = cv2.waitKey(10)

# if Esc key is press then break out of the loop

if key == 27: #The Esc key

break

# Stop video

webcam.release()

# Close all started windows

cv2.destroyAllWindows()

This tests the model with a defined number of dataset images.

train.py

# ---------------------------------------------------------

# Prediction:

# - Face Mask Detection

# -------------------

# - Train model algorithm

# - - - - - - - - - -

# December /021

# - - - - - - - - - -

# ---------------------------------------------------------

from keras.optimizers import RMSprop

from keras.preprocessing.image import ImageDataGenerator

import cv2

from keras.models import Sequential

from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout

from keras.models import Model, load\_model

from keras.callbacks import TensorBoard, ModelCheckpoint

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

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

import imutils

import numpy as np

# imports essential support libraries

model =Sequential([

Conv2D(100, (3,3), activation='relu', input\_shape=(150, 150, 3)),

MaxPooling2D(2,2),

Conv2D(100, (3,3), activation='relu'),

MaxPooling2D(2,2),

Flatten(),

Dropout(0.5),

Dense(50, activation='relu'),

Dense(2, activation='softmax')

])

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

TRAINING\_DIR = "./train"

train\_datagen = ImageDataGenerator(rescale=1.0/255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

# Training model object

train\_generator = train\_datagen.flow\_from\_directory(TRAINING\_DIR,

batch\_size=10,

target\_size=(150, 150))

VALIDATION\_DIR = "./test"

validation\_datagen = ImageDataGenerator(rescale=1.0/255)

validation\_generator = validation\_datagen.flow\_from\_directory(VALIDATION\_DIR,

batch\_size=10,

target\_size=(150, 150))

checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val\_loss',verbose=0,save\_best\_only=True,mode='auto')

history = model.fit\_generator(train\_generator,

epochs=10,

validation\_data=validation\_generator,

callbacks=[checkpoint])

This trains the model ready for the operation by subjecting it to the specified dataset.

The project execution is as follows

1. Import library support modules as shown in *train.py*

# Prediction:

# - Face Mask Detection

# -------------------

# - Train model algorithm

# - - - - - - - - - -

# December /021

# - - - - - - - - - -

# ---------------------------------------------------------

from keras.optimizers import RMSprop

from keras.preprocessing.image import ImageDataGenerator

import cv2

from keras.models import Sequential

from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout

from keras.models import Model, load\_model

from keras.callbacks import TensorBoard, ModelCheckpoint

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

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

import imutils

import numpy as np

# imports essential support libraries

1. Build the neural network

This consists of convolution network of two pairs of Conv and MaxPool layers essential to extract features from the dataset

model = Sequential([

Conv2D(100, (3,3), activation='relu', input\_shape=(150, 150, 3)),

MaxPooling2D(2,2),

Conv2D(100, (3,3), activation='relu'),

MaxPooling2D(2,2),

Flatten(),

Dropout(0.5),

Dense(50, activation='relu'),

Dense(2, activation='softmax')

])

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

1. Image data generation

This is also termed as augmentation

TRAINING\_DIR = "./train"

train\_datagen = ImageDataGenerator(rescale=1.0/255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

# Training model object

train\_generator = train\_datagen.flow\_from\_directory(TRAINING\_DIR,

batch\_size=10,

target\_size=(150, 150))

VALIDATION\_DIR = "./test"

validation\_datagen = ImageDataGenerator(rescale=1.0/255)

validation\_generator = validation\_datagen.flow\_from\_directory(VALIDATION\_DIR,

batch\_size=10,

target\_size=(150, 150))

1. Initialize a callback checkpoint to keep saving best model after each epoch while training

checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val\_loss',verbose=0,save\_best\_only=**True**,mode='auto')

1. Train the model

history = model.fit\_generator(train\_generator,

epochs=10,

validation\_data=validation\_generator,

callbacks=[checkpoint])

This is by executing the following command

python3 train.py

in the project working directory from the terminal

Hence using openCV we can run the project test to realize the performance by using the command

python3 test.py

**Discussion**

From the above project implementation procedure, it is true that the objective of designing a *Face mask detector* system can be well achieved using Python as a programming language for the models and *Keras* and *OpenCV* as thesupporting libraries for algorithm performance in *computer vision* and *image recognition.* This requires good performing system for data processing power as the model efficiency in giving the best results highly depends on the training level for different datasets its subjected to.